SC23-0059-5
File No. S370-36

Program Product

MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning

MVS/System Product - JES3
Version 2 Release 2.1

Program Number 5665-291

IBM

SC23-0059-5
File No. S370-36

Program Product

MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning

Program Number 5665-291

IBM

Sixth Edition (December 1987)

This is a major revision of SC23-0059-4. See the Summary of Amendments following the contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 2 Release 2.1 and to all subsequent releases of MVS/System Product-JES3 (5665-291) until otherwise indicated in new editions or Technical Newsletters. The previous edition still applies to the JES3 component of MVS/System Product Version 2 Release 1.5 (5665-291) and may now be ordered using the temporary order number ST00-2163. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products 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 in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This book is intended for any JES3 complex that runs Multiple Virtual Storage/Extended Architecture (MVS/XA). JES3 system programmers or anyone who is responsible for installing, initializing, customizing, or tuning JES3 should use this book. This book provides information about:

- Preparing for the installation of JES3
- Initializing and tailoring JES3
- Tuning JES3

Required Books

Before reading this book, you should read the following books:

- *MVS/Extended Architecture JES3 Introduction*, GC23-0049
- *MVS/Extended Architecture Conversion Notebook*, GC28-1143
- *MVS/Extended Architecture JES3 Conversion Notebook*, SC28-1412

Organization of this Book

This book contains two parts: ***Initialization and Tuning Guidelines,*** which contains the book's first eleven chapters, and ***Initialization Reference,*** which contains the twelfth chapter and two appendices.

Part I: Initialization and Tuning Guidelines

- *Chapter 1, "Planning for JES3"* lists what your installation plan should address and introduces you to the process of initializing and customizing JES3. This chapter introduces factors that affect JES3 performance and provides an overview of the JES3 tuning process.

- *Chapter 2, "JES3 Job Management"* describes the different phases of JES3 processing. It describes how JES3 initialization statements and user exits influence the way JES3 handles jobs during each phase of processing (except converter/interpreter (C/I) service, which the next chapter explains).

- *Chapter 3, "Defining and Managing C/I Service"* describes how to define the functions needed for JES3 converter/interpreter (C/I) service. This chapter also explains how to influence C/I service using user exit routines, how to update the procedure libraries used by C/I service, how to tune C/I service, and how to recover from C/I functional subsystem address space failures.

- *Chapter 4, "Defining and Managing Spool Data Sets"* describes how to allocate and format spool data sets, define spool space allocation units, and define spool partitions. This chapter also explains how to tune the spool configuration and how to recover from spool I/O errors.

- *Chapter 5, "Defining Consoles and Message Routing"* describes how to define and manage consoles. This chapter explains the types and functions of consoles in an MVS/JES3 installation. This chapter also describes how you can control message traffic.

- *Chapter 6, "Defining and Managing JES3 Resources"* describes how to define checkpoint data set(s), the JES3 step library, I/O devices, and volumes. This chapter also explains how to initialize the IBM 3850 Mass Storage System (MSS) and discusses how JES3 manages resources.

- *Chapter 7, "Defining and Managing JES3 Mains and Storage"* describes how to define and manage JES3 mains and storage. This chapter also discusses partitionable processor complexes.

- *Chapter 8, "JES3 Remote Job Processing"* discusses the use of and features of binary synchronous communication (BSC) remote job processing and systems network architecture (SNA) remote job processing.

- *Chapter 9, "JES3 Networking"* explains how to add your installation to a job entry network and how to initialize your installation to send and receive jobs over a network.

- *Chapter 10, "JES3 Start-Up and Initialization"* describes the different ways you can start JES3. This chapter also describes how to customize the JES3 start procedure, and how to test your JES3 initialization stream.

- *Chapter 11, "JES3 Recovery"* discusses the internal recovery procedures of JES3 and steps you must take to recover from hardware or software failures.

Part II: Initialization Reference

- *Chapter 12, "Initialization Statement Reference"* explains the format and parameters of JES3 initialization statements.
- *Appendix A, "RMT Option Statements"* describes the statements that you can code to generate a remote terminal program for programmable terminals.
- *Appendix B, "Remote Terminal Bootstrap (RTPBOOT)"* shows the object code for the program that you can use to load into storage the 1130 RTP program.

How to Use this Book

To use this book you need not read it from cover to cover. Depending on your knowledge of JES3 and your information needs, you can read selected chapters or even selected topics within a chapter. The first time you come to this book, however, you should read the preface, the table of contents, the figure list, and Chapter 1. This will help you understand the type of information that the book contains and the organization of the information.

The task you are doing will determine which parts of the book you should read:

- If you are preparing to install JES3, read Chapter 1, "Planning for JES3." It gives an overview of the installation requirements.
- If you are defining JES3 job management policy for your installation, read Chapter 2, "JES3 Job Management." This chapter explains how to use JES3 initialization statements to define policies for job interpretation, resource allocation, job scheduling and selection, and the processing of job output. The following chapter includes details about the information you need to define policies for job interpretation (converter/interpreter service).
- If you are defining or tuning converter/interpreter (C/I) service, read Chapter 3, "Defining and Managing C/I Service." This chapter explains how to define the JES3 global address space and/or C/I functional subsystem address spaces to handle JES3 C/I service without constraining private area virtual storage. It also explains how to update the procedure libraries used during C/I service.
- If you are defining spool data sets, read Chapter 4, "Defining and Managing Spool Data Sets." This chapter explains how to allocate and format spool data sets, and how to use spool partitioning to increase spool reliability, availability, and serviceability.
- If you are defining JES3 resources, such as the JES3 checkpoint data set(s), the JES3 step library, I/O devices, volumes, processors, and storage, read:
  - Chapter 5, "Defining Consoles and Message Routing"
  - Chapter 6, "Defining and Managing JES3 Resources"
  - Chapter 7, "Defining and Managing JES3 Mains and Storage"

  These chapters explain how to use initialization statements to define and tune those resources.

- If you are defining remote work stations, read Chapter 8, "JES3 Remote Job Processing." This chapter describes how to use JES3 initialization statements to define both binary synchronous communication (BSC) remote work stations and systems network architecture (SNA) remote work stations.

- If you are defining programmable remote work stations, besides reading Chapter 8, "JES3 Remote Job Processing," you should also read "Generating Remote Terminal Processing Programs," Appendix A, "RMT Option Statements," and Appendix B, "Remote Terminal Bootstrap (RTPBOOT)."

- If you are defining your installation as part of a job entry network, read Chapter 9, "JES3 Networking."

- If you are changing the JES3 start procedure, read the topics "Starting JES3" and "Initializing JES3" in Chapter 10, "JES3 Start-Up and Initialization."

- If you have changed the initialization stream, read "Testing Your Initialization Stream," in Chapter 10, "JES3 Start-Up and Initialization." This section describes how to use the initialization stream checker utility to test the validity of the new initialization stream.

- If you are defining JES3 recovery policy or are defining the dynamic system interchange (DSI) procedure, read Chapter 11, "JES3 Recovery." If you are trying to recover from spool I/O errors, read "Recovering From Spool I/O Errors," in Chapter 4, "Defining and Managing Spool Data Sets."

- If you are tuning JES3, refer to the chapter that describes the resource you are tuning.

Many tasks require that you code JES3 initialization statements. These statements, defined in Chapter 12, "Initialization Statement Reference," enable you to define to JES3 your installation's job management policies and the resources that JES3 can use. Within the chapter, statement descriptions appear in alphabetical order with the statement name appearing at the top of each page. This arrangement lets you quickly find a statement description without going to the index.

Many statement parameters influence or change the values of other initialization parameters. For example, a particular parameter coded on one statement may override a parameter coded on another statement. Figure 10-4, in Chapter 10, "JES3 Start-Up and Initialization," identifies statement parameters that affect each other.

If you specify an invalid subparameter on specific JES3 initialization statements, JES3 substitutes the subparameter's default value and continues to run. To find out which initialization statements and subparameters select defaults, see Figure 10-5 in Chapter 10, "JES3 Start-Up and Initialization."

As a general rule, do not use generic terms, such as ALL or NONE, as variable names when coding JES3 initialization statements.

Publications Referenced by this Book

Throughout the text, this book refers to other books. Most of these references use shortened book titles. For example, the shortened title *JES3 User Modifications and Macros* refers to the book *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.* The full book title and order numbers of all referenced books follow the shortened titles.

The first list contains the full title and order number of all referenced JES3 publications. Besides providing the name and order number of each book, this list also gives a short description of the information contained in the book.

Following the list of JES3 publications is a list of referenced non-JES3 books. This list shows the shortened title of each book (if applicable) and the book's full title and order number.

If this book references another book by its full title, the word *none* appears in the table where a shortened title would normally appear. In the column next to the word *none* is the book's full title and order number.

Related JES3 Publications

- *MVS/Extended Architecture JES3 Introduction*, GC23-0049

The *Introduction* describes JES3 from an external point of view, emphasizing what JES3 is and how you can use it. This book provides a knowledge base that makes the information in the rest of the JES3 library more meaningful.

- *MVS/Extended Architecture JES3 Logic Library Vols. 1 - 11:* LY28-1529, LY28-1531, LY28-1533, LY28-1535, LY28-1537, LY28-1539, LY28-1541, LY28-1543, LY28-1545, LY28-1547, LY28-1549

These eleven volumes can help you understand the details of JES3's organization and how individual JES3 modules work. *Vol. 1: Logic Overview* provides a high-level explanation of how JES3 works. Volume 1 can help you initialize and tuning JES3. *Vol. 11: Logic Reference* contains module description summaries, data area description summaries, and control block chaining diagrams, among other reference materials. You can use the *Logic Library* as a directory for getting into the program listings.

- *MVS/Extended Architecture Message Library: JES3 Messages*, GC23-0062

This book describes all JES3 messages. The messages are organized by message number. This book describes the causes of each message, accompanying actions by JES3, and suggests appropriate responses. This book also lists the names of the detecting, issuing, and containing modules for each message and, where applicable, includes problem determination information.

- *MVS/Extended Architecture Operations: JES3 Commands*, SC23-0063

This book describes how to operate JES3. It includes directions for using consoles; for starting, stopping, and restarting JES3; for controlling jobs, devices, volumes, and data; and for using utility programs. The book explains each JES3

operator command in detail. The book also contains an appendix that lists all JES3 commands, and shows where to look for commands.

- *MVS/Extended Architecture Operations: JES3 Command Syntax*, SX23-0012

This booklet contains a summary of all JES3 commands. Detailed descriptions are absent, but the booklet shows the complete syntax of every JES3 command.

- *MVS/Extended Architecture JES3 Diagnosis*, LC28-1370

This book describes tools that you can use for debugging JES3. Included are descriptions of commands that collect information about JES3 processing, details about how you can obtain dumps and traces, and explanations of the contents of dumps and traces.

- *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros*, LC28-1372

This book can help you tailor JES3 beyond the point you could reach by using JES3 initialization statements, commands, and control statements. The book includes descriptions of JES3 user exits, guidelines for writing dynamic support programs, and descriptions of JES3 macro-instructions.

Related Non-JES3 Books

Short Title	Full Title and Order Number
Assembler Language	*Assembler H Version 2 Application Programming: Language Reference*, GC26-4037
	Assembler H Version 2 Application Programming Guide, GC26-4036
Bulk Data Transfer Facility	*MVS/Bulk Data Transfer Facility: Initialization and Network Definition*, SC28-1314
	MVS/Bulk Data Transfer Facility: Operator's Guide, SC28-1322
Catalog Administrator's Guide	*MVS/Extended Architecture Catalog Administrator's Guide*, GC26-4138
Checkpoint/Restart	*MVS/Extended Architecture Checkpoint/Restart*, GC26-4139
Configuration Program Guide	*MVS/Extended Architecture Configuration Program Guide and Reference*, GC28-1335
Data Management	*MVS/Extended Architecture System Programming Library: Data Management*, GC26-4149
Data Management Macros	*MVS/Extended Architecture Data Management Macro Instructions*, GC26-4041
JCL	*MVS/Extended Architecture JCL User's Guide*, GC28-1351 *MVS/Extended Architecture JCL Reference*, GC28-1352
MVS Initialization and Tuning	*MVS/Extended Architecture System Programming Library: Initialization and Tuning*, GC28-1149
NetView Installation Guide	*NetView Installation and Administration Guide*, SC30-3360
Print Services Facility	*Print Services Facility/MVS System Programmer's Guide*, SH35-0091
Recovery and Reconfiguration	*MVS/Extended Architecture Planning: Recovery and Reconfiguration*, GC28-1160
RMF Monitor 1 and 2	*MVS/XA RMF Monitor 1 and 2 Reference and User's Guide*, LC28-1556
Service Aids	*MVS/Extended Architecture System Programming Library: Service Aids*, GC28-1159
System Commands	*MVS/Extended Architecture Operations: System Commands*, GC28-1206
System Generation Reference	*MVS/Extended Architecture System Generation Reference*, GC26-4148
System Logic Library Vol. 2	*MVS/Extended Architecture System Logic Library: Volume 2*, LY28-1210

Short Title	Full Title and Order Number
System Macros and Facilities	*MVS/Extended Architecture System Programming Library: System Macros and Facilities*, GC28-1150 and GC28-1151
System Management Facilities	*MVS/Extended Architecture System Programming Library: System Management Facilities*, GC28-1153
System Modifications	*MVS/Extended Architecture System Programming Library: System Modifications*, GC28-1152
TSO/E User's Guide	*TSO/E User's Guide*, SC28-1333
TSO Commands	*TSO/E Command Reference*, SC28-1307
Utilities	*MVS/Extended Architecture Utilities*, GC26-4150
Introduction to VTAM	*Introduction to VTAM*, GC27-6987
VTAM SPG	*Advanced Communications Function for VTAM Version 3 Programming*, SC23-0115
none	*IBM 3480 Magnetic Tape Subsystem Planning and Migration Guide*, GC35-0098
none	*IBM System/370 and 4300 Processors Bibliography*, GC20-0001
none	*IBM 3774 and 3775 Operating Procedures Guide*, GA27-3094
none	*Operator's Library: OS/VS2 Remote Terminals*, GC38-0228
none	*Systems Network Architecture General Information*, GA27-3102

Program Products Referenced by this Book

This book refers to the following program products:

- Advanced Communication Facility/Virtual Telecommunication Access Method (ACF/VTAM) Version 3 (5665-289) -- This book refers to this program product as VTAM.
- MVS/Bulk Data Transfer (5665-302) and MVS/Bulk Data Transfer Version 2 (5665-264) -- This book refers to these program products as MVS/BDT.
- NetView™[1] (5665-362)
- Print Services Facility (5665-275)
- Time Sharing Option (TSO) Extensions (5665-285) -- This book refers to this program product as TSO.
- Virtual Storage Personal Computing (VSPC) (5665-283)

Installed User Program Referenced by this Book

This book refers to the following Installed User Program:

- JES3 Monitoring Facility II (JMF II) (5796-PLW)

Your Role in Improving this Book

At the end of this book is a Reader's Comment Form. This form is your means to communicate with the writer of this book. Use this form to tell the writer about problems that you have when using the book, to send the writer suggestions about how to improve the book, or to tell the writer when there is something about the book that you like. Your comments will help improve the quality of this book.

---

1 NetView is a trademark of International Business Machines Corporation.

Contents

Part I: Initialization and Tuning Guidelines

Figures

Summary of Amendments

Summary of Amendments
for SC23-0059-5
MVS/System Product - JES3 Version 2 Release 2.1

This major revision applies to *MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning*, SC23-0059-5.

A new chapter, "Defining Consoles and Message Routing," is added in this revision. This chapter defines the types and functions of consoles and describes how to define and use consoles to control your installation. This chapter also provides information about how to control message traffic.

- MCS consoles can now coexist with JES3. That is, you can use MCS consoles attached to the JES3 global to enter most JES3 commands as well as MVS commands. Previously, you could not enter JES3 commands from MCS consoles.

- Information is added about defining consoles to MVS. You no longer define devices during system generation (SYSGEN). Instead, you define devices in members of SYS1.PARMLIB and in the input data set for the MVS configuration program.

- A new section, "Defining MCS Consoles," describes MCS console support. This section lists the subset of JES3 commands that you cannot enter from MCS consoles.

- Information is added about using MCS consoles to send MVS commands to another processor.

- Information is added about the JES3 and MVS action message retention facilities. You can use either action message retention facility to retrieve unresolved action messages.

- A new section, "Defining Message Routing," describes how you can control message traffic using JES3 destination classes and MCS routing codes. You can specify up to 95 JES3 destination classes to control the display of JES3 messages and as many as 128 routing codes to control the display of MVS messages. You can map MVS routing codes to JES3 destination classes to customize message traffic in your installation.

Information is added about creating Scheduler Work Area (SWA) space for jobs. You can now control which jobs will have space created above 16-megabytes in storage using the CIPARM initialization statement.

Information is added about overriding the JCL statement limit on the STANDARDS initialization statement using user exit IATUX41.

Information is added about using an automation product, such as the IBM NetView™[2] Program Product, to automate the processing of messages.

A new keyword, THWSSEP= is added to the STANDARDS initialization statement. This keyword allows you to establish device allocation preferences when a job uses high watermark setup or tape high watermark setup.

You can now suppress the display of JES3 action messages as well as the display of JES3 non-action messages using the MVS Message Processing Facility.

The following JES3 initialization statements are updated in this release:

- BUFFER
- CIPARM
- CONSOLE (MVS/BDT)
- CONSOLE (Non-RJP)
- CONSTD
- DEVICE
- DYNALLOC
- GROUP
- MAINPROC
- MSGROUTE
- OUTSERV
- RJPLINE
- RJPTERM
- SELECT
- SETNAME
- STANDARDS

---

2 NetView is a trademark of International Business Machines Corporation.

Summary of Amendments
for SC23-0059-4
MVS/System Product - JES3 Version 2 Release 1.5
as Updated March 31, 1986

This major revision applies to *MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning*, SC23-0059-3. The changes or additions to this manual support the SNA/NJE Enhancement, the truncation of blanks, and increased process mode capability.

To allow SNA/NJE networking capability, two new parameters (TYPE= and BDTID=) are added to the NJERMT initialization statement.

Chapter 9, "JES3 Networking," is reorganized to incorporate such SNA/NJE considerations as rerouting of jobs, conversion from BSC to SNA protocol, and effects of restarting JES3 and MVS/BDT.

Two new parameters (the TRUNC= parameter on the BUFFER initialization statement and the TRUNC= parameter on the SYSOUT statement) allow an installation to control the truncation of trailing blanks on SYSOUT data.

The DEVICE initialization statement is updated to accommodate changes to the function of the PM= parameter and to include the new ALTPM= parameter. Both parameters allow new flexibility when defining process modes.

Also, a new parameter (OUTSVFCT=) included on the OUTSERV initialization statement allows output service to process multiple jobs simultaneously.

Summary of Amendments
for SC23-0059-3
MVS/System Product - JES3 Version 2 Release 1.5
as Updated December 31, 1985

This major revision applies to *MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning*, SC23-0059-2 and includes minor technical changes and editorial changes, as well as information for the JES3 support of 31-bit addressing and some RAS improvements.

Added is information concerning:

- Running mixed JES3 release levels in the same complex
- Coding conventions for writing initialization statements
- Networking job numbers
- Console support
- User exits

In addition, the chapter which was entitled "Using the Dump Job Facility to Convert from JES3 SP1.3.1 to SP1.3.4" has been removed from this book and now appears as an appendix in *MVS/Extended Architecture JES3 Conversion Notebook*. Chapters are renumbered as a result of this change.

Part I: Initialization and Tuning Guidelines

Part I includes the following chapters:

- Chapter 1, "Planning for JES3"
- Chapter 2, "JES3 Job Management"
- Chapter 3, "Defining and Managing C/I Service"
- Chapter 4, "Defining and Managing Spool Data Sets"
- Chapter 5, "Defining Consoles and Message Routing"
- Chapter 6, "Defining and Managing JES3 Resources"
- Chapter 7, "Defining and Managing JES3 Mains and Storage"
- Chapter 8, "JES3 Remote Job Processing"
- Chapter 9, "JES3 Networking"
- Chapter 10, "JES3 Start-Up and Initialization"
- Chapter 11, "JES3 Recovery"

Chapter 1. Planning for JES3

You must make many decisions and perform many tasks to install, initialize, and customize JES3. You must:

1. Develop an installation plan
2. Define how JES3 manages resources and jobs
3. Develop performance objectives for JES3

This book helps you make the necessary decisions and perform the required tasks. Before installing JES3 however, you must make sure that your installation has the hardware needed to support JES3. You should also plan the layout of your I/O devices and learn how to install JES3. The *Program Directory*, a document you receive with JES3, explains how to install JES3.

If you are installing JES3 for the first time, you must define it to MVS. This means you must become familiar with the MVS system generation process and the macros that define JES3.

Developing Your Installation Plan

You should develop a well thought out plan to perform a smooth and orderly installation of JES3. Your plan should address questions such as:

- What hardware should I use and how should I configure it?
- Must I execute the MVS configuration program (MVSCP)?
- Must I change any members of SYS1.PARMLIB?
- How do I install JES3?

Only after you have developed this plan should you proceed to install JES3.

You must carefully plan the configuration of hardware and software required to satisfy your installation's needs. You should also consider ways to reconfigure your complex early in your planning. You can initially define MVS/XA processor complexes in such a way that will allow reconfiguration without having to restart JES3. For more information about reconfiguring a processor complex, see "Defining Mains" in Chapter 7, "Defining and Managing JES3 Mains and Storage." For information about planning an I/O configuration that supports reconfiguring and for instructions on the reconfiguration process, see *MVS/Extended Architecture Planning: Recovery and Reconfiguration.*

JES3 provides great flexibility in the location of equipment in your machine room. For example, you can use additional operator consoles to physically separate the operational functions (card I/O, printing, tape setup) across multiple systems and locate them in areas most convenient to your local work flow. You can locate your card readers, punches, and printers in the job dispatching area where programs are submitted for execution and output is returned. You can place your mountable I/O units in an area that is convenient to the tape and disk library. In addition, you can place an operator console at the tape and disk librarian's desk to receive library volume fetch requests. You can then place the processing units in some other area that is free of the congestion typical of the peripheral units.

Defining How JES3 Manages Resources and Jobs

You code JES3 initialization statements to define how you want JES3 to manage resources and jobs. These statements tell JES3 how to manage the following resources:

- Mains (global and local)
- I/O devices
- Main and external storage
- The system log
- Communication lines and/or protocols
- Operator communication

JES3 allows you to use a subset of the I/O devices that MVS supports. For a list of the I/O devices that JES3 allows you to use, see the following JES3 initialization statements in Chapter 12, "Initialization Statement Reference":

Statement	Devices listed
CONSOLE	consoles
DEVICE	consoles, magnetic tape drives, printers, punches, readers, direct access devices, and remote terminals
RJPTERM	remote binary synchronous communication (BSC) work stations

The job management information that you code on initialization statements specify how you want JES3 to:

- Process job input
- Interpret JES3 control statements
- Select and schedule jobs for execution
- Process job output
- Recover from failures

Developing JES3 Performance Objectives

Many factors that you can control affect JES3 performance. Among the factors that you should consider when stating your performance objectives are:

1. Hardware used
2. Availability of resources
3. JES3 work load
4. Job processing options selected

You should develop a set of performance objectives that state what you expect of JES3.

After you have developed your performance objectives and install JES3, you can tune it. Tuning, however, is not something that you do just one time. It is an iterative process; you measure performance and make adjustments, measure performance and make more adjustments, and so forth. Each chapter contains information about tuning the JES3 function that you are defining.

Installing JES3

Before you can initialize JES3, you must first initialize MVS. To initialize MVS you must:

- Perform a system generation
- Execute the MVS configuration program
- Update the MVS SYS1.PARMLIB data set
- Initialize MVS

Performing a System Generation

System generation is an MVS procedure that you use to construct the MVS operating system. When you perform a system generation, you must code the JES macro that defines the JES3 libraries to the generated system. For more information about the JES3 macro and the system generation process, see *MVS/Extended Architecture System Generation Reference.*

Executing the MVS Configuration Program

The MVS configuration program (MVSCP) provides a way for you to define I/O configurations to MVS. You can also use MVSCP to request I/O configuration data for the JES3 initialization stream checker.

MVSCP, with other program enhancements, enables you to maintain one copy of the MVS nucleus to support multiple I/O configurations. For more information about MVSCP, see *MVS/Extended Architecture MVS Configuration Program Guide and Reference.*

Initializing MVS

An operator can specify certain system parameters during MVS initialization or you can specify system parameters in an MVS data set named SYS1.PARMLIB. The purpose of SYS1.PARMLIB is to provide many initialization parameters in a pre-specified form in a single data set, and thus minimize the need for operator entry of parameters during MVS initialization.

To use JES3 as the primary job entry subsystem, you must specify JES3 in member IEFSSNxx of SYS1.PARMLIB. Otherwise MVS will default to IEFSSN00 which specifies JES2. You must also define all consoles in your installation in the CONSOLxx member of SYS1.PARMLIB to ensure console integrity. See *MVS/Extended Architecture System Programming Library: Initialization and Tuning* for information about how to use SYS1.PARMLIB.

Maintaining JES3

From time to time, you may receive updates or enhancements for your JES3 product. Although IBM delivers JES3 and MVS service in the same manner, the installation requirements can differ. You must apply JES3 service to all mains in your JES3 installation at the same time to avoid unpredictable results.

Chapter 2. JES3 Job Management

JES3 job management consists of the following phases:

- Input service
- Converter/interpreter service
- Resource allocation
- Job selection and scheduling
- Output service
- Purge

This chapter describes how each phase contributes to job management and discusses how you can use JES3 initialization statements and user exits to tailor job management. Although this chapter discusses the purpose of user exits, *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros* discusses how to code user exits.

Input Service

Input service, the first phase of JES3 job management, reads jobs and places each job into a queue for subsequent processing by other phases. Input service consists of two phases:

- Reader phase
- Control statement processing phase

Reader Phase

The reader phase reads jobs (JCL and input stream data) and stores the jobs on a spool data set. The only jobs not read by the reader phase are jobs from an internal reader and demand select jobs. These jobs are read directly by the control statement processing phase. Jobs can come from a card reader, a tape unit, a disk reader or from a remote work station. The reader phase treats jobs from a remote work station as though the job came from a card reader.

Figure 2-1 shows the flow of jobs to the reader phase.

Control Statement Processing Phase

After the reader phase completes execution, the control statement processing phase receives control. This phase analyzes JES3 control statements, checks RACF authorization, if required, and writes each job to the JES3 job queue. This phase also reads jobs from the internal reader.

If the job contains no //*PROCESS control statements, the control statement processing phase defines the job as requiring the standard sequence of scheduler elements (SEs). The standard sequence is:

- Converter/interpreter (CI)
- Main device scheduler and generalized main scheduling (MAIN)
- Output service (OUTSERV)
- Purge (PURGE)

If the job contains a //*MAIN control statement with the UPDATE parameter specified, the control statement processing phase adds the DISABLE SE (before the MAIN SE) and the ENABLE SE (after the MAIN SE) to the job's processing.

If the job contains one or more //*PROCESS statements, the control statement processing phase defines the job as requiring the sequence of scheduler elements named on the //*PROCESS statements.

You can use user exit IATUX17 to modify the sequence of scheduler elements. For information on using user exits, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Figure 2-1 shows job flow during the control statement processing phase.

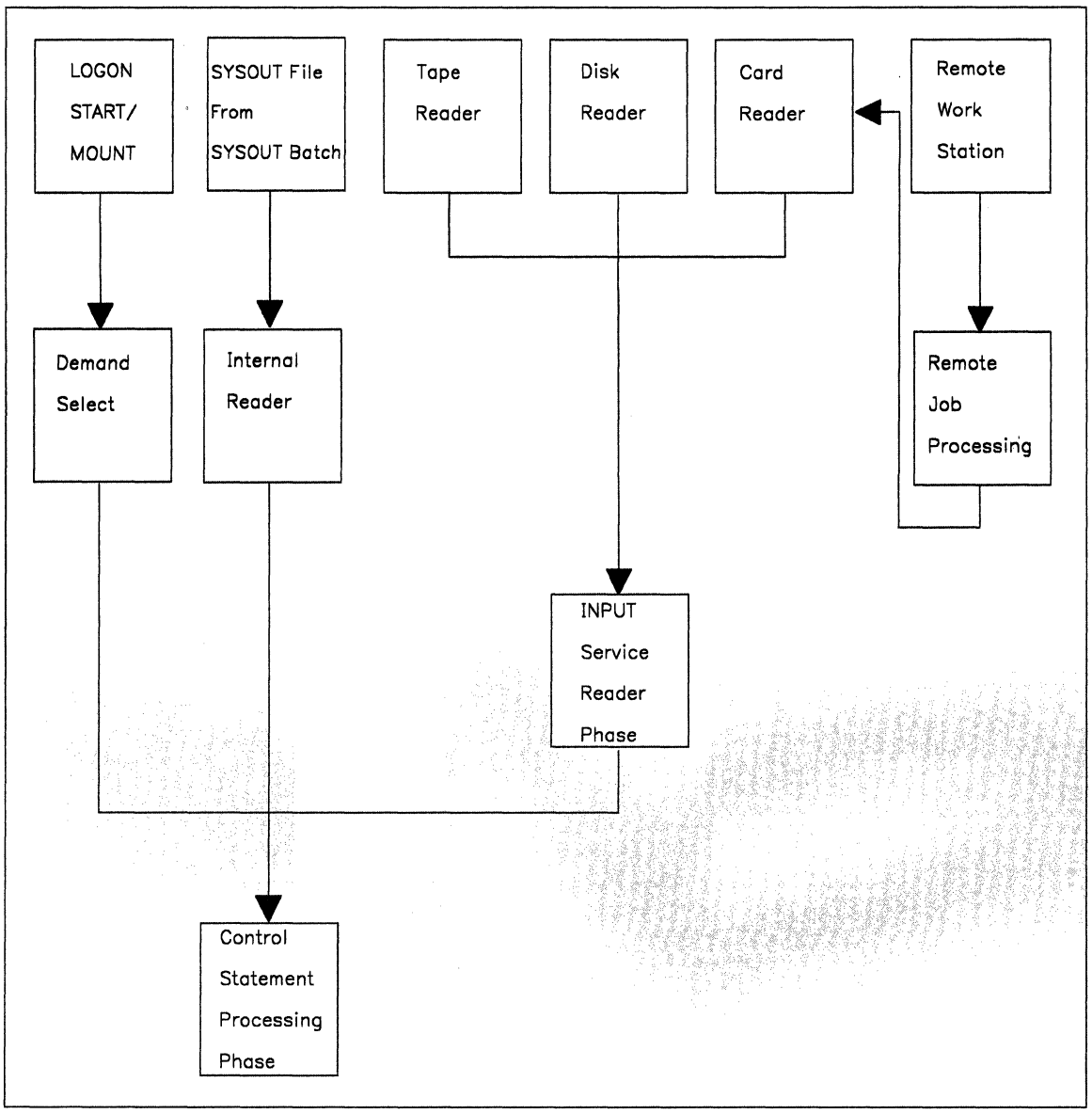


Figure 2-1. Overview of Job Flow to Input Service

The control statement processing phase also determines which procedure library should be used for the job. If the job contains a //*MAIN JES3 control statement with the PROC parameter specified, the control statement processing phase assigns the designated procedure library as the one to be used for the job. If the PROC parameter is not specified, the control statement processing phase assigns the default procedure library. The default procedure library is specified on the STANDARDS initialization statement as follows:

- for an internal reader job, the INTPROC parameter
- for a started task, the STCPROC parameter
- for a TSO LOGON job, the TSOPROC parameter

For a batch job, the standard procedure library (IATPLBST) is the default.

If the PROC parameter specifies an invalid procedure id (procid), the job is flushed from the system. If the job does not use any procedures (even if the PROC parameter is specified), JES3 does not assign a default procedure library to the job.

Selecting a Processor for Job Execution

Sometimes a user may want to select the processor on which a particular job will execute. For example, the job could need a feature or require an I/O device that is unavailable to some processors or not defined to JES3. To be sure the job executes on the right processor, the user can do one of the following:

- Code the SYSTEM parameter on the job's //*MAIN statement.
- Assign the job to a job class that is enabled on only the required processor.

If the user has specified a processor by coding the SYSTEM parameter on the //*MAIN statement, the control statement processing phase assigns the job to execute on the specified processor. Otherwise, the control statement processing phase assigns the job to execute on a processor eligible for the job's job class. The SYSTEM= parameter on the CLASS initialization statement defines the eligible processor(s) for a job class.

If no eligible processors are available, the job completes input service processing, then waits until an eligible processor becomes available before starting C/I processing.

Modifying JCL and JES3 Control Statements

After JES3 reads the input stream but before JES3 or MVS uses the information specified in the input stream, JES3 user exits can verify, modify, add, or delete information on:

- JOB, EXEC, DD, or OUTPUT JCL statements (except DD * or DD DATA statements)

- JES3 control statements (statements that begin with //*) except //*DATASET or //*ENDDATASET

To verify or change the JOB JCL statement, use user exit IATUX28. This exit also allows you to remove the job from the system.

To verify or change the EXEC JCL statement or JES3 control statements, use user exit IATUX33. This user exit also allows you to request that JES3 ignore the statement.

To verify or change DD JCL statements, use user exit IATUX34. This user exit also allows you to request that JES3 ignore the statement.

To verify or change an OUTPUT JCL statement or any JCL statements besides those listed above, use user exit IATUX44. This user exit also allows you to request that JES3 ignore the statement.

Examining Job Control Blocks

After input service has built the control blocks needed to process a job, you can use user exit IATUX29 to:

- Examine or modify the Job Control Table (JCT)
- Examine or modify the Job Description and Accounting Block (JDAB)
- Examine or modify the Job Management Record (JMR)
- Write to JESMSG (IATXISMG)

Through this user exit you can request that JES3 continue to process the job or you can request that JES3 cancel the job.

Converter/Interpreter Service

Converter/interpreter (C/I) service controls the conversion of JCL statements to internal text and then into control blocks. This service comprises primarily the JES3 CI and POSTSCAN DSPs, the C/I subtasks under which the MVS C/I routines run, and the initiator support routines.

C/I service controls the conversion and interpretation of a job's JCL. The three principal phases of C/I service are:

- *Converter/interpreter phase:* Uses the MVS C/I routines to convert and interpret the JCL into scheduler control blocks. At this time, the scheduler control blocks are created in the scheduler work area (SWA).

- *Prescan phase:* Creates job tables from the scheduler control blocks for use in the postscan phase. At the end of the prescan phase, the scheduler control blocks are moved from the SWA to the JES3 spool.

- *Postscan phase:* Creates the job summary table for use in JES3 main device scheduling (MDS).

You can move the converter/interpreter and prescan phases for some or all jobs into one or more C/I functional subsystem (FSS) address spaces. A C/I FSS address space contains many functions similar to a JES3 address space and can operate on the global processor or any local processor. A job is processed in the C/I FSS address space by a CI DSP through the prescan phase. Then, the postscan phase of the job's C/I processing takes place in the JES3 address space on the global processor (hereafter called the JES3 global address space) under a separate POSTSCAN DSP.

A POSTSCAN DSP also provides postscan processing for rescheduled jobs, such as dependent job control (DJC) jobs for which at least one predecessor job has not completed.

When a job enters the MAIN scheduler element, the JES3 initiator support routines move the scheduler control blocks from spool to the SWA in the user's address space. You can modify the scheduler control blocks before they are moved to the SWA using user exit routine IATUX26.

For detailed information about how you can set up and influence C/I service, see Chapter 3, "Defining and Managing C/I Service."

Converter/Interpreter Phase

The primary function of the converter/interpreter phase is to convert the JCL into internal text, then interpret the internal text and create scheduler control blocks.

When the job segment scheduler (JSS) has a job that can be scheduled for C/I processing, JSS determines whether a CI DSP in the JES3 global address space or a C/I FSS address space is available for the job. You can select the processor where the job will be scheduled for C/I service as well as whether the job is eligible for C/I processing in the JES3 global address space by using user exit IATUX46. If the initialization stream defines C/I FSS address spaces, you can override JSS's selection of an address space using user exit IATUX49. JSS schedules the job to an available CI DSP based on the user exit routines' responses.

A C/I subtask, operating under the direction of the CI DSP, links to the MVS converter to read the JCL and to convert it to internal text. If the number of JCL statements in a job exceeds the job JCL statement limit, the C/I subtask calls user exit routine IATUX41 to see if the job should be canceled. If so, the job is canceled from the system with print. (Operator messages refer to this type of cancellation as the job being "express canceled.") If the job is not to be canceled and there are no JCL errors, the C/I subtask links to the MVS interpreter to create the scheduler control blocks from the internal text.

The JES3 internal text and FIND routines are invoked as exits from the MVS converter. JES3 user exit IATUX03 is entered from the internal text exit, and IATUX02 is entered from the FIND routine. The JES3 SWA queue manager routine is invoked as an exit from the MVS interpreter and the JES3 SWA manager invokes the MVS SWA queue manager.

When the C/I subtask finishes its work, the job enters the prescan phase of C/I service.

Prescan Phase

The primary function of the prescan phase is to create the intermediate job summary table (IJS), the skeleton job volume table (JVT), and the data set name locate table (LVS). These tables reflect job resource requirements extracted from the scheduler control blocks created in the interpreter phase.

On entry to the prescan phase, the SETUP parameter on the STANDARDS initialization statement is examined:

- If SETUP=NONE is specified (that is, JES3 is to do no preexecution setup for the complex), JES3 does not build any tables.
- If you specify any value other that NONE on the SETUP= keyword, JES3 searches the scheduler control blocks to create the IJS, JVT, and LVS.

Job setup is discussed later in this chapter.

User exits IATUX04, IATUX05, and IATUX06 allow you to examine or change job, step, and DD information, respectively. You can use these exits to examine or change the information before processing begins (for example, you can examine where SWA is in the initiator). For an installation with the mass storage system (MSS) you may specify, via IATUX04, that unit requests for MSS staging drive groups (SDGs) are to be ignored for the job in question. You should modify IATUX04 to allow allocation by SDG when using VSAM IDCAMS.

During the prescan phase, the JCL for the job is examined for PGM=JCLTEST or PGM=JSTTEST. If PGM=JCLTEST is found on an EXEC statement, the JCL is interpreted and the job is then canceled-with-print on completion of the CI DSP. If PGM=JSTTEST is found on an EXEC statement, the job is processed through the prescan and postscan phases, a printed format of the job summary table (JST) is printed on the JESMSG data set, and the job is then canceled-with-print on completion of the CI DSP. For more information on JCLTEST and JSTTEST, see *MVS/Extended Architecture JES3 Diagnosis.*

At the end of the prescan phase, the scheduler control blocks, the IJS, JVT, and LVS are written to the JES3 spool. If the job is part of a DJC network and requires a predecessor job to run, the job waits until the predecessor job completes. Then, the job is rescheduled on the POSTSCAN DSP.

The IJS contains a header, step entries, and DD entries. One IJS DD entry is created for each DD statement referencing a JES3 unit type identified in a SETNAME or DEVICE initialization statement.

The skeleton JVT contains all volumes of any DD requests that were defined in the JCL for a job. The JVT is not completed until all volumes of cataloged data sets are known.

The LVS contains all data set names for which a catalog search must be made. If JOBCAT or STEPCAT DD statements are included in the JCL for a job, entries are created for each statement. JES3 uses this information to locate and set up the job and step catalogs.

If the job was being processed by a CI DSP in a C/I FSS address space (whether on the global processor or a local processor), the job is returned to the JES3 global address space for postscan processing. There the job is processed by a POSTSCAN DSP (unless the job was cancelled during the earlier phases).

Postscan Phase

The functions of the postscan phase are to resolve cataloged data set references, and to construct and modify the job summary table (JST) according to the installation's requirements.

Cataloged Data Set Resolution

Before the JST is created, JES3 accesses the system catalogs. If a LOCATE (catalog search) request fails to find a data set name, the postscan phase calls user exit IATUX07. Through this user exit, the system programmer can examine the available data set information and, if necessary, supply the unit and volume information.

When a job's JCL explicitly calls for private catalogs to be used through the use of JOBCAT and STEPCAT DD statements, C/I service utilizes MDS services to setup those catalog requests.

A request for an unavailable catalog volume causes the job to be canceled.

If a data set has been migrated (or is eligible to be migrated) by the Hierarchical Storage Manager (HSM), the LOCATE request for that data set causes JES3 to associate the data set with a set of volumes to which it can be recalled. HSM limits the choice of volumes eligible for recall during LOCATE processing in accordance with its space management algorithms. JES3 processing continues as though the data set is recalled to all eligible volumes. However, the actual recall does not occur until job execution when MVS issues a LOCATE request. At that time, HSM determines which volume is the best choice to recall the data set to and then recalls the data set to that volume.

In addition, whenever the response of a LOCATE request has been received, the system programmer can use user exit routine IATUX11 to inhibit printing of the LOCATE request/response in the JESMSG data set.

JST Construction

After all LOCATE requests have been processed, the postscan phase constructs the JST from the now complete IJS and JVT. The JST at this time contains a complete profile of the job. To allow the system programmer access to the JST before the CI or POSTSCAN DSPs relinquish control, the postscan phase calls user exit IATUX09. The user exit routine can examine the information in the JST and then continue or terminate the job.

If high watermark setup (HWS) is specified on the SETUP parameter of the STANDARDS statement, the postscan phase determines the minimum number of devices required to run the job and modifies the JST appropriately. If PGM=JSTTEST is specified on an EXEC statement, the postscan phase produces a formatted version of the JST in the JESMSG data set and cancels the job.

C/I Service Preparation for the Main Device Scheduler

Although the JES3 main device scheduler performs volume fetching and setup, JES3 must first build the job summary table and the job volume table according to the type of setup used for each job. You specify the type of job setup by coding the SETUP parameter on the STANDARDS initialization statement or the end user can override your specification by coding the SETUP parameter on the //*MAIN statement in a job's JCL.

You can have user exit IATUX08 examine the setup requirements for each job that uses job setup. Once coded, the user exit can modify the type of job setup or fail a job before the main device scheduler receives control. See *MVS/Extended Architecture System Programming Library: User Modifications and Macros* for a complete description of user exit IATUX08.

JES3 Resource Allocation

JES3 provides a device management facility called the main device scheduler (MDS) that can wholly or partially support the MVS allocation process. The purpose of MDS is to satisfy job resource requirements (the devices, volumes, and data sets needed) before and during job execution, thus allowing execution to proceed without allocation delays. MDS also allows controlled multisystem access to commonly accessible data sets in the loosely coupled environment.

You must choose whether to use MDS or use the operating system (which controls the job execution) for the entire allocation process as each step begins execution. If you choose MDS, you must then decide whether utilization of MDS is to be partial (set up some jobs, some resources) or total (set up all jobs, all resources).

System Allocation Compared with MDS Allocation

MDS and system allocation consider a job's resource requirements at different levels. System allocation considers job requirements one step at a time for the processor executing the job; MDS considers the resource requirements for all the steps in a job for all processors in the loosely-coupled complex. These two approaches lead to the following differences between MDS and system allocation.

System Allocation

In systems that do not use MDS, jobs are presented to the operating system based on criteria such as job class, priority, or workload mix. In these systems, a job's requirements are not known until the job entry subsystem selects the job for execution, and a system initiator begins the step allocation process. At each job step, system allocation attempts to satisfy the requirements for the step, in contention with every other job step currently executing on the same processor. If the requirements cannot be met, system allocation gives the operator the option of canceling the job or allowing it to wait for resources. Thus, in a system that does not use MDS, there may be jobs executing and other jobs waiting for resources.

The jobs waiting in system allocation hold critical resources (a system initiator, an address space, data sets, and possibly devices). Holding these resources longer than necessary makes it very difficult for the system programmer to determine how many initiators should be started to keep the system fully utilized, because at any given time, an unknown number of initiators may be waiting. MDS offers a solution to this problem.

Main Device Scheduler (MDS) Allocation

With MDS, the resources (data sets, devices, and volumes) that a job requires are already set up when the job is passed to MVS for execution. There should never be an idle initiator caused by a job waiting for these resources. Setup occurs while a job is in the JES3 address space, and the only system resource used while the job is waiting is the JES3 queueing space. MDS helps the system make maximum use of devices and allows jobs to run in a minimum amount of time once they are passed to the system for execution.

The main device scheduler requests and verifies the mounting of the initial volumes a job requires on each device before the job can be selected for execution (unless deferred volume mounting is specified in the JCL). After JES3 converter/interpreter service scans the JCL for required volumes and data sets and after it determines the volumes required (by accessing the system catalog), the volume fetch facility issues tape or disk volume request messages. If the system programmer specifies ALLOCATE=MANUAL on the SETPARAM initialization statement, JES3 puts the job into the volume-wait queue. The job stays in the queue until the operator releases it.

JES3 can schedule, for each processor, a combination of jobs that will execute without contention for both the sharable and nonsharable devices and data sets attached to that processor. Because MDS considers the volumes and data sets for a total job and for all systems in the complex, it has more information on their utilization than does system allocation. Thus, MDS can determine the volume and data set utilization for the combination of jobs running at one time on any processor.

The JES3 default method of reserving devices on a total job basis (Job Setup) may cause more devices to be used by a job than would be required under system allocation. For example, because devices are initially set up on a job basis, a device used in a later job step may be reserved (but not used) during all the prior job steps. System allocation avoids this problem by allocating at the job step level, and thus minimizing the number of devices used by a job.

Because setup occurs before job execution, JES3 cannot react to processing dependencies that can occur between different jobs and between different steps in the same job. This limitation is particularly important when considering the cataloging and passing of data sets. JES3 cannot determine whether any conditional job steps are skipped as a result of condition code processing. **JES3 assumes that all job steps will execute.** JES3 also counts the number of I/O devices needed by each step. Another consequence of this limitation is that if the VOLUME=(,RETAIN) parameter is specified in the JCL, all retained volumes are treated as public, even though PRIVATE may have been explicitly specified on the DD statement. Therefore, a private volume may end up being unloaded at the end of a step even though RETAIN was specified. In such cases, the MVS RETAIN message is issued.

Note: Mass storage volumes are not premounted by MDS; they are mounted by MVS.

The JES3 main device scheduler controls the volume fetching, allocation, mounting, and deallocation of I/O devices associated with job execution on all processors in a loosely-coupled complex.

MDS is divided into the following stages:

- Volume fetch
- Allocation
- Volume verification
- Breakdown

Volume Fetch

Volume fetch, the first phase of MDS, is performed for all jobs entering MDS. This phase determines the volumes required by the job and, if necessary, instructs the operator to get the volumes from the library. This phase also eliminates those processors on which the job cannot run.

During fetch processing, JES3 builds volume entries and issues messages for volumes that have no entries in the SETVOL table. The SETVOL table contains the volume serial number for each reference to a device managed by MDS.

Volume fetch messages are selected optionally by specifying FETCH=YES on the SETPARAM initialization statement. When the fetch option is used, JES3 issues volume fetch messages to indicate which volumes are required for specific jobs to execute. JES3 sends fetch messages to the console specified by the TAFETCH (for tape volumes) and DAFETCH (for direct-access volumes) parameters on the SETPARAM initialization statement. Volumes already mounted require no fetch processing, and volumes that have been fetched but not mounted get action-coded messages. Device types other than tape or disk do not require operator action. If the volume fetch option is not selected at initialization, jobs go directly into the allocation stage of MDS.

Allocation

The ALLOCATE parameter on the SETPARAM initialization statement controls the way in which jobs are processed during the MDS allocation phase. If the ALLOCATE parameter is specified (or defaulted) as ALLOCATE=AUTO, MDS sends incoming jobs directly into the allocation phase. If ALLOCATE=MANUAL is specified, the operator must issue the *START,SETUP command for each job requiring volumes to be fetched before the job can go through MDS allocation. Jobs which require volumes to be fetched are kept in the MDS WAITVOL queue; the contents of this queue can be obtained by issuing the *INQUIRY,S,W, command. The MDS queues are made of resident job queue (RESQUEUE) entries, grouped in subchains based on the MDS function to be performed. Setup is performed for dynamic allocation requests at the time each request is made. If the request cannot be satisfied, a return code is issued to dynamic allocation.

To start the allocation phase of MDS, a job is selected from the ALLOCATE queue. MDS examines the JST for the selected job and attempts to allocate the required devices, volumes, and data sets. When MDS initially tries to set up a job, it records the total device, volume, and data set requirements for the job. If a job cannot be set up because a resource is unavailable, the job will not be selected until the required resource becomes available.

Device selection (through initialization parameters) limits the number of processors that can execute a job whenever a needed I/O device is not shared among all eligible processors. Any processor that cannot allocate a requested device or satisfy the total resource requirement is ineligible to run the job.

A job that requests use of a volume that the operator has designated as "unavailable" (via the *F,S,VU= command) is placed on the volume unavailable queue as long as the job has not already completed the allocation process. A job that allocated a volume prior to that volume being made unavailable is allowed to complete normally.

Volume Verification

JES3 issues mount messages to direct the operators to mount the job's required volumes. User exit IATUX62 may be used if you wish to check the validity of mounts requested by JES3. This user exit routine, which is invoked after verification, can accept or override JES3's mount request.

The VERIFY function automatically obtains the volume serial number, label status, and other information for MDS once a job's volumes are mounted. User exit IATUX25 provides a way for the user to validate any nonstandard labels used in the installation. When all volumes are properly mounted, the job is ready for execution. Device types other than tape or disk do not require operator action. Premounting of anticipated MSS volumes on JES3-managed devices is not required and should not be done.

For more information about user exits IATUX25 and IATUX62, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Breakdown

The breakdown phase of MDS is automatically performed by JES3 when the resource (data set, volume, or device) is no longer required by a job. The resource is then available for use by other jobs.

JES3 issues messages to the operators to indicate whether volumes should be kept available for other jobs or demounted. The RETAIN and KEEP messages issued by MVS allocation apply only to the resources used within one job, while the RETAIN and KEEP messages issued by MDS consider volume usage by all jobs currently in the system that use JES3-managed or jointly-managed devices. In the event that both MVS and JES3 issue KEEP or RETAIN messages regarding a specific volume, the JES3 messages take priority.

Jobs that have had errors during the FETCH, ALLOCATION, or VERIFY phases of MDS, or that have failed MDS restart processing are placed on the MDS error queue. These jobs must be restarted or cancelled by the operator. User exit IATUX61 saves operator intervention by allowing the system programmer to selectively cancel jobs that would otherwise be placed on the error queue. This user exit routine can check the reasons why the job is failing, and according to specified conditions, cancel the job before it is placed on the queue.

For more information about user exit IATUX61, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Types of Setup

The JES3 setup type is defined by JES3 initialization statement parameters and JCL control statements, and by JES3 operator commands. This section describes some of the parameters needed to define setup to the system.

JES3 performs three different types of setup based on the installation and user requirements:

- Job setup
- High watermark setup
- Explicit setup

The system programmer can use JES3 initialization parameters and JES3 control statements to control allocation according to device types. For example, job setup can be used for disks, and high watermark setup for tapes.

General considerations for each type are discussed below; setup examples are described in *MVS/Extended Architecture JCL User's Guide*.

Job Setup

By using job setup you can cause JES3 to:

- Reserve all devices needed by a job before the job executes
- Mount all volumes needed by the job before the job executes

Thus, after the job starts to execute, it does not have to wait between job steps to have a device allocated or a volume mounted.

There are exceptions to the previous statements, however. The exceptions occur for:

- Deferred volume mounting
- Dynamic allocation
- The mounting of other than the first volume of a multivolume data set on the same device.

For a multivolume data set, if the unit count and the volume count are unequal (unit count and volume count are DD statement subparameters), JES3 mounts the number of volumes specified as the unit count. Thereafter, MVS mounts subsequent volumes as they are needed.

When job setup is used, devices, volumes, and data sets are available for use by other jobs as soon as the DD statement is deallocated in the last step using the resource. Disadvantages of job setup are inefficient device usage and reserving devices for a job during steps when they are not used.

Job setup is used when SETUP=JOB is specified on a //*MAIN control statement for the job. If the SETUP parameter is not specified on the //*MAIN statement, then job setup is used only when specified (or assumed by default) on the STANDARDS initialization statement. Setup for MSS virtual units is specified on the SETPARAM initialization statement.

High Watermark Setup

High watermark setup, as defined on the HWSNAME initialization statement, reduces the number of devices JES3 reserves for a job. To determine how many devices of a particular type to reserve for a job, JES3 considers the needs of each of the job's steps. In this way, JES3 determines which step needs the greatest number of devices of that type. JES3 then reserves that many devices of that type for the job. JES3 repeats this process for each device type the job needs. As a result, high watermark setup can cause premounting of all mountable volumes. Volume unloading and remounting may occur for both private and public volumes, even when RETAIN has been specified on the applicable DD statement.

High watermark setup for MSS virtual units may temporarily reserve more devices than the maximum number required by any single job step. This is the result of JES3 processing designed to improve the efficiency of MSS utilization by the entire job mix.

When high watermark setup is used, as in job setup, devices, volumes, and data sets are returned to JES3 for use by other jobs as soon as the DD statement is deallocated in the last step using the resources. When it is advantageous to use fewer devices for a job, high watermark setup is preferable to job setup.

High watermark setup is used when SETUP=THWS (for tapes only), SETUP=DHWS (for disks only), or SETUP=HWS (for tapes, disks, graphics, and unit-record devices), or MSS=HWS is specified on a //*MAIN statement for the job. If the SETUP or MSS parameter is not specified on a //*MAIN statement, then high watermark setup is used only when SETUP=THWS, SETUP=DHWS, or SETUP=HWS is specified on the STANDARDS initialization statement, when the specified setup is overridden by user exit IATUX08, or when MSS=HWS is specified on the SETPARAM initialization statement.

Explicit Setup

This setup provides a means for the application or system programmer to combine the execution advantages of job setup and the device usage advantages of high watermark setup. To specify explicit setup, use the SETUP parameter on the //*MAIN JES3 control statement.

When explicit setup is specified, the job's devices are allocated using job setup. Only the device premount characteristic is affected by specifying explicit setup. Explicit setup is mutually exclusive with any high watermark setup. The device allocation for job setup is the default when explicit setup is specified.

An advantage of explicit setup over high watermark setup is that volumes can be forced to remain mounted on devices until they are no longer needed. Job setup and high watermark can deallocate resources (device, volumes, and data sets) at the end of any step if the resources are no longer needed.

Explicit setup is used when the SETUP=(ddname[,ddname]...) parameter is specified on a //*MAIN statement for the job. To explicitly specify data sets that are not to be set up, the SETUP=/(ddname[,ddname]...) parameter should be specified.

Initializing MDS

The operation of MDS is affected by the parameters chosen for the following JES3 initialization statements: CLASS, DEVICE, GROUP, SELECT, SETACC, SETNAME, SETPARAM, SETRES, and STANDARDS.

CLASS Initialization Statement

The SDEPTH parameter on the CLASS statement limits the number of jobs of a specific class (requiring mountable devices) that can be active in setup at the same time. Thus a class restricted to a small number of devices (such as a test class) can be prevented from monopolizing mountable devices.

DEVICE Initialization Statement

The XUNIT parameter of the DEVICE statement specifies a console destination code for messages that pertain to the device being defined. You must assign this same destination code to the console that is to display these messages. To do this, specify the DEST= parameter on the CONSOLE statement that defines the console. For efficiency, the console should be near by the device. For example, given two 3330 facilities and one tape:

```
DEVICE,XTYPE=(3330,DA),XUNIT=(130,SY1,S6,ON)
          .
          .
          .
DEVICE,XTYPE=(3330,DA),XUNIT=(137,SY1,S5,ON)
DEVICE,XTYPE=(3330,DA),XUNIT=(230,SY1,S6,ON)
          .
          .
          .
DEVICE,XTYPE=(3330,DA),XUNIT=(237,SY1,S6,ON)
DEVICE,XTYPE=3400,XUNIT=(180,SY1,S7,ON)
```

Messages could be split specifying:

```
CONSOLE,JNAME=[illegible],...DEVL=5
CONSOLE,JNAME=[illegible],...DEST=S6

CONSOLE,JNAME=CNTAPE,...DEST=S7
CONSOLE,JNAME=MDSLOG,...DEST=SALL
```

The XTYPE parameter allows the specification of device class and volume removability. These subparameters, normally defaulted to tape (TA) and removable (RM), can be used to indicate a JES3-managed device that contains a permanently resident volume. For example:

```
DEVICE,XTYPE=(3330,DA,PR),XUNIT=(230,SY2,S5,ON)
```

would establish the volume on local processor SY2 unit 230 as permanently resident.

If a device has multiple access, it should be specified in the XUNIT parameter of the DEVICE statement for all processors. For example:

```
DEVICE,XTYPE=(3330,DA,PR),XUNIT=(230,SY1,S1,,132,SY2,S2)
```

GROUP Initialization Statement

Device pooling for job-class groups is controlled by the specification of either the DEVPOOL parameter or the device dedication positional subparameters in the EXRESC parameter. The basic difference between the two methods of dedication is that devices dedicated via the EXRESC parameter are dedicated when the group is allocated on the processor specified in the EXRESC parameter, while DEVPOOL-requested dedication is accomplished when the group is enabled on any processor. The allocation options of ANY or GROUP are used to determine whether a job that is not able to obtain all its required devices for volume mounting is allowed to go beyond the dedicated devices to satisfy its requirements. Normally, a group representing higher priority work would be allowed to go beyond the dedicated devices (ANY) if fewer than the total number required were dedicated. A group representing testing might be assigned dedicated devices to limit its impact on the throughput of production work by only allowing it to allocate dedicated devices (GROUP).

SELECT Initialization Statement

Several parameters on the SELECT statement affect the operation of MDS allocation on a processor basis (SDEPTH, SBAR, INCR, INCL, SAGER, SAGEL). Through these parameters, MDS allocation may be biased toward one processor (a larger SDEPTH), devices may be reserved but not entirely allocated to one processor (a higher SBAR), and jobs on a specific processor may be favored for selection (higher INCR, INCL, SAGER, and SAGEL parameters).

SETACC Initialization Statement

The SETACC statement is used to describe volumes that are found on permanently resident devices not totally shared by all processors in the complex. If the processor from which these volumes can be accessed has not been initialized, jobs requiring these volumes wait rather than request them to be mounted elsewhere. When a processor is initialized, the volumes found supersede those (if different) specified in the SETACC statement.

SETNAME Initialization Statement

The JES3 MDS algorithm uses the information specified on the SETNAME statement when searching for the proper device to be allocated for a DD request.

If a DD request is to be handled by JES3, the value (excluding the specific device number) specified in the UNIT parameter of the DD statement must also be specified in the NAMES parameter of the SETNAME statement. If the request specifies a device number in the UNIT parameter, the device number must also be specified in the XUNIT parameter of the DEVICE statement.

By specifying the same names in a different order for the same XTYPE, it is possible to create a preference order of device selection for requests requiring volume mounting. This preference might be to achieve channel path separation, if possible, while still allowing this job to run if separation cannot be achieved. For example, with devices on three channel paths:

```
DEVICE,XTYPE=(3330CH1,DA),XUNIT=(130,SY2,S1,ON)
DEVICE,XTYPE=(3330CH2,DA),XUNIT=(230,SY2,S1,ON)
DEVICE,XTYPE=(3330CH3,DA),XUNIT=(330,SY2,S1,ON)
SETNAME,XTYPE=3330CH1,NAMES=(DACH1,DISK1)
SETNAME,XTYPE=3330CH2,NAMES=(DACH1,DACH2,DISK2)
SETNAME,XTYPE=3330CH3,NAMES=(DACH1,DACH2,DACH3,DISK3)
SETNAME,XTYPE=3330CH1,NAMES=(DACH2,DACH3)
SETNAME,XTYPE=3330CH2,NAMES=(DACH3)
```

Requests for DACH1 would attempt allocation on channel path 1, then channel path 2, and finally on channel path 3. Similarly, requests for DACH2 would attempt allocation first on channel path 2, then channel path 3, then channel path 1. By using DISK1, DISK2, or DISK3, strict channel path separation could be achieved.

Although a name may appear in more than one SETNAME statement, all XTYPE parameters applying to the name must be the same type (all DA for example). All names on the SETNAME statement must be defined to MVS.

The maximum number of unique names specified on the SETNAME statement must not exceed 255. Each XTYPE should be defined to allow JES3 allocation to reference as large a collection of devices as possible to minimize the number of name/unit references that need to be examined by the JES3 allocation algorithm. The most frequently referenced names should be specified first (DASD, then tape, unit-record, and graphic devices).

The POOLNAMS parameter is provided to allow a convenient method of dedicating specific devices, even though there is no specific generic or esoteric name subset that exactly describes the attributes to be used to choose a specific set of devices. These names are allowed to be used only to dedicate devices and may not be used in the DD statement UNIT parameter.

To allow the use of devices outside of MDS control, define MVS names to include the desired devices and then omit these names from the SETNAME initialization statement.

SETPARAM Initialization Statement

The ADDRSORT parameter can be used to dictate the order in which MDS looks for mountable devices in attempting allocation. If ADDRSORT=NO is coded, the SETUNIT tables are ordered in the same sequence as the DEVICE statements in the initialization stream. This may be useful in cases when device locations are not physically ordered by ascending device number.

SETRES Initialization Statement

The SETRES statement is used to describe volumes that are to be MDS-mounted when found on removable direct-access devices during processor initialization. Any volume found may later be made removable by an MDS unload command (*MODIFY,S) issued by the operator.

STANDARDS Initialization Statement

The STANDARDS statement indicates the system standard for allocation of devices identified by the NAMES parameter on the SETNAME statement. The SETUP parameter on the STANDARDS statement specifies the type of setup processing.

Operator Control of MDS

The operator may use several commands to control the JES3 MDS process. For detailed information on the MDS commands, see *MVS/Extended Architecture Operations: JES3 Commands.*

Job Selection and Scheduling

Each time an MVS initiator requests work, generalized main scheduling (GMS) selects and schedules a job for execution. The job that GMS selects depends primarily upon initialization parameters that you have specified.

Deadline scheduling and dependent job control (DJC), additional GMS functions, enable you to control when jobs execute. With deadline scheduling, you specify a deadline by which you want the job executed. JES3 periodically increases the job's selection priority in an attempt to execute the job by the specified deadline. DJC allows you to create a network of related jobs.

Job Selection Algorithm

When a job has completed processing in MDS, the job is placed in the queue of jobs awaiting selection for execution. This queue is ordered by job priority, with the last jobs to arrive being placed last within the priority. Thus, the time at which a job completes setup processing partly determines its place in the queue.

When a request for work arrives from an initiator, the source of the request narrows the choice in two ways: (1) the group identification of the initiator limits the choice to jobs of that group, and (2) the processor on which the initiator is started limits the choice to jobs that can run on that processor.

The selection process first determines whether the select mode on the processor on which the request originated includes the IORATE parameter (from the CLASS initialization statement) as a factor. This determination is made on the basis of what CHOICE parameter on the SELECT initialization statement was specified. If CHOICE was BMIX or FMIX, then IORATE is a factor in this job selection. If IORATE is a factor in this selection, the job selection algorithm computes the best and alternate rates for later use. If the CHOICE parameter was other than

BMIX or FMIX, then IORATE is not a factor and best and alternates rates are not computed.

The best and alternate I/O rates are based on the total number of initiators with started jobs on the particular processor, and on the JOBMIX parameter specified on the SELECT initialization statement for this processor. For each number of active initiators, a set of jobs with low, high, and medium I/O rates is specified by the JOBMIX parameter. When the I/O rate computation begins, it determines the number of active initiators, and from this, the number of jobs for low, high, and medium I/O rates that the JOBMIX parameter specifies for this number of started initiators. The low, high, and medium numbers specified during initialization are the numbers of jobs that ideally should be executing for this total number of active initiators. The algorithm then computes which I/O rate (low, high, or medium) is farthest from the ideal, as specified during initialization. The I/O rate that most needs to be increased to meet the ideal becomes the best rate, and the one that needs to be increased second-most becomes the alternate rate. In case of ties, the choice favors low, then high, then medium for best or alternate. As far as I/O rate is a factor in job selection, the choice favors the I/O rate that most needs to be increased.

The job selection algorithm next moves to the queue of jobs ready for execution and looks up the first job in the group of the requesting initiator. The algorithm determines whether this first job is in hold status, either as the result of an operator command or because it is part of a DJC network. If the job is in hold status, the algorithm determines whether it has reached the end of the group's selection span. The JSPAN parameter on the GROUP initialization statement defines the span.

If the job being tried as a candidate for job selection could be executed on the processor from which the request originated, the algorithm determines the candidate's job class. The CLASS initialization statement parameters MDEPTH, MLIMIT, TDEPTH, and TLIMIT are all checked for this class. If any of these limits are met or exceeded for this job candidate, the scan checks for JSPAN and BAR parameters on the GROUP statement. If these are not exceeded, it moves to the next job candidate in this group and starts again.

The maximum number of job selection candidates examined in response to a job selection request is specified by the JSPAN parameter on the GROUP initialization statement. If the number of jobs scanned as candidates for selection reaches the value specified by JSPAN, the algorithm terminates the job selection pass. In this case, the initiator continues waiting, and the job-select request remains queued.

If the value specified by JSPAN has not been reached, the algorithm determines whether a priority barrier is effective at this point. If a priority barrier is reached, then the result is the same as if JSPAN has been reached: no more jobs in the queue can be scanned.

If all the CLASS statement limits were found acceptable, the selection algorithm next determines whether this job can fit into the available logical storage of this processor. If not, the algorithm checks whether the CHOICE parameter on the SELECT statement is set for FJOB (first job of the group in the queue). Specifying the FJOB parameter simply tries the first job of the group; if it can be executed, it is selected, and if it cannot, no job is selected.

If the job selection candidate fits into storage on the processor and the CHOICE parameter is not FJOB, the algorithm looks back to the earlier determination of whether IORATE is to be considered in this selection. If it is not, a suitable job has been found, and this job is returned to the requesting initiator. If IORATE is a factor (CHOICE is specified as BMIX or FMIX), the algorithm compares the I/O rate of this job with its earlier determination for best I/O rate. If the I/O rate of this job is the same as the best I/O rate, this job is returned to the initiator as the selected job. If the I/O rate of this job is the same as the alternate I/O rate and the CHOICE parameter is set for FMIX, this job is returned to the initiator as the job selected. If the job does not match the best or alternate I/O rate, the algorithm checks for JSPAN and BAR and starts examination of the next job of the group on the queue.

Deadline Scheduling

Deadline scheduling is a technique that allows a user to schedule a job by time-of-day, week, month, or year. The job's priority remains in force, but as the deadline approaches, JES3 increases the job's priority. Thus, deadline scheduling increases the likelihood that the job will be scheduled for execution by the specified deadline.

A deadline scheduling algorithm specifies how often and by what amount JES3 is to increase the job's priority. You must specify the algorithm on a DEADLINE initialization statement. You can specify a maximum of 36 deadline scheduling algorithms.

To specify deadline scheduling for a job, the user must code the DEADLINE parameter on a //*MAIN statement. This parameter specifies the time or date by which the user wants the job scheduled. It also specifies which deadline scheduling algorithm JES3 is to use for the job.

The DEADLINE DSP controls the scheduling of these jobs.

The DEADLINE DSP examines all the jobs in the deadline queue and issues an ATIME macro instruction for the shortest time interval until a job priority change is required. Unless another job with a shorter time interval is placed in the queue, the DEADLINE DSP waits until the time expires. Then, the priority of the waiting job is increased to increase the probability of the job being processed on time.

If the operator presses the STOP button or enters an MVS QUIESCE command, deadline scheduling stops. When the system is restarted, the operator must reinitialize the DEADLINE DSP by entering the *START,DEADLINE command. Reinitializing the DSP resets the deadline scheduling internal clock to the correct time. If the DSP is not reinitialized, the clock continues from the time at which deadline scheduling stopped and jobs in the deadline queue are delayed by the amount of time the system was stopped or quiesced.

If no jobs are in the deadline queue, the DEADLINE DSP sets the ATIME macro instruction to expire at midnight (as a default), and the DEADLINE DSP waits until it is canceled by the operator or until another deadline job is placed on the queue.

Dependent Job Control

Dependent job control (DJC) allows jobs to be executed in a specific order, as determined by job dependencies. Job dependencies may occur because of data dependencies or may be defined to achieve better device utilization or to manage job streams.

To define a DJC network, the user must include a //*NET control statement in the JCL stream for each job in the network. The //*NET control statement specifies the dependency that must be satisfied before the job can be scheduled for processing. Jobs normally must wait for scheduling until a predecessor job completes. A predecessor job is a job that must complete execution before this job can be scheduled. Jobs that have one or more predecessor jobs are called successor jobs. (Nonstandard DJC jobs are defined with the inclusion of //*PROCESS DJCPROC statements.)

For a complete description of the //*NET and //*PROCESS control statements, see *MVS/Extended Architecture JCL User's Guide.*

Early Dependent Job Control (DJC) JCL Scan

The CI DSPs process the JCL for all jobs in a DJC network through the prescan phase, regardless of the progress of predecessor jobs through the system. As a result, most JCL and control statement errors in those dependent jobs can be detected and corrected, and the job can be resubmitted prior to its release. Once the required predecessor jobs have indicated that dependent jobs can be released for execution, a POSTSCAN DSP is subsequently scheduled to complete postscan processing.

Initializing the DJC Job Network

The first job of a given DJC network entering the system causes the specified DJC network to be defined to JES3. All subsequent jobs with the same DJC network identification become members of that DJC network.

The first DJC job of a particular DJC network can use the DEVPOOL parameter of the //*NET control statement to reserve devices for the entire network. When reserving devices, the user can code the DEVPOOL parameter to refer to the requested devices by name. This parameter should refer to the names defined by the POOLNAMS parameter on the SETNAME initialization statement.

It is important to reserve devices for a DJC network if the DJC jobs pass data sets from one to another; this means that they have similar setup requirements. If devices are not reserved for a DJC network, the DJC jobs contend with other jobs in the system for the available devices when they enter setup. Since DJC jobs are normally held before setup and they are only released for setup when their predecessor jobs have completed, other jobs can take over the devices that the DJC network will soon need again. Both volume mounting operations and the time required by successor jobs to get through the system can be reduced by reserving the commonly required devices for the network. User exit IATUX24 allows you to examine information coded on a //*NET statement. You can examine the network id and the list of requested devices. A return code allows you to accept or reject the device request.

Scheduling the DJC Job Network

The NHOLD parameter on the //*NET control statement specifies the number of predecessor jobs that must complete before the job is eligible for scheduling. If no NHOLD parameter is specified, then the job is eligible for immediate scheduling. If the NHOLD parameter is specified or if the job is in an operator-hold state, only the converter/interpreter and prescan phases of C/I service are scheduled. Postscan processing is suspended until the job is released when all predecessors complete execution.

It is possible to make a job eligible for device setup before its predecessor jobs complete execution. To do this, code the NHOLD and RELSCHCT parameters on the job's //*NET statement. The values of these parameters determine when the job becomes eligible for device setup.

The job becomes eligible for device setup when its NHOLD value is equal to or less than its RELSCHCT value. JES3 reduces the NHOLD value by 1 each time:

- A predecessor job completes execution
- A job (the job need not be part of the DJC network) issues the following form of the DJC write-to-operator message:

  ```
  JESDJC1 jobname net-id
  ```

  (The variable 'jobname' refers to the name of the job to be terminated; NHOLD values for successor jobs will be decremented.)

For more information about using this write-to-operator message, see "DJC Completion Option" in this chapter.

If a job becomes eligible for device setup before its predecessor jobs complete execution, JES3 schedules the job up to but not including generalized main service. JES3 then places the job in DJC-hold status.

Modifying the DJC Job Network

Use the *MODIFY,N operator command to:

- Hold an entire DJC job network or a specific job within the network
- Release an entire DJC job network or a specific job within the DJC network
- Cancel an entire DJC network or a specific job within the DJC network

For additional information, see *MVS/Extended Architecture Operations: JES3 Commands.*

Terminating the DJC Job Network

A DJC job network is purged when all of the following conditions are satisfied:

- All jobs in the DJC network are completed
- There are no missing successor jobs in the DJC network
- There are no missing subnetworks
- The job pending count is equal to zero

The job pending count is the number of abended jobs that have been resubmitted, or have abended with the ABCMP=KEEP parameter specified on the //*NET statement. Specifying this parameter ensures that the DJC network will be retained in the system until the job is resubmitted and completed normally or until the DJC network is flushed by operator commands.

DJC Completion Option

A job completion option is available for the standard DJC job. A problem program may issue a write-to-operator (WTO) message which invokes DJC updating when the message is received. Thus, a job can become eligible for device setup before its predecessor jobs complete execution.

The WTO text format to invoke this option is:

```
JESDJCx jobname net-id
```

This format is positionally dependent. JES must begin in position 1; jobname, which is the name of the job to be terminated, must be 1 to 8 characters and must begin in position 9; net-id must be 1 to 8 characters and must begin in position 18. Comments must begin in position 26. The x in JESDJCx must be specified as 1 for normal job completion or as 2 for abnormal job completion.

Nonstandard DJC Job Processing

A nonstandard DJC job contains //*NET and //*PROCESS statements. For nonstandard DJC jobs, a //*PROCESS DJCPROC statement is required only when a //*PROCESS MAIN statement is *not* included in the job stream. If a //*PROCESS DJCPROC statement is included without a //*NET statement, an error message is issued and the job is flushed. The position of the //*PROCESS DJCPROC control statements indicates when the job is considered complete to DJC, that is, when its successor jobs should be considered for scheduling. A //*PROCESS DJCPROC statement has no parameters and must be preceded by a //*NET control statement. Note that in a standard job not issuing a DJC WTO message, DJC processing occurs after main service has completed.

With the use of the //*PROCESS DJCPROC statements or the DJC WTO message, job completion is always considered normal completion.

For more information on DJC, see *MVS/Extended Architecture JCL User's Guide*.

Controlling Job Selection

The JES3 job selection process can be used by the system programmer to tailor JES3 initialization to meet installation requirements.

Determining Main Eligibility

Main eligibility is determined in the following stages:

1. Input service determines the mains that can execute the job when the SYSTEM and TYPE parameters of the //*MAIN control statement and the job class are analyzed.

2. JES3 MDS selects mains by location of nonshared devices and permanently resident volumes and data sets.

3. The assignment of removable devices by the job can again restrict the mains eligible to run it.

For these reasons, jobs in a JES3 loosely coupled complex are selected for execution on the basis of processor eligibility (an implied attribute) as well as by the explicitly stated class and priority attributes for scheduling.

Determining Job Eligibility

After a job is set up, it becomes eligible for selection by GMS. GMS determines which jobs to select according to the SELECT mode under which the main is running. This gives the I/O rate (CHOICE) for jobs to select and the classes and job class groups eligible for selection for this processor. GMS considers all class constraints (MLIMIT, TLIMIT, TDEPTH, and MDEPTH) in further limiting the jobs that it can select.

Controlling the Job Mix on Each Processor

The system programmer can specify the IORATE parameter of the CLASS initialization statement to describe the I/O rate of jobs in the class. (A low, medium, or high I/O rate can be determined by SMF or performance measurement.) By properly mixing jobs with different I/O-to-processor ratios, the throughput can be increased over that obtained by random mixing.

Defining the Job Selection Environment

Through the use of the MVS JOB statement, the //*MAIN JES3 control statement, and several JES3 initialization statements, you can define the JES3 job selection environment. You can use the GROUP statement to define a job class group and to assign resources (main, initiators, and I/O devices) to that group. The CLASS statement allows you to define job classes, give them a priority, and specify them as members of job class groups. The user can use the MVS JOB statement or the //*MAIN statement to assign a job priority and a job class.

JES3 assigns each main a job selection mode. The SELECT parameter on the MAINPROC initialization statement tells JES3 which mode to assign. The job selection mode can be dynamically changed by a JES3 *MODIFY command if alternate selection modes are defined at initialization.

The effect of partitioning a processor complex should be considered when defining the job selection environment for a main. If one side of a processor complex is partitioned and taken offline, some jobs may not be able to run using the remaining resources. Less storage is available and particular devices, if only attached to the side that has been partitioned off, may no longer be accessible. Alternate selection modes should be defined and used to adjust the amount of batch work scheduled as resources are lost or regained through partitioning.

Figure 2-2 shows two jobs of class C5 in the input stream, and each of these jobs references a specific I/O device number as a DD UNIT parameter. One of these jobs references a nonshared device and, therefore, must execute on the JES3 local processor (designated WORKCPU) in this example. The other job of class C5 references a specific I/O device number that is shared between the JES3 global processor and the JES3 local processor. Since class C5 is assigned to group G2 and has execution resources allocated to both processors, the job may run on either processor.

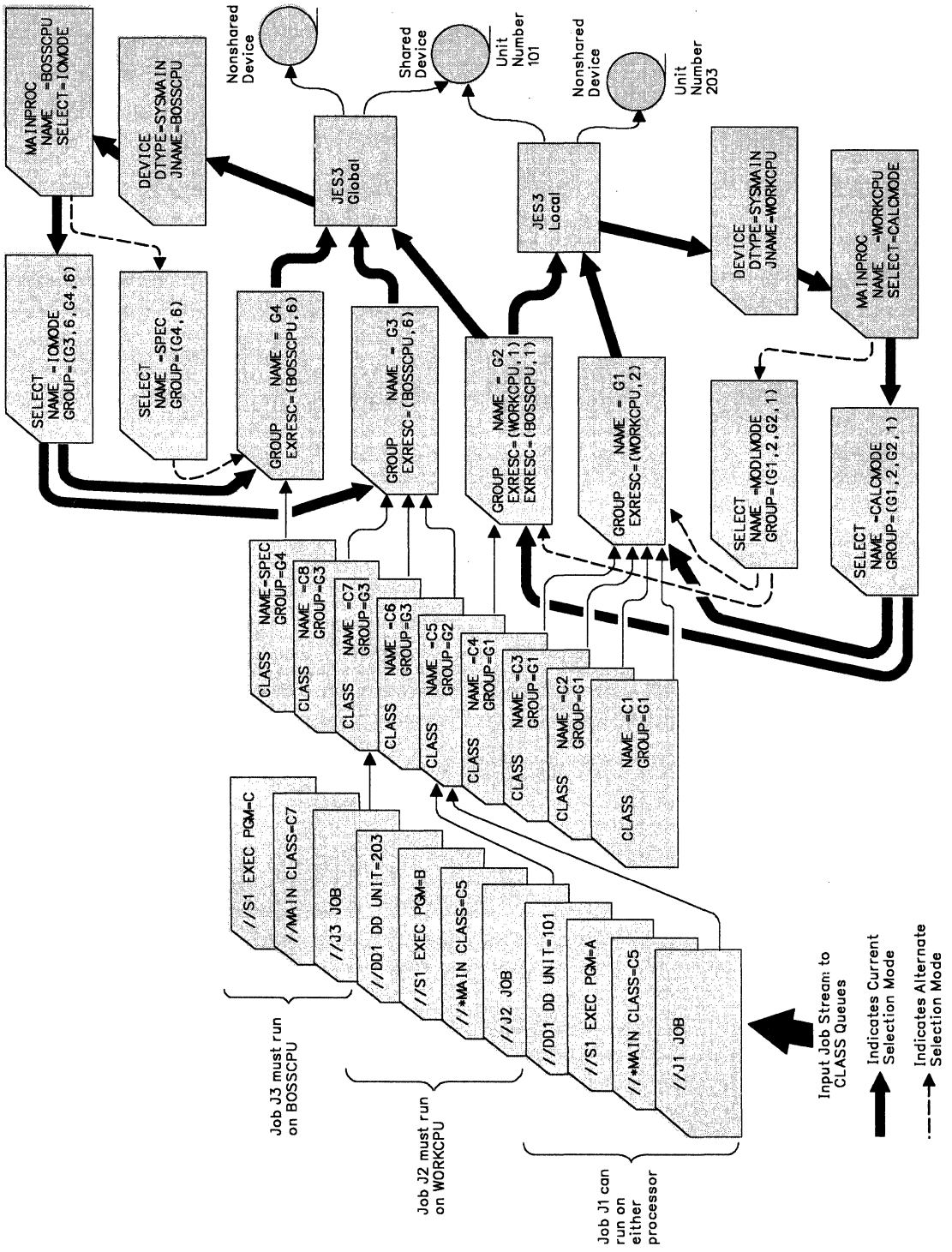


Figure 2-2. JES3 Job Selection Environment

Default Job Selection--Adding Initiators

Jobs are selected for execution by the GMS algorithms. The specific path followed through GMS for a job selection attempt is determined by the parameters specified in the MAINPROC, SELECT, GROUP, and CLASS initialization statements.

The default job selection environment is established when the SELECT, GROUP, and CLASS initialization statements are not specified. Jobs are then selected by priority only, in first-in-first-out order. In this case, two initiators are started on each processor, and default selection mode, group, and class are established with the name JS3BATCH. If more initiators are desired, the dedicated initiator count for the default group can be increased with a *MODIFY command, or a GROUP initialization statement for JS3BATCH can be specified with dedicated initiator counts for each processor.

Thus, the primary decision to be made is the number of initiators to be assigned to each group. This, in turn, determines the number of jobs that can run concurrently on each processor. The dedicated initiator counts, therefore, should not be set substantially above the job processing capability of a processor.

Starting and Stopping Initiators

The initiator allocation and deallocation options specified in the EXRESC parameter of the GROUP initialization statement control the conditions under which initiators are started and stopped.

Demand or dynamic initiator allocation occurs when jobs are available for execution and when the group is enabled. (Dynamic allocation is used to allocate all initiators when the first job of the group in the queue is eligible for main scheduling; demand allocation is used to allocate initiators as they are needed.) Demand or dynamic deallocation occurs when no jobs are available for execution.

If jobs are not continually available for execution, the overhead of starting and stopping initiators may be undesirable. In this case, the IPL allocation option should be used to allocate execution resources during processor connection, or the MANUAL allocation or deallocation option should be specified to indicate that all execution resources are not to be allocated or released until the operator disables the job class group with a *MODIFY command. The allocation and deallocation options should be chosen to maintain initiator availability and minimize start/stop overhead.

Defining Logical Storage for Processors

The LSTOR parameter on the SELECT initialization statement allows you to:

- Use dynamic real storage more efficiently
- Reduce the probability of excessive paging (thrashing)

JES3 schedules jobs on processors according to their available logical storage. When jobs are scheduled, their logical storage requirements are subtracted from the processor's logical storage; if sufficient logical storage is not available to schedule a job, it is not scheduled.

A job's logical storage requirements are determined either by the LREGION parameter on the //*MAIN control statement or by taking a percentage of the region size of the job's largest step. To specify the percentage, code the LSTRR parameter on the CLASS initialization statement.

Warning: Installations that run jobs specifying more than 16 megabytes for the REGION parameter on the JOB or EXEC JCL statement should be aware that JES3 logical storage scheduling considers the region size for those jobs to be 16 megabytes exactly. If jobs require a region size greater than 16 megabytes, logical storage scheduling should be disabled by specifying LSTRR=0 on the appropriate CLASS initialization statement(s).

The combination of methods chosen to control real storage utilization can lead to one of the following operating conditions:

- Real storage is underutilized, and throughput is limited regardless of any other tuning.
- Real storage is fully utilized, and paging is optimum. This condition allows maximum throughput.
- Real storage is fully utilized, but paging is excessive. Again, throughput is limited.

The third condition, excessive paging, can be caused by the system programmer specifying a high LSTOR value, allowing JES3 to schedule too many jobs on the processor. Excessive paging can also be caused by the application programmer specifying a low LREGION value or by defaulting to a low LSTRR percentage (defined by the system programmer).

To obtain maximum throughput, start with underutilization of real storage and gradually increase utilization. Until you optimize throughput, you should retain full control of the logical storage parameters. Application programmers should not use the LREGION keyword on the //*MAIN statement during this phase, and the system programmer should set the LSTRR keyword on the CLASS statement to 99. The maximum value that you can specify on the LSTOR keyword is 32,767.

These guidelines should provide reasonable real storage utilization while the actual real storage requirements (working sets) are determined for typical jobs and job classes. When working set measurements are obtained (through SMF data and other performance measurement tools) and identified for the various jobs and classes, the LREGION and LSTRR parameters can be used to fine-tune the processor throughput.

Job classes should be defined for low, medium, and high I/O rates. If this is not done, all jobs are given the default I/O rate (medium). To use the I/O rate, specify the job selection preference in the CHOICE parameter on the SELECT statement. If I/O rate is considered in scheduling, GMS attempts to select jobs that produce the desired job mix (as specified in the JOBMIX parameter of the SELECT initialization statement). Because the LREGION parameter on the //*MAIN JES3 control statement overrides the LSTRR parameter, use user exit IATUX33 to delete the LREGION parameter when you want logical storage scheduling disabled.

Controlling Job Scheduling

The GMS function of JES3 allows the system programmer to define a set of parameters that determine the scheduling of jobs on each processor in the JES3 complex. Each processor can have a unique set of scheduling parameters.

MAINPROC, SELECT, CLASS, and GROUP initialization statements allow the system programmer to control the key variables in the job scheduling and execution process. The dynamic interaction of these statements defines a job-selection mode that controls job selection in a specified way. The following parameter description provides an overview of these controls.

Assigning a Job Selection Mode

The MAINPROC statement describes each processor. The SELECT parameter on this statement specifies the name of the job selection mode to be assigned initially to the processor. The name designated with the SELECT parameter must also be specified in the NAME parameter of a SELECT initialization statement.

If the SELECT parameter is omitted, a selection mode is established using the SELECT statement default values. (The default selection mode is designated by JES3 as JS3BATCH.)

Defining Job Selection Parameters

The SELECT statement defines the job selection parameters for each job selection mode. The job selection mode is assigned to a processor by the SELECT parameter on the MAINPROC statement. After initialization, the system operator can change the association of processors with selection modes by using the *MODIFY command.

The system programmer can control the number of jobs that are candidates for allocation by setting the SELECT statement SBAR parameter. The SBAR parameter specifies a job priority that is a barrier to main device scheduling:

- If its full allocation requirements cannot be satisfied, a job with a priority greater than the barrier can reserve available JES3-managed resources (devices, volumes, and data sets) to prevent lower priority jobs from obtaining them.
- If a resource is reserved, only a job of the same priority or higher than the job that reserved the resource can allocate it.
- If a volume or data set is reserved, another job with compatible references (such as a share reference to a direct-access volume) can use it.
- If SBAR=PRTY is specified, the priority of the first job that cannot be set up is the barrier value.

The CHOICE parameter on the SELECT statement can be used to specify job selection criteria (based on the size of the job and its I/O rate) to control the order of job selection on a processor. JES3 uses the specified scheduling choice to select the most suitable jobs for execution.

Aging is the process of increasing a job's priority whenever an unsuccessful attempt is made to allocate the job's requirements. Although jobs flow through the priority queues in first-in first-out order, only the job that is first on a priority queue is eligible for aging. The system programmer controls aging by setting the following parameters on the SELECT statement:

- *MAGER:* Specifies the number of times a job must be eligible for aging (due to unsuccessful job selection) before its job priority is actually increased. (For example, MAGER=10 means that a job must be passed over for job selection 10 times while it is at the top of its priority queue before it is put at the bottom of the next higher priority queue.) If this parameter is omitted or if MAGER=0 is specified, no aging is performed.

- *MAGEL:* Limits the priority that can be reached (due to unsuccessful job selection attempts). For example, MAGEL=10 means that a job is not aged if its priority is 10 or greater.) If this parameter is omitted, jobs are not aged past priority 14.

- *SAGER:* Specifies the number of times that a job must be eligible for aging (due to unsuccessful resource allocation) before its job priority is actually increased. If this parameter is omitted or if SAGER=0 is specified, no aging is performed.

- *SAGEL:* Specifies an aging priority limit (0-15) beyond which a job cannot be aged during job setup.

The JES3-managed resource allocation (setup of devices, volumes, and data sets) for each job must be completed before the job is eligible for execution. The system programmer controls the execution queueing process by setting the following parameters on the SELECT initialization statement:

- *SDEPTH:* Specifies the maximum number of jobs (requiring mountable devices) that can be set up at one time for each processor. SDEPTH influences MDS allocation in two ways: First, SDEPTH can be used to limit the number of jobs set up for a processor; this should be done carefully to avoid delaying job execution. If SDEPTH is omitted, a value of 255 is assumed. Second, SDEPTH causes MDS to set up more work for one processor than another. MDS biases processor selection for setup toward the processor that is most below its SDEPTH. Therefore, by making the SDEPTH values different for each processor, a setup bias initially exists toward processors with higher SDEPTHs. MDS prefers the higher SDEPTH processors for setup until all processors are at equal differences from their SDEPTH values. Biasing setup toward certain processors may be desired because of device availability (if devices are not totally shared) or to keep a larger queue of jobs available for a fast executing processor.

- *INCR:* Specifies a number that is added to the priority of the job when it is set up. This parameter expedites the processing of jobs once devices have been assigned to them. (For example, if a job has a priority of 5 when it is set up, and INCR=4 is specified, the job's priority is increased to 9 after the devices have been allocated and set up.) If this parameter is omitted, a job's priority is increased by one after setup.

- *INCL:* Sets a limit to the priority assigned when the job is set up. If this parameter is omitted, job priorities are not increased past priority level 14 by setup.

Defining JES3 Job Classes

The CLASS statement defines the characteristics of the JES3 job classes. Up to 255 job classes can be defined. A CLASS statement must define each job class that appears on the JOB or //*MAIN control statement. If an undefined class is specified, the job is canceled. If a class is not specified on a JOB or //*MAIN control statement, the default class is used.

The system programmer controls class execution limits and limit dependencies by setting the following parameters on the CLASS statement:

- *SDEPTH:* Sets the maximum number of jobs requiring mountable devices (0 to 255) that can be set up at one time. If the SDEPTH parameter is omitted, no SDEPTH default is assumed for this class. (A typical use of SDEPTH would be to prevent overuse of devices by limiting the number of jobs that could be set up at one time in test class.)

- *TDEPTH:* Sets the maximum number of jobs of this class (0 to 255) that can execute in the total JES3 complex at one time. If the TDEPTH parameter is omitted, no TDEPTH default is assumed for this class.

- *MDEPTH:* Sets the maximum number of jobs of this class (0 to 255) that can execute on a given processor at one time. Each processor name specified in this parameter must also be specified on the NAME parameter of a MAINPROC initialization statement. If the MDEPTH parameter is omitted, no MDEPTH default is assumed for this class.

- *TLIMIT:* Specifies the maximum number of jobs of other job classes that can execute in the total JES3 complex and still allow jobs in this class to be scheduled. If any class limit is exceeded, no more jobs in this class are scheduled; that is, jobs in this class are scheduled only when the number of jobs running from other classes is equal to or less than the assigned limit. Each class name specified in this parameter must also be specified on the NAME parameter of another CLASS statement. If the TLIMIT parameter is omitted, no TLIMIT default is assumed for this class.

- *MLIMIT:* Specifies the maximum number of jobs of other job classes that can execute on a given processor and still allow jobs in this class to be scheduled. If any class limit is exceeded, no more jobs in this class are scheduled on the given processor; that is, jobs in this class are scheduled only when the number of jobs running from other classes is equal to or less than the assigned limit. Each class name specified in this parameter must also be specified on the NAME parameter of another CLASS statement. If the MLIMIT parameter is omitted, no MLIMIT default is assumed for this class.

Grouping JES3 Job Classes

The GROUP statement defines the resources available to a group of JES3 job classes. A maximum of 255 groups can be defined. The system programmer controls execution by specifying the EXRESC parameter on the GROUP statement. The EXRESC parameter defines the execution resources (initiators and devices) dedicated to a group. Devices assigned to the group satisfy requests for mountable volumes (not permanently resident) from jobs within the group. The EXRESC parameter should be repeated for each processor's execution resources.

The EXRESC parameter also defines the initiator allocation and deallocation options (see "Starting and Stopping Initiators" in this chapter).

The BAR parameter on the GROUP statement specifies a job priority used as a barrier to the generalized main scheduling process; it determines when no more jobs should be scheduled on an associated processor. Within a job class group, no jobs with priorities lower than the barrier are run until all jobs with priorities higher than the barrier have run. BAR can limit job scheduling due to processing requirements (such as, a job that cannot obtain sufficient storage on a processor or a job that is incompatible with the current job mix under a best-mix scheduling algorithm). If BAR=PRTY is specified, the priority of the first job that cannot be scheduled is the barrier value. BAR=16, the default, means that there is no priority barrier for the group.

Output Service

Output service executes on the global processor and processes SYSOUT data sets destined for print, punch, TSO, internal reader, and external writer. The output service driver receives control after a job completes breakdown in main service, after a job spins off an output data set, or after JES3 spins off an output data set.

JES3 output service performs three distinct functions:

- queueing output
- scheduling output
- writing output

Queueing Output

Normally, output data produced by a job is placed in one of three output service queues when the job terminates. Spin-off data sets are placed in an output queue while the job is still in execution. The three output queues are:

- *MVS/BDT work queue (Q=BDT):* This queue contains SNA/NJE networking job or networking SYSOUT streams. MVS/BDT sends these job or SYSOUT streams to the proper node within a SNA/NJE network for processing by JES3. Using commands, the operator may hold, release, or cancel the networking requests from the queue.

- *Output service writer queue (Q=WTR):* This queue contains data sets waiting for output processing. Such data sets are processed automatically by

output service, based on data set selection characteristics, such as output device-related requirements, output class, and output priority. Data sets in this queue may be temporarily placed in operator-hold status.

- *Output service hold queue (Q=HOLD):* This queue, sometimes called the hold-for queue, contains data sets that are to be processed by other than normal JES3 output services. These data sets must be processed by the function for which they are held (external writer, internal reader, or TSO). The function that processes the data set may then release it for JES3 processing or cause JES3 to purge it. If necessary, the operator can force a JES3 writer to process the data set. The operator can also delete held output data sets using the JES3 spool maintenance facility (JSM). For information about using JSM, see Chapter 4, "Defining and Managing Spool Data Sets."

The RESQUEUE entry for the job to be processed by the queueing function is placed on a queue of output service work. When the queueing function receives control, it dequeues the next job on the work queue.

The queueing function of output service accesses the job data set (JDS) for the job or for the spin data sets of a job. The queueing function builds output scheduling elements (OSEs) from the JDS. One OSE is built for each group of data sets that have unique writer requirements.

The information in an OSE for output data sets on the writer queue comes from:

- JCL parameters on the SYSOUT DD and OUTPUT JCL statements for the job (The installation can change these parameters during input service by coding user exit routine IATUX34 for the SYSOUT DD statement and user exit routine IATUX44 for the OUTPUT statement.)

- the //*FORMAT JES3 control statements for the job (The installation can change these statements during input service by coding user exit routine IATUX33.)

- the SYSOUT class table (defined by SYSOUT initialization statements)

Information from //*FORMAT control statements is not included in OSEs for data sets on the hold queue.

When moving data sets from the hold queue to the writer queue, all the original output characteristics may not be maintained. This occurs if JES3 is unable to determine which output JCL statement was used to build the OSE on the hold queue.

The OUTSERV initialization statement specifies default values for information not provided on other JES3 initialization statements or on JCL statements. If the OUTSERV initialization statement parameters are not overridden, the default values specified are applied to all jobs entered in the system.

The information in an OSE includes:

- Data set priority
- Data set destination
- Specific device types requested
- Forms requested

- Carriage tape name or FCB specified
- Train name specified
- Number of lines
- SYSOUT class
- Data set type (print, punch)
- External writer name, if specified
- Copy count for the 3800 or 3820 printer
- Forms flash ID for the 3800 printer
- Copy modification ID for the 3800 printer
- Character assignment tables for the 3800 or 3820 printer
- Stacker for the 3800 printer
- Data set processing mode
- Checkpoint-related information for the 3800-3 or 3820 printer
- TSO user ID
- Data set ID (for 3540 device only)

Override Sequences for Output Data Set Information

For the output parameters of a data set, you can override the OUTSERV initialization statement parameters. The method used to override the parameters depends on whether OUTPUT JCL statements and/or //*FORMAT JES3 control statements are specified for a data set, and whether the references are explicit or by default. JES3 does not merge the information from the OUTPUT and //*FORMAT statements that refer to the same data set. Under certain circumstance (explained below), JES3 creates separate OSEs from the two types of statements.

Override Sequence with "Direct" OUTPUT JCL Statements: A "direct" OUTPUT JCL statement is one explicitly referenced by the OUTPUT parameter of a SYSOUT DD JCL statement for a data set, or an OUTPUT JCL statement that specifies the JESDS parameter. If there is a direct OUTPUT JCL statement for the data set, the OUTSERV initialization statement parameters are overridden by the following sources:

1. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement.

2. *"Direct" OUTPUT JCL statement:* The values specified on the direct OUTPUT JCL statement override the values specified by source 1 (above).

3. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 and 2. This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.

Following is an example of this override process using the FORMS parameter on the OUTSERV initialization statement and "direct" OUTPUT JCL statements. In this example, the JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT
```

Under these conditions, the following job is run:

```
//PRTFORMA      JOB     MSGCLASS=F,PRTY=7
//OUTDEF        OUTPUT  BURST=Y,DEST=POK,FORMS=2PRT,DEFAULT=NO
//STEP1         EXEC    PGM=IEBPTPCH
//SYSPRINT      DD      SYSOUT=(F,,4PRT),OUTPUT=*.OUTDEF
//SYSUT1        DD      SYSOUT=F,OUTPUT=*.OUTDEF
                .
                .
                .
```

Figure 2-3 illustrates the override process for this situation.

Data Set	FORMS Initial Value in OUTSERV Statement	Order of Overrides				FORMS Final Value
		1. SYSOUT, CLASS=F	2. OUTPUT FORMS=2PRT	3. DD SYSOUT=(F,,4PRT), OUTPUT=*.OUTDEF	4. DD SYSOUT=F, OUTPUT=*.OUTDEF	
SYSPRINT	1PRT	3PRT	2PRT	4PRT	n/a	4PRT
SYSUT1	1PRT	3PRT	2PRT	n/a	none	2PRT
JESJCL	1PRT	3PRT	n/a	n/a	n/a	3PRT
JESMSG	1PRT	3PRT	n/a	n/a	n/a	3PRT
SYSMSG	1PRT	3PRT	n/a	n/a	n/a	3PRT

Figure 2-3. Output Parameter Overrides Using a Direct OUTPUT JCL Statement

JES3 builds one OSE for each direct OUTPUT JCL statement referring to a data set. The user receives a copy of the output data set for each OUTPUT statement. Each copy is formatted according to the processing options specified on the OUTPUT statement that produced it.

Override Sequence with "Direct" //*FORMAT JES3 Control Statements: A "direct" //*FORMAT JES3 control statement is one that uses the DDNAME parameter to explicitly reference a specific data set (that is, having the format //*FORMAT xx,DDNAME=xxxx). If there is a direct //*FORMAT statement, the OUTSERV initialization parameters are overridden by the following sources:

1. *//*FORMAT xx,DDNAME=, JES3 control statements:* When DDNAME=, is given and no ddname follows, the parameters specified on this statement become the defaults for the job and apply to all TSO, punch and print data sets.

2. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement. The values specified here override those specified by source 1.

3. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 and 2. This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.

4. *//*FORMAT xx,DDNAME=xxxx JES3 control statement:* The values specified here override those specified by sources 1, 2, or 3.

Following is an example of this override process using the FORMS parameter on the OUTSERV initialization statement and both direct and default //*FORMAT JES3 control statements. In this example, the JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT
```

Under these conditions, the following job is run:

```
//PRTFORMB          JOB  MSGCLASS=F,PRTY=7
//*FORMAT           PR,DDNAME=,FORMS=SPECIAL
//*FORMAT           PR,DDNAME=SYSMSG,FORMS=2PRT
//STEP1             EXEC     PGM=IEBPTPCH
//SYSPRINT          DD       SYSOUT=F
//SYSUT1            DD       *
                      .
                      .
                      .
                    (data)
/*
//SYSUT2            DD       SYSOUT=(F,,2PRT)
//SYSIN             DD       *
                    PRINT    MAXFLDS=100
                    RECORD   FIELD=(80)
/*
```

Figure 2-4 illustrates the override process for this situation.

Data Set	FORMS Initial Value in OUTSERV Statement	Order of Overrides				FORMS Final Value
		1. //*FORMAT PR, DDNAME=	2. SYSOUT, CLASS=F	3. DD SYSOUT= (F,,2PRT)	4. //*FORMAT PR, DDNAME= SYSMSG	
SYSPRINT	1PRT	SPECIAL	3PRT	n/a	n/a	3PRT
SYSUT2	1PRT	SPECIAL	3PRT	2PRT	n/a	2PRT
JESMSG	1PRT	SPECIAL	3PRT	n/a	n/a	3PRT
JESJCL	1PRT	SPECIAL	3PRT	n/a	n/a	3PRT
SYSMSG	1PRT	SPECIAL	3PRT	n/a	2PRT	2PRT

Figure 2-4. Example of Output Parameter Overrides Using Direct and Default //*FORMAT Statements

If there are more than one default //*FORMAT statements for the same data set, JES3 merges the information from them. JES3 lets parameters specified by later default //*FORMAT statements override those specified by earlier default //*FORMAT statements. The user receives the data set formatted only one way.

Note: **If both direct OUTPUT JCL statements and direct //*FORMAT JES3 control statements apply to the same data set,** JES3 formats the data set according to the specifications of *each* direct OUTPUT JCL statement and *each* direct //*FORMAT JES3 control statement. That is, JES3 creates a separate OSE for each direct OUTPUT JCL statement and each direct //*FORMAT JES3 control statement. Each OSE results in a formatted version of the data set, so that there are as many formatted versions of the data set as there are direct OUTPUT JCL statements and direct //*FORMAT JES3 control statements for that data set.

Override Sequence With Only "Default" OUTPUT JCL Statements: Next, if there is neither a direct OUTPUT JCL statement for the data set nor a direct //*FORMAT JES3 control statement, JES3 looks for a "default" OUTPUT JCL statement. A "default" OUTPUT JCL statement is a step-level or job-level OUTPUT JCL statement that specifies YES on the DEFAULT parameter. If there is a default OUTPUT JCL statement, the OUTSERV initialization parameters are overridden by the following sources:

1. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement.

2. *"Default" OUTPUT JCL statement:* If a step-level default statement applies, JES3 ignores the job-level default statement. The values specified here override those specified by source 1.

3. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 and 2. This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.

Note: "Default" OUTPUT JCL statements do not apply to system data sets, such as JESJCL, JESMSG, and SYSMSG, unless explicitly specified. To learn how to use the OUTPUT JCL statement with system data sets, see *MVS/Extended Architecture JCL User's Guide.*

Following is an example of this override process using the FORMS parameter on the OUTSERV initialization statement and a "default" OUTPUT JCL statements. In this example, the JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT
```

Under these conditions, the following job is run:

```
//PRTFORMC      JOB     MSGCLASS=F,PRTY=7
//OUTDEF        OUTPUT  DEFAULT=YES,BURST=Y,DEST=POK,FORMS=2PRT
//STEP1         EXEC    PGM=IEBPTPCH
//SYSPRINT      DD      SYSOUT=(F,,4PRT)
//SYSUT1        DD      SYSOUT=F
                  .
                  .
                  .
```

Figure 2-5 illustrates the override process for this situation.

Data Set	FORMS Initial Value in OUTSERV Statement	Order of Overrides				FORMS Final Value
		1. SYSOUT, CLASS=F	2. OUTPUT DEFAULT=YES, FORMS=2PRT	3. DD SYSOUT=(F,,4PRT)	4. DD SYSOUT=F	
SYSPRINT	1PRT	3PRT	2PRT	4PRT	n/a	4PRT
SYSUT1	1PRT	3PRT	2PRT	n/a	none	2PRT
JESJCL	1PRT	3PRT	n/a	n/a	n/a	3PRT
JESMSG	1PRT	3PRT	n/a	n/a	n/a	3PRT
SYSMSG	1PRT	3PRT	n/a	n/a	n/a	3PRT

Figure 2-5. Example of Output Parameter Overrides Using a Default OUTPUT JCL Statement

As with the direct OUTPUT JCL statement, JES3 builds a separate OSE for each default OUTPUT JCL statement of the same level (job or step) referring to the same data set. The user receives a copy of the data set for each OUTPUT statement, each formatted according to the characteristics specified by the OUTPUT statement that produced it. This process occurs *only* if there are no direct OUTPUT JCL statements for the data set. If direct OUTPUT JCL statements apply to the data set, JES3 ignores default OUTPUT JCL statements.

Override Sequence With Only "Default" //*FORMAT JES3 Control Statements: Finally, if there are no direct or default OUTPUT JCL statements and no direct //*FORMAT JES3 control statements, the OUTSERV initialization parameters are overridden by the following sources:

1. *//*FORMAT xx,DDNAME=, JES3 control statements:* When DDNAME=, is given and no ddname follows, the parameters specified on this statement become the defaults for the job and apply to all TSO, punch and print data sets that have no FORMAT statements.

2. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement. The values specified here override those specified by source 1.

3. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 and 2. This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.

Following is an example of this override process using the FORMS parameter on the OUTSERV initialization statement and a "default" //*FORMAT JES3 control statement. In this example, the JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT
```

Under these conditions, the following job is run:

```
//PRTFORMD      JOB     MSGCLASS=F,PRTY=7
//*FORMAT       PR,DDNAME=,FORMS=SPECIAL
//STEP1         EXEC    PGM=IEBPTPCH
//SYSPRINT      DD      SYSOUT=(F,,4PRT)
//SYSUT1        DD      SYSOUT=F
                .
                .
                .
```

Figure 2-6 illustrates the override process for this situation.

Data Set	FORMS Initial Value in OUTSERV Statement	Order of Overrides 1. //*FORMAT PR, DDNAME=	2. SYSOUT, CLASS=F	3. DD SYSOUT=(F,,4PRT)	4. DD SYSOUT=F	FORMS Final Value
SYSPRINT	1PRT	SPECIAL	3PRT	4PRT	n/a	4PRT
SYSUT1	1PRT	SPECIAL	3PRT	n/a	none	3PRT
JESJCL	1PRT	SPECIAL	3PRT	n/a	n/a	3PRT
JESMSG	1PRT	SPECIAL	3PRT	n/a	n/a	3PRT
SYSMSG	1PRT	SPECIAL	3PRT	n/a	n/a	3PRT

Figure 2-6. Example of Output Parameter Overrides Using a Default //*FORMAT Statement

Override Sequence for Held Output Classes: The following example shows override processing using the forms parameter on the OUTSERV and SYSOUT initialization statements. In this example, the initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,HOLD=TSO,FORMS=3PRT
```

In the following JCL, output from the SYSPRINT and SYSUT1 DD statements is directed to a held output class (F). Only the SYSPRINT DD statement references an OUTPUT JCL statement.

```
//HOLD      JOB     MSGCLASS=F,PRTY=7
//OUTDEF    OUTPUT  BURST=N,DEFAULT=NO
//STEP1     EXEC    PGM=IEBPTPCH
//SYSPRINT  DD      SYSOUT=F,OUTPUT=*.OUTDEF
//SYSUT1    DD      SYSOUT=F
```

Rule: Whenever a SYSOUT statement uses an OUTPUT statement (a direct or default type), JES3 uses values from the OUTPUT JCL statement, the OUTSERV initialization statement, and the SYSOUT initialization statement to establish the data set's output characteristics.

In the above example, SYSPRINT and SYSUT1 will both be placed in the output service hold queue for processing by TSO. However, the forms value for SYSPRINT becomes 3PRT because JES3 merges the values from the OUTPUT JCL statement with the values on the OUTSERV and SYSOUT initialization statements.

If a SYSOUT DD statement specifies a held output class and does not use an OUTPUT JCL statement, JES3 uses values from only the OUTSERV initialization statement and the SYSOUT DD statement to establish a data set's output characteristics. Therefore, the forms value for SYSUT1 remains 1PRT since values from the SYSOUT initialization statement are not used. If a //*FORMAT card had been coded for the SYSUT1 DD statement, it also would not have been used.

For information about how to use the JCL statements and JES3 control statements to achieve the output results you desire, see

As each data set is selected, a temporary OSE is constructed containing all scheduling requirements. A call is then made to user exit IATUX19 to allow the system programmer to examine and, if desired, to change the information before the OSE is spooled.

When control is returned from the user exit, if the OSE copy count is nonzero, the OSE is added to the job's OSE spool file. Then, the master OSE (MOSE) pool, which is in main storage, is searched. If an identical master OSE is found, an output scheduling summary (OSS) entry is chained onto that master OSE. If no identical master OSE is found, a new master OSE is created and an OSS is also created, chained to the new MOSE, and added to the pool.

After all the above processing for the current job has completed, the queueing function posts the start writer function to determine which jobs are eligible for writer scheduling. The start writer function tries to associate each master OSE of a job with an output device or with an active writer that is waiting for more work.

Scheduling Output

JES3 output service schedules OSEs to writers in one of two ways:

- An OSE is used to scan the 'writers waiting for work' queue and the available-devices queue to find a device that can process the OSE.

- A set of writer scheduling parameters is used to search the OSEs for the first perfect-fit OSE or for the OSE which best fits the requirements of the writer requesting a job. You can specify these parameters on the DEVICE or OUTSERV initialization statement by coding the WC and WS parameters. The operator can change these parameters when calling, starting, or restarting a writer. The operator does this by specifying the WC and WS parameters on the *X, *S, or *R commands.

  If two or more OSEs fit the requirements of this writer equally well, JES3 schedules the OSE with the highest JES3 job queue priority. The JES3 job queue priority is based on the job priority specified on the JCL JOB statement.

Writing Output

JES3 writer support consists of a writer driver, writer scheduling (selection) routines, device-dependent routines, command-processing routines (also called message-processing routines) and spool-access routines (for print and punch writers).

In most cases, the writer support is provided within the JES3 global address space. Certain devices, however, use device-dependent routines that operate in a separate address space called an output writer functional subsystem (FSS) address space. In this case, the writer driver and the command-processing routines operate in the JES3 global address space and communicate with the output writer FSS using the functional subsystem interface (FSI). The device-dependent routines, also called a functional subsystem application (FSA), and the spool-access routines operate in the output writer FSS. To define an output writer FSS, see "Running a Printer Under an Output Writer Functional Subsystem," in Chapter 6, "Defining and Managing JES3 Resources."

Note: The spool-access routines in an output writer FSS read and write spool data from USAM protected data buffers (PBUFs) in CSA. JES3 does not release a PBUF until all the records in the PBUF have been copied to a private area buffer in preparation for passing them to the FSA. To allocate enough pages of storage for PBUFs used by all the output writer FSSs in the JES3 complex, use the PRTPAGE parameter on the MAINPROC statement.

Two methods of controlling the starting and stopping of writing are provided. The start/stop criteria identify the writer as a hot writer or a dynamic writer:

- *Hot writer:* The operator controls the writer and its associated devices via the *CALL, *START, *RESTART, or *CANCEL commands. The writer notifies the operator when it is waiting for work and remains available for processing.

- *Dynamic writer:* The starting and stopping of the writer and its associated devices is controlled by JES3 output service based on the availability of output devices and output data set requirements that exist at any given time. When no more data sets with defined characteristics for the writer are available for processing, the writer is automatically terminated.

  Caution: Running an output writer FSS as a dynamic writer could slow output processing, because an address space must be brought up or down each time the writer is started or stopped.

The output service writers process system output data sets destined for print, punch, external writers or for the internal reader:

- *Print:* The print (hot or dynamic) writer processes any output data sets for which SYSOUT classes were defined during JES3 initialization as TYPE=PRINT and any additional data sets described by //*FORMAT PR control statements. The printer writer accepts data in EBCDIC format, ready for printing with either MVS channel command forms control or the MVS-supported extended ASCII channel command code, or with no carriage control. Output to the printer is command-chained.

- *Punch:* The punch (hot or dynamic) writer processes any output data sets for which SYSOUT classes were defined during JES3 initialization as TYPE=PUNCH, and any additional data sets described by //*FORMAT PU control statements. The punch output writer ignores ASCII and MVS channel-command stacker selection characters. Also, the punch output writer does not support column binary mode (DCBMODE=C).

- *External writer:* For a description of external writer usage, see the following subsection.

- *MVS internal reader:* The MVS internal reader acts as a JES3 writer that passes submitted jobs to input service.

External Writers

External writer routines execute in an address space other than the JES3 address space. This type of writer is functionally independent of JES3 and operates as a completely separate MVS job. However, the external writer interacts with JES3, via the subsystem interface, to request data sets for processing. A subset of the output service scheduling function called PROCESS SYSOUT is invoked as a result of this kind of request.

No attempt is made by output service to schedule external writers as a result of constructing OSEs requiring their services; it is the responsibility of the operator to start external writers as required.

For more information on external writers, see *MVS/Extended Architecture System Programming Library: System Modifications.*

NJERDR

The NJE reader accepts incoming SNA networking job or SYSOUT streams from MVS/BDT. The reader interacts with JES3 by using the PROCESS SYSOUT scheduling function to request work from output service.

Data sets with the destination 'NJERDR' are passed to the reader. The reader determines whether the networking streams can be processed at the home node or be forwarded to another node. The networking streams are spooled and a JES3 job is created to process them.

The NJE reader is started and terminated by the operator using the *CALL,NJERDR and *CANCEL,NJERDR commands, respectively.

Internal Reader

Internal reader routines allow TSO jobs or application programs to submit job streams to JES3 via output data sets. When a job stream enters the system, input service assigns the data sets directly to an internal reader. If an internal reader is not available, the system dynamically creates one. When JES3 schedules the internal reader, input service can proceed to process the data set as an input stream.

As mentioned, an application program can create a job and pass it to the JES3 internal reader. To pass the job to the internal reader, the program must write the job to a data set for which SYSOUT=(class,INTRDR) has been specified. The internal reader reads the job, then gives it to JES3 where the job is processed like other reader-submitted jobs.

To display a list of all the internal readers in the system at any one time, issue the *INQUIRY,A,D=INTRDR command. As the workload slows, input service will automatically terminate internal readers when they are no longer required. The operator can also stop the internal reader by issuing a *CANCEL,INTRDR or a *CANCEL,J=jobno command. For additional information about commands that control the internal reader see *MVS/Extended Architecture Operations: JES3 Commands.*

A check is made for a /*EOF and /*DEL statement. If either is found, an ENDREQ request is issued via SVC 111 to spin off the data set.

Following is an example of the DD specification and application program code needed to use the internal reader:

```
//SYSUT2    DD       SYSOUT=(X,INTRDR),DEST=PRINTER2

PGMX        CSECT
            OPEN     (SYSUT2,OUTPUT)     OPEN JCL DATA SET
            PUT      ('CARDS')           CREATE JOB STREAM
            CLOSE    (SYSUT2)            SPIN OFF DATA SET

CARDS       DATASTART
            //TEST1 JOB
            //STEP1 EXEC     PGM=IEFBR14
            /*
            //TEST2 JOB
            //STEP1 EXEC     PGM=IEFBR14
            /*
            DATAEND
            END
```

In the example, SYSOUT class X is specified on the DD statement. However, any class may be specified. In this example, class X becomes the MSGCLASS for the job unless the MSGCLASS parameter has been specified on the JOB statement. Sometimes, the spin-off data set is dynamically allocated without a MSGCLASS specified, as in the case of a TSO SUBMIT created job. When this occurs, the MSGCLASS of the submitting job, or TSO user, becomes the default MSGCLASS. TSO LOGON jobs and started tasks, such as initiators, are assigned the default MSGCLASS defined in the initialization deck.

The DEST parameter on the DD statement specifies the destination for print/punch output from submitted jobs. If no destination or an invalid destination is specified, print/punch output is routed to any printer or punch directly attached to the global processor (DEST=ANYLOCAL).

Specifying FREE=CLOSE on a SYSOUT DD statement causes the data set to be deallocated and made available for further system processing at the time the data set is closed. JES3 creates a separate track allocation table (TAT) for this type of data set. This allows the spool space allocated to be freed independently of the job with which the data set is associated. This type of data set is called a spin data set. A spin data set must be purged from the system before its associated job is purged.

The FREE=CLOSE parameter should not be specified for a data set that is opened and closed more than once during a job step. If the data set is reopened, the job step abnormally terminates unless there is an intervening dynamic allocation.

TEST1 and TEST2 are passed to JES3 as a spin data set. This spin data set then becomes available for processing by the internal reader. To invoke the internal reader, enter *X,INTRDR on the global processor. The first available internal reader data set will be returned. The internal reader passes this data set to input service to be processed as a job stream.

The spin data set is treated as a batched input stream. Job processing does not begin until the data set is closed. When the data set is closed, a job number is assigned to the spin data set. When input service processes the first job of the data set, the preassigned job number is given to the job. Subsequent jobs in the data set are assigned job numbers when input service processes the job.

If the data set has an independent TAT to control spool space assigned to the data set, the spin data set facility releases the spool space occupied by the data set once the output processing of the data set is complete. If the data set does not have an independent TAT, the space associated with the spin data set is retained until all JES3 processing for the job is complete and the job is purged.

Abnormal closing of the data set or closing after a write error causes deletion of the last job in the input stream. A /*DEL statement may be used to explicitly delete the job.

For more information about using the internal reader, see *MVS/Extended Architecture System Programming Library: System Modifications.*

Accessing Job Output Through TSO

It is possible for a TSO user to access the output--job control language (JCL), system messages, and system output (SYSOUT) data sets--of a batch job. The user that submits the batch job must first make the job's output available to TSO. There are two ways the user can do this:

- Assign the job's output that is to be accessed to a SYSOUT class for which you have specified HOLD=TSO.
- Specify on the // JOB statement MSGCLASS parameter a SYSOUT class for which you have specified TYPE=RSVD; and assign the job's SYSOUT data sets to a SYSOUT class for which you have also specified TYPE=RSVD.

You must have previously specified HOLD=TSO and TYPE=RSVD on the appropriate SYSOUT initialization statements.

To access the output of a batch job, which is on the JES3 spool, the TSO user must issue the TSO command, OUTPUT. How to use this command is described in *TSO/E Command Reference.*

Following is an example of the use of reserved classes:

```
//JOB1          JOB         ...,MSGCLASS=T
//S             EXEC        ...
//DD1           DD          SYSOUT=A
//DD2           DD          SYSOUT=E
//DD3           DD          SYSOUT=F
```

The following initialization statements are included in the initialization stream:

```
SYSOUT,CLASS=A,TYPE=PRINT
SYSOUT,CLASS=E,TYPE=(PRINT,RSVD)
SYSOUT,CLASS=F,TYPE=PRINT,HOLD=TSO
SYSOUT,CLASS=T,TYPE=(PRINT,RSVD)
```

In this example, SYSOUT data from DD1 is processed by output service as normal SYSOUT. SYSOUT data from DD2 is placed in HOLD for return to the TSO user (because the MSGCLASS SYSOUT class of the job is also assigned to a RSVD SYSOUT class). SYSOUT data from DD2 is processed by output service as normal SYSOUT if the MSGCLASS SYSOUT class is changed to class A (because the MSGCLASS SYSOUT class is no longer assigned to a RSVD class). SYSOUT data from DD3 is placed in HOLD for return to the TSO user (because class F is specified as unconditional HOLD for TSO).

Virtual Storage Personal Computing: Output Requirement

You must assign output from the VSPC (Virtual Storage Personal Computing) program product to a SYSOUT class whose SYSOUT initialization statement specifies HOLD=EXTWTR or TYPE=RSVD.

Output Service User Exits

Output service provides five user exits, which are summarized in Figure 2-7.

User Exit Routine	Purpose
IATUX19	Allows the user to access the contents of the temporary OSEs
IATUX20	Allows the user to access the information to be written on job header pages
IATUX21	Allows the user to access the data that is to be written on the data set header pages
IATUX22	Allows the user to alter the forms alignment
IATUX23	Allows the user to access the data to be written on the job trailer pages
IATUX45	Allows the user to modify the job information for data sets processed by output writer functional subsystems

Figure 2-7. Output Service User Exits

The output writer functional subsystem application (FSA) in an output writer FSS includes several non-JES3 user exits, which are explained in *Print Services Facility/MVS System Programmer's Guide*. Refer to that document for all information on user exits from the output writer FSA.

Purge

Purge is the last processing function for a job in the JES3 system. It releases all JES3 DASD space assigned to the job, making it available for allocation to subsequent jobs. A message is issued to the operator indicating that the job has been purged from the system.

Since purge is the last processing segment for a job, it is at this point that an installation accounting program can be placed to summarize the accounting data for a job.

At the conclusion of purge, the JES3 installation accounting routine writes out a type 26 SMF record. This record contains times and dates of the processing on the processor writing the record. Since the clocks on the various processors are not synchronized, elapsed times should all be taken from the same types of records (for example, from type 26 records). Otherwise, timing differences in the records may result.

See *MVS/Extended Architecture System Programming Library: System Management Facilities* for a complete description of record type 26.

Chapter 3. Defining and Managing C/I Service

Converter/interpreter (C/I) service controls the conversion of JCL statements to internal text and then control blocks. The three principal phases of C/I service are:

- *Converter/interpreter phase:* Uses the MVS C/I routines to convert and interpret the JCL into scheduler control blocks. At this time, JES3 (through the MVS interpreter) creates the scheduler control blocks in the scheduler work area (SWA) in the JES3 address space on the global processor (hereafter called the JES3 global address space) or C/I functional subsystem address space(s).

- *Prescan phase:* Creates 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 creates the job summary table for use in JES3 device setup.

In the JES3 global address space, the CI DSP controls all three of the above phases. More than one copy of the CI DSP can be active at a time. You specify the number of CI DSPs that may be active in the JES3 global address using the STANDARDS initialization statement.

C/I service can also take place in a C/I functional subsystem (FSS) address space. A C/I FSS address space contains many functions similar to a JES3 address space and can operate on the global processor or any local processor. Where the address space operates depends on what you specify for the FSS on its FSSDEF initialization statement.

The CI DSPs that run in a C/I FSS address space process jobs only through the converter/interpreter and prescan phases of C/I service. You specify the maximum number of CI DSPs that may be active in a C/I FSS address space using the FSSDEF statement defining that address space.

After a job passes through the converter/interpreter and prescan phases in a C/I FSS address space, C/I service returns the job to the JES3 global address space for the postscan phase. There, the job's postscan processing takes place under a separate POSTSCAN DSP. You specify the number of POSTSCAN DSPs that may be active in the JES3 global address space using the STANDARDS initialization statement.

This chapter includes the following sections:

1. "Setting Up C/I Service" explains how to set up C/I services to make the best use of the address spaces available to JES3.

2. "Controlling Jobs through C/I Service" briefly describes the user exit routines and other means available to influence a job's C/I processing.

3. "Managing the Scheduler Work Area" explains how to avoid storage constraints in the scheduler work area (SWA).

4. "Monitoring and Modifying C/I Service" lists the operator commands that you can use to monitor and change the C/I setup after JES3 initialization.

5. "Managing Procedure Libraries" explains how to update the procedure libraries used by C/I service.

6. "Recovering from C/I FSS Address Space Failures" describes what actions to take if a C/I FSS address space fails. It also describes the actions taken by JES3 if a C/I FSS address space fails.

Setting Up C/I Service

JES3 interprets each job's JCL using a separate copy of the CI DSP. For jobs processed in C/I FSS address spaces or for rescheduled jobs, JCL interpretation occurs in a CI DSP and then in a POSTSCAN DSP. Thus, for every job in C/I service, there is one CI or POSTSCAN DSP allocated to the job. JES3 defines each CI and POSTSCAN DSP to handle either batch or demand select jobs. Demand select jobs are started tasks and TSO LOGON jobs.

To set up C/I service, you must first determine whether to use C/I FSS address spaces and, if so, where to place them. Then, decide how many CI DSPs to define for each address space in which C/I processing can take place. Finally, determine how many POSTSCAN DSPs should run in the JES3 global address space.

Advantages to Using C/I FSS Address Spaces

C/I service uses large quantities of the scheduler work area (SWA) and storage in subpool 0. Because JES3 support of 31-bit addressing relieves virtual storage constraint, an installation may choose to increase CSA and consequently reduce the size of the JES3 region. In this case, an installation running many CI DSPs in the JES3 global address space may encounter private virtual storage constraint. When virtual storage is severely constrained, one of three actions occurs:

- JES3 functions terminate with out-of-storage abend codes (such as 80A)

- C/I service fails jobs that would cause out-of-storage abends

- Address space JCL statement limit processing causes CI DSPs and jobs to wait

To relieve actual or foreseen private virtual storage constraint in the JES3 global address space, you can establish C/I FSS address spaces. For every CI DSP you define in the C/I FSS address space, you can reduce by one the number of CI DSPs in the JES3 global address space. This reduction means C/I service in the JES3 global address space uses less virtual storage. If private virtual storage becomes constrained in the C/I FSS address space and causes out-of-storage abends, JES3 global address space functions are not affected.

Another advantage to having C/I FSS address spaces is that you can define more CI DSPs in the JES3 complex than the JES3 global address space alone can handle. Thus, C/I service can process more jobs in the JES3 complex at a time.

Deciding How Many C/I FSS Address Spaces to Use and Where to Put Them

How many, if any, C/I FSS address spaces your installation needs depends how constrained virtual storage is in the JES3 global address space. An installation with no virtual storage constraint and no expected need for more CI DSPs needs no C/I FSS address spaces. An installation with severe virtual storage constraint and a heavy work load should set up C/I FSS address spaces. That installation should also decrease the number of CI DSPs in the JES3 global address space, perhaps eliminating them altogether. (Remember that postscan processing always takes place in the JES3 global address space, whether under a CI DSP or a POSTSCAN DSP.)

Where should a C/I FSS address space be placed? Placing one on a local processor requires CTC communication with the global processor. CTC communication causes more overhead than placing a C/I FSS address space on the global processor. However, placing a C/I FSS address space on the global processor uses processing capability that other JES3 functions that must run on the global processor could use. Placing C/I FSS address spaces on the global processor, then, reduces performance for jobs being processed in the JES3 global address space. Placing C/I FSS address spaces on a local processor reduces performance for jobs being processed by those C/I FSS address spaces.

If you establish C/I FSS address spaces on one or more local processors, you must ensure that they can access the same system procedure libraries. To ensure that all CI DSPs use identical procedure libraries, place the procedure libraries on DASD devices that are defined to MVS as shared among all processors eligible for C/I FSS address spaces. Also define the DASD devices as jointly-managed by JES3 and MVS (that is, defined to MVS as permanently resident).

Deciding how many C/I FSS address spaces to define for a particular processor requires additional calculation. Each C/I FSS address space requires some space in the common service area (CSA) and the system queue area (SQA) for control blocks. (See the following subsection for the space requirements.) The control blocks either reduce the amount of available CSA and SQA or require the installation to define more SQA, thus reducing the amount of private area virtual storage. Also, C/I FSS address spaces can degrade system performance because they are non-swappable.

Avoiding Common Storage Constraints

The JES3 support for 31-bit addressing provides storage constraint relief for the C/I FSS address spaces because much of the JES3 code and many control blocks reside above 16 megabytes. However, you may still want to examine potential areas of constraint.

Some data areas associated with the C/I FSS address spaces occupy common storage. Factor the amount of common storage occupied by these data areas (given below) into your calculations of how many C/I FSS address spaces to establish on each processor.

- IATYBAL (BALJ) - For every C/I FSS address space active in the processor, there is one BALJ data area in SQA. The BALJ data area occupies 64 bytes plus one byte per page of JSAM buffer space. For example, suppose you specify a 250-page JSAM buffer pool for FSS address spaces. If 2 C/I FSS address spaces are active in a given processor, that processor will have 2 BALJ data areas, each occupying 314 bytes of SQA.

- IATYDMC (DMC) - For every data buffer block (DAT) defined in the JSAM buffer pool of every active C/I FSS address space, a data management control block (DMC) occupies 128 bytes of CSA in subpool 231. The DMC data area cannot cross page boundaries, and each C/I FSS address space has its own DMC pool. For example, suppose you specify a JSAM buffer pool size of 250 pages for FSS address spaces and a buffer size of half a page. If 2 C/I FSS address spaces are active in a given processor, that processor will have 1000 DMC data areas occupying a total of 128k of CSA.

- IATYRAB (RAB) - For every job in the converter/interpreter or prescan phase of C/I service, a record allocation block (RAB) occupies 112 bytes of CSA.

- IATYDSS (DSS) - For every C/I subtask, there are 3 data set status blocks (DSSs) in CSA. Each DSS occupies 120 bytes.

MVS Performance Considerations

C/I FSS address spaces are not swappable because they use JSAM to perform I/O operations. Non-swappable address spaces occupy real storage at all times and therefore may adversely affect MVS performance. To avoid adverse effects on MVS performance, tune the number of C/I FSS address spaces after initialization using operator commands. For a list of those commands, see "Monitoring and Modifying C/I Service" in this chapter.

Defining a C/I FSS Address Space

If you decide that establishing C/I FSS address spaces will help your installation, you must supply a cataloged procedure for starting the C/I FSS address spaces. Then, define each C/I FSS address space you want to establish using the FSSDEF initialization statement. (You cannot define or add a C/I FSS address space using operator commands.) Follow the steps given below.

1. Include a procedure for starting a C/I FSS address space in the appropriate procedure library for your installation. The default C/I FSS start procedure resides in SYS1.PROCLIB in the member named JES3CI. The JCL statements included in this procedure are:

```
//JES3CI  EXEC  PGM=IATINTKF,DPRTY=(15,14)
//STEPLIB   DD  DSN=SYS1.JES3LIB,DISP=SHR
//CHKPNT    DD  DSN=SYS1.JES3CKPT,DISP=SHR
//CHKPNT2   DD  DSN=SYS1.JES3CKP2,DISP=SHR
//JES3OUT   DD  SYSOUT=A
//JES3SNAP  DD  SYSOUT=A
//SYSABEND  DD  SYSOUT=A
//JESABEND  DD  SYSOUT=A
```

   Using this default procedure as a guide, you can change the data set names and the SYSOUT classes to meet your installation's needs. Always specify the same checkpoint data set(s) as specified in the JES3 start procedure and share the data set(s).

   In order to ensure an adequate region size, you may want to add a REGION= keyword to the EXEC JCL statement. The JES3CI procedure provided by IBM does not include a REGION= keyword. Do not include JCL statements to define procedure libraries in the C/I FSS start procedure.

2. Include the appropriate FSSDEF statement(s) in your initialization stream, one for every C/I FSS address space you wish to create.

   - Specify TYPE=CI.
   - The PNAME parameter should give the name of the cataloged procedure that starts C/I FSS address spaces, which is in the procedure library used for started tasks.
   - Use the SYSTEM parameter to specify the JES3 processor on which you want to establish the C/I FSS address space.
   - Use the START parameter to specify whether you want JES3 to automatically start this C/I FSS address space when an eligible processor is available and both subparameters of the DSPCNT parameter (see below) do not equal 0.
   - Use the DSPCNT parameter to specify the maximum number of CI DSPs that may operate in the C/I FSS address space. Specify this number by giving the maximum number of DSPs for batch jobs and/or the maximum number of DSPs for demand select jobs that may operate concurrently. See "Specifying the Number of CI DSPs in a C/I FSS Address Space" for guidelines on setting values for this parameter.

- Use the MAXASST parameter to specify the maximum number of JCL statements all CI DSPs may process concurrently in this C/I FSS address space. See "Selecting the Address Space JCL Statement Limit" for guidelines on setting a value for this parameter.

Defining the Maximum Number of CI and POSTSCAN DSPs

You can specify the maximum number of CI DSPs that can run in the JES3 global address space using the STANDARDS initialization statement. You can also specify the maximum number of CI DSPs that can run in a C/I FSS address space using the appropriate FSSDEF initialization statement. To specify the maximum number of POSTSCAN DSPs (which can run only in the JES3 global address space), use the STANDARDS statement.

The combined maximum number of CI and POSTSCAN DSPs in all address spaces equals the maximum number of jobs that C/I service can process at any time. Increasing the number of CI DSPs in any address space allows C/I service to process more jobs concurrently in that address space and in the JES3 complex. However, a CI DSP in a C/I FSS address space that has finished processing a job is considered "in use" until a POSTSCAN DSP becomes available. The number of POSTSCAN DSPs, therefore, can limit the number of CI DSPs in a C/I FSS address space that are actively processing jobs.

Each CI DSP uses at least 13k of private area virtual storage, in addition to the private area virtual storage occupied by reentrant modules used by all CI DSPs. To calculate how many CI DSPs you should define for each address space, you must know two things:

- How much private area virtual storage is available
- How much of the private virtual storage is not being occupied by other necessary functions

The following subsections describe how to specify and choose the appropriate maximum number of CI and POSTSCAN DSPs for an address space.

Specifying the Number of CI DSPs in the JES3 Global Address Space

Use the STANDARDS initialization statement to specify the maximum number of copies of the CI DSP that can operate in the JES3 global address space at any time. The CICNT parameter defines the maximum number of CI DSPs that can process (1) batch and (2) demand select (started task and TSO LOGON) jobs. The combined maximum number of CI DSPs active in the JES3 global address space at any time cannot exceed 255.

If you specify CICNT=(0,0), no CI DSPs will run in the JES3 global address space. If you do not specify the CICNT parameter, JES3 defines 2 C/I DSPs assigned to batch jobs and 1 CI DSP assigned to demand select jobs.

To determine how many CI DSPs to run in the JES3 global address space, consider how much private area virtual storage is available in that address space. CI DSPs for demand select jobs use less SWA space than CI DSPs for batch jobs because demand select jobs normally have fewer JCL statements. If private area virtual storage is constrained, define demand select CI DSPs but no batch CI DSPs.

Note: If you define C/I FSS address spaces for your installation, JES3 creates a special C/I subtask in the JES3 global address space for starting C/I FSS address spaces; it cannot be used for starting other tasks or TSO LOGONs. This subtask allows a C/I FSS address space to be started even if you specify CICNT=(0,0) in the initialization stream. The DSP counts for the JES3 global address space do not include this subtask, but you should account for the subtask when determining how much private area virtual storage is available.

Specifying the Number of CI DSPs in a C/I FSS Address Space

Use the FSSDEF initialization statement that defines a C/I FSS address space to specify the maximum number of CI DSPs that can operate in that C/I FSS address space. The DSPCNT parameter defines the maximum number of CI DSPs that can process (1) batch and (2) demand select (started task and TSO LOGON) jobs.

No more than 255 CI DSPs may be active in a C/I FSS address space at one time. Just as too many CI DSPs in the JES3 global address space can cause private virtual storage constraint, too many CI DSPs within a given C/I FSS address space can cause private virtual storage constraint. However, if a C/I FSS address space fails because it has no more virtual storage available, the JES3 global address space is not affected.

Avoiding Private Area Virtual Storage Constraint Within a C/I FSS Address Space: To avoid private virtual storage constraint within a C/I FSS address space, you must know how much storage C/I service requires. You must also know how much common storage is defined on the processor where the C/I FSS address space executes.

The amount of storage required for each C/I FSS support item is listed below. From these numbers, you can determine how many CI DSPs may run concurrently within a C/I FSS address space without constraining private virtual storage.

- **Nucleus support services**, those service routines required to execute the C/I routines, require roughly 103K of storage in subpool 251. These routines also support the rest of the JES3 functions available in the C/I FSS address space.

- **Resident reentrant modules**, required when one or more CI DSPs are active in a C/I FSS address space, occupy approximately 30k of storage in subpool 0. The amount of storage varies depending on the size of the user exit routines.

- **Resident CI DSP private area control blocks**, required when one or more CI DSPs are active in a C/I FSS address space, occupy about 2K of storage. The exact amount of storage depends on the size of the procedure library tables and the HWS, CIPARM, and RESDSN tables.

- **DSP-specific data areas** occupy about 6k of storage for every CI DSP. Data areas used by the CI DSP on behalf of a job occupy about 50 bytes in subpool 0 and 384 bytes in subpool 236 or 237 for each DD statement.

- **C/I subtask data areas** occupy about 28k of storage in subpool 229 for every C/I subtask.

- **Control blocks for the C/I subtasks to access the JESJCL, JCLIN, and SYSMSG data sets** require the following storage space for each data set to be accessed:

  - IATYDSB (DSB), the data set block: 176 bytes in subpool 230

  - IATYDSS (DSS), the data set status block: 120 bytes in subpool 241

  - IEZDEB (DEB), the data extent block: 32 bytes in subpool 0

  - IATYDAT (DAT), the USAM spool data buffer block: 2 pages in subpool 229. To determine how many DATs fit in the 2 pages, divide 8192 bytes (2 pages) by the number of bytes in one buffer, as specified using the BUFSIZE parameter on the BUFFER statement.

  - IATYDMC (DMC), the data management control block: 128 bytes in subpool 229 for every DAT (see above)

- **The RESQUEUE cell pool** requires 320 bytes for each CI DSP defined on the FSSDEF statement using the DSPCNT parameter (or as modified by the *F,F command), rounded upwards to the nearest page. For example, if the CI DSP count has not been modified and is specified as DSPCNT=(1,2), then the number of required bytes would be 3 times 320, or 960. That number would be rounded upwards to 4096 bytes, for one page. The RESQUEUE cell pool is in subpool 6.

- **The preallocated FCT entries** require 496 bytes for each CI DSP defined on the FSSDEF statement using the DSPCNT parameter (or as modified by the *F,F command). This value is not rounded upward to the nearest page. The preallocated FCT entries are in subpool 0.

In sum, a C/I FSS address space contains at least 171k of private area virtual storage for modules and data areas, plus storage for the JSAM buffer pool and job-related data areas. The functional subsystem interface (FSI) modules and control blocks occupy about 4k of additional storage.

Specifying the Number of POSTSCAN DSPs

Use the STANDARDS initialization statement to specify the maximum number of POSTSCAN DSPs that can operate in the JES3 global address space at any time. A POSTSCAN DSP performs postscan processing for each job processed by a CI DSP in a C/I FSS address space and for each rescheduled job. (Rescheduled jobs include dependent job control (DJC) jobs for which predecessor jobs have not completed.) The PSTCNT parameter defines the maximum number of POSTSCAN DSPs that can process (1) batch and (2) demand select (started task and TSO LOGON) jobs.

To choose values for the PSTCNT parameter, add the maximum number of batch jobs that each C/I FSS address space can process. Then, separately add the maximum number of demand select jobs that each C/I FSS address space can process. If your installation uses dependent job control (DJC), increase the batch value to account for rescheduled DJC jobs. You can use these batch and demand select values as the initial PSTCNT values.

Then, consider the effect of the number of POSTSCAN DSPs on the number of catalog locate requests being processed and the number of catalog volumes set up for postscan processing. Each job and step catalog data set for a job causes a setup call for the volume, which may require an operator to mount a volume. A volume mount causes a significant delay in processing. Even if the volume does not need mounting, volume setup and allocation to the job or step catalog library increases processing time. If your installation experiences delays in processing because of jobs waiting for locate processing or catalog volume setup, reduce the number of POSTSCAN DSPs using the *MODIFY,X command.

Controlling Jobs through C/I Service

You can use user exit routines to ensure that jobs going through C/I service comply with installation standards or to approve exceptions. Another way to affect C/I service is to define one or more options lists for use by the MVS converter. The options lists can specify various job-related requirements and default values.

This section describes, in brief, the user exit routines and initialization statements available to you for controlling jobs through C/I service.

Controlling Job Flow with User Exits

The user exit routines available for controlling job flow through C/I service are summarized in Figure 3-1. Some of these exits are taken during input service, before C/I processing. The input service exits influence C/I service by allowing the user to change a job's JCL statement values. The C/I service user exits let you write routines to examine and change the results of C/I processing. One user exit (IATUX26) is taken at MVS execution time (after C/I processing) but lets the exit routine make changes to the results of C/I processing. User exits IATUX03 - IATUX09, IATUX11, IATUX26, IATUX28, and IATUX41 let you decide whether to continue processing a job or remove it from the system. For detailed information on the user exit routines, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

User Exit Routine	Purpose
IATUX02	Allows you to determine what action to take when JES3 cannot find a procedure name in the in-storage BLDL list (if any)
IATUX03	Allows you to access JES3 internal text created from JCL by the MVS converter
IATUX04	Allows you to access JOB statement information from scheduler control blocks
IATUX05	Allows you to access EXEC statement information from scheduler control blocks
IATUX06	Allows you to access DD statement and SMF job management record information from scheduler control blocks
IATUX07	Allows you to make a decision on an unsuccessful catalog LOCATE request
IATUX08	Allows you to determine whether a job can use job setup as specified, go through high watermark setup, or should fail
IATUX09	Allows you to access final job status and the job summary table (JST) and job volume table (JVT) at the end of interpreter service
IATUX10	Allows you to supply a message when the message number specified on the IATXIWT macro is not in a predefined table
IATUX11	Allows you to inhibit the writing of the LOCATE request/response into a job's JESMSG data set
IATUX26	Allows you to access MVS scheduler control blocks (during job execution) before they are moved to the scheduler work area (SWA)
IATUX28	Allows you to verify or change information specified on the JOB JCL statement (during input service)
IATUX33	Allows you to verify or change (during input service) EXEC JCL statements or JES3 control statements (except //*DATASET and //*ENDDATASET)
IATUX34	Allows you to verify or change (during input service) DD JCL statements (except DD * or DD DATA statements)
IATUX41	Allows you to cancel a job that contains more JCL statements than allowed by the job JCL statement limit or to override the job JCL statement limit
IATUX44	Allows you to change any JCL statements other than the JOB, EXEC, and DD statements (during input service)
IATUX46	Allows you to select the processors where the conversion and interpretation of a job's JCL can take place and specify whether the JES3 global address space can be used, if you have defined C/I FSS address spaces
IATUX49	Allows you to override JES3's selection of an address space for C/I processing of a job's JCL, if you have defined C/I FSS address spaces

Figure 3-1. User Exits for Monitoring JCL Interpretation

Assigning Jobs to the Appropriate Processor and Address Space for C/I Service

If you have defined C/I FSS address spaces, two user exits let you control where jobs get processed through C/I service. If you have defined C/I FSS address spaces on local processors, remember that C/I processing of jobs might take place on a local processor. Also, remember that in a complex having processors with different release levels of MVS or different JCL definitions, jobs may use JCL parameters or statements that require conversion and interpretation as well as execution on a particular system. You must ensure that JES3 routes those jobs to the proper system.

Before the job enters C/I processing, you can use user exit IATUX46 to examine and limit the processors eligible for selection for the job's C/I processing. You can also specify whether the JES3 global address space is eligible. You can use user exit IATUX49 to override JES3's choice of address space (and, thus, the processor) for the job's C/I processing. For detailed information on the user exit routines, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Within the limits imposed by user exits IATUX46 and IATUX49, JES3 schedules jobs to a CI DSP using a priority scheme that considers whether jobs are batch or demand select jobs.

JES3 schedules demand select jobs to demand select CI DSPs in the following priority:

1. in the JES3 global address space
2. in the FSS address space with the greatest number of available demand select CI DSPs on the global processor
3. in the FSS address space with the greatest number of available demand select CI DPSs on a local processor

JES3 attempts to schedule batch jobs to batch CI DSPs in the following priority:

1. in the FSS address space with the greatest number of available batch CI DSPs on the global processor
2. in the FSS address space with the greatest number of available batch CI DSPs on a local processor
3. in the JES3 global address space

Defining a Converter/Interpreter Options List

The MVS converter/interpreter runs under a C/I subtask during the converter/interpreter phase of C/I service. You can define one or more options lists for use by the MVS converter/interpreter. The options list provides installation defaults for certain JCL keywords. Each options list can specify:

- Whether a JOB statement must specify a programmer name or an account number, or whether the scheduler work area is located above 16-megabytes when a job executes
- The maximum execution time for a job step
- The default region size for a job step
- Whether MVS is to bypass label processing
- Whether the MVS C/I routines are to write the following information to the system message data set:
  - JOB statement
  - Input JCL (including in-stream procedures)
- Whether the MVS allocation routines are to write allocation/termination messages to the system message data set
- The message class default

To define and name an options list, code a CIPARM initialization statement. Use the PARMID= parameter on this statement to name the list. You must code a separate CIPARM statement for each options list.

Each time the operator calls a disk reader, a card reader, or a tape reader, the operator can select one of the options lists that you defined. Thereafter, each time the MVS C/I routines process a job read from that particular reader, the MVS C/I routines use the selected options list.

To call a reader and select an options list, the operator must issue one of the following commands:

- *CALL,CR,...PARMID=xx
- *CALL,DR,...PARMID=xx
- *CALL,TR,...PARMID=xx

The variable xx is the name of the options list the operator wishes to select. This must be the same name you used when you defined the options list.

To specify which options list the MVS C/I routines should use for internal reader jobs, started tasks, and TSO LOGON jobs, use the INTPMID, STCPMID, and TSOPMID parameters on the STANDARDS initialization statement.

Managing the Scheduler Work Area

The scheduler work area (SWA) of the JES3 global address space or the C/I FSS address space serves as storage for the scheduler control blocks derived from a job's JCL statements during the converter/interpreter phase. At any time, the SWA contains the scheduler control blocks for all jobs that CI DSPs are processing at that time, from the converter/interpreter phase through the prescan phase. When a CI DSP finishes processing a job, JES3 or the C/I FSS writes that job's scheduler control blocks to the spool. JES3 or the C/I FSS then frees the SWA space occupied by that job's scheduler control blocks.

If a CI DSP starts to process a job but there is not enough SWA space to store the job's scheduler control blocks, C/I service cancels the job, or JES3 or the C/I FSS address space processing the job may terminate abnormally.

There are two reasons why there might not be enough space in the SWA for a job's scheduler control blocks.

- The job itself contains so many JCL statements that the control blocks derived from the JCL will not fit in the SWA.
- JCL statements from other jobs that CI DSPs are processing concurrently have caused scheduler control blocks to temporarily fill most or all of the SWA. A single large job (that is, a job with many JCL statements) or many smaller jobs that CI DSPs are processing could cause this condition.

You can limit the amount of SWA space generally used by C/I service in three ways:

- Balance the C/I service work load among the JES3 global address space and C/I FSS address spaces so that no one address space requires more SWA space than it has available.
- Limit the number of JCL statements allowed for each job, thus preventing a large job from dominating the SWA.
- Limit the number of JCL statements that can be processed by the JES3 global address space or a C/I FSS address space at any time. This limit prevents the address space from running out of SWA space and abnormally terminating.

For further information about balancing the work load, see "Setting Up C/I Service."

Creating SWA Space

SWA space is created twice for each job. It is first created during C/I processing in the global address space or in the C/I FSS address space. This initially-created SWA space is always located above 16-megabytes.

Although you need not be concerned with storage constraint in extended storage, you can still specify a limit on the number of address space JCL statements that can be processed concurrently. The address space JCL statement limit indicates the number of JCL statements that all CI DSPs in a given address space may

process concurrently. See "Preventing Abnormal Termination of JES3 or a C/I FSS Address Space" later in this chapter for more information.

The second time SWA space is created is when the job executes. For job execution, SWA space is located in the private area of the job's address space above or below 16-megabytes. To control which jobs will have SWA space created in extended storage, use the CIPARM initialization statement.

Because jobs with a large number of JCL statements can cause storage constraint problems, you may want to specify a job JCL statement limit. The job JCL statement limit indicates the number of JCL statements a job may have and continue into MVS interpretation. See the following topic, "Preventing a Job from Dominating the SWA" for further information.

Preventing a Job from Dominating the SWA

To prevent a job with many JCL statements from dominating the SWA, use the job JCL statement limit. If a job contains more JCL statements than the job JCL statement limit allows and you have not provided an exit routine for user exit IATUX41, JES3 cancels the job. If you provide the exit routine, JES3 allows the exit routine to decide whether to cancel the job or to let the job continue.

Selecting the Job JCL Statement Limit

To select the job JCL statement limit, determine the number of JCL statements that are in the largest job you want to run at your installation. Specify that number on the MAXJOBST= parameter of the STANDARDS initialization statement. The next time you do a warm start or cold start using the initialization stream containing that STANDARDS statement, JES3 uses the job JCL statement limit you specified.

You can override the effects of this limit on a particular job by writing an exit routine for user exit IATUX41. This user exit lets you decide whether a job that exceeds the job JCL statement limit should continue processing. To find out how to write an exit routine for user exit IATUX41, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

To set or change the job JCL statement limit without restarting JES3, issue the *F,X,D=CI,MAXJOBST=nnn command. This command becomes effective for the next job to enter C/I processing and remains in effect across a hot start. To display the current value of the job JCL statement limit, issue the *I,X,D=CI command.

If the initialization stream does not specify a job JCL statement limit, JES3 uses the default value of 0. When the job JCL statement limit is 0, there is no limit on the number of allowable JCL statements. JES3 does not check to see if jobs have too many JCL statements to fit in the SWA.

Preventing Abnormal Termination of JES3 or a C/I FSS Address Space

To prevent abnormal termination of the JES3 global address space or a C/I FSS address space when there is not enough SWA space to store the scheduler control blocks derived from a job's JCL, specify an address space JCL statement limit. The address space JCL statement limit defines the maximum number of JCL statements that all CI DSPs in an address space can process concurrently.

For the JES3 global address space, specify the address space JCL statement limit using the MAXASST parameter on the STANDARDS initialization statement or the MAXSYSDD= operand on the *F,X,D=CI operator command. For a C/I FSS address space, specify the address space JCL statement limit using the MAXASST parameter on the FSSDEF initialization statement or using the MAST= operand on the *F,F operator command.[1]

After you set the address space JCL statement limit(s), JES3 allows a CI DSP to process a job only if both of the following are true:

1. The number of JCL statements for the job is less than the address space JCL statement limit

2. The sum of the number of JCL statements for the job plus the number of JCL statements all CI DSPs are processing at that time in the address space is less than the address space JCL statement limit

If the first condition is not true, JES3 cancels the job. If the second condition is not true, JES3 makes the job wait. In addition, all jobs that have not yet begun MVS converter processing wait, while jobs that are into or past MVS interpretation continue processing.

After JES3 writes a job's JCL (as scheduler control blocks) to the spool, JES3 frees the SWA space. The total number of JCL statements that all CI DSPs are processing in the address space decreases by the number of JCL statements in the job. When this total decreases to a point where the waiting jobs can be processed without exceeding the address space JCL statement limit, JES3 allows the jobs to continue processing.

---

[1] The address space JCL statement limit and the job JCL statement limit do not affect demand select (started task and TSO LOGON) jobs. This means that:

- A demand select job can contain more JCL statements than either JCL statement limit allows.
- When counting the total number of JCL statements that all CI DSPs are processing, JES3 does not count the JCL statements for demand select jobs.

However, demand select jobs generally use few JCL statements. Most installations will not encounter virtual storage constraint while converting or interpreting demand select jobs.

Selecting the Address Space JCL Statement Limit

Figure 3-2 describes a procedure for selecting and setting the address space JCL statement limit. The left column of the figure explains each step of the procedure; the right column shows an example of the procedure. The example shows how to use the procedure.

Procedure	Example
1. Determine how much space is available in private virtual storage for SWA space to store scheduler control blocks derived from the job's JCL statements.	Assume there are 60000 bytes available.
2. Divide the amount of available SWA space by 600. (600 is the average number of bytes of storage needed for one JCL statement.) (see Note 1) The quotient represents the maximum number of JCL statements whose scheduler control blocks can be stored in the SWA at one time. Ignore the remainder.	• 60000/600 = 100 with a remainder of 0 • 100 JCL statements can be stored in scheduler control blocks in the SWA at one time
3. To set (or change) the address space JCL statement limit using the initialization stream: • For the JES3 global address space, code the parameter MAXASST=nnn on the STANDARDS initialization statement. • For a C/I FSS address space, code the parameter MAXASST=nnn on the FSSDEF initialization statement. **Note:** Substitute the value you got in step 2 for nnn. Restart JES3 with a warm start.	STANDARDS,MAXASST=100,... FSSDEF,TYPE=CI,FSSNAME=fssname, MAXASST=100,...
4. To set (or change) the address space JCL statement limit without restarting JES3: • For the JES3 global address space, issue the command *F,X,D=CI,MAXASST=nnn. • For a C/I FSS address space, issue the command *F,F,FSS=fssname,MAST=nnn. **Note:** Substitute the value you got in step 2 for nnn.	*F,X,D=CI,MAXASST=100 *F,F,FSS=fssname,MAST=100
Note 1: Each DD statement generates an SIOT (192 Bytes), a JFCB (192 Bytes), and almost always a JFCBX (192 Bytes). This gives 576 total bytes. Add to that a 'share' of the STEP and JOB related control blocks and you get at least 600 bytes required per DD statement.	

Figure 3-2. Procedure for Selecting and Setting an Address Space JCL Statement Limit

If you change an address space JCL statement limit, the change remains in effect across a hot start.

If the initialization stream does not specify an address space JCL statement limit, JES3 uses the default value of 0. The default value of 0 means the number of JCL statements for the address space has no limit. In this case, JES3 does not

prevent a CI DSP from processing a job even though there is not enough SWA space for the job's scheduler control blocks. If there is not enough SWA space, the address space processing the job will abnormally terminate or the job will be canceled by the MVS interpreter.

To display the current value of the address space JCL statement limit in the JES3 global address space, issue the command *I,X,D=CI. To display the current value of the address space JCL statement limit for a C/I FSS address space, issue the command *I,F,FSS=fssname.

Monitoring and Modifying C/I Service

After you have set up C/I service for your installation, you may need information about jobs in C/I processing or about the CI DSPs themselves. You might use such information to judge how well C/I service meets the installation's needs and the needs of individual jobs. If any problems become evident, you might also need to change the C/I service setup. The sections below briefly list the operator commands that let you monitor and change C/I service.

For details about the particular commands and operands, see *MVS/Extended Architecture Operations: JES3 Commands.*

Keeping an Eye on C/I Service

The *INQUIRY command lets you display the status of jobs and the status of CI DSPs, as described below.

Finding Out The Status of Jobs

Use the *INQUIRY (*I) command to learn what phase of C/I processing a job or jobs are in and which C/I FSS address space (if any) is processing the job.

- For a specific job:

  I,J=jobname|jobno|jj

- For jobs of a specific priority:

  *I,P=prty,N=nnn|ALL

- For jobs in the job queue:

  *I,Q,J=jobno,N=nnn|ALL

Finding Out The Status of CI and POSTSCAN DSPs

Use the *INQUIRY command to display information about the status of CI and POSTSCAN DSPs.

- For jobs waiting to be scheduled for a CI DSP:

  *I,Q,D=CI

- For jobs currently being processed by CI DSPs:

  *I,A,D=CI

- For the status (held or released) of C/I scheduling:

  *I,X,D=CI

- For jobs waiting to be scheduled for a POSTSCAN DSP:

  *I,Q,D=POSTSCAN

- For jobs currently being processed by POSTSCAN DSPs:

  *I,A,D=POSTSCAN

- For the number of jobs currently being processed by or waiting for CI and POSTSCAN DSPs:

  *I,B

- For the complex-wide job JCL statement limit and the JES3 global address space JCL statement limit for CI DSPs:

  *I,X,D=CI

- For the maximum CI DSP count and use count for the JES3 global address space:

  *I,X,D=CI

- For the maximum DSP count and the address space JCL statement limit for a C/I FSS address space:

  *I,F,FSS=fssname

Modifying C/I Service

After you have set up C/I service, you can change the configuration without restarting JES3 by using operator commands.

- To stop (HOLD) or resume (RELEASE) scheduling jobs for C/I processing throughout the complex:

  *F,X,D=CI,HOLD|RELEASE

- To change the number of batch and demand select CI DSPs in the JES3 global address space:

  *F,X,D=CI,MC=(maxbatch,maxdemsel)

  (This command works even if you specified no CI DSPs on the CICNT parameter of the STANDARDS initialization statement.)

- To change the number of batch and demand select CI DSPs in the specified C/I FSS address space:

  *F,F,FSS=fssname,DSPC=(maxbatch,maxdemsel)

  Note: If you specify DSPC=(0,0) on the above command, the C/I FSS address space will not be started or terminates after all work completes.

- To change the maximum number of JCL statements that all the CI DSPs in the JES3 global address space may process concurrently:

  *F,X,D=CI,MAXASST=nn

- To change the maximum number of JCL statements that all the CI DSPs in the specified C/I FSS address space may process concurrently:

  *F,F,FSS=fssname,MAST=nn

- To change the maximum number of JCL statements a job may have before C/I processing fails the job:

  *F,X,D=CI,MAXJOBST=nn

- To change the maximum number of batch and demand select POSTSCAN DSPs that may operate concurrently in the JES3 global address space:

  *F,X,D=POSTSCAN,MC=(maxbatch,maxdemsel)

- To change whether JES3 automatically starts the C/I FSS address space or to restart a C/I FSS that has terminated:

  *F,F,FSS=fssname,ST=Y|N

  Note: This change of the C/I FSS address space's characteristics carries over a hot start but not a warm start. Also, JES3 starts the address space automatically only if the DSPCNT parameter value is not 0.

- To cancel a job in C/I service:

  *F,J=jobno,C|CP or *C,J=jobno

- To reduce the number of C/I FSS address spaces on a given processor, change the C/I FSS address space DSP counts to zero using the *MODIFY command, or cancel the C/I FSS address space using the MVS CANCEL operator command.

- To fail a C/I FSS, the C/I driver, or the CI DSP that is processing a job, use the *FAIL command.

You cannot add a C/I FSS address space using operator commands. You must add a FSSDEF statement defining the C/I FSS address space to the initialization stream and perform a warm start.

Managing Procedure Libraries

Procedure libraries are partitioned data sets used to hold pre-defined sets of JCL statements called procedures. You can specify those procedures by name in EXEC JCL statements. A procedure library can be one partitioned data set or several partitioned data sets concatenated together. A partitioned data set can be a member of more than one procedure library.

The user identifies a partitioned data set as a member of a procedure library using a ddname of IATPLBxx on either:

- a DD JCL statement in the JES3 start procedure (see "Initializing JES3")
- the DYNALLOC initialization statement

The last two characters of the ddname are called the procedure library id (procid).

The user must define a **standard, or default, procedure library** with the ddname IATPLBST. The IBM-supplied start procedure defines the standard procedure library as including the partitioned data set SYS1.PROCLIB. SYS1.PROCLIB contains IBM-supplied procedures, to which the user can add procedures. The user can also change the standard procedure library to a procedure library of the user's choosing.

When a job requires a procedure in a procedure library, the PROC parameter of the //*MAIN JES3 control statement can be used to specify the procid identifying the procedure library. If the PROC parameter is not specified, JES3 uses the default procid appropriate to the job type, as specified on the STANDARDS initialization statement (the INTPROC, STCPROC, or TSOPROC parameter). The default for batch jobs is the standard procedure library, IATPLBST. If the job does not use any procedures (even if the PROC parameter is specified), JES3 does not assign a default procedure library to the job.

If you have C/I FSS address spaces on local processors, they need access to all procedure libraries. To ensure that all CI DSPs use identical procedure libraries, place the procedure libraries on DASD devices shared among all processors eligible for C/I FSS address spaces. The DASD devices must also be jointly-managed by JES3 and MVS (that is, defined to MVS as permanently resident or reserved).

The procedure libraries must be cataloged in the MVS catalogs for all processors sharing access.

Note: Converter/Interpreter functional subsystems obtain unit and volume information for the procedure libraries from the MVS catalog. For these FSSs,

JES3 ignores unit and volume information that you specify on in the JES3 start-up procedure or on a DYNALLOC initialization statement.

Updating Procedure Libraries

If you want to update one or more procedure libraries, you must ensure the integrity of the procedure libraries by letting JES3 know which data set(s) your job will update. List the names of the data set(s) to be updated on the UPDATE parameter of the //*MAIN JES3 control statement for that job. Before the job enters setup processing, JES3 disables all procedure libraries containing those data sets. JES3 and the C/I FSS address spaces, if any, deallocate the procedure libraries so that C/I service cannot use them. However, another job can update those procedure libraries if the job updates different data sets in the library concatenation.

The procedure libraries remain inaccessible (disabled) to C/I service until the updating job(s) finish executing. Other jobs needing those procedure libraries for C/I service must wait. To avoid making other jobs wait for long periods, keep updating jobs brief, with simple setup requirements. Jobs using other procedure libraries are not affected.

When the updating job(s) finish executing, JES3 reallocates the procedure libraries to JES3 and the C/I FSS address spaces. JES3 enables the procedure libraries when no more data sets in the procedure libraries are being updated.

If a job updating a procedure library is placed in spool hold over a restart, the procedure library remains disabled until the job is released from hold and all the processing described above finishes.

If a job updating a procedure library moves any procedure library data set to another volume, the job must update the catalog entry for that data set on all processors having C/I FSS address spaces. Otherwise, C/I service cannot find the correct catalog entry to enable the procedure libraries after the job finishes executing. When the procedure libraries are enabled, JES3 updates or verifies other information about the updated procedure library, such as the BLDL list entries and block size data.

Warning: Make sure that at least one initiator for the appropriate job class has been started before allowing any jobs to enter the system that update the procedure libraries used by the initiators. Otherwise, a deadlock will occur: the procedure libraries are disabled, the job is waiting for an initiator, and the initiator is waiting for the procedure libraries to be enabled. If this situation arises, the updating job must be canceled and resubmitted.

To prevent new jobs from updating the procedure library, change the DISABLE DSP maximum use count to 0 or issue the *F,X,D=DISABLE,HOLD command. To resume updating, increase the DSP maximum use count or issue the *F,X,D=DISABLE,RELEASE command.

Displaying the Status of Procedure Library Data Sets

You can display the status of the procedure libraries using the *I,PROCLIB[,ID=procid] command. This command shows whether a procedure library concatenation is enabled or disabled and shows the individual data set names in the procedure library concatenation with the job number and job name of the job updating each data set (if any). If you specify the procedure library id, the command displays this information only for the specified procedure library.

Recovering from C/I FSS Address Space Failures

Failure of a C/I FSS address space does not cause any JES3 address space or other C/I FSS address spaces to fail. If a C/I FSS address space encounters an error and is able to recover, no other address spaces, including JES3, even become aware of the problem. For recoverable errors, the system operator sees messages from the C/I FSS failsoft routines. The messages are similar to those the operator sees if a JES3 address space encounters an error.

If a C/I FSS address space does fail, there are two ways the JES3 global address space becomes aware of the failure:

- The C/I FSS address space disconnects through the functional subsystem interface (FSI).
- If the C/I FSS address space fails without ending communication with JES3, JES3 becomes aware of the failure at job termination. (The C/I FSS runs as a demand select job.)

JES3 never automatically restarts a C/I FSS address space that terminates. (When a C/I FSS address space abnormally terminates, JES3 changes the START value, defined by the FSSDEF statement, to NO.) However, JES3 automatically reschedules all jobs that were active in the C/I FSS address space at the time of failure. Jobs restart at the beginning of C/I service.

To restart the address space, use the *F,F,FSS=fssname,ST=Y operator command.

If the JES3 global address space abnormally terminates, all C/I FSS address spaces continue operating until they run out of work. Then they are idle until the JES3 global address space restarts. If the FSSDEF statement for the C/I FSS address space specifies TERM=YES, an *RETURN or *DUMP command for the JES3 global address space terminates the C/I FSS address space.

If a C/I FSS address space terminates during an IPL of a processor, JES3 will restart the C/I FSS provided:

- the processor is connected and online
- the DSPCNT is not zero
- the START option is specified as YES

Chapter 4. Defining and Managing Spool Data Sets

Before initializing JES3, decide how many spool data sets you should allocate and then allocate them. During initialization, format any unformatted spool data sets. You can also define spool space allocation units suited to the type of work handled by your JES3 complex. Then, define how many of these allocation units are to be allocated (on each request) to jobs that use a particular processor, job class, or SYSOUT class.

Optionally, you can define spool partitions, which are logical groups of spool data sets. Further, you can specify the spool partitions JES3 is to use for each processor, for each job class, and for each SYSOUT class.

To adjust the spool configuration in your JES3 complex as the installation's work load and other needs change, you can:

- Balance the work load across spool partitions using operator commands
- Free spool space permanently or temporarily, as needed, using the JES3 spool maintenance facility and the dump job facility
- Add or delete spool data sets
- Keep track of I/O errors on a spool data set using operator commands and BADTRACK statements
- Temporarily remove or permanently replace a spool data set having permanent I/O errors

Defining Spool Data Sets

Defining spool data sets requires:

- Determining how many data sets you should allocate
- Determining where to allocate a spool data set
- Allocating spool data sets
- Formatting spool data sets

This section includes guidelines and procedures for these tasks.

Determining How Many Spool Data Sets You Should Allocate

Several factors determine the number and size of spool data sets that you should allocate:

- The amount of spool space that all jobs in the complex may need at any one time. Allocate enough spool space to handle peak usage.
- The number of processors in the complex competing for available control units and devices. The more processors competing for control units and devices, the more spool data sets needed for reasonable performance. A spool data set should not be busy more than 30 to 40 percent of the time or there will be too much contention for the data set.
- The type of work being carried out in the complex. Jobs that include large amounts of output handled by JES3 need more spool space than jobs with little output, such as TSO jobs.
- The number of spool partitions in the complex. Each active partition must include at least one spool data set. Spool partitions cannot share spool data sets.

You cannot allocate more than 1024 spool data sets.

After you initially allocate a number of spool data sets, you may need to adjust that number as the installation's work load changes. See "Managing Spool Space" to learn how to adjust the number of spool data sets.

Allocating Spool Data Sets

To allocate a spool data set, include a DD statement for the data set in the JES3 start procedure. To dynamically allocate the spool data set, omit the DD statement and include a DYNALLOC statement for the data set in the JES3 initialization stream. Dynamic allocation provides an easier method for changing your spool configuration than allocating the spool data sets through the JES3 start procedure with DD statements.

You can allocate a spool data set on a 3330, 3340, 3350, 3375, or 3380 device. The volume on which a spool data set is allocated must be accessible to the global processor and to all local processors. Each spool data set must be contained in a single extent. (A single extent is one adjoining group of tracks or cylinders.) You cannot allocate any secondary extents.

To avoid degradation of JES3 performance and possible lockouts, do not allocate more than one spool data set per volume. When a volume contains more than one spool data set, the average seek time to access the data increases. Similarly, do not allocate data sets to a volume that JES3 rarely accesses.

JES3 spool data sets are location-dependent and unmovable. Utility programs that defragment should not be allowed to move these data sets. Spool data sets can be marked unmovable by coding DSORG=PSU on the DCB parameter of the DD statement.

Formatting Spool Data Sets

Before JES3 can use a spool data set, you must format the spool data set. Two ways to do this are:

- Format it during JES3 initialization by including a FORMAT statement in the JES3 initialization stream.
- Format it by executing the utility program IEBDG as a batch job. This fills the extent with hexadecimal 'FF' data.

If you format a spool data set during JES3 initialization, JES3 can use the spool data set after initialization completes. If you use IEBDG to format a spool data set, you must then do a warm start or cold start so JES3 can use the data set.

Formatting During JES3 Initialization

To format a spool data set during JES3 initialization, include a FORMAT statement for the spool data set in the JES3 initialization stream. Then start JES3 using that initialization stream.

The type of start you use depends on why you are formatting the spool data set:

- If you have changed the BUFSIZE= parameter on the BUFFER statement, use a cold start (C). (In this case, you must format all spool data sets.)
- If you are replacing a spool data set, use a warm start to replace a spool data set (WR). If you also want an analysis of the jobs in the job queue, use a warm start with analysis to replace a spool data set (WAR).
- If you are adding a spool data set, use a warm start (W) or a warm start with analysis (WA).

For a discussion of each of these methods of starting JES3, see "Starting JES3" in Chapter 10, "JES3 Start-Up and Initialization."

After JES3 processes the initialization stream, replace the FORMAT statement with a TRACK statement. If the FORMAT statement contained the STT or STTL parameter, also code this parameter on the TRACK statement.

If you use a warm start and the initialization stream contains a FORMAT statement for a spool data set that is already formatted, JES3 issues a warning message. JES3 continues with initialization, however, and does not reformat the spool data set.

Formatting with IEBDG

You can use the utility program IEBDG to format a data set that you plan to use as a spool data set. The book *MVS/Extended Architecture Utilities* explains how to use IEBDG. Figure 4-1 shows a sample job that uses IEBDG to format a spool data set.

```
//JOB1       JOB   .....
//FORMAT     EXEC  PGM=IEBDG
//SPXTNT     DD    DSN=spool data set name,DISP=OLD,
//                 UNIT=SYSDA,VOL=SER=serial number,
//                 DCB=(RECFM=U,BLKSIZE=nnn)
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD  *
  DSD          OUTPUT=(SPXTNT)
  FD           NAME=SPOOL,FILL=X'FF',LENGTH=nnn
  CREATE       NAME=(SPOOL),QUANTITY=9999999
END
/*
```

Figure 4-1. Sample Job Using IEBDG to Format a Spool Data Set

The value of the variable nnn must equal the value of the BUFSIZE= parameter on the BUFFER initialization statement. The variable nnn appears on both the SPXTNT DD statement and on the FD utility program control statement.

The value of the QUANTITY= parameter on the CREATE statement determines how much of the spool data set IEBDG formats. To ensure that IEBDG formats the entire data set, specify QUANTITY=9999999.

If IEBDG successfully formats the entire spool data set, the formatting job ends with an abend code of D37. In addition, MVS issues message IEC031I. Ignore the corrective action specified in the message. The formatting job abnormally terminates because the value of the QUANTITY= parameter causes IEBDG to try to format more records than the data set can contain.

If the job ends without the D37 abend code and the message, IEBDG may not have formatted the entire spool data set. Do not use the spool data set. Instead, find the problem, fix it, and then rerun IEBDG.

After the spool data set has been formatted, include a TRACK statement for it in the initialization stream. To make the spool data set available to JES3, restart JES3.

The type of restart you use depends on why you are formatting the spool data set:

- If you have changed the BUFSIZE= parameter on the BUFFER statement, use a cold start (C). (In this case, you must format all spool data sets.)
- If you are replacing a spool data set, use a warm start to replace a spool data set (WR) or a warm start with analysis to replace a spool data set (WAR).
- If you are adding a spool data set, use a warm start (W) or a warm start with analysis (WA).

For a discussion of each of these methods of starting JES3, see "Starting JES3" in Chapter 10, "JES3 Start-Up and Initialization."

Reformatting a Spool Data Set

Once formatted, reformat a data set only when:

- the spool data set has been damaged
- you change the BUFSIZE= parameter on the BUFFER initialization statement (in this case, you must reformat all spool data sets)

To reformat a spool data set, use either of the procedures just described.

Using Spool Partitions

A spool partition is a logical grouping of spool data sets. You control five factors:

- the number of spool partitions used
- the number of spool data sets that are in each spool partition
- the work load distribution across spool partitions
- the type(s) of spool data to be included in each spool partition
- the size of a track group for each partition

These factors influence the reliability, availability, and serviceability (RAS) of spool data sets and the performance impact of accessing a spool data set.

The following discussion assumes that you allocate no more than one spool data set per volume (as previously recommended).

Advantages to Spool Partitioning

Most JES3 installations can benefit from using several spool partitions. If used properly, spool partitioning can provide many advantages. However, if these advantages do not apply to your installation, one spool partition will serve as well. The major advantages to using spool partitions are:

- If a spool data set fails, the failure affects only a subset of the jobs in the JES3 complex. That is, the failure affects only those jobs that have data in the spool partition including the failed spool data set, not jobs that have data in other spool partitions. (The failure may not affect all jobs in that spool partition, however. Some jobs may not have had any data on the failed data set.) Thus, spool partitioning improves spool RAS.
- By spreading the use of spool partitions across jobs, job classes, and SYSOUT classes, you can limit the number of processors that compete for each partition. If processor competition for spool data sets is an actual or potential problem for your installation, spool partitioning could improve system performance.

- By specifying track group size on a partition by partition basis, you can tailor spool space allocation to the requirements of jobs using that partition. Efficient use of spool space minimizes spool access time and can improve performance. (See "Determining the Size of a Track Group" later in this chapter.)

- By isolating the JES3 initialization data in its own spool partition, you can prevent the infrequently-accessed initialization data from occupying the track groups that have the best performance characteristics in the default partition or any other partition.

- By isolating critical work in specific spool partitions, you can ensure that spool space is available for critical jobs and users. At the same time, you can ensure that spool space requirements of noncritical applications do not interfere with spool space requirements of critical applications.

- By isolating certain types of work in specific partitions, you can better determine what action to take if a spool data set fails. Refer to the section below entitled "Isolating Different Spool Data Types" for details.

If none of the above advantages apply to your installation, you need not define any spool partitions. In that case, JES3 assumes that all spool data sets belong to a single spool partition.

If spool partitioning seems advantageous for your installation, consider carefully which data sets you assign to each partition. To get the best balance of high RAS and high performance, distribute the installation's work load evenly across the spool partitions. You can adjust the balance *after* your installation has set up the spool configuration for the first time. See "Managing Spool Space" for information on how to tune your spool configuration.

Isolating Different Spool Data Types

Spool partitioning allows you to isolate different types of spool data. Isolating spool data in separate partitions can help you improve spool performance, spool recovery procedures, and spool space management.

- To improve spool performance, group SYSOUT data sets with similar characteristics in one spool partition. For example, you could put spool data for TSO LOGON jobs in one partition and spool data for batch jobs in a different partition. Contention for processors, data sets, and other resources can thus be reduced.

- To improve spool recovery, keep critical spool data separate from noncritical data. (Your installation might consider spool data generated while running a payroll job to be critical, but data generated while compiling programs might be easily replaced and therefore noncritical.) If a spool data set fails, you will know whether you must try to recover the data or can simply replace the data set. You will also be better able to judge how quickly you must reinstate the data set.

- To improve spool space management, you can isolate data controlled by the system programmer or operator from data controlled by an end user. If the installation needs more spool space, you will know which data sets you control. Then you can release stored data in those data sets for printing or other final processing.

Defining Spool Partitions

Using SPART initialization statements, you can define up to 1024 spool partitions. Additionally, you can identify one of the partitions as the default partition by specifying DEF=YES on a SPART statement.

You need not define spool partitions or specify a default partition. If you define no partitions--you include no SPART statements in the initialization stream--JES3 defines one partition and names it JES3PART. If you define partitions but not a default partition, JES3 uses as the default partition the partition defined on the first SPART statement in the initialization stream.

The default spool partition always contains:

- JES3 spool access method (JSAM) single and multirecord files
- job input (SYSIN) data
- JES3 control blocks created by input service

It may also contain output spool data for:

- jobs requesting a spool partition with no free space that overflows into the default partition
- jobs requesting a spool partition that has been deleted
- jobs that do not request a spool partition and for which the job class and processor have no spool partition designation

JES3 selects the partition used for a job's data when the job is ready to execute. If the requested partition and its overflow partitions do not have enough space available, JES3 bypasses selection of the job until enough space becomes available.

Defining Spool Partition Overflow

To provide for occasions when a requested spool partition is full, you can specify where each spool partition's overflow data should go. To do this, use the OVRFL parameter on the SPART initialization statement:

- To specify that spool data directed to a particular spool partition should overflow into another named spool partition, specify OVRFL=name on the SPART statement that defines the requested spool partition.
- To specify that spool data directed to a spool partition should overflow into the default partition, specify OVRFL=YES. This specification is also the default.

- To specify that spool data directed to a spool partition should not overflow into another partition, specify OVRFL=NO. In this case, the requestor waits until track groups within the requested partition become available.

- **The default partition cannot overflow.** It always has the attribute of OVRFL=NO.

A spool partition that accepts overflow from another partition may, in turn, overflow into a third partition, and so on. This can continue until the overflowing spool data reaches a partition that allows no overflow. For example, consider the following SPART initialization statements:

```
SPART,NAME=TSODATA,OVRFL=SMLBATCH
SPART,NAME=SMLBATCH,OVRFL=YES
SPART,NAME=DEFPART,DEF=YES
SPART,NAME=BIGBATCH,OVRFL=NO
```

The TSODATA partition contains TSO output data, the SMLBATCH partition contains output data from small batch jobs, and the BIGBATCH partition contains output data from large batch jobs. If the TSODATA partition becomes full, the TSO output data overflows into the SMLBATCH partition. If the SMLBATCH partition becomes full, it overflows into the default partition, DEFPART. The default partition cannot overflow. Neither can the BIGBATCH partition, because its SPART statement specifies OVRFL=NO.

If an overflow partition begins running low on space, JES3 suspends job selection for jobs requesting that partition or for any partition that overflows into that partition. (The *marg* subparameter of the SPLIM parameter on the SPART initialization statement for that partition or on the BUFFER initialization statement defines when the partition is running low on space.) If an overflow partition reaches a critical space shortage, JES3 suspends all SYSOUT buffer processing for jobs requesting that partition or any partition that overflows into that partition. (The *min* subparameter of the SPLIM parameter on the SPART initialization statement for that partition or on the BUFFER initialization statement defines when the partition has a critical space shortage.)

CAUTION

1. **Avoid allowing too many partitions to overflow into the default partition.** If the default partition begins running low or critically short on space, JES3 takes the same actions as for any overflow partition that runs low or critically short on space. With a critical spool space shortage, JES3 also prevents any new work from starting because the default partition has no room for a job's input data (SYSIN).

2. **The initialization stream must not define a circular spool partition overflow.** Below is an example of SPART statements defining a circular overflow:

   ```
   SPART,NAME=TSODATA,OVRFL=SMLBATCH
   SPART,NAME=SMLBATCH,OVRFL=TNYBATCH
   SPART,NAME=TNYBATCH,OVRFL=TSODATA
   ```

   If the TSODATA partition overflows into the SMLBATCH partition, and the SMLBATCH partition overflows into the TNYBATCH partition, the TNYBATCH partition cannot overflow into the already-overflowing TSODATA partition. JES3 detects the circular overflow condition on the

SPART statement for the TNYBATCH partition and changes the overflow specification to OVRFL=NO. If this situation arises in your initialization stream, use the *INQUIRY and *MODIFY commands to respecify the spool partition overflow.

Specifying Spool Data Sets as Members of Spool Partitions

For each spool partition, you can specify the data sets that are to be members of that partition. You do this by specifying the name of the spool partition on the FORMAT or TRACK statement associated with each spool data set. A spool data set, however, can be a member of only one spool partition at any time.

If you do not name some spool data sets as members of a spool partition, JES3 makes them members of the default spool partition. If you have defined no spool partitions, JES3 makes all spool data sets members of the spool partition named JES3PART. JES3 defines and names the JES3PART spool partition.

Except for the default spool partition, it is not necessary for every spool partition to have spool data sets. For the default spool partition, you must specify at least one spool data set as a member of that partition, or you must let JES3 make a spool data set a member by default. During JES3 initialization, if the default spool partition has no spool data sets, JES3 terminates with an abend code of DM012. Any spool partition without a spool data set is called a "dummy partition."

You can also specify which spool partition you want JES3 to use when it writes the initialization data that it needs to perform hot starts and local starts. Isolating the initialization data in this way causes JES3 to allocate the most accessible track groups to that data. To define a spool partition for initialization data, specify INIT=YES on the SPART statement that defines the spool partition you want to use.

For better performance, keep spool data sets within a partition about the same size.

Specifying a Spool Partition for Spool Data

You can specify which spool partition JES3 is to use for allocating spool space for:

- Output from jobs that execute on a specific processor
- Output from jobs that belong to a specific job class
- Specific SYSOUT classes

To do this, use the SPART parameter on the MAINPROC, CLASS, or SYSOUT initialization statements:

- To specify a spool partition for output from jobs that execute on a specific processor, specify the partition's name on the SPART parameter of the MAINPROC statement that defines the processor.

- To specify a spool partition for output from jobs that belong to a specific job class, specify the partition's name on the SPART parameter of the CLASS statement that defines the job class.

- To specify a spool partition for a specific SYSOUT class, specify the partition's name on the SPART parameter of the SYSOUT initialization statement that defines the SYSOUT class.

Each SYSOUT data set in the SYSOUT class for which you specify a specific spool partition must have its own track allocation table (TAT). To give each SYSOUT data set in a SYSOUT class its own TAT, specify TYPE=DSISO on the SYSOUT initialization statement that defines the SYSOUT class.

You can specify the name of a "dummy partition" on the SPART parameter of the MAINPROC, CLASS, or SYSOUT initialization statement. However, JES3 will not allocate any space for jobs requesting a "dummy partition" (unless the SPART statement defining the "dummy partition" specified an overflow partition to which spool data sets are assigned). The operator receives a message that the requested partition is full and does not overflow. Then the job waits until space becomes available in the partition.

Determining the Order of Spool Partition Overrides

Each time a job is ready to execute on a processor, JES3 decides which spool partition to use for the job's spool output. Consider this situation, for example. You specify that JES3 is to use partition A for jobs that execute on processor PROC1 and partition B for jobs in job class CL1. If a user submits a job assigned to job class CL1 and that job executes on processor PROC1, JES3 must decide whether to use spool partition A or B.

In making this type of decision, JES3 uses the following order of initialization statement overrides for job output:

- SYSOUT overrides CLASS and MAINPROC
- CLASS overrides MAINPROC

JES3 always writes a job's input data (SYSIN) to the default spool partition.

Figure 4-2 shows how the hierarchy of overrides works. It shows to which spool partition JES3 writes a job's spool data for different combinations of the SPART parameter on the MAINPROC, CLASS, and SYSOUT statements. To use this figure, make the following assumptions about the job:

- The job has data in two or more SYSOUT classes

- When a MAINPROC statement specifies a spool partition, the job executes on the processor defined by that statement

- When a CLASS statement specifies a spool partition, the job belongs to the class defined by that statement

If you specify a spool partition for:			
Data in a SYSOUT class (SYSOUT statement)	**Jobs on a processor (MAINPROC statement)**	**Jobs in a job class (CLASS statement)**	**JES3 writes a job's spool data to partitions as follows:**
X	-	-	• Data that belongs to the SYSOUT class for which a partition is defined is written to that partition. • All other data is written to the default partition.
-	X	-	• Input data (SYSIN) is written to the default partition. • All other data is written to the partition specified on the MAINPROC statement.
- -	- X	X X	• Input data (SYSIN) is written to the default partition. All other data is written to the partition specified on the CLASS statement.
X	X	-	• Input data (SYSIN) is written to the default partition. • Data that belongs to the SYSOUT class for which a partition is defined is written to that partition. • All other data is written to the partition specified on the MAINPROC statement.
X X	- X	X X	• Input data (SYSIN) is written to the default partition. • Data that belongs to the SYSOUT class for which a partition is defined is written to that partition. • All other data is written to the partition specified on the CLASS statement.

Figure 4-2. Spool Partition Overrides

How the User Can Request a Spool Partition

When submitting a job, the user can request that JES3 write the job's spool data to a specific spool partition. To do this, the user specifies the name of the spool partition on a //*MAIN JES3 control statement. This allows the user to override partition names specified on MAINPROC or CLASS statements. The user, however, cannot override partition names specified on SYSOUT statements. Instructions for coding the //*MAIN statement are in *MVS/Extended Architecture JCL User's Guide*.

The installation can override the spool partition specified by the user by coding user exit routine IATUX29 or IATUX33.

Example of Spool Partitioning

The following example illustrates most of the spool partitioning concepts discussed to this point. The JES3 initialization statements in the example:

- Define five partitions, PARTA, PARTB, PARTC, PARTD, and PARTE
- Define a default partition, PARTA
- Assign the spool data for jobs in job class IMSBATCH to partition PARTB
- Assign the spool data for jobs that will execute on a specific processor (SY2) to partition PARTC
- Assign data in SYSOUT class S to partition PARTD

The //*MAIN statement in the third job shows how the user can request a specific spool partition for output data.

```
* JES3 Initialization Statements

SPART,NAME=PARTA,DEF=YES
SPART,NAME=PARTB
SPART,NAME=PARTC
SPART,NAME=PARTD
SPART,NAME=PARTE
ENDJSAM
CLASS,NAME=BATCH,DEF=YES
CLASS,NAME=IMSBATCH,SPART=PARTB
*
MAINPROC,NAME=SY1,.......
MAINPROC,NAME=SY2,SPART=PARTC
*
SYSOUT,CLASS=N,....
SYSOUT,CLASS=S,SPART=PARTD,TYPE=(PRINT,DSISO)

*   JOBS

//JOB1    JOB  ......
//*MAIN   ......
//STEPA   EXEC   .....
//OUT1    DD     SYSOUT=N
//OUT2    DD     SYSOUT=S
//
//JOB2    JOB  ......
//*MAIN   CLASS=IMSBATCH,.....
//STEPA   EXEC   .....
//OUT1    DD     SYSOUT=N
//OUT2    DD     SYSOUT=S
//
//JOB3    JOB  ......
//*MAIN   CLASS=IMSBATCH,SPART=PARTE
//STEPA   EXEC   .....
//OUT1    DD     SYSOUT=N
//OUT2    DD     SYSOUT=S
//
```

Figure 4-3 refers to the previous example. Part 1 shows to which spool partition JES3 writes the jobs' spool data when the jobs execute on processor SY1. Part 2 shows the same thing when the jobs execute on processor SY2.

The text following Part 1 and Part 2 of the figure explains why JES3 writes the spool data to certain partitions.

	JES3 writes the job's spool data to this partition:		
When this job runs on processor SY1	input (SYSIN) (see note 1)	SYSOUT class N	SYSOUT class S (see note 2)
Job 1	PARTA	PARTA	PARTD
Job 2	PARTA	PARTB	PARTD
Job 3	PARTA	PARTE	PARTD

Figure 4-3 (Part 1 of 2). Spool Partitions Used in Spool Partition Example

Notes:

1. *JES3 writes input (SYSIN) for all jobs to the default spool partition, PARTA, because JES3 always writes input data to the default partition.*

2. *The SYSOUT statement that defines SYSOUT class S also specifies spool partition PARTD. Therefore, JES3 writes all data in SYSOUT class S to spool partition PARTD.*

JOB1:

JES3 writes the data in SYSOUT class N to the default partition because:

- The SYSOUT statement that defines class N specifies no spool partition.
- The MAINPROC statement that defines processor SY1 specifies no spool partition.

JOB2:

JES3 writes the data in SYSOUT class N to spool partition PARTB because the CLASS statement that defines job class IMSBATCH specifies spool partition PARTB. (The //*MAIN statement assigns this job to job class IMSBATCH.)

JOB3:

JES3 writes the data in SYSOUT class N to spool partition PARTE because the //*MAIN statement specifies partition PARTE. (A spool partition specified on a //*MAIN statement overrides a spool partition specified on a CLASS statement.)

	JES3 writes the job's spool data to this partition:		
When this job runs on processor SY2	input (SYSIN) (see note 1)	SYSOUT class N	SYSOUT class S (see note 2)
Job 1	PARTA	PARTC	PARTD
Job 2	PARTA	PARTB	PARTD
Job 3	PARTA	PARTE	PARTD

Figure 4-3 (Part 2 of 2). Spool Partitions Used in Spool Partition Example

Notes:

1. *JES3 writes input (SYSIN) for all jobs to the default spool partition, PARTA, because JES3 always writes input data to the default partition.*

2. *The SYSOUT statement that defines SYSOUT class S also specifies spool partition PARTD. Therefore, JES3 writes all data in SYSOUT class S to spool partition PARTD.*

JOB1:

JES3 writes the data in SYSOUT class N to spool partition PARTC because the MAINPROC statement that defines processor SY2 specifies PARTC.

JOB2:

JES3 writes the data in SYSOUT class N to spool partition PARTB because the CLASS statement for job class IMSBATCH specifies PARTB. (The user assigned the job to job class IMSBATCH.)

JOB3:

JES3 writes the data in SYSOUT class N to spool partition PARTE because the //*MAIN statement specifies PARTE. (A spool partition specified on a //*MAIN statement overrides a spool partition specified on a CLASS statement.)

Defining Spool Space Allocation Units

The basic unit of spool space allocation is called a **track group**. A track group is a group of spool records, with the size of each spool record equal to the size of a JES3 buffer. JES3 buffer size is defined by the BUFSIZE parameter on the BUFFER initialization statement (see "Determining the Size of the JES3 Buffers"). You can define the number of spool records in a track group using the GRPSZ parameter on the BUFFER statement or on the SPART statement.

JES3 allocates one or more track groups to a job when the job needs spool space. The number of track groups allocated on each request depends on the number defined using the TRKGRPS parameter for the job's SYSOUT class, //*MAIN JES3 control statement, job class, or assigned processor.

Defining a Track Group

A track group is the number of records that JES3 treats as a unit when allocating spool space. You can specify the size of a track group using the GRPSZ parameter, choosing a value from 1 to 999. The GRPSZ parameter can be used on two initialization statements:

- the BUFFER statement, if you want to define a "default" track group size for spool partitions that do not have an explicit track group size specification, or if you do not define any spool partitions
- the SPART statement, if you want to override the BUFFER statement and tailor the track group size to the type of data in the spool partition being defined

Determining the Size of a Track Group

The GRPSZ value on the BUFFER statement should correspond to the spool space requirements of a typical job in the JES3 complex. For example:

- If the workload consists mainly of small jobs, specify a GRPSZ value of less than half the number of records per cylinder for the spool device type. The small value uses spool space efficiently and reduces access time on moveable head devices.

- If the workload consists mainly of jobs producing much output, specify a GRPSZ value of roughly the number of records per cylinder. The large value reduces the number of requests for additional spool space and thus reduces the amount of time the job must wait to be allocated additional space. (The number of allocation requests also depends on the track group allocation size defined by the TRKGRPS parameter. Track group allocation sizes are discussed later in this chapter.)

If you have defined spool partitions to isolate different types of data, specify a "tailored" GRPSZ value on the appropriate SPART statement. Tailoring the GRPSZ value this way helps especially when part(s) of the installation's workload requires much less or much greater spool space than average.

Example: Suppose an installation processes a significant number of small files, such as TSO interactive data transmission files. However, the installation's average job requires much larger spool space allocations. To satisfy the space requirements of an average job, the system programmer specifies a GRPSZ value on the BUFFER statement of moderate size; say, 30 records of 4k each. Then, the programmer defines a spool partition to hold the small files. On the SPART statement defining that partition, the programmer specifies a GRPSZ value equal to the average size of a small file; say, 3 records of 4k each. Setting up a separate spool partition for small files and defining a track group size for that partition uses spool space efficiently, thus improving performance when accessing that data.

Suppose the installation also runs some critical jobs that produce large quantities of output. The system programmer wants to improve the chance that these jobs will continue processing during a failure of the JES3 global address space or during a dynamic system interchange. By creating a spool partition with large track groups (say, 300 records of 4k each), these jobs might not need secondary allocations of spool space. Then the jobs could run to completion without needing to wait for the JES3 global address space to become available.

CAUTION

A spool data set keeps its original track group size even if you move it to a spool partition with a different track group size (unless the data set is reformatted). Try not to move a spool data set to a partition having a different track group size. Having spool data sets with different track group sizes within one partition could result in performance problems.

To change the track group size of a data set, replace the data set. See *MVS/Extended Architecture Operations: JES3 Commands* for instructions on how to replace a data set.

Relating Track Group Sizes to Physical Tracks: Another consideration for determining track group size is the physical track size of the device type on which the spool data sets reside. JES3 rounds the number of records specified by the GRPSZ parameter to the next track boundary. Suppose the initialization stream includes the following BUFFER statement:

```
BUFFER,GRPSZ=192
```

For a spool data set residing on a 3380 device with a record size of 4k, the number of records per track is 10. JES3 rounds the number of records to the next track boundary, resulting in a track group size of 200 records. If you want your GRPSZ parameter specification to exactly match the number of records JES3 allocates to a track group, specify a GRPSZ value that is a multiple of the number of records per track on the device type. (See Figure 4-4.)

If your installation uses more than one device type for spool data sets, you can take one of two approaches to defining track group size:

1. Mix different device types within spool partitions so that one GRPSZ value applies to all the devices. Specify one GRPSZ value on the BUFFER statement. As a result, JES3 rounds the number of records per track group as appropriate for each device type. The track group size will be *approximately* the same for all the devices.

2. Separate different device types into different spool partitions. Then specify a GRPSZ value on the SPART statement for each partition that is a multiple of the number of records per track for the device type. In this case, JES3 does not need to round the number of records per track group.

The approach you choose depends on whether you want your installation to have a single GRPSZ value for all spool data sets (the first approach) or to use spool space as efficiently as possible (the second approach).

In the following chart, "Record Size" is approximated; 2k refers to the minimum allowable BUFSIZE value of 1952 bytes.

Device	Record Size	Records/ Track	Records/ Cylinder
3380	4k 2k	10 19	150 285
3375	4k 2k	8 14	96 168
3350	4k 2k	4 9	120 270
3340-70	4k 2k	2 4	24 48
3340-35	4k 2k	2 4	24 48
3330-11	4k 2k	3 6	57 114
3330-1	4k 2k	3 6	57 114

Figure 4-4. Record, Track, and Cylinder Characteristics for DASD Devices

Determining Track Group Allocation Sizes

When a job needs spool space for the first time, JES3 allocates one or more track groups to the job. You can specify how many track groups JES3 should allocate by using the TRKGRPS parameter on the MAINPROC, CLASS, and SYSOUT initialization statements. The person submitting a job can specify the track group allocation size on the //*MAIN JES3 control statement.

When a job uses up its first, or "primary," allocation of spool space, JES3 allocates more track groups. You can also specify, using the TRKGRPS parameter, how many units JES3 may allocate for each additional, or "secondary," allocation. JES3 continues allocating secondary quantities of spool space until the job needs no more space.

To determine the track group allocation size, then, you determine two things: how much spool space JES3 should allocate to a job initially, and how much JES3 should allocate thereafter.

1. **Primary TRKGRPS allocation**. Use the following general equation to determine the value for your primary TRKGRPS allocation:

$$\frac{\text{spool space required for average job (in bytes)}}{\text{BUFSIZE}} \times \frac{1}{\text{GRPSZ value}} = \text{primary TRKGRPS value}$$

   You can determine how much spool space an average job in your installation requires by using a resource monitoring program, such as the JES3 Monitoring Facility or the system management facilities (SMF).

   Tailor this equation to the particular initialization statement on which you specify the TRKGRPS parameter. For example, if you specify the TRKGRPS parameter on the CLASS statement, use the value for the spool space required for an average job in that job class. If the installation uses spool partitions on a job class basis, use the GRPSZ value from the appropriate SPART statement, not from the BUFFER statement.

   Note that the primary TRKGRPS allocation value cannot exceed 9.

2. **Secondary TRKGRPS allocation**. The number specified for secondary allocations depends on whether jobs that are larger than average are much larger or only somewhat larger. If the jobs are much larger, you need to specify a larger secondary allocation than if the jobs are only slightly larger. Set the value to keep as low as possible both the number of times JES3 allocates spool space and the amount of unused spool space.

   Note that the secondary TRKGRPS allocation value also cannot exceed 9.

Notice that you have greater control over spool space allocation when you use spool partitioning. You can vary track group size, and thus primary and secondary allocation sizes, partition by partition. You can also use spool space more efficiently by assigning large jobs to one spool partition and small jobs to another partition and selecting their GRPSZ and TRKGRPS parameters accordingly.

See the section entitled "Using Spool Partitions" earlier in this chapter for information about defining spool partitions.

Track Group Allocation Overrides

As noted above, you can specify the TRKGRPS parameter on the MAINPROC, CLASS, and SYSOUT initialization statements and on the //*MAIN JES3 control statement. The order of overrides for job spool space allocation, beginning with the statement that overrides the others, is:

1. SYSOUT initialization statement (SYSOUT class basis)
2. //*MAIN JES3 control statement (for a specific job)
3. CLASS initialization statement (job class basis)
4. MAINPROC initialization statement (for a specific processor)

Note: User exit routine IATUX33 can override the specification on the //*MAIN JES3 control statement.

If you do not specify the TRKGRPS parameter on any of the above statements, JES3 uses default values of 1 for primary allocation and 2 for secondary allocation.

Managing Spool Space

JES3 provides several facilities for managing spool space. These facilities help you balance the workload across spool partitions and gain additional spool space when available space runs low.

If you need to allocate more spool data sets, or if you have allocated too many spool data sets, you can add or delete a spool data set over a warm start. You can also use operator commands to determine the amount of spool space remaining in particular spool data sets or partitions and to redistribute the workload.

To gain additional spool space without adding data sets, the JES3 spool maintenance facility (JSM) lets you delete held SYSOUT data sets that are no longer needed. If you need more spool space but also need to save the jobs or SYSOUT on spool, the dump job (DJ) facility lets you copy spool data to tape. Then you can use the dump job facility to restore the data when space becomes free.

Another way to gain additional spool space without adding data sets is to replace existing data sets with larger ones. Replacing a spool data set requires a warm start. "Replacing a Spool Data Set" later in this chapter gives the steps to follow.

Adding or Deleting a Spool Data Set

You can increase or decrease your installation's spool capacity without performing a cold start by adding or deleting a spool data set. To know whether your installation's spool capacity is appropriate for your installation, you can monitor spool usage using the *I,Q,S operator command. To monitor the use of channel paths, control units, and spool data sets, use a system monitoring facility such as system management facilities (SMF), resource measurement facility (RMF), or JES3 monitoring facility (JMF).

To add a spool data set over a warm start, follow the guidelines given in "Allocating Spool Data Sets" and "Formatting Spool Data Sets" earlier in this chapter. To delete a spool data set over a warm start, remove the appropriate DD statement from the JES3 cataloged start procedure or DYNALLOC statement from the initialization stream. Also remove the TRACK or FORMAT statement for the spool data set. *MVS/Extended Architecture Operations: JES3 Commands* explains the operator activities required to add or delete a spool data set during a warm start.

If you delete a spool data set, JES3 cancels all jobs in the system that have spool data or allocation tables on the affected data set. Try not to delete a data set that contains important information (for example, the single track table (STT) or the JESNEWS data set). If this information is lost, the system issues messages giving you the opportunity to take appropriate actions.

Balancing the Work Load Across Spool Partitions

To help you determine the work load distribution across the spool partitions, JES3 provides commands that you can use to:

- Determine the amount of space remaining in each spool data set (*I,Q,DD = ddname)
- Determine the amount of space remaining in each spool partition (*I,Q,SP = spart)
- Determine where spool data overflows to when each spool partition becomes full (*I,Q,SP = spart,O)
- Determine if any spool partitions are overflowing, in a minimum or marginal spool space condition, or out of space (*I,R,JSAM)

Using the SPLIM parameter on the SPART and BUFFER initialization statements, you can have JES3 notify you when space in a spool partition begins running low and when it runs critically low.

```
BUFFER,SPLIM = (min,marg)
SPART,NAME = partitionname,SPLIM = (min,marg)
```

The *min* (minimal) and *marg* (marginal) subparameters define percents of the total number of track groups in the partition that remain available. The value specified on a SPART statement overrides the value specified on the BUFFER statement.

A good value for the *marg* subparameter is the point at which spool performance begins falling because of increased seek time. Seek time increases as JES3 writes and reads data farther from the middle of the volume. A good value for the *min* subparameter is the point at which spool performance becomes severely degraded. Another good value is the point at which active jobs will likely need to overflow into another partition.

JES3 issues messages indicating when a spool partition reaches a marginal or minimal condition.

If you want to redistribute the work load, there are commands that let you:

- Respecify a spool data set as a member of another spool partition:

  *F,Q,DD=ddn,SP=spart...

- Respecify the spool partition that JES3 is to use for specific processors:

  *F,G,main...

- Respecify the spool partition that JES3 is to use for specific job classes:

  *F,C=cls...

- Respecify the spool partition that JES3 is to use for the overflow of spool data when a spool partition becomes full:

  *F,Q,SP=spart,O=spart

  If this command would result in circular overflow, JES3 prevents the modification and issues a message rejecting the command.

During the next warm start, the initialization statements will override changes made using these commands. To retain the changes, make the same changes to the initialization stream. During a hot start, changes made using these commands remain in effect, except the *F,G and *F,C=cls commands.

Over a warm or cold start, you can make additional changes to the spool configuration by changing the initialization stream. You may add or delete partitions, change the use of any partition, or move spool data sets from one partition to another. (Try not to move a spool data set to a partition having a different track group size. Having spool data sets with different track group sizes within one partition could result in performance problems.) Any spool partition created without any spool data sets (a "dummy partition") should overflow. Otherwise, if a job requests the dummy partition, JES3 will never select the job for execution.

Deleting Held Output Data Sets Using JSM

You can delete output data sets from the output service hold queue by using an operator command that calls the JES3 spool maintenance (JSM) facility. The operator command, *X JSM, allows you to specify the criteria by which JSM selects data sets for deletion. Selection criteria include the data sets' SYSOUT class and age and the TSO userid of the data sets' creator. For details on using the *X JSM command, see *MVS/Extended Architecture Operations: JES3 Commands.*

The installation can override the operator command parameters that select the data sets to be deleted by coding user exit routine IATUX47. The JSM DSP calls IATUX47 after JSM selects the data sets. For information on IATUX47, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Freeing Spool Space Using the Dump Job Facility

You can free spool space without losing jobs already in the system by using the dump job (DJ) facility. Use this facility to copy one or more jobs to tape, then let JES3 purge the job. When spool space becomes available, use the dump job facility to restore the job(s) to the system. Call the dump job facility as a DSP, using operator commands that are explained in *MVS/Extended Architecture Operations: JES3 Commands.*

Recovering From Spool I/O Errors

It is possible to recover from many kinds of spool I/O errors without the need to perform a cold start. This includes, for example, errors caused by defective tracks on a spool volume or errors caused by a failing I/O device or control unit. The type of error indications you receive from JES3 can help you to determine the corrective action to take.

When an I/O error occurs on a spool data set, JES3 adds an entry to the BADTRACK table. Entries in the BADTRACK table prevent JES3 from allocating the track group containing the track with the I/O error. JES3 does not, however, create BADTRACK table entries for the following types of I/O errors:

- Read errors
- I/O error retry failures
- Write errors that can be attributed to some cause other than failure of the spool device (for example, channel errors and machine checks)
- temporary I/O errors

If the error caused JES3 to create a BADTRACK entry, you use the *INQUIRY,Q,BT operator command to display:

- the location of the track having the error

- the exact time JES3 found the error, if JES3 found the error while performing I/O to the track

- whether the BADTRACK entry for the track having the error was added to the BADTRACK table during formatting of the spool data set or by a BADTRACK statement during initialization

BADTRACK table entries that JES3 adds dynamically are lost during a warm start. To avoid further allocation of these tracks, system operators must inform you of I/O errors. Operators should also save the track address information given in the message stating that JES3 has added an entry to the BADTRACK table. Then, before performing a warm or cold start, update the initialization stream with BADTRACK statements as appropriate.

Intermittent I/O Errors

If you receive one or a few I/O error messages and DM711 or DM725 abend codes and JES3 continues to execute, the error is probably intermittent. Such an error might be caused by a defective track on a spool volume.

If a defective track caused the error, JES3 dynamically adds an entry to the BADTRACK table identifying the defective track. JES3 also issues a message indicating the ddname, cylinder, and track described by the new entry. As stated above, the system operator should save the information in the message. (The operator can also get this information at a later time using the *I,Q,BT command, if JES3 is active.) Then you must update the initialization stream with a BADTRACK statement for each I/O error before the next warm or cold start. The operator should also keep track of the frequency of errors. Frequent I/O errors suggest that the spool data set needs replacing.

Permanent I/O Errors

If you receive DM711 or DM725 abend codes, several I/O error messages, and JES3 functions no longer execute, the I/O error is probably permanent. (You can tell when JES3 functions no longer execute because you will no longer receive any JES3 messages.) To try recovery, use the following procedure:

1. Issue the command *F,Q,DD=ddname,STOP. This command requests that JES3:

   - suspend scheduling of jobs that have track groups allocated to the affected spool data set

   - stop jobs that are executing and have track groups allocated to the affected spool data set

   - stop all JES3 writers that are writing on the affected spool data set and reschedule them for later processing, beginning from the last checkpoint.

   - stop allocating track groups to the affected spool data set

2. If JES3 accepts the command, you can try to correct the problem that caused the I/O error.

3. If you correct the problem, issue the *F,Q,DD=ddname,RELEASE command. This command requests that JES3:

   - resume scheduling of jobs that have track groups allocated to the affected spool data set
   - resume allocating track groups to the affected spool data set

   JES3 uses the job failure options to determine how to process jobs that were executing at the time you issued the command *F,Q,DD=ddname,STOP.

4. If JES3 does not accept the *MODIFY,Q,DD=ddname,STOP command, or if you cannot quickly correct the problem that caused the I/O error, let JES3 continue execution without the affected spool data set.

   - If you used a DD statement in the JES3 cataloged procedure to allocate the affected spool data set, remove that DD statement from the procedure and perform a hot start.
   - If you used a DYNALLOC statement in the initialization stream to allocate the affected spool data set, issue the MVS VARY command on the global to vary offline the volume that contains the affected spool data set and perform a hot start.

     Note: Changes made to the JES3 spool configuration require a special operator dialog to complete the initialization process. Read the section *Removing and Reinstating a Spool Data Set* in *MVS/Extended Architecture Operations: JES3 Commands* before performing a JES3 hot start.

     If JES3 issues message IAT4102 during the hot start, the spool data set that you are attempting to remove contains the checkpointed initialization stream. You must perform a warm start to recreate the checkpointed initialization data.

     If you continue having problems because of the data set on the local processors, issue the MVS VARY command on the local processors. Perform a local start for each processor.

     If you need to restart JES3 before you can restore the spool data set, reissue the MVS VARY command on every processor to which you want the volume offline. If you cannot repair the volume quickly, you may want to remove the DYNALLOC statement for that spool data set from the initialization stream, perform a warm start on the global processor, and restart the local processors.

5. JES3 is now executing without the spool data set that caused the I/O errors. JES3 maintains information about the spool data set, including its size, its device characteristics, and the volume serial number of the volume on which it resides. However, JES3 considers the spool data set unavailable for use. Removing the spool data set in this manner does not release the spool space of jobs with data on the unavailable data set, unless the jobs have been cancelled. You may now repair the spool data set.

To restore the repaired data set to the JES3 complex:

1. If you removed a DD statement from the JES3 cataloged procedure, reinsert it and perform a hot start.

2. If you issued the VARY command on one or more processors to take the volume offline, issue the VARY command on those processors to bring the volume online. Perform a hot start on the global processor and a local start on the appropriate local processors.

3. If you removed the DYNALLOC statement from the initialization stream, reinsert the statement and perform a warm start.

4. Issue command *F,Q,DD=ddname,RELEASE. This command requests that JES3 resume scheduling of jobs that have track groups allocated to the affected spool data set. JES3 will use the job failure options to determine how to handle the jobs that were executing on a processor when you issued the command *F,Q,DD=ddname,STOP.

Replacing a Spool Data Set

If a permanent I/O error occurs on a spool data set and you cannot recover the data (for example, there is a head crash on a direct access device), you can replace the affected spool data set. To replace the data set, perform a warm start and follow the procedures outlined below. You may create the new data set on a volume or device type different from the one being replaced. You may also change the size of the data set and redefine the single track table (STT) range using the STT or STTL parameter on the TRACK or FORMAT initialization statement.

Be aware that when you replace a spool data set, JES3 cancels all jobs with data on the replaced spool data set. Other risks include the possible loss of JES3 control blocks, STT extents, checkpoint records, and the JESNEWS data set, which may have been on the damaged spool data set. If these losses occur, the system will issue messages giving you the opportunity to take appropriate actions.

If you cannot immediately perform a warm start (for example, if it takes some time for you to make the changes needed to replace the spool data set), you can cancel jobs that have track groups allocated on the spool data set being replaced. To cancel the jobs, issue the command *F,Q,DD=ddname,CANCEL. After you cancel the jobs, the user can resubmit them. You can then replace the spool data set at the time most convenient for your installation.

When you replace the spool data set, you must use the same ddname for the new spool data set as for the old.

To replace a spool data set, use the following procedure:

1. If you allocated the old spool data set by using a JES3 cataloged procedure, update the DD statement in the cataloged procedure to reflect information about the new data set. You may need to change the data set name, device number, device type, or volume serial number. Do not change the ddname.

   If you allocated the old spool data set by including a DYNALLOC statement in the initialization stream, update the optional parameters as necessary. Do not change the ddname.

2. If the old spool data set is cataloged, replace its catalog entry with an entry for the new spool data set.

3. If the new spool data set is unformatted and your initialization stream currently includes a TRACK statement for the old spool data set, replace it with a FORMAT statement. Otherwise, leave your TRACK or FORMAT statement alone.

4. Perform a warm start. Specify WR or WAR as the restart mode. JES3 will prompt you to enter the ddnames of replaced spool data sets (message IAT4009 for unformatted spool data sets and message IAT4008 for formatted spool data sets). JES3 will then cancel all jobs that have track groups allocated to the spool data sets being replaced.

Moving a Spool Data Set to Another DASD Volume

If you must move the contents of a spool data set to another DASD volume, perform a hot start with the data set not allocated or the DD statement for the data set removed from the JES3 start procedure. During JES3 initialization, JES3 considers the spool data set unavailable. After moving the data to the new DASD volume, perform a hot start with the data set (on the new volume) allocated or with the DD statement for the data set included in the JES3 start procedure. JES3 now considers the data set available.

Chapter 5. Defining Consoles and Message Routing

This chapter provides information about defining and tailoring consoles that you use to control your operating system. It also describes how to control where and how messages appear in your installation.

Defining consoles and message routing requires you to coordinate definitions in the MVS configuration program, the MVS SYS1.PARMLIB data set, and your JES3 initialization stream. These statements and the books in which you can reference them are cited throughout this chapter.

Defining Consoles

Consoles are devices that you use to enter commands and receive messages from JES3, MVS and application programs. Consoles fall into one of the following classes:

- JES3 consoles
- Multiple console support (MCS) consoles
- Remote job processing (RJP) consoles
- MVS/Bulk data transfer (MVS/BDT) consoles
- Network Job Entry (NJE) consoles
- Subsystem-allocatable consoles

JES3 consoles are devices that you can physically attach only to the JES3 global processor. You can use them to control your entire JES3 installation.

MCS consoles are devices that you can physically attach to global or local processors. Because MCS consoles are known only to the processor to which they are attached, communication is generally limited to within that processor. However, you can issue most JES3 commands from MCS consoles attached to the global and display messages from any processor in your installation. To give an MCS console attached to the global the ability to control your entire installation (that is, send commands to local processors), you must define a logical association to an MCS console on each local processor during JES3 initialization.

RJP consoles are devices that you attach to the JES3 global as part of a remote workstation using telecommunications lines. RJP permits you to transfer jobs to and from workstations that reside at some distance from your installation.

BDT consoles are logical devices that you define for the MVS/bulk data transfer facility. Refer *MVS/Bulk Data Transfer Facility: Initialization and Network Definition* for information about using MVS/BDT.

NJE consoles are logical devices that you define for a job entry network. Refer to Chapter 9, "JES3 Networking" for for information about using NJE.

Subsystem-allocatable consoles are logical devices that you define for purposes such as console associations, which permit processor-to-processor communication. Refer to "Establishing Logical Associations" on page 5-10 later in this chapter for a discussion about logical associations.

The functions of JES3 and MCS consoles frequently overlap. To properly configure consoles in your installation, be aware of the distinctions that are created depending on how you define them:

- **JES3-only console:** This console accepts only JES3 commands; you cannot use it to enter MVS commands because you do not establish console associations for this device (refer to "Establishing Logical Associations" on page 5-10 for information about defining logical associations). You cannot use this console to send commands to local processors. However, this console can display messages that originate from any processor in your installation. You define JES3-only consoles in your JES3 initialization stream using JES3 CONSOLE and DEVICE initialization statements. You can omit the command prefix when entering JES3 commands (usually the digit 8 or an asterisk (*)) because all commands go to JES3.

- **JES3-managed or MCS-managed console:** You define this console to MVS in the CONSOLxx member of SYS1.PARMLIB ***and*** to JES3 by coding a device type on the TYPE= keyword of a JES3 CONSOLE initialization statement. You can initialize this console as either a JES3-managed console or as an MCS-managed console and can dynamically convert this console between console managers:

  - **When JES3-managed,** this console accepts both JES3 and MVS commands; you cannot use it to send commands to local processors. JES3-managed consoles can display messages that originate from any processor in your installation.

  - **When MCS-managed,** this console communicates only with the processor to which it is attached. You can enter all MVS commands from this console. If the console is attached to the JES3 global, you can also enter most JES3 commands and receive messages that originate from any processor in your installation. If attached to a JES3 local, you can enter only a subset of JES3 commands. See the topic, "Defining MCS Consoles" on page 5-6, later in this chapter for a list of commands that an MCS-managed console accepts.

- **JES3-managed console with a logical association:** This console can communicate with JES3 and MVS on one or more local processors in your installation, as well as on the global. Communication with local processors is possible because of logical associations that you define during initialization. Refer to "Establishing Logical Associations" on page 5-10 for information about defining logical associations. You define this console as an MCS console during MVS initialization and as a JES3 console during JES3 initialization. You can dynamically convert this console to MCS-management using operator commands. If you dynamically convert this console to an MCS-managed console, the console retains its logical associations to other processors.

- **MCS-only console:** You define this console during MVS initialization as an MCS console; therefore, it can communicate only with MVS on the processor to which it is attached. You do *not* define it during JES3 initialization; however, you can enter most JES3 commands if the console is attached to the global. You can issue only a subset of JES3 commands if the console is attached to a local processor. See the topic, "Defining MCS Consoles" on page 5-6, later in this chapter for a list of JES3 commands that an MCS-only console accepts.

- **MCS-managed console with a logical association:** You define this console as an MCS console during MVS initialization. You also define it to JES3 by coding TYPE=MCS on a JES3 CONSOLE initialization statement. Coding TYPE=MCS and the UNIT= parameter establishes a logical association with a console on another processor. A logical association allows you to send commands from the MCS console on the global to a local using the JES3 *SEND command. As with all MCS consoles, you can enter most JES3 commands and display messages that originate from any processor if you attach the console to the JES3 global. If you attach this console to a local processor, you can enter only a subset of JES3 commands. This console differs from an MCS-managed console in that you *cannot* dynamically modify it to become JES3-managed.

- **Subsystem-allocatable console:** You define this console as an MCS console using the CONSOLE statement in the CONSOLxx member of the MVS SYS1.PARMLIB data set. This console is reserved for use by a subsystem, such as JES3. To JES3, subsystem-allocatable consoles are logical devices. This book often refers to logical consoles used by JES3 as ***MCS DUMMY consoles***. For example, you can use a subsystem-allocatable console to create a logical association between a real console on your global with a logical console on the local instead of, or in place of a real console device.

- **JES3 Dummy Console:** This is an internally-defined console for use by JES3 dynamic support programs (DSPs) that do not have access to a console. DSPs use this console to submit commands to the operating system. Do not confuse the JES3 DUMMY console with an MVS subsystem-allocatable console which is called an MCS DUMMY console.

MVS allows you to define as many as 99 consoles (real and subsystem-allocatable) for each processor in your installation.

Figure 5-1 shows a simple installation with JES3-managed and MCS-managed consoles.

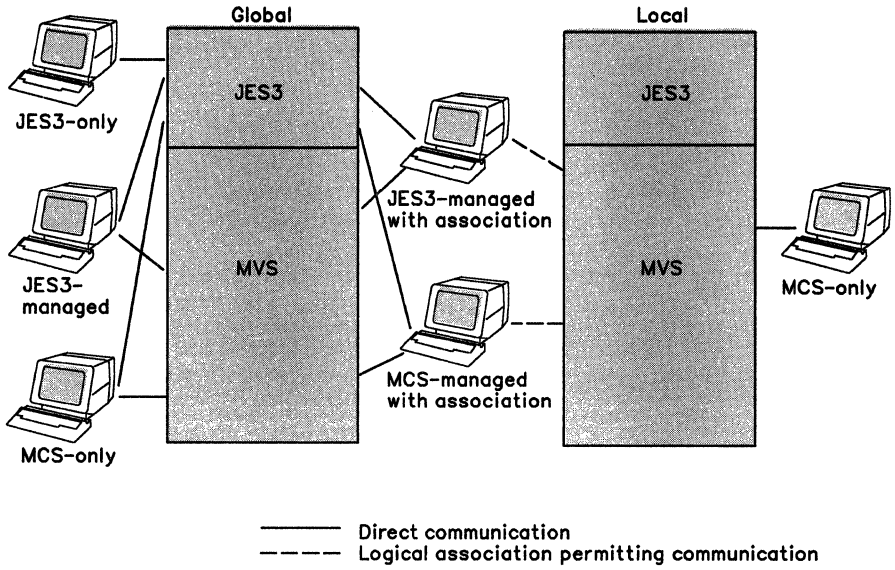


Figure 5-1. Complex With JES3-managed and MCS-managed Consoles

Defining JES3 Consoles

The term ***JES3 consoles*** means both JES3-only consoles and JES3-managed consoles. JES3-managed consoles are often referred to as ***full-function consoles*** because you can use them to communicate with both JES3 and MVS. Use the following guidelines when defining JES3 consoles:

- Define all devices that you plan to use as JES3 and/or MCS consoles (except RJP devices) on IODEVICE statements. You specify IODEVICE statements in the input data set for the MVS configuration program.

- You must define all JES3 consoles in your JES3 initialization stream using JES3 CONSOLE initialization statements. You should also define all JES3 consoles to MVS in the CONSOLxx member of the MVS SYS1.PARMLIB data set.

- You must also include a JES3 DEVICE statement in your JES3 initialization stream for each JES3 console. If you plan to use the device only as a console, specify DTYPE=CNSxxxx on the DEVICE statement. If you want to use the device for some other purpose, as well as a console, specify a non-console device type such as a printer (DTYPE=PRTxxxx). Do not include a DEVICE statement for MCS-managed consoles that have logical associations; you can never make these devices JES3-managed.

- If you plan to use a device as both a JES3 console and for some other purpose (such as a printer), and you want to initialize the device as a JES3 console, specify a valid destination class on the DEST= keyword of the device's CONSOLE statement. If you want to initialize the device as a non-console device, specify DEST=NONE and a non-console device type on the DTYPE= keyword such as a printer (DTYPE=PRTxxxx). You can use the JES3 *VARY command to change the function of such a device.

- Code the UNIT= keyword of the JES3 CONSOLE initialization statement to create a logical association. You can associate a JES3-managed or MCS-managed console attached to the global with a real or subsystem-allocatable console attached to a local processor. Refer to "Establishing Logical Associations" for information about logical associations.

- JES3 cannot acquire a device and initialize it as a console if it is in use by another component. For example, if MVS is currently using a device that you want to use as a JES3 console, you must first release the device by varying the device online to MVS (varying the device online to MVS makes the device available to all subsystems, including JES3).

- Before JES3 attempts to acquire a device, it checks the device's ***JES3*** online/offline status (which is independent of its ***MVS*** online/offline status). If the device is online to both JES3 and MVS and *not* in use by another component, JES3 can acquire the device and initialize it as a JES3 console. Use the JUNIT= keyword of the JES3 CONSOLE initialization statement to define a console's initial JES3 online/offline status.

Initializing JES3 Consoles

Because you define consoles to both MVS and JES3, MVS can 'steal' a device that you intend to initialize as a JES3 console unless you prevent MVS from acquiring it during MVS initialization. If you want JES3 to acquire a device as a JES3 console, you should:

1. Make the device offline before MVS initialization using the IODEVICE statement that you include in the input data set for the MVS configuration program. MVS cannot acquire a device as a console during MVS initialization if you set its initial status as offline to MVS.

2. After MVS initializes, vary the device online to MVS using the MVS VARY ddd,ONLINE command[1] (at this point, it is too late for MVS to automatically acquire the device as a console). You perform this step because JES3 cannot acquire a device as a console unless it is online to MVS and not in use by MVS or another subsystem.

3. Initialize JES3. JES3 can now acquire the device as a JES3-managed console if you define its JES3 status as online.

---

1 You can automatically issue this command during MVS initialization by placing it in the SYS1.COMMNDxx member of the MVS SYS1.PARMLIB data set.

Defining MCS Consoles

The term ***MCS consoles*** means both MCS-only consoles and MCS-managed consoles. You must define all MCS consoles in the CONSOLxx member of the MVS SYS1.PARMLIB data set. MCS consoles accept all MVS commands based on their MCS authority level.

The MCS master console is the principal device for communication with its MVS processor, and is required by MVS on each processor. The master console has the highest MCS authority level assigned to it. Master consoles can issue MVS commands of any authority level, and can receive all messages. MCS secondary consoles cannot issue all MVS commands or respond to all action messages; these consoles receive only those messages specifically routed to them.

If attached to the global, MCS consoles accept all JES3 commands except the following, which affect console screen presentation:

- *DELAY
- *ERASE
- *FREE (non-directed)
- *START,CNT [2]

If an MCS console is attached to a local processor as the master console, you can issue only the following JES3 commands:

- *CALL, *START, *CANCEL,DSI
- *CALL, *START, *CANCEL,VARYL
- *RETURN (requires a password)
- *DUMP (requires a password)

In addition to the commands listed above, MCS-only consoles do not accept the following commands:

- *SEND
- *INQUIRY,O,MAIN
- *MODIFY,M,MAIN

You can also assign a ***JES3*** authority level to MCS-managed consoles with a logical association (master or secondary) by coding the LEVEL= keyword on the JES3 CONSOLE initialization statement. You can also change the JES3 authority level of these consoles using the JES3 *MODIFY,O command. The MCS command authority level that you specify in the CONSOLxx member of the MVS SYS1.PARMLIB data set determines the JES3 authority level for MCS-managed consoles without logical associations. Refer to "Authorizing JES3 Commands" on page 5-14 for information about defining console authority.

To send commands directly to a local processor from an MCS console on the global processor, you must create an association with a console on each local processor with which communication is desired. To associate MCS-managed consoles, you must specify TYPE=MCS in conjunction with the UNIT= parameter on a JES3 CONSOLE initialization statement. Refer to "Establishing Logical Associations" on page 5-10 for information about defining logical associations.

---

[2] Output from the console test DSP can not be directed to an MCS console.

JES3 DSPs that you call into execution retain the ID of the console from which they were called. You should symmetrically define your MCS-only consoles. That is, if you have defined an MCS-only console as the fifth valid entry in the CONSOLxx member of SYS1.PARMLIB, you should define that console as the fifth valid entry in the CONSOLxx member for all processors. If you do not symmetrically define your MCS-only consoles, then messages about those DSPs may not appear on the expected console after a DSI or hot start with IPL.

For full flexibility in initiating the JES3 console environment, you should generate as many MCS consoles and subsystem-allocatable MCS consoles (combined) as the maximum number of JES3 consoles planned for attachment to each processor. You must define subsystem-allocatable MCS consoles during MVS initialization through statements in the CONSOLxx member of SYS1.PARMLIB. Subsystem-allocatable MCS consoles do not have a device number and do not correspond to any physical device. On a JES3 CONSOLE initialization statement, specify DUMMY on the UNIT= parameter for those mains on which you want to use the console. For additional information about defining subsystem-allocatable consoles see *MVS/Extended Architecture System Programming Library: Initialization and Tuning.*

Defining RJP Consoles

Remote job processing consoles can be either bisynchronous communication (BSC RJP) consoles or system network architecture (SNA RJP) consoles. You define a BSC RJP console during initialization using a JES3 CONSOLE initialization statement. The name you specify on the JNAME= keyword of the JES3 CONSOLE initialization statement must match the name you specify on the N= keyword of the JES3 RJPTERM initialization statement.

If you want a work station to have the facilities of a JES3 console, you must code the work station name and console options on the JES3 CONSOLE initialization statement for that console.

You can define simulated consoles for work stations that do not have real consoles. In this case, you enter console commands through the card reader, and receive messages on the terminal's printer.

You also define a SNA RJP console during initialization using a JES3 CONSOLE initialization statement. The name you specify on the JNAME= keyword of the CONSOLE statement must match the name you specify on the N= keyword of the RJPWS statement.

JES3 Console Management

JES3 console service manages communication between consoles and JES3. In managing console communication, console service:

- Provides operator communication with JES3 functions
- Manages the buffers used for communication
- Provides processing for input and output messages
- Coexists with the multiple console support (MCS) facility

Operator Communication

You communicate with JES3 dynamic support programs (DSPs) using JES3 commands. When a JES3 DSP initiates execution, it must identify itself to console service. You can refer to JES3 DSPs by the name or number of the device assigned to the dynamic support program.

Console Message Buffer Pool

JES3 uses a preallocated buffer pool for both input and output console messages. Each time a buffer is required, JES3 obtains it from this buffer pool. When the buffer is no longer needed, JES3 returns it to the pool.

To define the number of buffers in the pool, use the JES3 CONSTD initialization statement. To define buffer depth, the number of messages that a DSP may queue for a specific console, code the DEPTH= keyword on the JES3 CONSOLE initialization statement.

Input Processing

Input commands which you initiate are directed to the operating system.

You can enter all JES3 commands (except *DUMP and *RETURN) from a card reader (CR), tape reader (TR), or disk reader (DR). You can use the tape or disk reader to enter repetitive commands based on system requirements (such as shift change). Any output messages generated from a card reader, tape reader, or disk reader are displayed at the console from which you called the reader.

You can enter a pause command from any reader through the use of the //*PAUSE control statement. JES3 recognizes this statement only if the statement appears before the first //JOB statement in the job's input stream. Once the //*PAUSE statement is recognized, the reader issues a message and waits for a reply. (For example, if *CALL and *START DSP commands are entered through the reader, the //*PAUSE statement can be used to stop the reader after the *CALL,dsp command is issued. This allows the DSP to be readied before the *START command is executed. When the DSP is ready, you can start the reader to have the next command executed.) The use of the //*PAUSE statement is intended primarily for system checkout and test.

Output Processing

Output messages are initiated by JES3 DSPs on any main, output writer functional subsystems, C/I functional subsystems, MVS, or any user program that issues a write-to-operator or write-to-operator reply (WTO/WTOR) to a JES3 console. These messages may be job status messages, replies to operator inquiries, or operator action-required messages.

All output messages are prefixed with a time stamp. In addition, JES3 messages contain the component identification IAT. MVS messages that are displayed on JES3 consoles contain the name of the processor from which the message originated and the name of the job (if applicable) that issued the message. Refer to "Defining Message Routing" on page 5-21 for information about controlling message traffic.

JES3 supports the display of messages in 4 colors on 3279 consoles and the intensification of important messages on 3277 and 3278 consoles. For information on the meaning of the colors or intensification, see *MVS/Extended Architecture Operations: JES3 Commands.*

MCS Console Management

JES3 allows you to use the multiple console support (MCS) facility of MVS. MCS provides the following support:

- ***Backup console service:*** When a console fails, you can specify an alternate console to process messages that were being sent to the original console. In this case, the routing codes of the two consoles are merged.
- ***Operator action messages:*** On MCS consoles configured in conversational mode or in roll-deletable mode, action messages remain on the screen until deleted by the program issuing them or until you delete them manually. (If the MCS console was defined as an output-only console, the messages can be deleted from the screen by entering a system command from a JES3 console associated with that MCS console.)
- ***Screen-oriented displays on display consoles:*** At system generation, the MCS console screen can be divided into multiple screen areas for receiving these displays in out-of-line or nonmessage screen areas. If operator action is unnecessary, MCS consoles can also be designated as output-only consoles.
- ***Authority levels:*** You can assign authority levels to allow or restrict the types of MVS commands that an operator can enter at a console; however, you must set up the MCS master console on each processor to accept all MVS commands. For MCS-managed consoles with logical associations, you can also specify an authority level for JES3 commands using the LEVEL= keyword of the JES3 CONSOLE initialization statement. Refer to "Authorizing JES3 Commands" on page 5-14 for information about defining console authority.
- ***Log facilities:*** You can use a real device as a hard-copy log or you can use the system log facilities.
- ***Enhanced Display Capability:*** You can control several characteristics of MCS console such as reverse video, extended highlighting and seven color support.

Establishing Logical Associations

A logical association is an internal communication path that permits you to send commands to MVS on a local processor from a JES3-managed or MCS-managed console attached to the global. An association also permits you to receive responses to those commands.

To send commands to a local processor from a JES3-managed or MCS-managed[3] console from the global, you must associate that console with an MCS real or subsystem-allocatable console on each local processor with which communication is desired. You define an association using the UNIT= keyword of the JES3 CONSOLE initialization statement. This keyword identifies all MCS consoles with which the console you are defining can communicate. Use the JES3 *SEND command to route commands to another processor after you establish a logical association.

You can associate a JES3-managed console with one MCS real or subsystem-allocatable console defined to each local processor. You can also associate a JES3-managed console to a subsystem-allocatable console or to itself on the global.

You can associate an MCS-managed console[3] with only one MCS or subsystem-allocatable console defined to each local processor, other than the global.

How to Define Logical Associations

Use the UNIT= keyword of the JES3 CONSOLE initialization statement to define logical associations. How you code the UNIT= keyword depends on the type and purpose of each console. For example, if you have a global processor named SY1 and two local processors named SY2 and SY3, you can define logical associations as follows:

- **JES3-managed consoles with logical associations:** You can associate a JES3-managed console with actual console numbers. For example, if you specify:

  ```
  CONSOLE,JNAME=CN1,UNIT=(SY1,3E1,SY2,3E1,SY3,3E1)...
  DEVICE,CNS3277,JNAME=CN1,JUNIT=(3E1,SY1)
  ```

  This definition allows you to send commands from console 3E1 on SY1 to MVS on processors SY2 and SY3 and receive responses to those commands. These associations are retained if you dynamically convert this console to MCS management. Specifying actual device numbers rather than subsystem-allocatable (DUMMY) or no association (NONE) allows you to issue all JES3 commands other than the *SEND command when you make it MCS-managed. Specifying the same device number (3E1) across all mains also prevents problems when you perform a dynamic system interchange (DSI).

---

[3] This console is known as an MCS-managed console with logical associations. You define it to MVS in the CONSOLxx member of the SYS1.PARMLIB data set ***and*** to JES3 by specifying TYPE=MCS on a JES3 CONSOLE initialization statement.

- **MCS-managed console with logical associations:** You can also associate an MCS-managed console with actual device numbers. For example:

```
CONSOLE,TYPE=MCS,UNIT=(SY1,3E1,SY2,3E1,SY3,3E1)
```

  Commands and commands responses appear on both the console from which you enter the command ***and*** on the associated consoles on the global. You can avoid this condition by defining three separate console statements for this console. For example,

```
CONSOLE,TYPE=MCS,UNIT=(SY1,3E1,SY2,DUMMY,SY3,DUMMY),JNAME=MCS1
CONSOLE,TYPE=MCS,UNIT=(SY1,DUMMY,SY2,3E1,SY3,DUMMY),JNAME=MCS2
CONSOLE,TYPE=MCS,UNIT=(SY1,DUMMY,SY2,DUMMY,SY3,3E1),JNAME=MCS3
```

  Defining your console in this fashion permits you to establish logical associations with local processors without echoing commands and command responses on actual consoles attached to those processors. If you use this method, use unique names on the JNAME= keyword so that the system log contains the console name that you are currently using. Both examples shown above allow you to retain your associations across a DSI.

Rules for Defining Logical Associations

- You can specify mains in any order on the UNIT= keyword of a JES3 CONSOLE initialization statement. However, you can not specify a main more than once.

- You can use a console only once for an association within a given processor. For example, if you associate a JES3 console with an MCS console, you cannot use that MCS console in an association for another console attached to the global.

- If you are associating a JES3-managed console with a real MCS console attached to the global, you must use the device number specified on the JUNIT= keyword of the JES3 DEVICE initialization statement for that JES3 console when defining the association.

- You must specify the actual device number of an MCS-managed console on the UNIT= keyword for the global device number. For example, if SY1 is your global, you must specify UNIT=(3E1,SY1...) on the JES3 CONSOLE initialization statement. You cannot specify DUMMY or NONE for the global.

- The console with which you create an association on the target processor determines the MCS command authority level for commands that you send to that processor. If you create an association with a subsystem-allocatable console, you cannot control which subsystem-allocatable console you get. Therefore, the authority level is unpredictable unless you assign the same authority level to all subsystem-allocatable consoles in your installation.

Sending Commands From JES3-managed Consoles

You can send commands to another processor without using the *SEND command if you establish a logical association with the target processor and include the MAIN= keyword on the console's JES3 CONSOLE initialization statement. If you omit the MAIN= keyword, you must use the *SEND command. You cannot specify the MAIN= keyword for MCS-managed consoles that have logical associations; you must always use a JES3 *SEND command from these consoles.

If you dynamically migrate a JES3-managed console to MCS-management, you must use the *SEND command afterwards. Refer to *MVS/Extended Architecture Operations: JES3 Commands* for information about using the *SEND command.

Changing the Status of a Console

You are not required to have any JES3-managed consoles in your JES3 installation because you can use MCS consoles to enter JES3 commands and receive JES3 messages. Of course, if you want to hardcopy messages to MLOG, rather than DLOG, you need at least one JES3-managed console.

You can permanently migrate a JES3-managed console to MCS management by changing the TYPE= keyword on the CONSOLE statement to TYPE=MCS. However, this change requires a warm start of JES3. You must use this method if you have not defined the JES3 console with an association to an MCS console.

You can use commands to dynamically change the status of a console. Figure 5-2 summarizes the command(s) required to change the status of a console. Refer to *MVS/Extended Architecture Operations: JES3 Commands* for the syntax of these commands.

Device's current MVS/JES3: state	Vary the device online to MVS	Vary the device offline to MVS	Make the device an MCS console	Vary the device online to JES3 (AV)	Vary the device offline to JES3 (OFF)	Disable the device as a JES3 console (CON)	Enable the device as a JES3 console (CON)
MVS ONLINE & not a MCS console	current state	VOFFLINE	VCONSOLE	Determine current JES3 state and use corresponding commands in this column	Determine current JES3 state and use corresponding commands in this column	*VCONSOLE *SWITCH *DISABLE	*VCONSOLE
MVS OFFLINE & not a MCS console	VONLINE	current state	VCONSOLE	Determine current JES3 state and use corresponding commands in this column	Determine current JES3 state and use corresponding commands in this column	VONLINE *VCONSOLE *SWITCH *DISABLE	VONLINE *VCONSOLE
MVS ONLINE & a MCS console	VONLINE	VOFFLINE	current state	Determine current JES3 state and use corresponding commands in this column	Determine current JES3 state and use corresponding commands in this column	VONLINE *VCONSOLE *SWITCH *DISABLE	VONLINE *VCONSOLE
ONLINE to JES3 (AV)	VONLINE[1]	VOFFLINE[1]	VCONSOLE[1]	current state	*VOFFLINE	Determine current MVS state and use corresponding commands in this column	Determine current MVS state and use corresponding commands in this column
OFFLINE to JES3 (OFF)	VONLINE[1]	VOFFLINE[1]	VCONSOLE[1]	*VONLINE	current state	Determine current MVS state and use corresponding commands in this column	Determine current MVS state and use corresponding commands in this column
JES3 console disabled (CON)	VONLINE[1]	VOFFLINE	VCONSOLE	*VCONOFF *VONLINE	*VCONOFF	current state	*ENABLE *SWITCH
JES3 console enabled (CON)	VONLINE[1]	*SWITCH *DISABLE *VCONOFF VOFFLINE	*SWITCH *DISABLE *VCONOFF VCONSOLE	*SWITCH *DISABLE *VCONOFF *VONLINE	*SWITCH *VOFFLINE	*SWITCH *DISABLE	current state

Figure 5-2. Changing the State of a Console

Note 1: You do not need to enter this command if the device is currently in the desired MVS state. Issue an MVS **D U,ddd,1** command to determine the device's MVS status.

Abbreviation	Meaning
VONLINE	MVS vary online command
VOFFLINE	MVS vary offline command
VCONSOLE	MVS vary console command
*VONLINE	JES3 vary online command
*VOFFLINE	JES3 vary offline command
*VCONSOLE	JES3 vary console command
*VCONOFF	JES3 vary console,off command
*SWITCH	JES3 switch command
*ENABLE	JES3 enable command
*DISABLE	JES3 disable command

Figure 5-3. Key to Abbreviations

Defining Console Authority

MVS and JES3 allow you to control which commands operators can enter at consoles. MVS and JES3 perform independent authority checking. JES3 checks all JES3 commands entered at JES3 and MCS consoles. MVS checks all MVS commands entered at JES3 and MCS consoles.

Authorizing JES3 Commands

JES3 groups its commands into command authority levels that range from 1 to 15. You can assign a JES3 authority level to a JES3 or MCS console with a logical association by specifying the LEVEL= keyword on the JES3 CONSOLE initialization statement for each console.

Every time you issue a JES3 command, JES3 checks the authority of the console from which you entered the command. If you authorize the console to issue the command, JES3 processes the command; otherwise, JES3 rejects the command and displays an error message. Figure 5-4 lists the JES3 commands associated with each authority level:

Authority Level	Allowable JES3 Commands
0-4	*ERASE *INQUIRY *MESSAGE
5-9	All of the commands listed above plus: *CALL *CANCEL *RESTART *SEND *START
10-14	All of the commands listed above plus: *DELAY *DISABLE *ENABLE *MODIFY *SWITCH *TRACE *VARY *FREE
15	All of the commands listed above plus: *FREE,con *DUMP *FAIL *RETURN

Figure 5-4. JES3 Authority Levels

After initialization you can alter a console's authority level by issuing a *MODIFY,O command. Use the *INQUIRY,O command to display a console's current authority level. You can use the JES3 user exit IATUX18 to redefine authority levels in specific cases. See *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros* for information about IATUX18.

Notes:

- You must authorize a least one console in your MVS/JES3 installation to issue authority level 15 commands.
- You should always specify the highest JES3 authority level when defining consoles for use by automation packages, such as the IBM NetView™[4] program product. Specifying a lower authority level may prevent your automation package from issuing certain commands.

The MCS authority level that you define on the AUTH parameter of the CONSOLE statement in the CONSOLxx member of the MVS SYS1.PARMLIB data set determines which MVS commands are allowable from JES3-managed as well as MCS-managed consoles. The MCS authority level also determines which ***JES3*** commands are allowable from MCS-managed[5] or MCS-only consoles as follows:

MVS Authority Level (Command Group)	Corresponding JES3 Authority Level	Allowable JES3 Commands
Informational	0	*ERASE *INQUIRY *MESSAGE
System Control	5	All of the commands listed above plus: *CALL *CANCEL *RESTART *SEND *START
Console Control I/O Control	10	All of the commands listed above plus: *DELAY *DISABLE *ENABLE *MODIFY *SWITCH *TRACE *VARY *FREE
Master	15	All of the commands listed above plus: *FREE,con *DUMP *FAIL *RETURN

Figure 5-5. JES3 Commands Allowed from MCS-managed and MCS-only Consoles

Figure 5-6 on page 5-16 shows the commands and authority levels for RJP consoles (BSC and SNA).

---

4 NetView is a trademark of International Business Machines Corporation. All rights reserved.

5 Excluding MCS-managed consoles with logical associations. These consoles obtain their authority level from the LEVEL= keyword on the JES3 CONSOLE statement.

JES3 Authority Level	Allowable JES3 Commands	Restrictions
0-4	No remote console support exists.	
5-9	*START *RESTART *CANCEL *INQUIRY *MESSAGE	For BSC terminals, the *START, *RESTART, and *CANCEL must specify a remote terminal name which is part of the remote work station. For SNA terminals, the *START, *RESTART, and *CANCEL must specify a logical device name which belongs to the SNA work station.
10-14	All of the commands listed above plus: *CALL *VARY *MODIFY	The input (IN=) and output (OUT=) keywords specified on a *CALL command must be devices associated with the remote work station. You can specify the long form (such as IN=REMOTRD1) or short form (IN=RD1). If you omit both keywords on a *CALL command for a function in which card input is expected, (that is *X,CR), the remote card reader is assumed. The restrictions discussed previously for the *START, *RESTART, *CANCEL and *VARY commands also apply for this authority level. Three *MODIFY commands are permitted: modify job (*F,J=...), modify output (*F,U,...) and modify RJP (*F,T,...). The modify job command allows the remote operator to modify jobs submitted from that terminal group. If you omit group or terminal name from the T= keyword, JES3 inserts those parameter values into the message. The SNA remote console operator can enter *START, *RESTART, and CANCEL commands, but each must specify a logical device name which belongs to the SNA work station. The SNA remote console operator can also enter *INQUIRY, *MODIFY, and *MESSAGE commands, but only two *MODIFY commands (MODIFY,J=jobname and *MODIFY,T,T=termname) are permitted.
15	All commands except: *SWITCH *DUMP *RETURN *ENABLE *DISABLE	JES3 does not provide any default parameters.

Figure 5-6. Authority Levels for Remote Consoles

Authorizing TSO Commands

If TSO user exit IKJEFF53 is not performing authority checking, you can use JES3 user exit IATUX30 to authorize the use of the TSO CANCEL, STATUS, or OUTPUT commands. JES3 invokes IATUX30 each time a user issues one of these TSO commands.

Entering Commands

You can enter JES3 commands from all JES3 consoles. You can enter MVS commands and most JES3 commands from MCS-managed consoles with logical associations.

You cannot enter JES3 commands to inquire about or modify the status of MCS consoles. You must use the equivalent MVS commands instead. For example, MCS-managed consoles with logical associations reject JES3 *ERASE, *DELAY, and *FREE commands (however, you can issue *FREE,con which is also known as a directed *FREE command from MCS-managed consoles with logical associations if the target console is a JES3 console.)

You can issue only the following JES3 commands from master MCS consoles that are attached to local processors:

- *RETURN (with password)
- *DUMP (with password)
- *CALL, *START, and *CANCEL,VARYL
- *CALL, *START, *CANCEL,DSI

During JES3 initialization, you should not enter any JES3 commands other than the *DUMP, *RETURN, and *START JSS commands. You should always use equivalent MVS commands.

You can enter MVS commands from all MCS consoles. You can also enter MVS commands from JES3-managed consoles for which you have defined logical associations, but you must precede them with the *SEND[6] command if you direct them to a processor other than the one to which the console is attached.

Defining Program Function Keys

You can enter commands using program function keys on JES3 consoles if you prepare beforehand using JES3 initialization statements. You define program function keys for MCS consoles in the PFKTABxx member of the MVS SYS1.PARMLIB data set.

To prepare, you must name and define one or more PFK tables. You can specify up to 24 program function keys in each table and can assign up to 12 commands with each key. Use the JES3 PFK initialization statement to name and define a PFK table. Each statement defines the commands you want to assign to each program function key. The name you specify on the N= keyword is the name of the PFK table to which JES3 will assign the PFK statement.

After defining a PFK table, you can assign a table to each JES3 console by specifying the table name (the same name you specified on the N= keyword) on the PFK= keyword of the JES3 CONSOLE initialization statement. You can associate any number of consoles with one PFK table. However, you can assign only one table at a time to a given console.

After you assign a PFK table to a console and you depress a program function key (or select keys 1-12 with a selector pen), JES3 executes the commands assigned to that key. If you depress or select a key for which no commands are defined, JES3 cancels the current input line.

Optionally, you can use the E= keyword of the PFK statement to request that JES3 display the commands associated with a key when you depress or select the key. This allows you to add parameters to the displayed list. When you have finished modifying the list of parameters, you can depress the ENTER key or a similar key and JES3 will execute the command.

You can dynamically redefine individual program function keys using the *MODIFY,K command. The *MODIFY,O command, assigns a different PFK table to a console. If you enter JES3 commands that modify program function keys from a console when it is MCS-managed, you affect only the JES3 program function key settings, not the MCS program function keys. Use MVS commands to change program function keys for MCS-managed consoles. Refer to *MVS/Extended Architecture: System Commands* for information about using MVS commands to change program function keys for MCS consoles.

---

6 You can omit the *SEND command from JES3-managed consoles if you include the MAIN= keyword on the JES3 CONSOLE initialization statement.

Defining Alternate Consoles

You can define an alternate console for each locally-attached JES3 console in your installation. Use the ALTCON= keyword of the JES3 CONSOLE initialization statement to define an alternate console.

JES3 switches console operations to the alternate console if the primary console encounters an I/O error or if the console goes out of ready status. For example, a console can go out of ready status if it reaches the message limit that you specify on the DEPTH= keyword of the JES3 CONSOLE initialization statement.

The following rules apply when defining and using alternate consoles:

- You cannot use the following consoles as alternate consoles, nor can they have alternates themselves:
  - RJP consoles
  - JES3 DUMMY consoles
  - MCS-managed consoles with logical associations

  Note: Although remote consoles cannot have alternates, JES3 will not prevent you from defining one on the ALTCON= keyword. JES3 simply ignores the value in the event of a console failure.
- If you do not define an alternate console, or you specify an invalid console, such as DUMMY, JES3 switches to the first active, locally-attached JES3 console (other than itself) defined in the initialization stream.
- If a failing console cannot locate an alternate, it uses itself as the alternate and attempts to correct the problem.
- To prevent automatic switching, specify the same name on the ALTCON= keyword that you specify on the JNAME= keyword for that console.

Defining a Time Limit for Console Messages

You can limit how long I/O operations can take for those I/O operations that send messages to consoles. If an I/O operation exceeds the time limit, JES3 switches to the alternate console (JES3 assumes that a failure in the original console caused the time limit to expire). You can use the IOWAIT= keyword of the CONSOLE statement to specify a time limit for each JES3 console. The following factors determine the time limit you should specify:

- **For all consoles:** The more activity there is on the channel path to which the console is attached, the more time you should specify. Use performance measurement tools such as the MVS Resource Monitoring Facility (RMF) to measure channel path activity.
- **For a console without a buffer:** Base the time limit on the size of the largest message that will be sent to the console (the larger the message, the more time you should specify).

- **For a console with a buffer:** Base the time limit on the time it takes to write a full buffer to the console (the more time it takes, the more time you should specify).

- **For printers defined as consoles:** Consider the printer speed (the faster the printer, the less time the I/O operation should take) and the carriage return time (for printers with a moveable carriage).

Defining the System Log

The system log is a device or pair of devices that record message traffic in your installation. The log can be a hardcopy device, such as a printer, a graphic console, or a disk data set which resides on spool.

JES3 uses the term ***master log*** (MLOG) to identify a JES3-managed console (usually a printer or a graphic console defined to receive destination class MLG), and ***disk log*** (DLOG) to identify the disk data set log. The MLOG and DLOG are collectively referred to as the ***system log***.

JES3 manages logging devices. JES3 requires you to have at least one type of log active at all times. However, if you do not use MLOG, then all JES3 and MVS messages originally destined for MLOG are sent to DLOG.

JES3 and MVS record all message traffic in the system log. Each log entry includes a time stamp, a date stamp, and the text of the message. Each entry in the log can also include:

- The JES3 destination class to which the message was routed.
- The JES3 or MCS console from which a command was entered[7].
- The JES3 or MCS console to which a message was routed.
- The name of the job associated with the message.
- Whether or not the system suppressed the message from console display.
- The system ID for messages issued by another processor.
- Flags indicating unusual conditions or marking lines of special interest.

Figure 5-7 shows sample log entries.

```
      MASTER  87311  1908053 +I,O
MLG   MASTER  87311  1908055  IAT8541  NAME      ADDR    LV  ALT      MAIN     SWITCH      DEPTH  DEPQD
MLG   MASTER  87311  1098055  IAT8542  CN3E1     (3E1)   15  CN310    SY1      DISABLED    050    00000
MLG   MASTER  87311  1908055  IAT8542  MASTER    (3E0)   15  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  MCS15     (320)   15  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  MCS10     (320)   10  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  MCS05     (3DC)   05  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  MCS00     (3DD)   00  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  MCS02     (302)   15  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  MCS03     (303)   15  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  AUTOMCS    ( )    15  -------  -------  TYPE=MCS    ---    -----
MLG   MASTER  87311  1908055  IAT8542  CN310     (310)   15  CN3E1    SY1      CN3E1       ---    -----
MLG   MASTER  87311  1908055  IAT8542  CN311     (311)   15  CN3E1    SY1      CN3E1       ---    -----
```

Figure 5-7. Sample Log Entries

---

7 Entries for MCS-managed consoles used by automation packages will show an MCS console number in the log until the automation package issues a *SEND command. The *SEND command activates the console's logical associations, after which the console's JES3 name (JNAME) appears in the log.

Rules

- You must have both MLOG and DLOG active if you want to direct the *FREE command to a hardcopy device that has failed or run out of paper.
- If you specify HARDCOPY=MLOG or HARDCOPY=(DLOG,MLOG), JES3 sends the message log to consoles that have a destination class of MLG. Therefore, assign this destination class to only those consoles that you want to receive passwords if you are logging security messages (routing code 9) and/or write-to-programmer messages (routing code 11).
- For MLOG, a JES3 hard-copy console whose active destination classes include MLG (that is, you specified DEST=MLG on the JES3 CONSOLE initialization statement), records the system log. If you choose MLOG and no such console exists, JES3 assigns the MLG destination class to the first hard copy device defined in the initialization stream.
- For DLOG, the system log is spooled and periodically printed by JES3 output service. By default, the log is printed every 500 lines to output class A. To change these defaults, code the LOGLIM and LOGCLS parameters of member IEASYSnn of the MVS SYS1.PARMLIB data set. You can also print the disk log by entering the MVS WRITELOG command from the global.
- You can select either a hardcopy log or a disk log or both during JES3 initialization. Through the use of *MODIFY,O commands, you can enable or disable either log. However, JES3 rejects any command that would leave your installation without some form of system log.
- If the disk log fails, and a hardcopy log is not currently active, JES3 will enable the first hardcopy console that you defined during initialization.
- You must ensure that the hard copy device on which JES3 prints the system log is fast enough to print at the message rate your installation requires.
- You can specify which MVS commands you want recorded in the system log using the HARDCOPY parameter in the CONSOLxx member of the MVS SYS1.PARMLIB data set. For information about SYS1.PARMLIB see *MVS/Extended Architecture System Programming Library: Initialization and Tuning*.

Defining Message Routing

Your system issues messages for many reasons. For example, MVS and JES3 issue messages to inform you of your system's status, on behalf of jobs that require resources, in response to commands, or to instruct you to perform some type of action.

MVS and JES3 together determine where and how messages are routed and presented in your installation. You can use many facilities to control message traffic including:

- JES3 initialization statements
- Statements in members of the MVS SYS1.PARMLIB data set
- The MVS message processing facility (MPF)
- Automation packages, such as NetView
- Macros used to issue messages
- JES3 and MVS user exits
- JES3 and MVS commands.

This chapter provides information about how to control message routing using JES3 initialization statements. Overviews of non-JES3 facilities such as MPF and automation packages, are provided to introduce you to those topics. You should refer to the appropriate book for detailed information about using those facilities. The following books also contain information about message routing:

JES3 Conversion Notebook
JES3 User Modifications and Macros
JES3 Commands
JES3 Logic Library Volume 7: Complex Management
MVS Initialization and Tuning

To control message routing in a JES3/MVS environment, you must first understand the following basic concepts:

- Where and how messages originate
- Where messages can go
- The general path of messages
- JES3 destination classes
- MVS routing codes
- Message routing exceptions

The following sections describe the tasks of controlling message traffic using JES3 initialization statements, message retention facilities, the MVS message processing facility, and automation packages, such as the IBM NetView program product.

Where and How Messages Originate

JES3, MVS, and application programs can originate messages on both global and local processors. You can also issue messages from user exits and user-written dynamic support programs using the following macros:

- The JES3 MESSAGE macro
- The MVS WTO or WTOR macro.

The JES3 MESSAGE Macro: Most JES3 messages are issued using the MESSAGE macro. You can specify many routing and display options using this macro, including the following characteristics:

- Routing
- Logging
- Retention
- Presentation
- Deletion

The MESSAGE macro always converts messages into WTOs or WTORs before message routing begins. The following sections generically refer to messages issued by a WTO or WTOR macro as WTOs. Refer to *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros* for additional information about using the JES3 MESSAGE macro.

The MVS WTO and WTOR Macros: Both JES3 and MVS can use the MVS write-to-operator (WTO) or write-to-operator-with-reply (WTOR) macros to issue messages.

You can specify information on the WTO and WTOR macro that is similar to that on the MESSAGE macro. In addition, you can also specify whether subsystems can or cannot modify the message's original routing information. Refer to *MVS/Extended Architecture System Programming Library: System Macros and Facilities* for information about using the WTO or WTOR macro.

Where Messages Can Go

MVS and JES3 present messages in many places. Notice that the term *present* is used rather than *display*, because message destinations can be internal, such as disk logs and automation packages in addition to, or instead of external destinations, such as operator display consoles and hardcopy logs. Depending on the JES3/MVS routing algorithms and the routing decisions you make, the system may or may not present messages to:

- JES3 and/or MCS consoles
- The system log (MLOG or DLOG)
- The MVS message processing facility (MPF)
- The MVS action message retention facility
- The JES3 action message retention facility
- Automation packages, such as NetView

Understanding the General Path of a Message

A message can pass through many functions of MVS and/or JES3 along the route to its final destinations. Many of these functions can add, change, or delete a message's original routing and presentation characteristics. Figures Figure 5-8 and Figure 5-9, illustrate the general path of a message. This figure includes optional functions such as the MVS message processing facility (MPF), user exits and automation packages. There are special exceptions to this path that are discussed in the following sections:

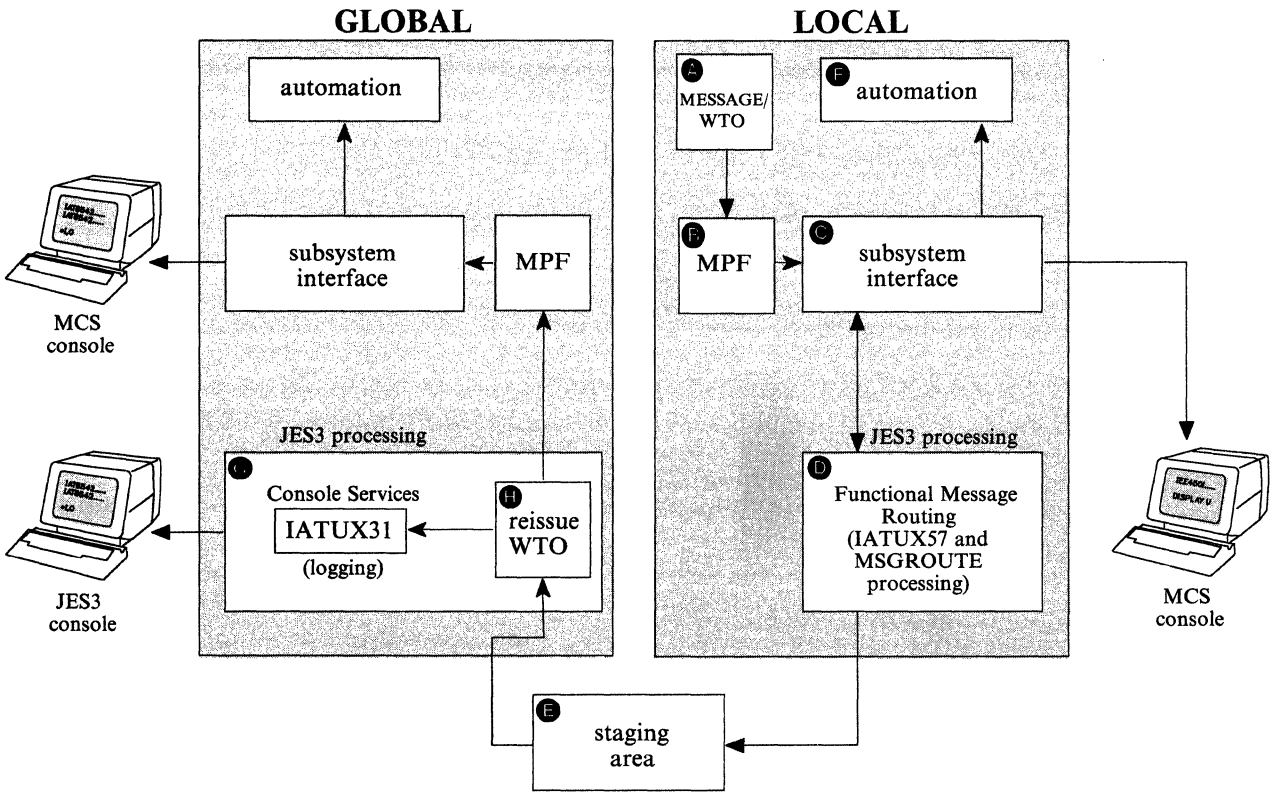


Figure 5-8. Simplified Path of a Message Issued from a Local Processor

1. When a message originates (A) from a local processor, MPF (B) can alter the message's routing and presentation characteristics.
2. Next, MVS places the message on the subsystem interface (SSI). The SSI (C) is the portion of the system on which other subsystems, such as JES3, can access and alter the path of a message.
3. While on the SSI, JES3 (IATUX57 and MSGROUTE processing specifically) can modify the message's original routing information (D) if the issuer allows subsystem modification (using SUBSMOD= keyword of the WTO macro). The sections "Routing MVS Messages to Consoles" describes this portion of message processing in greater detail.
4. Depending upon the results of JES3's processing, the message can be packaged in a staging area and sent to the global (E) for further processing. The message is also placed back on the SSI for processing by an automation package (F) if present, or for display on local MCS consoles.
5. The staging area is sent to the global where JES3 reissues the message (G) using a WTO macro and sends it to both JES3 and MCS for final processing.

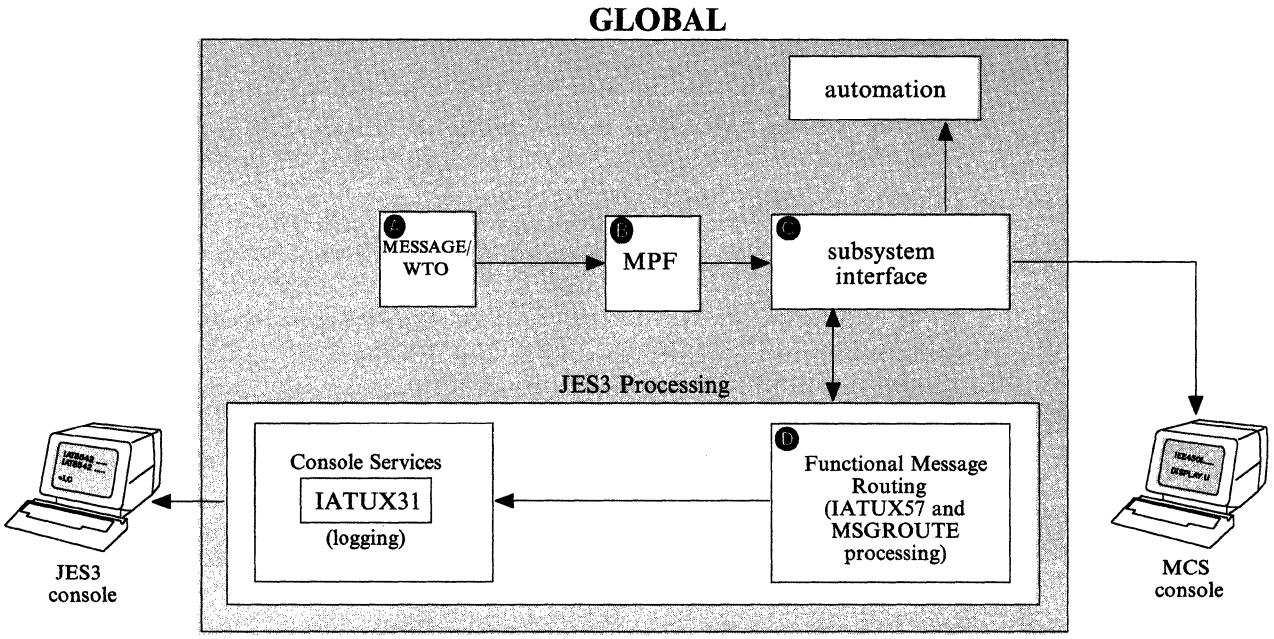


Figure 5-9. Simplified Path of a Message Issued from a Global Processor

Path of a Message Issued from a Global Processor

Messages that originate on a global processor travel a path similar to that of a message issued on a local processor except that the message is not reissued as a WTO on another processor.

1. When a message originates Ⓐ on the global processor, MPF Ⓑ can alter the message's routing and presentation characteristics.
2. Next, MVS places the message on the subsystem interface (SSI). The SSI Ⓒ is the portion of the system on which other subsystems, such as JES3, can access and alter the path of a message.
3. While on the SSI, JES3 (IATUX57 and MSGROUTE processing specifically) can modify the message's original routing information Ⓓ if the issuer allows subsystem modification (using SUBSMOD= keyword of the WTO macro). The sections "Routing MVS Messages to Consoles" describes this portion of message processing in greater detail.
4. The message is then sent to both MCS consoles and JES3 consoles attached to the global.

Refer to *MVS/Extended Architecture JES3 Logic Library Volume 7: Complex Management Logic* for additional information about message processing.

JES3 Destination Classes and MVS Routing Codes

JES3 uses 95 destination classes to route messages to JES3 consoles. You can define which messages you want displayed on a JES3 console by specifying one or more destination classes on the DEST= keyword of the JES3 CONSOLE initialization statement.

MVS uses 128 routing codes to route messages to MCS consoles. You can define which messages an MCS console displays by specifying routing codes on the ROUTCODE keyword of the CONSOLE statement in the CONSOLxx member of the MVS SYS1.PARMLIB data set. Each destination class corresponds to an MVS routing code. However, no JES3 equivalent exists for routing codes 1, 2, 4, 5, 6, and 11 to 40. Figure 5-10 shows the 95 JES3 destination classed and their corresponding MVS routing codes.

JES3 Destination Class	Equivalent MVS Routing Code/Function	Destination Class Purpose
All	Broadcast	Messages intended for MCS consoles on the global that display broadcast messages and all JES3 consoles except those whose CONSOLE statement specifies DEST=NONE.
ERR	10	Equipment failure and JES3 failsoft and problem messages.
JES	42	General information about JES3.
LOG	41	General information about jobs.
MLG	Hardcopy	All input and output messages.
SEC	9	All security messages.
TAP	3	Messages about JES3-controlled tape requirements.
TP	8	Messages about teleprocessing.
UR	7	Messages about JES3-controlled unit-record equipment.
DALL or D1-D22	43-64	Messages about a user-defined console configuration. The exact JES3 destination class-route code mappings are: D1=43 D4=46 D7=49 D10=52 D13=55 D16=58 D19=61 D22=64 D2=44 D5=47 D8=50 D11=53 D14=56 D17=59 D20=62 D3=45 D6=48 D9=51 D12=54 D15=57 D18=60 D21=63
MALL or M1-M32	65-96	Messages unique to a JES3 main. Use the MDEST parameter of the MAINPROC statement to define the destination class for messages about specific mains. The exact JES3 destination class-route code mappings are: M1=65 M5=69 M9= 73 M13=77 M17=81 M21=85 M25=89 M29=93 M2=66 M6=70 M10=74 M14=78 M18=82 M22=86 M26=90 M30=94 M3=67 M7=71 M11=75 M15=79 M19=83 M23=87 M27=91 M31=95 M4=68 M8=72 M12=76 M16=80 M20=84 M24=88 M28=92 M32=96
SALL or S1-S32	97-128	Messages pertaining to JES3 device setup. Use the XUNIT or JUNIT keywords of the DEVICE statement to define the consoles to receive device related messages. The exact JES3 destination class-route code mappings are: S1=97 S5=101 S9= 105 S13=109 S17=113 S21=117 S25=121 S29=125 S2=98 S6=102 S10=106 S14=110 S18=114 S22=118 S26=122 S30=126 S3=99 S7=103 S11=107 S15=111 S19=115 S23=119 S27=123 S31=127 S4=100 S8=104 S12=108 S16=112 S20=116 S24=120 S28=124 S32=128

Figure 5-10. Valid Destination Classes and their Corresponding Routing Codes

The following destination classes are also valid, but are not considered part of the 95 JES3 classes:

- *NONE:* No messages
- *OUTPUT:* All messages except MLG messages
- *TOTAL:* All messages.

You can control the routing of MVS-issued messages by mapping MVS routing codes to JES3 destination classes using the JES3 MSGROUTE initialization statement. You cannot control the routing of most JES3-issued messages (that is, those messages issued with the IATxxxx prefix) even though JES3 messages travel through most of the message path as WTOs. The following sections explain how to control the routing of messages.

Two Types of Messages

You can group messages into one of two basic categories when controlling the routing of messages in an MVS/JES3 environment, regardless of the originator:

- Messages whose routing can be changed (subsystem-modifiable).
- Messages whose routing cannot be changed.

Most JES3-issued messages fall into the non-modifiable category. The routing information of most MVS-issued messages can be modified. The following sections describe how to control the routing characteristics of these messages.

Routing JES3 Messages to Consoles

You cannot change the routing of most JES3-issued messages, (that is, messages that begin with the IATxxxx prefix) because **most JES3 messages prohibit subsystems from modifying their original routing information**. For these messages, the original destination class specified on the JES3 MESSAGE macro is converted to its MVS equivalent routing code when it becomes a WTO or WTOR. MCS consoles on both global and local processors can display the message if you have defined them to receive that equivalent routing code. JES3 converts that code back to its original destination class for display on JES3 consoles.

If you issue a message with multiple routing codes during JES3 initialization, JES3 ignores the destination class you specify and uses only the specified routing codes to route messages to MCS consoles. After initialization, JES3 ignores the routing codes and uses only the destination class to route messages.

If you issue a message using the WTO/WTOR macro that prohibits subsystem modification, JES3 selects a single routing codes using the following algorithm:

1. MCS consoles on the global display the message using the original set of routing codes you assigned to the message on the WTO/WTOR macro.

2. JES3 selects the highest routing code (128 being the highest) that has an equivalent JES3 destination class.

3. If none of the original routing codes correspond to a JES3 destination class, JES3 routes the message using the destination class you specify on the MDEST= keyword of the JES3 MAINPROC initialization statement.

Refer to *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros* for information about using macros to issue messages.

Routing MVS Messages to Consoles

Selecting a Single Routing Code

If an MVS-issued message has multiple routing codes (excluding MVS routing code 11), JES3 invokes user exit IATUX57 to select a single code so that JES3 can translate that code to a single destination class for display on JES3 and MCS consoles attached to the global. If you omit IATUX57 and a message contains multiple routing codes, JES3:

1. Discards routing code 11 if present.
2. Selects the highest routing code below 16 if one exists or,
3. Selects the lowest routing code between 17 an 128.

For additional information about user exit IATUX57 and the default selection algorithm see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Using MSGROUTE to Control Message Routing

You can use the JES3 MSGROUTE initialization statement to route messages:

- directly to JES3 consoles or MCS-managed consoles with associations
- to only MCS consoles attached to the local processor
- to only JES3 and MCS consoles attached to the global
- to JES3 consoles, as well as local and global MCS consoles
- to only the system log

The routing instructions you place on the MSGROUTE statement **affect only those messages that allow subsystems to modify their routing information**, such as most MVS-issued messages. MSGROUTE provides two methods to route MVS messages to JES3:

1. You can map an MVS routing code directly to a console name.
2. You can map an MVS routing code to a JES3 destination class.

For example if you want all MVS security messages (routing code 9) displayed on JES3 consoles that receive JES3 security messages, you could use the MSGROUTE statement to map routing code 9 to destination class SEC. Afterwards, all JES3 consoles that display destination class SEC messages would also display MVS security messages.

You can define one MSGROUTE statement for each processor in your installation. The following example shows the use of the MSGROUTE statement and the various processing that occurs:

```
MSGROUTE,MAIN=SY1,1=(M1,CN1,J),2=(M28,CN1),3=(S1,,J),8=(TP),9=(,,J)
```

Note: If you omit one or more routing codes from a MSGROUTE statement, those messages are sent only to the system log and to the originating processor's MCS consoles.

Processing if SY1 is the Local:

- Messages assigned routing code 1 will be displayed on console CN1 and all JES3 consoles defined to receive M1 messages. MCS consoles attached to the global display those messages only if you have defined them to display the routing code **equivalent** of M1 (which is routing code 65). The J parameter prevents these messages from being displayed on MCS consoles attached to the originating processor.

- Messages assigned routing code 2 will be displayed on console CN1 and all JES3 consoles defined to receive M28 messages. MCS consoles on the global processor display those messages if you have defined them to display the routing code **equivalent** of M28 (which is routing code 92). MCS consoles on the originating processor are also eligible to display these messages using the message's original set of routing codes.

- Messages assigned routing code 3 will be displayed on all JES3 consoles defined to receive S1 messages. MCS consoles attached to the global display those messages only if you have defined them to display the routing code **equivalent** of S1 (which is routing code 97). The J parameter prevents these messages from being displayed on MCS consoles attached to the originating processor.

- Messages assigned routing code 8 will be displayed on all JES3 consoles defined to receive teleprocessing (TP) messages and on all MCS consoles attached to the global defined to display the routing code **equivalent** of TP (which is routing code 8). MCS consoles on the originating processor are also eligible to display these messages using the message's original set of routing codes.

- Messages assigned routing code 9 are sent only to the log (MLOG and/or DLOG) if you have defined that routing code on the ROUTCODE keyword of the HARDCOPY statement in the MVS SYS1.PARMLIB data set console defined to display MLG messages. Otherwise, these messages are not displayed or logged.

- Messages assigned to routing codes that you omit from the MSGROUTE statement are available for display on only the originating processor's MCS consoles. The messages are also received by the system log (MLOG and/or DLOG) if you've defined those codes on the ROUTCODE keyword of the HARDCOPY statement in the MVS SYS1.PARMLIB data set.

Processing if SY1 is the Global:

- Messages assigned routing code 1 will be displayed on console CN1 and all JES3 consoles defined to receive LOG messages. MCS consoles attached to the global display those messages only if you have defined them to display the routing code **equivalent** of LOG (which is 41).

- Messages assigned routing code 2 will be displayed on console CN1 and all JES3 consoles defined to receive LOG messages. MCS consoles on the global processor display those messages if you have defined them to display either the routing code **equivalent** of LOG (which is 41), or one of the message's original routing code(s).

- Messages assigned routing code 3 will be displayed on all JES3 consoles defined to receive S1 messages. MCS consoles attached to the global display those messages only if you have defined them to display the routing code **equivalent** of S1 (which is routing code 97).

- Messages assigned routing code 8 will be displayed on all JES3 consoles defined to receive teleprocessing (TP) messages and on MCS consoles attached to the global defined to display either the routing code **equivalent** of TP (which is routing code 8), or one of the message's original routing code(s).

- Messages assigned routing code 9 are sent only to the log (MLOG and/or DLOG) if you have defined that routing code on the ROUTCODE keyword of the HARDCOPY statement in the MVS SYS1.PARMLIB data set and to JES3 consoles defined to display MLG messages. Otherwise, these messages are not displayed or logged.

- Messages assigned to routing codes that you omit from the MSGROUTE statement are displayed only the originating processor's MCS consoles if you've defined them to display at least one of the message's original routing codes. The messages are also received by the log (MLOG and/or DLOG) if you've defined those codes on the ROUTCODE keyword of the HARDCOPY statement in the MVS SYS1.PARMLIB data set.

Coding Rules for the MSGROUTE Statement

- If you omit one or more of the 128 MVS routing codes on the MSGROUTE statement, messages assigned those codes will be sent to the log (MLOG and/or DLOG). These messages are also available to the originating processor's MCS console(s) using the message's original set of routing codes.

- Specify the J parameter to make messages available for display on only JES3 and MCS consoles attached to the global. If you omit the J parameter, messages assigned the specified routing code are eligible for display on consoles attached to the originating processor as well as consoles attached to the global.

- Define a routing code with only the J parameter (do not specify a destination class or a console name) if you want messages with that routing code sent only to MLOG and/or DLOG. However, you must also define that routing code on the ROUTCODE keyword of the HARDCOPY statement in the MVS SYS1.PARMLIB data set if you want the log to receive those messages.

Usage Notes for the MSGROUTE Statement:

- MSGROUTE processing occurs only on the processor that originates the message. For example, if a local processor originates a message, that message will undergo MSGROUTE processing only on the local. MSGROUTE processing does not occur on the global if the message is sent there.

- MSGROUTE does not affect messages that prohibit subsystem modification (such as JES3 IATxxxx messages).

Message Routing Exceptions

There are several types of messages that undergo special message routing. For example, messages issued from functional subsystems or device-related messages are subject to message routing in addition to, or other than the routing you define on the JES3 MSGROUTE initialization statement. The following sections explain each of these special message types:

Action Messages That Must Be Displayed

The master MCS console attached to the global displays all action messages that no other console (JES3 or MCS) displays. For example, if MVS issues an action message with a routing code that you haven't defined to any console, then the master console attached to the global displays that message.

If an action message originates on a local processor, the MCS consoles attached to the originating processor can also display the message unless you specify the J parameter for that routing code on that processor's MSGROUTE statement.

Hardcopy Only Messages

You can specify that a message be sent only to the hardcopy log. For example, if a message is eligible for subsystem modification, you can use the MVS message processing facility (MPF), a user exit, or the macro used to issue the message to suppress the message display. If you specify hardcopy only, the system sends the message only to the hardcopy log. No additional message routing is performed.

Deleted Messages

You can use the MVS IEAVMXIT to prevent messages from being displayed or logged. These messages however, travel the entire message path and are presented to internal functions such as automation packages (if installed). These messages are subject to a subset of message routing but are not available on the global for display or logging. This type of message is often referred to a ***deleted message***. These messages are also written to the JES3 JESMSG data set. No additional message routing is performed.

Routing and Descriptor Codes Absent

If a message is issued without routing information, or you use a facility such as MPF to remove all routing information, and you specify DEFAULT=NONE on the CONSOLE keyword in the CONSOLxx member of SYS1.PARMLIB, the message is displayed only on the issuing processor's MCS console and on any JES3 consoles that receive messages about that processor. No additional message routing is performed.

Messages That Originate From Functional Subsystems

Use the JUNIT= keyword of the JES3 DEVICE initialization statement to assign a destination class to messages that originate from an output writer functional subsystem (FSS). The destination class that you specify overrides the destination class selected by other message processing (such as the JES3 MSGROUTE initialization statement for modifiable messages, and default processing for non-modifiable messages).

If you define a destination class for messages that originate from an output writer functional subsystem, the following rules apply:

- JES3 consoles attached to the global display messages based on the destination class that you specify on the JUNIT= keyword of the DEVICE statement for the FSS.
- MCS consoles attached to the global display messages based on the equivalent routing code of that destination class.
- MCS consoles attached to the originating processor display messages based on the message's original set of routing codes.

JES3 routes all messages concerning an output writer FSS to the console from which you called the FSS into execution, unless the message is a command response, which is routed to the console from which you enter the command.

Broadcast Messages

Messages assigned the MVS *broadcast* function are available for display on MCS consoles attached to the local and global and all JES3 consoles except those that specify DEST=NONE on their JES3 CONSOLE initialization statement.

Suppressed Messages

You can use the MVS message processing facility (MPF) to suppress the display of messages. These messages are routed only to the system log (MLOG and/or DLOG) and to consoles that receive the MLG destination class or its equivalent routing code. Refer to "Suppressing the Display of Messages" for information about using MPF.

Messages That Specify Only an MVS Routing Function

Messages that specify only a routing function, such as *monitor jobnames*, and a descriptor code other than 8 or 9 are displayed on consoles attached to the global using the destination class that you specify on the MDEST= keyword of the JES3 MAINPROC initialization statement and its equivalent routing code. Consoles attached to the originating processor also display those messages if you have defined them to display the routing function assigned to the message.

Job Status Messages Without Routing Codes

Messages about the status of a job (descriptor code 6) are assigned the destination class you specify on the MDEST= keyword of the JES3 MAINPROC initialization statement for the main on which the job executes. The message is also sent to a specific console if the original message specifies one.

Special Messages

Printer Setup Messages: Messages about local JES3 printers which require a response are automatically responded to by JES3 and then written to the system log. Otherwise, the message is treated as a device-related message which is described below.

Device-Related Messages: The destination class you specify on a JES3 DEVICE initialization statement overrides all other message routing for messages about the device you are defining. If you do not specify a destination class, the routing selected during MSGROUTE processing is used to route the message.

Job Termination Messages: Messages about failed jobs (message IEF402I specifically), are sent only to the system log and to any console defined to display messages assigned the MLG destination class or its equivalent MVS routing code. The original routing of messages about command failures is not affected.

Main Processor Messages: Messages about a specific main are sent to the destination class you specify on the MDEST= keyword of the JES3 MAINPROC initialization statement. If you do not specify a destination class, JES3 routes messages to consoles receiving destination class M1 or its equivalent MVS routing code.

RJP Messages: Messages about RJP-submitted jobs that have started, ended, or failed are sent to the remote console from which the job was submitted if the message does not specify a console.

Diagnosing Misrouted Message Traffic

You can use the MVS generalized trace facility (GTF) to diagnose message routing problems. Refer to *MVS/Extended Architecture: JES3 Diagnosis* for information about invoking and using GTF in your JES3 installation. You can also refer to *MVS/Extended Architecture System Programming Library: Service Aids* for general information about GTF.

Retaining Action Messages

JES3 retains both JES3 and MVS action messages in the JES3 action message retention queue.

MVS also retains JES3 and MVS action messages in its own action message retention facility. The following rules apply when using message retention:

- To retrieve action messages from the JES3 action message retention queue, use the JES3 *INQUIRY,R command. To retrieve action messages from the MVS action message retention queue, use the MVS DISPLAY R command.

- To delete action messages from only the JES3 action message retention queue, use the *MODIFY,O,MAIN= command. To delete action messages from both the JES3 and MVS action message retention queues, use the CONTROL C,A command. See *MVS/Extended Architecture Operations: JES3 Commands* and/or *MVS/Extended Architecture Operations: System Commands* for information about using these commands.

- JES3 retains all messages with descriptor codes 1, 2, and write-to-operator-with-reply (WTOR) messages in its action message retention queue. JES3 does not retain messages with descriptor codes 3 and 11. If you want JES3 to retain those messages, you must code JES3 user exit IATUX31. You can also use IATUX31 to override retention decisions by other facilities, such as MPF. For a description of IATUX31 see *MVS/Extended Architecture System Programming Library:User Modifications and Macros.*

- MVS retains all messages with descriptor codes 1,2,3,11, and messages issued using the WTOR macro. See *MVS/Extended Architecture Message Library: Routing and Descriptor Codes* for a complete list of descriptor codes and their meanings.

- You can use the MVS message processing facility (MPF) to prevent specific messages from being retained by either action message retention facility. See *MVS/Extended Architecture System Programming Library: Initialization and Tuning* for information about how to prevent messages from being retained.

Suppressing the Display of Messages

To reduce the number of messages an operator must read, you can suppress the display of specific messages.

There are three ways you can suppress message displays:

- Message processing facility (MPF)
- MVS user exit IEAVMXIT
- JES3 user exit IATUX31 (JES3 consoles only)

Using MPF, you can suppress message displays from any processor in your JES3 installation:

- You can prevent messages from being displayed on any console in your installation by using MPF on the processor from which the messages originate.

- You can use MPF on the global to suppress messages that originate from a local processor. However, you can only suppress those messages from display on MCS consoles attached to the global. JES3 consoles continue to display these messages.

JES3 records each message that you suppress in the system log (MLOG and/or DLOG). JES3 flags these messages to help you identify them.

You can use the MVS user exit IEAVMXIT to suppress messages. This user exit can run on any or all processors in the installation, and can be used to change or suppress message displays.

You can also use the JES3 user exit IATUX31; however, IATUX31 does not receive the message until after processing by user exit IEAVMXIT. This means that a message could be deleted or changed before the JES3 user exit receives it. For information about the functions and use of user exit IATUX31, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Specifying Messages for Display Suppression

First, determine the messages that you want to suppress when they are issued from the global. Specify the message ids of these messages in a MPFLSTxx member of the global processor's SYS1.PARMLIB data set. You can also specify messages that originate from a local processor. However, these messages can be suppressed only from display on MCS consoles attached to the global. You cannot use MPF on the global to suppress from JES3 consoles those messages that originate on a local. Next, determine the messages that you want to suppress for each local processor. Specify their message ids in a MPFLSTxx member of the local processor's SYS1.PARMLIB data set.

To specify a message in a MPFLSTxx member, use from 1 to 8 characters of the message identifier. This allows you to identify specific messages or groups of messages. To specify a message, use the complete message identifier (for example, IAT2000). To specify a group of messages, use one or more contiguous characters of the message identifiers starting with the leftmost character followed by an asterisk (*). The asterisk identifies the id as a partial id. For example, to specify all messages that begin with IAT0, you would specify IAT0*. For more information on how to specify message identifiers in the MVS SYS1.PARMLIB data set, see *MVS/Extended Architecture System Programming Library: Initialization and Tuning.*

You may want to suppress the display of a different set of messages at different times. In this case, specify each set of message identifiers in different MPFLSTxx members. You can then enter an MVS SET command to change the active MPFLSTxx member.

In each MPFLSTxx member, you can specify the flag that the system is to use for flagging suppressed messages in the hardcopy log. By using a different flag for each MPFLSTxx member, you can determine which processor issued the suppressed messages. For instructions on how to specify a flag, see *MVS/Extended Architecture System Programming Library: Initialization and Tuning.* If you do not specify a flag, the system uses an ampersand sign (&) to flag the messages.

Controlling Message Suppression from a Local Processor

To control the message processing facility from a local processor, issue the MVS SET MPF=xx command, from an MCS console attached to that local processor. Thereafter, when that local processor issues a message that you specified in MPFLSTxx, MPF suppresses the message display.

To terminate control of message suppression from the local processor, enter the MVS SET MPF=NO command from an MCS console attached to that local processor.

Controlling Message Suppression from the Global Processor

To suppress messages from the global, issue the MVS SET MPF=xx command from a console attached to the global. Thereafter, when the global issues a message that you have specified in MPFLSTxx, MPF suppresses the message display on all consoles.

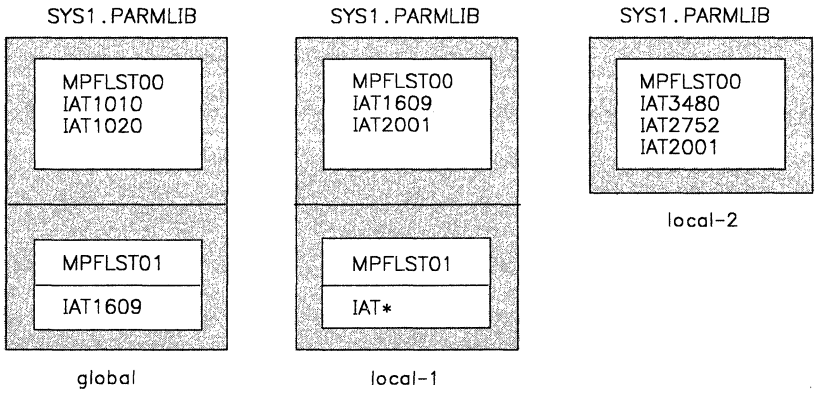
When a local processor issues a message that you have specified for suppression using only MPF on the global, MPF suppresses the message display on only MCS consoles attached to the global. MCS consoles attached to the originating processor and JES3 consoles continue to display the message unless you use MPF on the originating processor to suppress the message.

To terminate control of message suppression from the global, enter the MVS SET MPF=NO command from a console attached to the global. (The system will then display all messages except those that a local processor is suppressing.)

Example of Message Suppression

These examples show how the contents of the MPFLSTxx member(s) of the MVS SYS1.PARMLIB data set and the SET MPF commands being issued on different processors at different times affect message suppression.

The examples use a JES3 installation consisting of a global processor (Global) and two local processors (local-1 and local-2). Each processor has one or more MPFLSTxx members in its MVS SYS1.PARMLIB data set:



In the following examples, the name in parentheses before the MVS SET MPF= commands identifies the processor that issued the command. MPF on all processors is initially set to MPF=NO.

Example 1: Controlling message suppression from a local processor.

(local-1)	SET MPF=00	• When local-1 issues message IAT1609 or message IAT2001, the system suppresses the message display. The system displays all other messages issued by local-1. • The system displays all messages issued by local-2 or the global.
(local-1)	SET MPF=NO	• The system now displays all messages.

Example 2: Simultaneously controlling message suppression from the global processor and from a local processor.

(local-1)	SET MPF=01	• The system suppresses the display of all JES3 messages (IAT*) issued by local-1. The system displays all messages issued by local-2 or the global.
(global)	SET MPF=01	• When the global issues message IAT1609, the system suppresses the message display. However, when local-2 issues message IAT1609, MPF suppresses the display from only the MCS consoles attached to the global. The system displays all other messages issued by the global or by local-2. • The system continues to suppress the display of all JES3 messages issued by local-1.
(local-1)	SET MPF=NO	• When the global issues message IAT1609, the system suppresses the message display. However, when local-1 or local-2 issues message IAT1609, the message can be displayed by MCS consoles attached to the local and on JES3 consoles. MPF on the global suppresses the display of message IAT1609 from only the MCS consoles attached to the global. The system displays all other messages.
(global)	SET MPF=NO	• The system now displays all messages issued by all processors.

Example 3: Suppressing the display of a different set of messages at different times.

(local-1)	SET MPF=01	• The system suppresses the display of all JES3 messages (IAT*) issued by local-1. • The system displays all messages issued by the global or by local-2.
(local-1)	SET MPF=NO	• The system now displays all messages.
(local-1)	SET MPF=00	• When local-1 issues messages IAT1609 or IAT2001, the system suppresses the message display. The system displays all other messages issued by local-1. • The system displays all messages issued by the global or by local-2.

Using MPF to Screen Messages for Automation

Use MPF to specify which messages you want eligible for processing by an automation package, such as the IBM NetView program product. You can improve the performance of your automated operations and prevent bottlenecks by sending to your automation program only those messages that you've identified for automation.

Use the AUTO parameter on the MPF ***msgid***, **.DEFAULT**, or **.NO_ENTRY** control statements to control which messages you want sent to your automation subsystem. For specific information and examples about using these control statements, see *MVS/Extended Architecture System Programming Library: Initialization and Tuning*. The contents of an MPFLSTxx member that controls which messages are sent to an automation program might be:

```
.NO_ENTRY,AUTO(NO)
.DEFAULT,AUTO(YES)
IAT9191
IAT1600
IAT2000,AUTO(NO)
IAT2002
```

In the above example, messages IAT9191, IAT1600, and IAT2002 will be suppressed from console display and sent to the automation subsystem. MPF will suppress IAT2000 but not send it to the automation subsystem. MPF will prevent all other messages from being sent to the automation subsystem.

The use of the AUTO parameter does not affect whether or not messages are suppressed from console displays. You must specify message suppression independently of the automation option.

Automating Message Processing

You can automate the processing of messages in your JES3 installation by installing an automation product, such as the IBM NetView program product. You can use NetView to automatically respond to action or informational messages issued by JES3, MVS or application programs. You can also suppress the display of unsolicited messages that you select for automation. Unsolicited messages are all messages except those that a program issues in response to commands.

Using NetView to Automatically Respond to Messages

When you install NetView on the JES3 global, you can pre-define lists of commands (CLISTs) that NetView uses to automatically respond to messages issued by JES3, MVS, or application programs. For information about how to define NetView CLISTs see *NetView Installation and Administration Guide*.

NetView examines each message to determine whether or not it can issue a CLIST in response to that message. However, NetView issues automatic responses only for those messages that you define to NetView. NetView displays messages for which you have not defined a CLIST on its own terminal, as well as on other consoles defined to display those messages.

You can set up NetView to automatically issue all JES3 commands except console-specific commands (such as *ERASE, *DELAY, and *FREE).

You need install NetView only on the global to process all messages issued by JES3, MVS and application programs since JES3 routes all messages, global as well as local, through the JES3 global (except those messages that you delete from a local processor using MVS user exit IEAVMXIT). However, you can also install NetView on a local main to provide automation capabilities from a local if you so desire.

NetView installed on a local processor can issue only the following subset of JES3 commands in response to MVS or application messages:

- *CALL, *START, and *CANCEL,DSI
- *CALL, *START, and *CANCEL,VARYL
- *DUMP
- *RETURN

How to Reduce Message Traffic Using NetView and MPF

Use the MVS Message Processing Facility (MPF) to suppress the display messages that you have defined to NetView for automation. Refer to "Suppressing the Display of Messages" on page 5-33 or see *MVS/Extended Architecture System Programming Library Initialization and Tuning* for information using MPF.

How to Define NetView to JES3

Before you can use NetView on the global to send commands (MVS or a subset of JES3 commands) to a local main, you must first create a logical association between NetView and each main in your installation. This logical association establishes a communication path between NetView and the local main(s). To create an association, specify TYPE=MCS in conjunction with the UNIT= keyword on the console's JES3 CONSOLE initialization statement. You must specify all DUMMY or NONE values on the UNIT= keyword. For example, if you have three mains named SY1, SY2, and SY3, and add the following initialization statement:

```
CONSOLE,JNAME=MCSAUTO1,TYPE=MCS,UNIT=(SY1,DUMMY,SY2,DUMMY,SY3,NONE)
```

then NetView could send MVS commands or a subset of JES3 commands to SY1 and SY2, but not SY3.

Coding Rules:

- You must code DUMMY for the first main that you specify on the UNIT= keyword.
- If you do not include a JES3 CONSOLE initialization statement with all DUMMY or NONE values on the UNIT= keyword and NetView attempts to send a command to a local main, JES3 issues error message IAT7117 that notifies the operator that no communication path exists.
- You can also specify an authority level using the LEVEL= keyword when defining a console for automation. However, it is recommended that you specify the highest JES3 authority level (LEVEL=15) so that your automation package can issue all JES3 commands. If you omit the LEVEL= keyword, JES3 automatically assigns authority level 15 to that console.

Chapter 6. Defining and Managing JES3 Resources

You are responsible for allocating and defining resources for JES3. Resources include data sets, libraries, catalogs, devices, processors, volumes, and main storage. To allocate and define resources use JES3 initialization statements and the JES3 cataloged procedure.

For a discussion on defining and managing processors and storage, see Chapter 7, "Defining and Managing JES3 Mains and Storage."

JES3 Data Sets

Before starting JES3, you must allocate space for the JES3 job library and for the spool data sets. Optionally, you can define spool partitions and specify how JES3 is to use these partitions. Chapter 4, "Defining and Managing Spool Data Sets" explains the many considerations for defining and managing spool data sets.

You can improve JES3 performance by identifying data sets that are resident on all systems. By knowing where the data sets are located, JES3 will not have to search for them. You can also improve performance by the way you configure system catalogs.

JES3 data sets should be marked as unmovable (DSORG=PSU on the DCB parameter of the DD statement). JES3 data sets are physical, sequential, and contain location-dependent information.

Allocating JES3 Data Sets

There are two ways to allocate most JES3 data sets: include a DD statement for the data set in the JES3 cataloged procedure or dynamically allocate the data set by including a DYNALLOC statement in the JES3 initialization stream. You should ensure that a dynamically allocated data set is accessible to all processors in the JES3 complex. If a dynamically allocated data set is unavailable to one or more processors, operator intervention will be required during JES3 initialization. You cannot dynamically allocate dummy data sets, data sets cataloged in a private catalog, data sets required for a C/I FSS address space, or data sets defined by the ddnames:

- JES3LIB
- STEPLIB
- CHKPNT or CHKPNT2
- JES3IN

If you dynamically allocate a data set, do not include a DD statement for that data set. JES3 assigns a disposition of SHARE to dynamically allocated data sets.

Allocating the JES3 Checkpoint Data Set(s)

The JES3 checkpoint data set(s) let(s) you warm start or hot start the JES3 system with little or no loss of system information. The checkpoint data sets contain many data areas described in "JES3 Checkpoint Data Set(s)," in Chapter 11, "JES3 Recovery."

One or both JES3 checkpoint data sets must be defined in the JES3 start procedure using ddnames of CHKPNT and/or CHKPNT2. (See Chapter 10, "JES3 Start-Up and Initialization.") If your installation defines any C/I FSS address spaces, define the JES3 checkpoint data set(s) in the C/I FSS start procedure exactly as the JES3 start procedure defines them, except substitute DISP=SHR for DISP=OLD. (To see the IBM-supplied C/I FSS start procedure, see "Defining a C/I FSS Address Space," in Chapter 3, "Defining and Managing C/I Service.") You cannot dynamically allocate the checkpoint data sets. You must allocate and catalog one or both checkpoint data sets before JES3 operation.

At least one of the checkpoint data sets must be available to JES3 during initialization processing. You can add or replace either checkpoint data set over a JES3 restart of any kind with no effect on JES3 processing. Both checkpoint data sets contain identical information; to ensure against loss of checkpointed data, allocate both data sets.

Note: If you allocate only one checkpoint data set and it develops a severe permanent I/O error, you must perform a cold start. If you allocate both checkpoint data sets and one develops a severe permanent I/O error, JES3 can continue. For recovery procedures, see "Recovering from Permanent Errors on the JES3 Checkpoint Data Sets," in Chapter 11, "JES3 Recovery."

Determining the Size and Placement of the Checkpoint Data Set(s)

Each checkpoint data set must be allocated as a single extent which begins and ends on a cylinder boundary. The size of each checkpoint data set should be at least two cylinders on a direct access storage device. To determine how much space your installation's checkpoint data set(s) require, consider the following factors:

- Each checkpoint record type begins on a track boundary. Each track contains a 128-byte track header record.
- The checkpoint data set track map, the complex status record, the initialization checkpoint record, and the JESCKPNT checkpoint record each need one track.
- The dynamic allocation checkpoint record requires 44 bytes plus an additional 92 bytes for each DYNALLOC and JES3LIB initialization statement.

- The spool volumes checkpoint record requires 64 bytes plus an additional 80 bytes for each TRACK and FORMAT initialization statement. Add additional bytes as reserve for spool expansion.

- The spool partition checkpoint record requires 64 bytes plus an additional 96 bytes for each SPART initialization statement.

- The partition TAT checkpoint record space requirement is calculated using a complex algorithm involving many different factors. Allow about 512 bytes for each spool data set.

- The BADTRACK checkpoint record requires 44 bytes plus an additional 64 bytes for each BADTRACK initialization statement. Every entry in the BADTRACK checkpoint record requires an additional 64 bytes. Thus, the size of this data area varies with the number of tracks having I/O errors at one time.

- Allocate enough free space to permit growth in the complex without reallocating the checkpoint data sets.

If either checkpoint data set runs out of space, the data set must be replaced. Recalculate the amount of space the checkpoint data set needs and allocate a new checkpoint data set that is larger than the old one.

To minimize the effect of the loss of any one DASD volume, control unit, or channel path, allocate the checkpoint data sets on different volumes, channel paths, and control units. The volumes may be different device types.

Dynamically Allocating the JES3 Step Library

Use JES3LIB initialization statements to dynamically allocate from 1 to 16 private JES3 step libraries. These libraries replace libraries allocated by a STEPLIB DD statement in the JES3 or C/I FSS procedures.

JES3 uses the ddname JES3LIB to dynamically allocate the data set specified on the first JES3LIB statement. Thereafter, JES3 uses a ddname of the form JS3Dxxxx to dynamically allocate data sets specified on additional JES3LIB statements. The suffix xxxx is a four-digit decimal number and has a different value for each data set. You cannot use a ddname of the form JS3Dxxxx in the JES3 procedure because JES3 uses it.

Determining the Size of the JCT Data Set

The JES3JCT data set contains the job control table (JCT). JES3 determines the maximum number of jobs that can be in the system at the same time by choosing the smallest of the following:

- the number of entries in the JCT

- the range of job numbers

- the value specified for the third subparameter of the JOBNO parameter on the OPTIONS initialization statement

If the data set is larger than required, JES3 uses only that portion needed to hold the maximum number of jobs. However, the larger the JES3JCT data set, the longer JES3 initialization takes. For optimum performance when restarting JES3, the size of the JCT data set should be minimized as much as possible.

Note: The JCT data set should be allocated on a cylinder boundary.

The SE parameter on the OPTIONS initialization statement specifies the maximum number of scheduler elements for a job. The third subparameter of the JOBNO parameter on the OPTIONS initialization statement specifies the maximum number of jobs that may be in the JES3 complex at any given time.

Use the following algorithm to calculate the size of a JCT control block:

```
record size = (4 x max SEs) + 348 + the size of the SRF
              header prefix
```

Where:

- the value of max SEs is specified in the SE parameter on the OPTIONS initialization statement,
- 348 bytes is a fixed size, and
- the size of the SRF header prefix is mapped using the IATYSRF macro with the PREFIX parameter.

JES3 always reserves up to 64 JCT records in the JES3JCT data set for write error recovery.

Identifying Resident Data Sets

During JES3 initialization you can identify data sets that are resident on all systems. Thereafter, when the names of these data sets appear as catalog references on a DD statement for a job, JES3 will bypass LOCATE processing. Thus, you improve JES3 performance. To specify the names of the resident data sets, use the RESDSN initialization statement.

Using System Catalogs

You can eliminate the need for a job to execute on a specific processor to satisfy a catalog dependency by sharing catalogs among all processors. To ensure that JES3 does not assume unnecessary processor dependencies because of a catalog search, use the RESDSN statement to identify catalogs that are resident on all processors. MVS, not JES3, will then process references to these catalogs.

If a job needs a catalog that is not available to all processors, the user must ensure that the job executes on a processor that has access to the catalog. The user can do this by specifying the SYSTEM parameter on the //*MAIN statement or by specifying a job class that belongs to a group that will execute on the appropriate processor.

To prevent a catalog from being altered between the time JES3 accesses it while scheduling the job and the time MVS accesses it while executing the job, the user should use dependent job control (DJC). In this way, the user can ensure that the jobs will access the catalog in the correct sequence. For example, if job A creates a catalog entry required by job B, the user should define job B as being dependent on job A.

If a job references a data set that is cataloged in a control volume catalog (CVOL), the operator must mount the volume that contains the CVOL before JES3 can access the dataset. JES3 fails any job that requires access to an unmounted CVOL. See *MVS/Extended Architecture Catalog Administration Guide* for a complete description of CVOLs and their uses.

To access data in an unmounted VSAM catalog, use a JOBCAT or STEPCAT DD statement. If a catalog request needs an unmounted VSAM user catalog and the user has not provided a JOBCAT or STEPCAT DD statement, JES3 fails the job.

I/O Devices

You must define to JES3 each I/O device that JES3 is to use or manage, except devices used for data sets defined in the JES3 cataloged start procedure or defined using the DYNALLOC initialization statement. If your JES3 complex uses the mass storage system (MSS), you must execute the programs that initialize and define MSS. In addition, if your installation uses any remote terminal packages, you must generate those packages.

Because JES3 varies assignable devices and all execution devices online to MVS as well as to JES3 at initialization (based on the JUNIT or XUNIT parameter values), at least one path to those devices must be online before initializing JES3. Also, you must satisfy all conditions necessary for the MVS VARY command to execute properly before initializing JES3. (For a discussion of assignable devices, see "JES3 Devices" below.)

Defining I/O Devices to JES3

Before JES3 can use an I/O device, the device must be attached to the global processor or to a local processor and you must define it to JES3. To define a device to JES3, code a DEVICE initialization statement.

The DEVICE statement specifies device characteristics and tells JES3 how to use that device. There are three ways to define a device to JES3 using the device statement:

- You can specify that JES3 is to use a device to satisfy JES3 functions, such as DSP requests, input processing, and output processing. This is called a **JES3 device**.
- You can specify that JES3 may allocate a device to MVS to execute user jobs. This is called an **execution device**.

  An execution device that is defined to JES3 and MVS as permanently resident or reserved is called a **jointly-managed device**.

- You can specify that JES3 can use the device as either a JES3 device or as an execution device. This is called a **shared device**.

JES3 cannot use restricted devices. Therefore, do not define a restricted device on a JES3 DEVICE initialization statement. For information about restricted devices see the MVS IODEVICE macro in *MVS/Extended Architecture MVS Configuration Program Guide and Reference.*

JES3 Devices

To define a JES3 device, specify the DTYPE, JNAME, and JUNIT parameters on the DEVICE statement. If applicable, you should also specify the DGROUP parameter and any printer or punch parameters.

JES3 devices must be attached to the global. The only exception is a device driven by an output writer functional subsystem (FSS), which you can define as a JES3 device and attach to the global or a local main.

Use unique device numbers for JES3 devices when you execute the MVS configuration program. When the same device is attached to more than one main, the device should have the same device number on each main. On the other hand, if different JES3 devices attached to different mains have the same device number, JES3 considers the device number "ambiguous" because JES3 cannot always determine which device is being requested. Therefore, JES3 may reject commands that refer to the devices by device number. Commands for those devices must refer to them using that you specify on the JNAME parameter of the DEVICE statement.

You can define as JES3 devices:

- Consoles
- CTC adapters
- Network lines
- Printers
- Card punches
- Card readers
- Remote terminals
- Tape drives

An "assignable device" is a JES3 device such as the 3480 tape drive that JES3 can reserve using the MVS assign/unassign facility. When an assignable device is connected to more than one processor, the assign/unassign facility lets the operator use the *VARY command to reserve the device for one or more processors within a JES3 complex. However, you must use the JES3 VARYL dynamic support program to unassign an IBM 3480 tape drive when all of the following conditions exist:

- The global has failed
- You want to use the IBM 3480 to take a stand-alone dump
- The IBM 3480 is assigned to more than one main

For additional information about using the VARYL dynamic support program see "Recovering an IBM 3480 Tape Drive for a Stand-Alone Dump" or see *MVS/Extended Architecture Operations: JES3 Commands.*

Execution Devices

To define an execution device, specify the XTYPE and XUNIT parameters on the DEVICE statement. When any execution device (except a MSS virtual unit or staging drive group) is initialized, JES3 varies the device online or offline to MVS as well as to JES3 based on the XUNIT parameter specification. You can attach execution devices to the global processor or to any local processor.

Shared Devices

To define a device as a shared device, specify the DTYPE, JNAME, JUNIT, XTYPE, and XUNIT parameters on the DEVICE statement for that device. If applicable, you may also specify the DGROUP parameter and any printer or punch parameters.

Defining Process Modes

Defining device process modes allows you to specify what types of data sets you can schedule to that device. If a user specifies a data set process mode (specified on the OUTPUT JCL statement) that matches one of the process modes defined for a device, then JES3 can schedule that data set to the device. To define process modes for a device, specify the PM= keyword on the DEVICE initialization statement. You can specify as many as eight process modes per device. Unless you modify the list of process modes or the operating mode of the device using operator commands, the process modes that you define in your initialization stream remain in effect.

You can also define alternate process modes for some devices.[1] These process modes become effective only when you change the device operating mode (using an operator command) from what was originally defined (or defaulted to) on the MODE= keyword of the DEVICE statement. In other words, the you can switch between the initial set of process modes and an alternate set.[2] To define alternate process modes for a device, specify the ALTPM= keyword on the DEVICE statement. You can specify as many as eight alternate process modes for a device.

Two values frequently defined on the PM= and ALTPM= keywords are *LINE* and *PAGE*. Specifying PM=LINE indicates that data sets assigned a process mode of 'line' in their JCL can be scheduled to this device. Under normal circumstances, these *are* line mode data sets, meaning that they are formatted line by line. Specifying PM=PAGE indicates that data sets assigned a process mode of 'page' in their JCL can be scheduled to this device. Again, these data sets normally *are* page mode data sets, meaning those formatted as a composed page.

Note: If PM=(PAGE,...) is specified for a device that does not support composed page formatting, or 'all-points-addressable' function as it is sometimes

---

1 You can define alternate process modes for certain devices, such as the 3800 Model 3 printer, that may run under the control of an output writer in the JES3 global address space or under the control of an output writer functional subsystem.

2 You can switch between sets of process modes using the *MODIFY,FSS,D= command.

called, that device will select and attempt to print page mode data sets with unpredictable results. Line mode data is printed normally.

Using Process Mode Defaults

Device modes and process modes are interdependent. If you change a printer's device mode, its process mode must be compatible (for example, you cannot print PAGE mode data when operating your printer in COMP mode).

You can code the ALTPM= keyword on the DEVICE initialization statement to define which process mode(s) take effect when you change a printer's device mode.

If you do not define alternate process modes and you change a printer's device mode (from MODE=COMP to MODE=FSS or the reverse), JES3 uses default values shown on the following figures.

Initial process mode value	Default process mode(s) that JES3 uses if you switch to FSS mode	Process mode(s) that JES3 uses if you switch back to COMP mode
LINE	(LINE,PAGE)	LINE
PAGE	(LINE,PAGE)	PAGE
(LINE,PAGE)	(LINE,PAGE)	(LINE,PAGE)

Figure 6-1. Default Value(s) for ALTPM= when MODE=COMP

Initial process mode value	Default process mode(s) that JES3 uses if you switch to COMP mode	Process mode(s) that JES3 uses if you switch back to FSS mode
LINE	LINE	LINE
PAGE	LINE	PAGE
(LINE,PAGE)	LINE	(LINE,PAGE)

Figure 6-2. Default Values for ALTPM= When MODE=FSS

Running a Printer Under an Output Writer Functional Subsystem

Most printers run under the control of a writer DSP that operates entirely in the JES3 global address space. Certain printers, however, can run under routines that operate in a non-JES3 address space called an output writer functional subsystem (FSS). The routines are device-specific and let the printer use functions that the JES3 global address space does not support. If you want one of these printers to run under the control of an output writer FSS, you must specify certain parameters on the DEVICE initialization statement for the printer. You can also define characteristics of the output writer FSS address space using the FSSDEF initialization statement. Otherwise, JES3 creates an output writer FSS address space to control the printer using default values.

The following subsections describe how to define the 3800 model 3 printer to run under the control of an output writer FSS and how to define an output writer FSS.

Defining the 3800 Model 3 Printer

The 3800 model 3 printer may be run like any other 3800 printer, under the control of a writer DSP in the JES3 global address space. Running the printer in this way is known as running it in *compatibility mode*. A printer running in compatibility mode formats data line by line.

To use all the functions of the 3800 model 3, however, the printer must run under the control of an output writer FSS. The output writer FSS allows the printer to process data containing control characters that format the data as a composed page. Running the printer in this way is known as running it in *FSS mode*.

To define the 3800 model 3 to run in compatibility mode, specify MODE=COMP on the DEVICE initialization statement and specify any other parameters relevant to defining a 3800 printer. If you want to switch the printer to run in FSS mode later and control which output writer FSS the printer will run under, specify the FSSNAME parameter on the DEVICE statement.

To define the printer to run in FSS mode, define the printer on the DEVICE initialization statement as a shared device and specify the following other parameters:

- Use the FSSNAME parameter to specify the installation-defined name of the FSS under which the printer is to run. The name of the FSS is defined using the FSSDEF initialization statement, which the next section explains. If you choose to let JES3 define the output writer FSS under which the printer is to run, let the FSSNAME parameter default.
- Specify the MODE=FSS parameter.
- Use the PM (processing mode) parameter to specify what types of data sets you want the printer to handle. Specifying PM=PAGE allows data sets with a process mode of 'page' to be printed; specifying PM=(PAGE,LINE) allows data sets with process modes of both 'page' and 'line' to be printed. The default allows the printer to handle both types.

You should also specify any other parameters on the DEVICE statement that apply to defining a 3800 printer.

You can change the 3800 model 3 processing mode if the printer is offline and not currently allocated. Use the *F,F operator command with the M= operand (see *MVS/Extended Architecture Operations: JES3 Commands*).

Defining the 3820 Printer

The 3820 printer runs only under the control of an output writer FSS. To define the 3820 printer to JES3, specify DTYPE=PRT3820 and MODE=FSS on the DEVICE initialization statement. Use the FSSNAME= parameter to specify the name of the functional subsystem that is to be associated with the 3820 printer.

Because the 3820 printer is not directly channel-attached to the processor, do not code an address on the JUNIT= parameter; instead, code a comma since this parameter is positional. Also, do not code the XUNIT= and XTYPE= parameters.

Although the 3820 printer is not channel-attached to processors, the controlling functional subsystems may still require specific resources, such as VTAM, that are available only on certain processors. Therefore, carefully consider where functional subsystems are to execute when coding the SYSTEM= parameter on the FSSDEF initialization statement.

Defining an Output Writer Functional Subsystem

As stated above, the 3820 printer or the 3800 model 3 printer running in FSS mode must run under the control of an output writer FSS. Each output writer FSS handles one printer at a time and operates in its own address space. An output writer FSS may operate on the JES3 global processor or any JES3 local processor, as defined on the FSSDEF initialization statement. The output writer FSS, in turn, runs under the control of a FSS writer DSP operating in the JES3 global address space.

Attach the printer to be run in FSS mode to the processor on which the output writer FSS operates. The data processed by the printer may originate from any processor within your JES3 complex or at any node in a job entry network.

Before you can run a printer under the control of an output writer FSS, you must include the cataloged procedure for starting the output writer FSS in SYS1.PROCLIB. The name of the IBM default procedure for the 3800-3 printer is APSWPROC; for the 3820 printer, the default procedure is APSWPROS. To learn about this procedure and how to change it, see *Print Services Facility/MVS System Programmer's Guide.*

Parameters that you define on JES3 initialization statements override the corresponding parameters in the catalogued procedure.

You can define an output writer FSS for each printer eligible to be run in FSS mode or you can let JES3 define the output writer FSS by default. If JES3 defines the FSS, its name will be the same as the name specified for the JNAME parameter on the DEVICE statement for the printer. In this case, you can change the default FSS's characteristics using the *MODIFY command, but you cannot specify its initial characteristics.

If you would rather define the output writer FSS for each printer yourself, include a FSSDEF statement in your initialization stream.

To change the definition of an output writer FSS, use the *F,F command. You cannot start, restart, or cancel the output writer FSS directly. Instead, you may call, start, restart, or cancel the FSS writer DSP that supervises it, and the DSP will carry out the desired function. Use the same operator commands as for any output writer (for example, *X,WTR), specifying the 3820 or 3800 model 3 printer as the device.

To allocate an I/O device that JES3 needs during initialization, include a DD statement for the device in the JES3 start procedure. To dynamically allocate the device, omit the DD statement and include a DYNALLOC statement for the device in the JES3 initialization stream.

Before you dynamically allocate a device, however, there are two restrictions you should be aware of:

- You cannot specify unit affinity for a dynamically allocated device
- You cannot use the ddname of a dynamically allocated device to assign unit affinity to a device defined in the JES3 start procedure

Defining the Mass Storage System (MSS)

Initialize the JES3 mass storage system (MSS) support in two steps. In the first step, execute the MSS Table Build program. This program receives input from the Mass Storage Control (MSC) Table Create program. The MSS Table Build program produces the data set defined by the JSMSSTAB DD statement. The second step is either a JES3 cold start or warm start.

Before doing a warm start, allow all jobs that are active on the main scheduler element to complete execution. To determine which jobs these are, use the command *I,B,M. If you do not let these jobs complete execution, some of them might fail when they try to execute after the warm start. The jobs that will fail are:

- Jobs that the converter/interpreter processed before the warm start.
- Jobs that request virtual volumes that are not mounted or that are mounted on JES3-managed virtual units.

The MSS Table Build program is implemented as an independent program. It is not a part of JES3 initialization. The MSS Table Build program is described next; for more information on the MSC Table Create program, see *OS/VS Mass Storage Control Table Create.*

MSS Table Build Program

The MSS Table Build program is executed following the execution of the MSC Table Create program. Execute these programs before JES3 initialization to ensure that the correct device and unit-name configurations will be known to JES3.

The MSS Table Build program produces the data set referred to by the JSMSSTAB DD statement. This data set provides input to JES3 cold-start and warm-start initialization. If the MSS configuration changes physically, then you must execute the MSC Table Create and the MSS Table Build programs again. However, if the MSS configuration to JES3 changed logically, you need execute only the MSS Table Build program.

An installation may have multiple data sets to be used with different MVS configurations. However, all JES3 local and global processors will use the same data set during any cold start or warm start processing.

Functional Characteristics: The MSC Table Create program produces output statements (called J-UNITNAME statements) which, combined with user-supplied statements, are the input file to the MSS Table Build program. Using the J-UNITNAME statements and user-supplied statements, the MSS Table Build program produces tables. These tables are written to the data set defined by the JSMSSTAB DD statement. These tables then become input to JES3 cold-start and warm-start initialization processing.

Job Control Statements. The MSS Table Build program requires six job control statements:

- JOB
- EXEC
- SYSPRINT DD
- JSMSSTAB DD
- JSMSSWRK DD
- JSMSSUNT DD

These statements are described in Figure 6-3.

Statement	Use	Example
JOB	Initiates the job.	name JOB
EXEC	Specifies the program name.	//EXEC PGM=IATSATAB
SYSPRINT DD	Defines a sequential message data set.	//SYSPRINT DD SYSOUT=A

JSMSSTAB DD	Defines a sequential output data set to contain the newly constructed MSS tables. This data set must be allocated to a direct access device which is shared by all of the JES3 processors in the complex during JES3 cold-start and warm-start initialization.
	
	Note: *This data set should be cataloged for access by JES3 initialization.*
		//JSMSSTAB DD DSN=dsname,
		// UNIT=3330,
		// VOL=SER=volume-serial,
		// DISP=(NEW,CATLG),
		// SPACE=(TRK,(5,1),RLSE),
		// DCB=(RECFM=U)

JSMSSWRK DD	Defines a temporary work data set used by the MSS Table Build program.	//JSMSSWRK DD UNIT=3330, // SPACE=(TRK,5)

JSMSSUNT DD	Defines a sequential 80-column input data set containing the statement produced by the MSC Table Create program, the MSS statements, the DEFSYS statements, the user UNITNAME statements, and the HOSTEXCL statements. This data set can be an input stream data set (DD *) or reside on any appropriate device that can be allocated to MSS TABLE Build program.	//DSMSSUNT DD *,
		// DCB=BLKSIZE=
		
		Note: *See the following section on MSS definition statements for a description of the input stream that must follow the JSMSSUNT statement.*

Figure 6-3. The MSS Table Build Program Job Control Statements

MSS Definition Statements. There are five statement types that you may use when defining an MSS configuration: (1) MSS, (2) J-UNITNAME, (3) DEFSYS, (4) user-UNITNAME, and (5) HOSTEXCL. This collection of statements defines a single MSS configuration, while multiple MSS configurations are defined by repeating this collection of statements. Following is a sample input stream of a multiple configuration:

```
label       MSS         NAME=mssname
JCPU0xyy    UNITNAME    NAME=SDGyy,UNIT=(ddd [[,ddd]
                           ...[,(ddd,n)]...]),
                        CPUID=cpuid
                .
                .
                .
label       DEFSYS      SYS=sysid
label       UNITNAME    NAME=name,UNIT=(ddd)
                .
                .
                .
label       HOSTEXCL    NAME=(name)
                .
                .
                .
label       MSS         NAME=mssname
JCPU0xyy    UNITNAME    NAME=SDGyy,UNIT=(ddd),
                        CPUID=cpuid
                .
                .
                .
```

MSS Statements. The first statement in this input stream must be an MSS statement. If it is not, JES3 prints an error message. The MSS statement begins the definition of an MSS configuration and invokes the routine which performs basic initialization for the MSS Table Build program. The format for the MSS statement is:

```
[label]  MSS  NAME=mssname
```

label

Specifies 1 to 8 alphameric characters, the first of which must be alphabetic. It must begin in column one and have one or more blanks following it. It is an optional field.

NAME =mssname

A unique name from 1 to 8 alphameric characters, the first of which must be alphabetic. This name will be used to identify this MSS definition.

J-UNITNAME Statements. The J-UNITNAME statements are produced by the MSC Table Create program. They identify each virtual unit in the MSS configuration with its processor serial number and staging drive group. J-UNITNAME statements should follow the MSS statement for an MSS definition. If more than four different processor serial numbers are encountered in the J-UNITNAME statements for any MSS configuration definition, a message

will be written and the MSS tables will not be written to the output data set. The maximum number of different names in the NAME field of the J-UNITNAME statement for a single MSS is 28. The format for this statement is:

```
label  UNITNAME NAME=SDGyy
       ,UNIT=( {ddd    } [,ddd    ]  [,...])
               {(ddd,n)}  [,(ddd,n)]
       ,CPUID=cpuid
```

label

Specifies 8 alphameric characters in the form JCPU0xyy. J identifies this statement as a J-UNITNAME statement; CPU0x specifies the logical CPU identifier (this number is not used for any processing); and yy specifies the staging drive group (range 00-27). You must specify this label.

NAME =SDGyy

Specifies the staging drive group (range 00-27), where yy must be identical to the last 2 characters in the label.

UNIT =(ddd)

Specifies the device number(s) of the virtual unit(s) to be associated with the name assigned.

(ddd,n)

Specifies the lowest device number of a group of sequential numbers being specified. n is the number of sequential device numbers being assigned, the maximum being 255. If the form (ddd,n) is used as the only subparameter, double parentheses must be used. For example, UNIT=((180,5)) would create a group of devices that have the numbers 180, 181, 182, 183, and 184.

CPUID =cpuid

Specifies a 10-digit processor identifier (a 6-digit processor serial number, followed by a 4-digit model number).

Note: For a model number with 3 digits, specify 0xxx (for example, Model 168AP would be 0168).

DEFSYS Statements. A DEFSYS statement identifies the beginning of the user-supplied input. This input is defined by and begins with a DEFSYS statement followed by user-UNITNAME and/or HOSTEXCL statements. If neither user-UNITNAME or HOSTEXCL statements are used for this MSS definition, then the DEFSYS statement is not needed. The format for this statement is:

```
[label]  DEFSYS  SYS=sysid
```

label

Specifies 1 to 8 alphameric characters, the first of which must be alphabetic. It must begin in the first column and have one or more blanks following it. This is an optional field.

sysid

Specifies the 10-character processor identifier to be associated with the user-UNITNAME statements. It must match at least one of the processor identifiers specified in the CPUID parameter on the J-UNITNAME statements for this definition.

User-UNITNAME Statements. The user-UNITNAME statements let JES3 manage user group names on a unit count basis with dynamic association of 3330V units to unitnames. JES3 will treat these devices as class 3 devices. This statement is optional. However, if used, it must be preceded by a DEFSYS statement. The user-UNITNAME statements are grouped by processor serial numbers. The format for this statement is:

```
[label] UNITNAME NAME=SDGyy
        ,UNIT=( {ddd     } [,ddd     ]     [,...])
                {(ddd,n) }  [,(ddd,n) ]
```

label

Specifies 1 to 8 alphameric characters, the first of which must begin in column one and have one or more blanks following it. It is an optional field.

NAME =name

Specifies a 1- to 8-character name which must be different than any name previously specified on a J-UNITNAME statement for this definition.

UNIT

Specifies the device number of the MSS device. This number must have been previously specified on the J-UNITNAME statement with the same processor serial number as specified in the DEFSYS statement. The number of unique names specified on the J-UNITNAME statement and user-UNITNAME statement cannot exceed 128.

HOSTEXCL Statements. The HOSTEXCL statement may be optionally submitted. If used, it requests that the names it references be treated as host exclusive. It must be preceded, however, by one or more user-UNITNAME statements and the UNITNAME statement it references. The format is:

```
[label]  HOSTEXCL  NAME=(name[,name])
```

label

Specifies 1 to 8 alphameric characters, the first of which must be alphabetic. It must begin in column one and have one or more blanks following it. This is an optional field.

NAME =name

Specifies a 1- to 8-character name which must match a name specified on a previous UNITNAME statement.

Following is an example of an MSS Table Build program job stream.

```
(1) //SAMPLE      JOB          ......
(2) //MSTBLD      EXEC         PGM=IATSATAB
(3) //SYSPRINT    DD           SYSOUT=A
(4) //JSMSSWRK    DD           UNIT=3330,SPACE=(TRK,5)
(5) //JSMSSTAB    DD           DSN=dataset-name,DISP=(NEW,CATALG),
                                UNIT=3330,VOL=SER=volume-serial,
                                SPACE=(TRK,5)
(6) //JSMSSUNT    DD           *
(7) MSSCARD1      MSS          NAME=PRODMSS
(8) *             CPU00 refers to CPUID 0202413084
(9) *             CPU01 refers to CPUID 1202413084
(10) *            CPU02 refers to CPUID 0217423033
(11) *            CPU03 refers to CPUID 0201503081
(12) JCPU0200     UNITNAME     NAME=SDG00,UNIT=((E8E,20)),
                               CPUID=0217423033
(13) JCPU0201     UNITNAME     NAME=SDG01,UNIT=((EA2,30)),
                               CPUID=0217423033
(14) JCPU0300     UNITNAME     NAME=SDG00,UNIT=(((6CE,20)),
                               CPUID=0201503081
(15) JCPU0301     UNITNAME     NAME=SDG01,UNIT=((6E2,30),
                               CPUID=0201503081
(16) JCPU0300     UNITNAME     NAME=SDG00,UNIT=((68E,20)),
                               CPUID=0201503081
(17) JCPU0301     UNITNAME     NAME=SDG01,UNIT=((6A2,30)),
                               CPUID=0201503081
(18) JCPU0000     UNITNAME     NAME=SDG00,UNIT=((68E,20)),
                               CPUID=0202413084
(19) JCPU0001     UNITNAME     NAME=SDG01,UNIT=((6A2,30)),
                               CPUID=0202413084
(20) JCPU0100     UNITNAME     NAME=SDG00,UNIT=((6CE,20)),
                               CPUID=1202413084
(21) JCPU0101     UNITNAME     NAME=SDG01,UNIT=((6E2,30)),
                               CPUID=1202413084
(22) JCPU0002     UNITNAME     NAME=SDG02,UNIT=((606,2),(60E,20)),
                               CPUID=0202413084
(23) JCPU0003     UNITNAME     NAME=SDG03,UNIT=((622,30)),
                               CPUID=0202413084
(24) JCPU0302     UNITNAME     NAME=SDG02,UNIT=((606,2),(60E,20)),
                               CPUID=0201503081
(25) JCPU0303     UNITNAME     NAME=SDG03,UNIT=((622,30)),
                               CPUID=0201503081
(26) JCPU0102     UNITNAME     NAME=SDG02,UNIT=((646,2),(64E,20)),
                               CPUID=0202413084
(27) JCPU0103     UNITNAME     NAME=SDG03,UNIT=((662,30)),
                               CPUID=0202413084
(28)              DEFSYS       SYS=0217423033
(29)              UNITNAME     NAME=MSS1,UNIT=(E8E,E90,E93)
(30)              DEFSYS       SYS=0201503081
(31)              UNITNAME     NAME=MSS1EXCL,UNIT=(6A2,6AC)
(32)              DEFSYS       SYS=0202413084
(33)              UNITNAME     NAME=MSS1EXCL,UNIT=(6A2,6AC)
(34)              HOSTEXCL     NAME=MSS1EXCL
(35) /*
```

The statements in this example have been numbered (1-35) for reference purposes:

- Statements 1 through 6 are JCL statements prepared by the user.
- Statement 7 is an MSS statement, which specifies the beginning of an MSS definition.
- Statements 8 through 11 are comments created by the MSC Table Create program. They associate each processor with its processor identifier number.
- Statements 12 through 27 are J-UNITNAME statements. These statements were created by the MSC Table Create program.
- Statements 28, 30, and 32 are DEFSYS statements, which indicate that the user is also supplying UNITNAME statements.
- Statements 29, 31, and 33 are user-UNITNAME statements.
- Statement 34 is a HOSTEXCL statement, which specifies that MSS1EXCL be treated host exclusive. This statement must follow the UNITNAME statement containing the referenced name.
- Statement 35 indicates end-of-data.

In the example, CPUIDs 0202413084 and 1202413084 represent the two sides of a 3084 multiprocessor. If you want to run the 3084 MP in single image mode or partitioned mode and want to have one interface to the MSC, it is necessary to have two Table Build output data sets.

A symmetric device is defined only once for either processor on the JCPU0xyy statements by MSC Table Create. If the two processors were to be defined as uniprocessors, the symmetric devices must appear on JCPU0xyy UNITNAME statements for both processors. (Not all multiprocessors can be configured and defined as uniprocessors. The 3081 processor complex is an example of a multiprocessor that cannot be defined as uniprocessors.) This is accomplished by running the MSC Table Create program defining the two processors as separate processors, which would result in the specification of all defined devices on the J-UNITNAME statements for each processor.

MSS1 will provide implicit processor affinity, and MSS1EXCL will provide processor exclusive mass storage volume usage.

Requesting a Specific Staging Drive Group

To enable a user to select a specific staging drive group (SDG), you must write an exit routine for user exit IATUX04. Without this exit routine, JES3 or MSS selects the SDG. To find out how to write this exit routine, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Mounting or Unloading a Virtual Volume

JES3 does not recognize the MVS mount attribute of **reserved** for virtual volumes. Therefore, if you use a VATLST entry; the MVS MOUNT command or UNLOAD command; or the JES3 *MODIFY,S,M= or *MODIFY,S,U= commands for a reserved virtual volume, follow these rules:

- If you use a VATLST member to assign a virtual volume the reserved attribute, also list the volume on a SETRES initialization statement.
- If you list a volume on a SETRES initialization statement, also use a VATLST member to assign the reserved attribute to the volume.
- If you use the MVS MOUNT command to mount a volume, also use the JES3 command *MODIFY,S,M= to mount the volume.
- If you use the JES3 command *MODIFY,S,M= to mount a volume, also use the MVS MOUNT command to mount the volume.
- If you use the MVS UNLOAD command to unload a volume, also use the JES3 command *MODIFY,S,U= to unload the volume.
- If you use the JES3 command *MODIFY,S,U= to unload a volume, also use the MVS UNLOAD command to unload the volume.

MVS/Extended Architecture System Programming Library: System Modifications describes VATLST entries.

Multiprocessor Considerations

Nonpartitionable Multiprocessors: When both processors of a multiprocessor (that you have defined as a multiprocessor) are connected to the MSS and the virtual I/O unit configuration is symmetric, the MSC Table Create Program creates J-UNITNAME statements for only one CPUID. You may use these statements without change.

When both processors of a multiprocessor (that you have defined as a multiprocessor) are connected to the MSS and the virtual I/O unit configuration is asymmetric, the MSC Table Create Program creates two sets of J-UNITNAME statements. One set contains the CPUID of one side while the other set contains the CPUID of the other side. Before using these statements, change the CPUID on one set of statements (it doesn't matter which set) so they match the CPUID on the other set of statements.

Partitionable Multiprocessors: The CPUID keywords on the J-UNITNAME statements identify which JES3 mains are attached to the MSS. Multiprocessors that cannot be defined as separate physical partitions have unique processor serial numbers, therefore their CPUIDs (serial number plus model number) can be used directly as input to the Table Build program.

To allow partitioning of processor complexes, however, the same serial number is coded on more than one MAINPROC statement. Consequently, CPUIDs used during Table Create may not be unique. To reflect which mains communicate with the MSS, these J-UNITNAME statements must be modified.

Since only one set of J-UNITNAME statements is produced, you must first duplicate this set for the other partition of the processor complex. Then, on the MAINPROC statement for each main connected to the MSS, add a unique, alias CPUID. This CPUID should not actually exist in the complex. Prior to running the Table Build program, substitute this alias CPUID for the one generated by Table Create.

Notes on the Use of MSS

1. If you want JES3 to manage the MSS, JES3 should manage all virtual units on all processors attached to the MSS.

2. If another system that is not part of the JES3 complex (for example, an MVS system with JES2) is also managing the MSS, the systems cannot share a virtual volume.

3. It is recommended that you do not divide the management of virtual units between JES3 and MVS within a single processor. However, if you do divide the management of the virtual units, you must observe the following restrictions:

   - You must define and use two esoteric unit names: one for the set of virtual units managed by JES3 and one for the set of virtual units managed by MVS.
   - You must ensure that each virtual volume is used exclusively on MVS-managed virtual units or exclusively on JES3-managed virtual units, but not on both.
   - You must ensure that JES3 manages all nonspecific volume requests. MVS defaults are used if the UNIT= parameter has not been defined to JES3.
   - You cannot use the catalog for MVS-managed virtual volumes because JES3 will manage all data sets that are on virtual volumes and are retrieved through the catalog.

4. To relinquish JES3-management of the MSS, use a warm start.

Grouping I/O Devices

You can group JES3 devices by using generic and esoteric names, which you define in the following manner:

- Use the IODEVICE system generation macro to describe the characteristics of an I/O device and its system requirements. The UNIT parameter specifies the generic device type (for example, 3330).
- Use the UNITNAME system generation macro to assign an esoteric name to a collection of I/O devices. The NAME parameter specifies the esoteric name to be assigned to the group of devices (for example, SYSDA). The UNIT parameter specifies the device numbers of the devices that will be recognized by the assigned name. In order for a device to be referred to by other than its generic name or its specific number, the device must be specified in the UNITNAME macro instruction.

MVS divides generic device types into subgeneric groups, as illustrated in Figure 6-4. The use of subgeneric groups allows MVS allocation to serialize a subset of units within a generic name. For example, using the figure, if 3330A is requested, MVS allocation needs to serialize only subgeneric group 5 rather than all 3330 devices. 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.

Generic Device Type	3400				3350				3330				
Esoteric Names (Defined by the Installation)					SYSDA				SYSDA				3330A
Device Numbers	131	132	133	134	181	182	183	184	191	192	193	194	195
Subgeneric Groups	1				2	3	2		4		5		

Figure 6-4. Subgeneric Groups

The guidelines by which MVS determines subgeneric groups are:

- If an esoteric name (for example, 3330A) includes only a subset of the units in a generic name, that subset is a subgeneric group. (For 3330A in the figure, units 193, 194, and 195 belong to a subgeneric group.)
- If an esoteric (for example, SYSDA) includes different generic device types, the units in each generic name belong to different subgeneric groups. (For SYSDA in the figure, units 181, 183, and 184 belong to one subgeneric group; units 191 and 192 belong to another subgeneric group; and units 193, 194, and 195 belong to a third group.)
- The units not contained in the intersection of a generic group and its new subgeneric groups constitute a subgeneric group. (For SYSDA and 2314 in the figure, unit 182 comprises a new subgeneric group.)

If one unit of a subgeneric group is defined to JES3, all units of the subgeneric group must be defined to JES3. (For subgeneric group 2 in the figure, if unit 183 is assigned to JES3, units 181 and 184 must also be assigned to JES3.) Thus, a subgeneric group cannot have system-managed devices mixed with JES3-managed and jointly managed devices.

When the devices assigned to an esoteric or generic name are to be managed by JES3, one or more of the subgeneric groups that constitute that name must be defined to JES3 via DEVICE initialization statements. For example, SYSDA in the figure is composed of subgeneric groups 2, 4, and 5. For SYSDA devices to be managed by JES3, at least one of the subgeneric groups, 2, 4, and 5 must be defined to JES3. For SYSDA devices in subgeneric group 4 (191 and 192), or all the devices in subgeneric group 2 (181, 183, and 184), all the devices must be defined to SYSDA on the NAMES parameter of the SETNAME initialization statement. Thus, an esoteric or generic name may comprise JES3-managed, jointly managed, and system-managed devices.

Since a subgeneric group can belong to different generic or esoteric names, it is possible that the subgeneric group could be managed both by JES3 and MVS. For example, 3330A in the figure might be MVS-managed, whereas subgeneric group 5 might be JES3-managed. If a device is being managed both by JES3 and by MVS, MVS does not allocate to any device that does not have a permanently resident volume (jointly managed).

Specifying Device Fencing

You can specify device fencing (sometimes called device pooling) for job-class groups by specifying either the DEVPOOL parameter or the device dedication subparameters in the EXRESC parameter of the GROUP statement. The difference between the two methods of dedication is that devices dedicated via the EXRESC parameter are dedicated when the group is allocated on the processor specified in the EXRESC parameter and DEVPOOL-requested dedication is accomplished when the GROUP is enabled on any processor. See the GROUP initialization statement description in Chapter 12, "Initialization Statement Reference" for more information on the DEVPOOL parameter.

Using an MSS Drive as a Real DASD

You can specify that staging drives be used as real DASDs through the use of the DEVICE and SETNAME initialization statements. All of the existing operands and defaults are applicable on both statements, with the exception that the XUNIT subparameter on the DEVICE statement must specify OFF. This indicates that the initial status of the drive is offline.

Dynamically Reconfiguring I/O Devices

The operator can move a tape volume or direct access volume that is in use from one device to another device without terminating jobs that are using the volume. The operator might want to move a volume, for example, to allow a device to be serviced.

To move a volume, the operator first issues the MVS command, SWAP. With this command the operator specifies two device numbers: (1) the device number where the volume is now mounted and, (2) the device number to which the volume will be moved. A message then tells the operator when to physically move the volume. The SWAP command is described in *MVS/Extended Architecture Operations: System Commands*. This book also lists the devices from which and to which volumes can be moved.

When selecting a device to which a volume is to be moved, the operator must follow these rules:

- A volume on an MVS-managed device may be moved to only another MVS-managed device.
- A volume on a JES3-managed device may be moved to only another JES3-managed device.

- If a job is using a device from which a volume is to be moved, the device to which the volume will be moved must be online to both:
  - The processor on which the job is executing
  - The processor on which JES3 setup the job
- A tape volume may be moved from either a shared or nonshared tape drive to another shared or nonshared tape drive.
- A tape volume may be moved to a tape drive that is allocated to a job that has not started execution.
- A tape volume may be moved across device fence boundaries.
- A direct access volume may be moved from a nonshared drive to another nonshared drive only.

MVS can also request that a volume be moved, providing the operator previously issued the command SWAP ON. MVS makes the request after a permanent I/O error occurs on a device from which a volume can be moved.

When MVS requests that a volume be moved, it also selects the device to which the volume is to be moved. A message then tells the operator which device MVS selected. The operator must approve the MVS selection or select a different device. If the operator selects a different device, the operator must follow the rules mentioned earlier.

Volumes

The VATLSTxx member of SYS1.PARMLIB, an MVS operator command, a JES3 operator command, and three JES3 initialization statements allow you to define and manage volumes in the JES3 complex.

Every volume has mount and use attributes. The mount attribute controls volume demounting while the use attribute helps determine volume allocation. You can specify mount and use attributes via the VATLSTxx member. You can also specify use attributes on the MVS MOUNT command. For more information about the purpose of mount and use attributes and how to assign them, see *MVS/Extended Architecture System Programming Library: System Modifications*.

To identify a permanently resident volume, use the VATLSTxx member or specify XTYPE=(name,DA,PR) on the DEVICE statement that defines the I/O device on which the volume resides. If you use the VATLSTxx member for a volume that is shared among processors, be sure that the VATLSTxx member on each processor specifies the same mount attributes for the volume.

If you reserve a volume for MVS, use the VATLSTxx member to assign the volume the reserved status.

For a JES3-managed direct access storage device, the use attribute is always treated as private, regardless of the use attribute specified on the MOUNT command.

To keep a volume mounted, use the SETRES initialization statement. When you include the SETRES statement in the initialization stream, the operator does not need to issue the MOUNT command for the volume. The operator can also keep a volume mounted on a direct access device by issuing the *MODIFY,S command.

You must identify uninitialized processors in the JES3 complex that have direct-access volumes or MSS volumes that are permanently resident and not shared. This prevents JES3 from setting up a job that needs the volume until the processor is initialized. To identify the processor and the volumes, use the SETACC initialization statement.

If volumes are to be accessed from more than one processor at a time they should be made nonremovable (either JES3-mounted or permanently resident) on a device that is attached to the accessing processors and specified in the XUNIT parameter of the DEVICE statement. For example:

```
DEVICE,XTYPE=(3330,DA,PR),                                    X
XUNIT=(230,SY1,S1,230,SY2,S2)
```

If this is not done, jobs referencing the volume cannot be set up and can remain in the MDS allocation queue.

How Resource Definition Affects JES3 Resource Management

How you define JES3 resources -- whether you provide a DEVICE statement, the information you specify in the VATLSTxx member and so forth -- influences many of the decisions that JES3 makes as it manages its resources. For example, to determine whether JES3 or MVS will manage an I/O device, JES3 looks for the presence of a DEVICE statement and examines certain volume mount attributes of the volume that is mounted on the I/O device.

Resource definition affects:

- I/O device management
- Data set management
- Data set integrity
- Dynamic allocation
- JES3 console management

Either JES3, MVS, or both manage I/O devices in the JES3 complex. Two factors determine how a device will be managed: whether the device is specified on a JES3 DEVICE initialization statement; and whether the device contains permanently resident volumes. Figure 6-5 shows how to determine which system will manage a device.

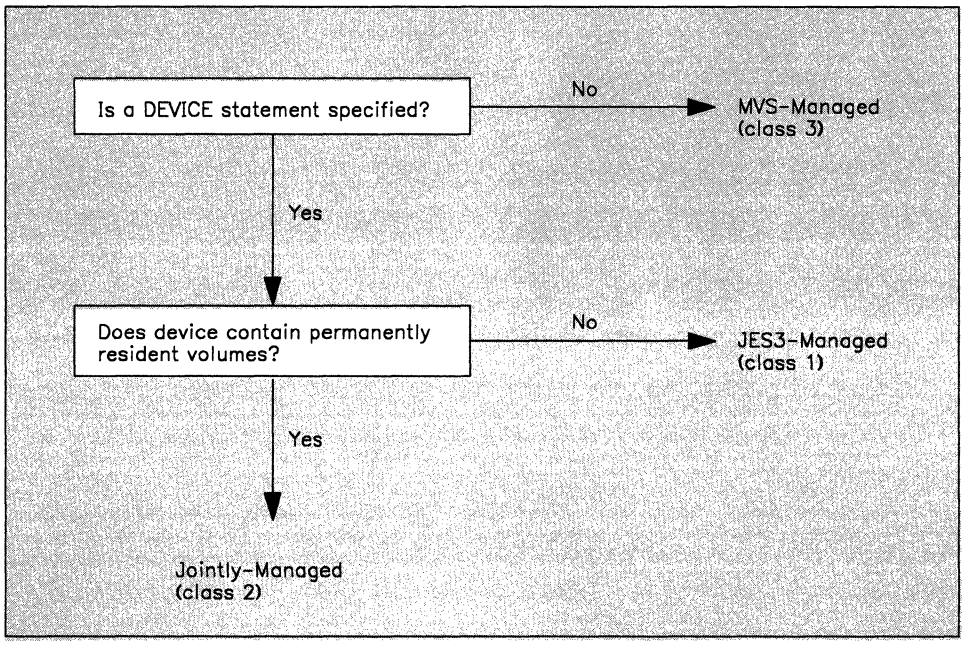


Figure 6-5. JES3-Managed, MVS-Managed, and Jointly Managed Device Determination

JES3 places each device type into one-of-three classes. The class identifies the system that will manage the device.

The three device classes are defined as follows:

- *Class 1 devices:* Tape, unit-record, and graphic devices and removable DASDs are class 1 devices. The removable DASD may still be MDS-mounted or MVS-reserved. MVS does not allocate these devices unless they are MVS-reserved. JES3 manages these devices.

- *Class 2 devices:* Class 2 devices are DASDs defined as permanently resident. These devices are managed by both JES3 and MVS. For specific or private requests, JES3 allocates the device. For nonspecific or nonprivate requests, MVS allocates the device.

- *Class 3 devices:* Class 3 devices are not defined to JES3 -- they have no DEVICE initialization statements in the initialization stream. These devices are managed by MVS only. In order to reference these devices, one or more esoteric or generic names must have been assigned to them in the MVS SYSGEN. However, these names must not be specified in any SETNAME initialization statement.

Data Set Management

To determine whether JES3 or MVS will manage access to a data set, JES3 considers:

- Whether the data set is resident
- Whether a JES3 unit name is associated with the device type on which the data set is mounted
- The type of request

Figure 6-6 shows the decision making process that JES3 uses to determine who will manage access to a data set.

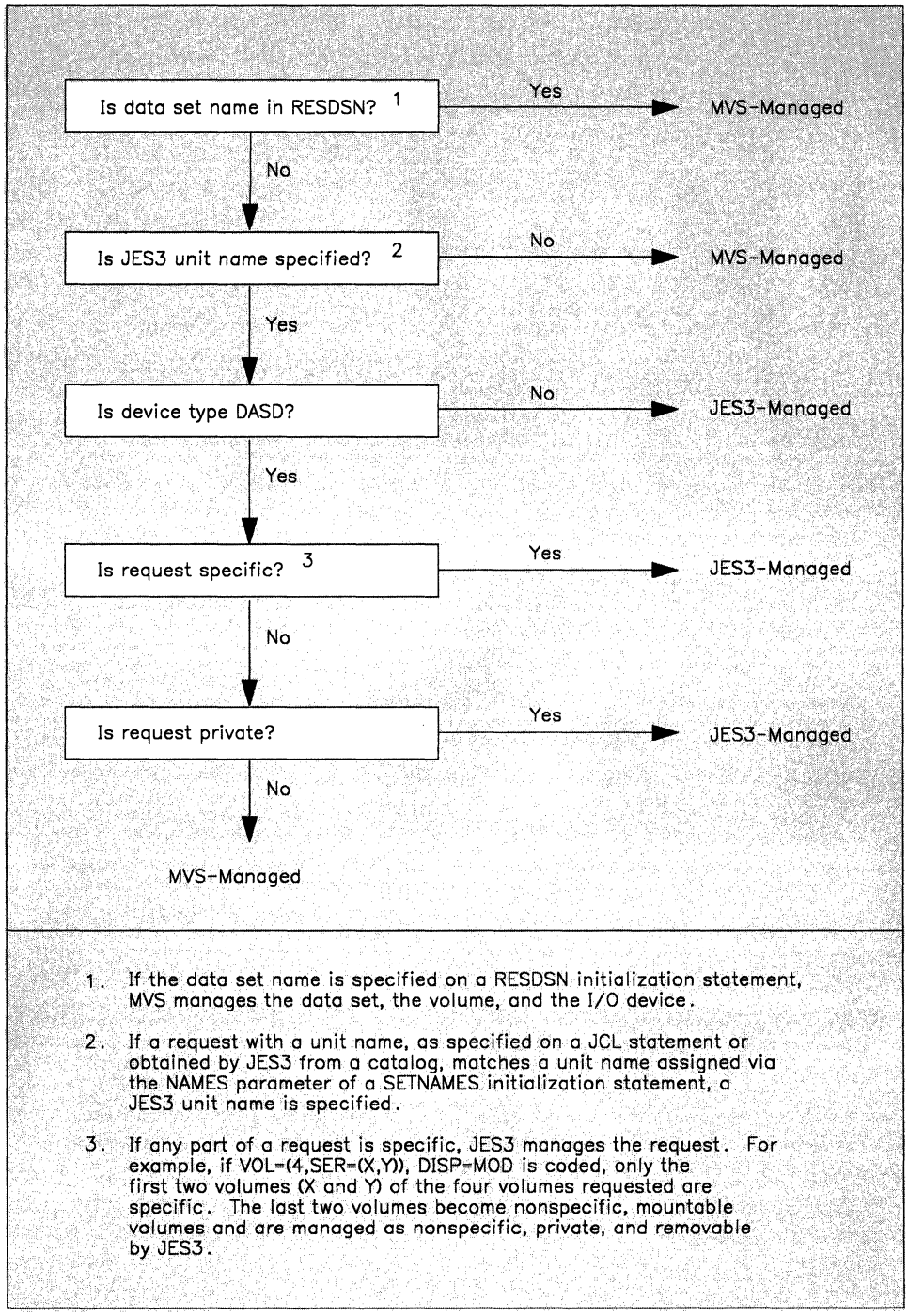


Figure 6-6. JES3-Managed and System-Managed Data Set Determination

Data Set Integrity

In a JES3 complex both MVS and JES3 provide data set integrity. Data set integrity ensures that:

- While one job is reading a data set another job does not change that data set
- Only one job at a time uses a sequentially accessed device that is attached to one or more processors

MVS Data Set Integrity: What it Does

Before allocating data sets to a job, MVS enforces data set integrity for MVS-managed data sets. To ensure data set integrity, MVS does not allocate a data set to a job if:

- The job requests non-shared access to an allocated data set
- The job requests shared access to a data set that is allocated as non-shared

MVS does not begin the allocation process until the integrity of all the data sets in the job has been enforced. Once integrity has been established, MVS then begins allocating the resources needed for the job, one step at a time.

MVS provides integrity for data sets within a single MVS system or across multiple MVS systems. For details about how MVS provides data set integrity, see "Global Resource Serialization" in *MVS/Extended Architecture System Programming Library: System Macros and Facilities.*

JES3 Data Set Integrity: What it Does

JES3 enforces data set integrity for:

- JES3-managed data sets that are requested via DD statements
- JES3-managed data sets that are requested dynamically
- JES3-managed sequentially accessed devices that are attached to one or more processors

If a job requests a data set via a DD statement, JES3 enforces data set integrity before scheduling the job for execution. For dynamically requested data sets, JES3 enforces data set integrity at the time of the request.

To ensure data set integrity, JES3 denies a job's request for a data set if:

- The request is for non-shared access to an allocated data set
- The request is for shared access to a data set that is allocated non-shared

What happens to a job that is denied an allocation request by JES3? This depends on how the job requested the data set. If it was requested via a DD statement, JES3 does not schedule the job for execution at that time. The job must wait until all of the resources it needs are available.

If the data set was dynamically requested, MVS will not allocate the data set to the job at that time. The job, however, can continue to execute.

JES3 also enforces data set integrity for data sets on sequentially accessed devices managed by JES3. Examples of such devices are card readers and magnetic tape drives. To ensure data set integrity, JES3 allows only one job at a time to use such a device.

JES3 does not provide full data set integrity for uncataloged generations of generation data groups. Since these data sets may be referenced validly by two different names, integrity can only be maintained by ensuring that no two jobs running concurrently reference the same generation data group.

JES3 does not provide data set integrity for DASD data sets that are not JES3-managed. For example, requests for new data sets on non-specific, non-private DASD volumes will not be handled by JES3; this type of data set is managed by MVS. See "Data Set Management" earlier in this chapter for more information.

Differences Between MVS and JES3 Integrity Checking

JES3 and MVS both check data set integrity at the data set level. JES3 also checks it at the volume level. This means that JES3 knows which volume a data set is on while MVS does not. This becomes important when there are identically named data sets on different volumes. To understand why this is important, consider the following situation.

Assume there are two identically named data sets on different volumes. Two jobs enter the system. One job requests non-shared access to one of the data sets; the other job requests access (shared or non-shared, it doesn't matter) to the other data set.

JES3, aware that the data sets are on different volumes, can schedule both jobs for allocation. To MVS, however, it looks like both jobs want to access the same physical data set. Therefore, MVS does not allow the jobs to execute on the same processor simultaneously.

Bypassing JES3 Integrity Checking

To bypass JES3 data set integrity checking for dynamically allocated data sets or permanently-resident volumes mounted on JES3-managed devices, code the data set names an a DYNALDSN initialization statement. If you bypass JES3 setup for permanently resident data sets, you also bypass JES3 data set integrity checking for those data sets. This happens for data sets specified on a RESDSN initialization statement. Although you bypass JES3 data set integrity checking, MVS data set integrity checking remains in effect.

The following examples illustrate the effects of data set integrity on job processing. In the examples:

- The data set named DIRECT is a JES3-managed data set.
- The model 3400-3 tape drive is a JES3-managed device.

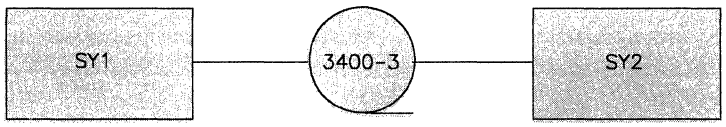
Example 1: Both jobs request data set DIRECT and specify shared access. JES3 will allow both jobs to be scheduled for resource allocation at the same time.

```
//JOBA   JOB .....
//STEP1  EXEC PGM=JOBA
//DD1    DD   DSN=DIRECT,DISP=(SHR,...),...
/*

//JOBB   JOB .....
//STEP1  EXEC PGM=JOBB
//DD1    DD   DSN=DIRECT,DISP=(SHR,...),...
/*
```

Example 2: Both jobs request data set DIRECT. JOBA requests it shared, JOBB requests it non-shared. Therefore, JOBB implies that it will update DIRECT. JES3 will not schedule these jobs at the same time: one job must finish using DIRECT before JES3 will schedule the other job.

```
//JOBA   JOB .....
//*MAIN  SYSTEM=SY1
//STEP1  EXEC PGM=JOBA
//DD1_   DD   DSN=DIRECT,DISP=(SHR,...),...
/*

//JOBB   JOB .....
//*MAIN  SYSTEM=SY2
//STEP1  EXEC PGM=JOBB
//DD1    DD  DSN=DIRECT,DISP=(OLD,...),...
/*
```

Example 3: There is one model 3400-3 tape drive shared between processors SY1 and SY2.



Both jobs need the tape drive. Therefore, JES3 will not schedule these jobs at the same time: one job must finish using the tape drive before JES3 will schedule the other job.

```
//JOBA   JOB .....
//*MAIN  SYSTEM=SY1
//STEP1  EXEC PGM=JOBA
//DD1    DD  UNIT=3400-3
/*

//JOBB   JOB .....
//*MAIN  SYSTEM=SY2
//STEP1  EXEC PGM=JOBB
//DD1    DD  UNIT=3400-3
/*
```

Volume Management

Either JES3 or JES3 and MVS together manage volumes that are mounted on JES3-managed devices. Figure 6-7 shows who manages a volume for different combinations of mount and use attributes.

	Use Attributes		
Mount Attribute	**Public**	**Private**	**Storage**
Permanently resident	Jointly managed; MVS can allocate nonspecific, temporary,nonprivate data sets only.	JES3-managed	Jointly managed; MVS can allocate nonspecific, nonprivate data sets only.
Reserved	Jointly managed	JES3-managed	Jointly managed
Removable	JES3-managed	JES3-managed	Not applicable

Figure 6-7. JES3 Volume Management

After submitting a job for JCL interpretation, JES3 schedules the job for main device scheduling. One of the factors used by JES3 in selecting devices for volume mounting is the ADDRSORT parameter on the SETPARAM initialization statement. This parameter specifies that devices are either allocated in the same order as the DEVICE statement defining them or allocated by the order of their device numbers.

After all the devices are assigned and any initial volumes required are mounted on each of the devices, JES3 makes the job eligible for job selection. Once the job is scheduled on an MVS processor, MVS allocation calls JES3 for each data set request to determine if JES3 has already selected a device for this request. If JES3 has scheduled the devices, MVS allocates them. If JES3 has not scheduled the devices, MVS selects devices; if volume mounting is required, MVS selects devices not controlled by JES3.

Figure 6-8 lists the actions taken by JES3 or by MVS to handle the requests when the volumes containing the data sets are currently located on JES3-managed, MVS-managed, or jointly-managed devices.

	Allocated By:	
Volume*	**Non-JES3 Unit Name**	**JES3 Unit Name**
Not mounted	MVS allocation allocates devices on the processor where the job is scheduled.	JES3 setup is on any processor eligible to process the job.
Resident on MVS managed device	MVS allocation will find device on processor that must be defined by //*MAIN SYSTEM or JOB CLASS.	JES3 setup assumes a volume is not mounted and schedules job on any processor unless //*MAIN SYSTEM or JOB CLASS is specified. JES3 asks the operator to mount; operator must cancel the job.
Mounted on MVS managed device	MVS allocation will find device on processor that must be defined by //*MAIN SYSTEM or JOB CLASS.	JES3 setup assumes a volume is not mounted and schedules job on any processor unless //*MAIN SYSTEM or JOB CLASS is specified. JES3 asks the operator to mount; operator can let a job wait and then satisfy mount or cancel the job.
Resident on jointly managed device	MVS allocation will find device on processor that must be defined by //*MAIN SYSTEM or JOB CLASS.	JES3 setup locates required volume and processor.
Mounted on JES3-managed device	Allocation fails.	JES3 setup locates required volume and processor.
Note: The category "mounted on jointly managed device" is omitted because only permanently resident volumes are contained on jointly managed devices. The category "resident on JES3-managed device" is omitted because permanently resident volumes on JES3-managed devices are defined as jointly managed devices.		

Figure 6-8. JES3 and System Handling of Allocation Requests

If a request is made for a volume on a MVS-managed device using a unit name assigned to JES3, one of two actions can occur:

- If the request is a static allocation request made prior to initiation and does not specify deferred mounting, the job is held until the volume is unloaded and mounted on a JES3-managed device.
- If the request is a dynamic allocation request or a static allocation request that specifies deferred mounting, the job waits in allocation until the volume is unloaded and mounted on a JES3-managed device.

Dynamic Allocation

Dynamic allocation obtains resources as needed. For an explanation on how to use and specify dynamic allocation, refer to *MVS/Extended Architecture System Programming Library: System Macros and Facilities.*

For each invocation of dynamic allocation on a global or local processor, JES3 identifies those dynamic allocations that are a single request; that is, the request has a single SIOT/JFCB pair. The following requirements must also be met:

- Request is for a single unit (equivalent to specifying UNIT=nnnnnnnn or UNIT=(nnnnnnnn,1) on the DD statement)

- Request is for a specific volume (equivalent to specifying VOL=SER=serial number on the DD statement)

- Request is for a volume that is permanently-resident or reserved on a real DASD.

Once JES3 identifies a request that meets the requirements described, it sends the request to the DYNAL DSP.

Those requests which do not meet the requirements are passed to MDS for dynamic allocation under the SETUP FCT.

Use the ALWIO parameter on the SETPARAM statement to specify the current number of asynchronous I/O requests that can be processed at the same time by the DYNAL DSP. Use the MAXIO parameter on the SETPARAM statement to specify the maximum number of asynchronous I/O requests that can be processed simultaneously by the DYNAL DSP.

If you assign all allocation processing and all device names to JES3, identify all devices as either permanently resident or removable JES3-managed devices. JES3 then processes all requests, including dynamic allocation requests.

Note: All nonspecific, nonprivate requests requiring DASD space must be allocated from permanently resident, or reserved devices. If insufficient space is available on such permanently-resident devices, allocation fails and no scratch volume is mounted.

To improve the performance of dynamic allocation, you can bypass integrity protection for selected data sets. JES3 then bypasses all MDS processing on the global processor for the selected data sets. To specify data sets that require integrity protection and those that do not, use the DYNALDSN initialization statement.

To reduce the number of spool I/O requests needed by dynamic allocation, suppress the messages that are sent to the JESMSG file for TSO users. To suppress these messages, code the JESMSG parameter on the STANDARDS initialization statement.

Chapter 7. Defining and Managing JES3 Mains and Storage

Processor, Mains, and Storage

As part of JES3 initialization, you must define the ***mains*** functioning on the ***processors*** that you want JES3 to use. A processor is the hardware unit that interprets and processes instructions. A main is the configuration of resources, supporting a single operating system on which jobs can execute.

You can also define storage for your system. You can define the size of JES3 buffers, and the size of the JES3 buffer pool. In addition, you can limit the amount of common service area (CSA) that JES3 uses by allocating specific buffers in the JES3 auxiliary address space.

Defining Mains

You must code a MAINPROC statement and a DEVICE statement for every main and processor in the JES3 complex. The MAINPROC statement defines each main to JES3. For a description of the parameters on the MAINPROC statement, see Chapter 12, "Initialization Statement Reference."

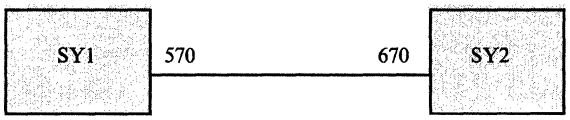
The DEVICE statement defines the CTC connections between the processor on which a main runs and the other processors in a JES3 complex. You must specify DTYPE=SYSMAIN. When DTYPE=SYSMAIN is specified, the JUNIT parameter must specify the CTC adapter for every main in the configuration. The CTC adapter address is specified in the ctcaddr subparameter. This describes if and by what means the main specified in the JNAME parameter can communicate with all other mains in the configuration. If they cannot communicate, then NONE must be specified. Each main representing a local processor must have unique CTC addresses to the global processor. Alternate CTC connections should always be used to improve availability or to allow communication when mains are partitioned.

Note: Ensure that all mains in your JES3 complex are at the same JES3 maintenance level.

To ensure successful dynamic system interchange (DSI) processing, each processor containing a main that is eligible for DSI must be defined with connections (via CTC adapters) to all other processors in the configuration.

Example:

The following hardware configuration includes two processors, SY1 and SY2, connected via a CTC adapter. The adapter address at SY1 is 570 and at SY2, it is 670.



Each processor must be defined by a DEVICE statement and by a MAINPROC statement.

For SY1:

```
DEVICE,DTYPE=SYSMAIN,JNAME=SY1,                                    X
JUNIT=(NONE,SY1,,,670,SY2,,ON)
```

NONE,SY1
Specifies that SY1 does not communicate with itself via a CTC adapter.

670,SY2
Specifies that SY2 communicates with SY1 via CTC adapter address 670 at SY2.

```
MAINPROC,NAME=SY1,SYSTEM=JES3,CPUID=(099999)
```

For SY2:

```
DEVICE,DTYPE=SYSMAIN,JNAME=SY2,                                    X
JUNIT=(570,SY1,,ON,NONE,SY2)
```

570,SY1
Specifies that SY1 communicates with SY2 via CTC adapter address 570 at SY1.

NONE,SY2
Specifies that SY2 does not communicate with itself via a CTC adapter.

```
MAINPROC,NAME=SY2,SYSTEM=JES3,CPUID=(023813)
```

Partitioning

Some multiprocessors, called processor complexes, can be reconfigured in such a way as to form multiple physical partitions. Each physical partition consists of a CPU and hardware resources that can support a single operating system. Forming multiple partitions from a processor complex is called partitioning.

To enable partitioning, you must define to JES3 each possible configuration of mains. You do this by coding MAINPROC statements. You can define an entire processor complex as one JES3 main, or you can define each physical partition as a separate JES3 main.

If you have defined multiple configurations of mains, then you may choose which of these configurations to run in your JES3 complex. You can partition a processor complex into multiple mains, or you can regroup the partitions into a single main at any time without warmstarting JES3.

Planning for a JES3 complex which contains partitionable processor complexes is very important. You must decide what configuration can best handle your workload. Consider how alternate definitions for "backup" mains can be used, and what DASD or MSS connections are needed. Also consider which main will function best as the global and which should be locals. CTC connections are required between the global and each local. If partitioning provides DSI candidates, consider what CTC connections are needed.

You must provide operators and users with adequate instructions on the available facilities. Procedures in run books should detail how and when reconfigurations are to be done on a step-by-step basis. Services and hardware resources that are available on particular mains, and the schedules for when mains are to be active should be detailed.

Activating a processor complex that can be partitioned requires two JES3 starts. The first start (warm or cold) simply adds the main definition; the second start begins using the main or mains defined.

General Considerations

Consider the following when planning your configuration:

- Which main will be the global. Establish DSI capability on other mains.
- The total number of mains. A maximum of 8 MAINPROC statements is allowed.
- The number of CTC connections. Connections from the global processor to each local processor should have a unique set of CTCs.
- The MSS/JES3 connections. A maximum of 4 connections is allowed.
- The job selection environment. Alternate selection modes may be required.
- The impact of users specifying particular mains. With the //*MAIN SYSTEM= facility, the user can specify the mains on which his job is to run. He may also specify the job class or group in which his job is to be processed, which may be defined to a specific main.

Allowable Configurations for a Processor Complex

There are a number of ways that mains for a processor complex can be defined to JES3. The number of processors, the number of mains that you want available to JES3, the attached hardware, and CTC connections all affect how you code the MAINPROC and DEVICE statements.

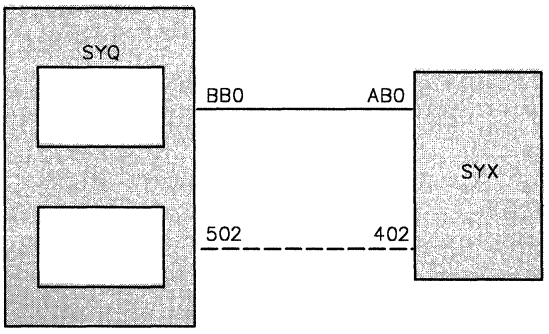
CONFIGURATIONS THAT IBM RECOMMENDS AND SUPPORTS ARE DESCRIBED BELOW. ALTHOUGH DEFINING OTHER CONFIGURATIONS FOR PARTITIONABLE PROCESSOR COMPLEXES IS POSSIBLE, SPOOL DAMAGE MAY RESULT. SOME CONFIGURATIONS ARE RESTRICTED AND SHOULD BE AVOIDED. DEFINING A SINGLE-IMAGE MAIN AND AN INCOMPLETE SUBSET OF THAT DEFINITION IS SUCH A RESTRICTION. DO NOT IPL A PARTITION OF A PROCESSOR UNLESS THE MAIN REPRESENTING THAT PARTITION IS INACTIVE.

Included below are the supported configurations. Connections to other hardware, such as MSS, spool volumes, shared data set volumes, consoles, and remote devices, are not represented in the following examples, but should be considered when setting up your JES3 complex. For detailed information about configuring hardware, see *MVS/Extended Architecture Planning: Recovery and Reconfiguration.*

1. A Processor Complex Defined as One Main: This configuration represents one processor complex, running as a single-image main, connected to another processor. A processor complex has multiple CPUIDs, but when defined this way, functions as *one* main and supports a single operating system. Define a processor complex this way if you require significant processing power, but want to reserve only one main. No reconfigurations are possible; this processor complex cannot be partitioned without warmstarting JES3.

A primary CTC connection is required to communicate with the global main. To ensure alternate capability in the event that one of the partitions becomes inoperable, you should provide a second CTC connection from the opposite partition to the global main.

In the figure a 3084 processor complex, defined as main SYQ, is connected to the global, main SYX.



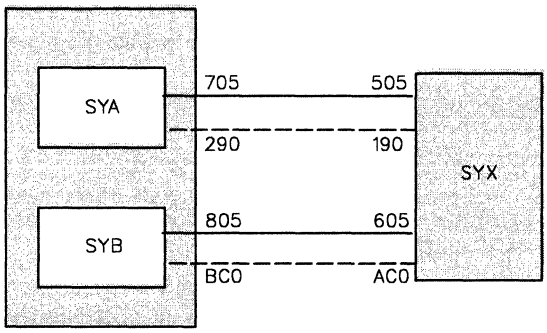
All four CPUIDs from the processor complex are required in the MAINPROC statement:

```
MAINPROC,NAME=SYQ,SYSTEM=JES3,CPUID=(044444,144444,244444,344444),...
MAINPROC,NAME=SYX,SYSTEM=JES3,CPUID=(099999),...

DEVICE,DTYPE=SYSMAIN,JNAME=SYQ,JUNIT=(NONE,SYQ,,,(AB0,402),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYX,JUNIT=(NONE,SYX,,,(BB0,502),SYQ,,ON)
```

2. A Processor Complex Defined as Two Mains: This configuration represents a partitioned processor complex with two mains defined to JES3. The CPUIDs from each partition are included in separate MAINPROC statements. Define a processor complex this way only if you wish to maintain the partitions as two separate mains. Each partition can be IPLed as a global or local processor; however, reconfiguration into a single-image main is not possible without warmstarting JES3.

To communicate with the global main, a unique set of CTC connections is required from each partition to the global.

In the figure a partitioned 3084 processor complex, defined as mains SYA and SYB, is connected to the global, main SYX.



Two partitioned mains, SYA and SYB

The following MAINPROC and DEVICE statements are required:

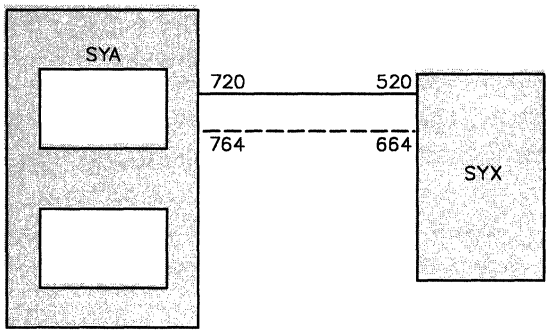
```
MAINPROC,NAME=SYA,SYSTEM=JES3,CPUID=(055555,255555),...
MAINPROC,NAME=SYB,SYSTEM=JES3,CPUID=(155555,355555),...
MAINPROC,NAME=SYX,SYSTEM=JES3,CPUID=(099999),...

DEVICE,DTYPE=SYSMAIN,JNAME=SYA,                                    X
JUNIT=(NONE,SYA,,,NONE,SYB,,,(505,190),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYB,                                    X
JUNIT=(NONE,SYB,,,NONE,SYA,,,(605,AC0),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYX,                                    X
JUNIT=(NONE,SYX,,,(705,290),SYA,,,(805,BC0),SYB)
```

3. A Processor Complex Defined as Two Single-Image Mains: This configuration represents one processor complex, defined as two single-image mains. All four CPUIDs are included on each of the MAINPROC statements for SYA and SYB. When the processor complex is to run as as a single-image main, the operator must specify the name of the main that is to be initially active. While running in single-image mode, the processor complex can be partitioned, and the multiprocessor that is partitioned off can be IPLed, using the other main name. The original main retains its single-image name, even though processing capacity is reduced, and resources attached to the partitioned multiprocessor are no longer available.

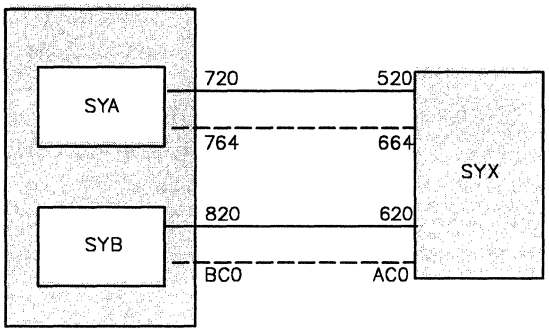
Including all the CPUIDs on each of the two MAINPROC statements also enables you to IPL each partition separately as a global or local processor. While running in partitioned mode, you can reconfigure into a single-image main at any time without having to restart JES3.

Both mains should have a unique set of CTC connections. If the processor complex is partitioned, alternate CTC connections can be lost.

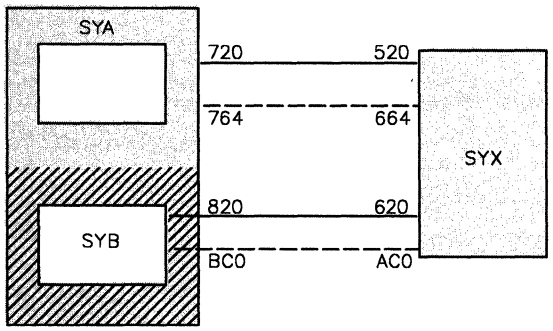
The figure illustrates some possible ways to run JES3 on a 3084 processor complex that is defined as two single-image mains. In the figure, SYX is the global main.



A single-image main, SYA (or SYB)



Two partitioned mains, SYA and SYB



A single-image main, SYA, with one side offline. A partitioned main, SYB, is then IPLed

The following MAINPROC and DEVICE statements are required for running all of the above configurations.

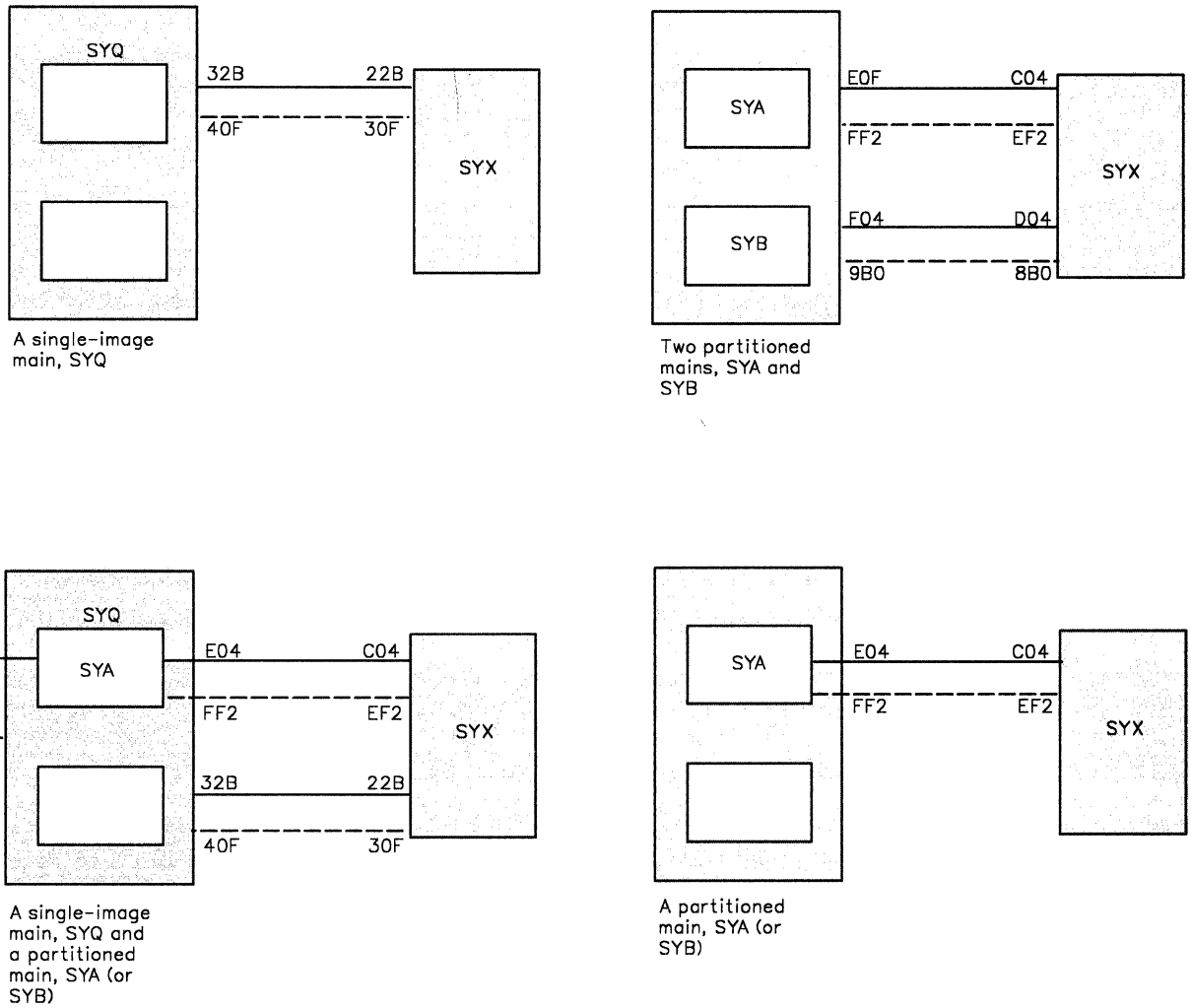
```
MAINPROC,NAME=SYA,SYSTEM=JES3,CPUID=(066666,166666,266666,366666),...
MAINPROC,NAME=SYB,SYSTEM=JES3,CPUID=(066666,166666,266666,366666),...
MAINPROC,NAME=SYX,SYSTEM=JES3,CPUID=(099999),...

DEVICE,DTYPE=SYSMAIN,JNAME=SYA,                                    X
JUNIT=(NONE,SYA,,,NONE,SYB,,,(520,664),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYB,                                    X
JUNIT=(NONE,SYB,,,NONE,SYA,,,(620,AC0),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYX,                                    X
JUNIT=(NONE,SYX,,,(720,764),SYA,,,(820,BC0),SYB)
```

4. A Processor Complex Defined as One Single-Image Main and Two Partitioned Mains: This configuration represents a processor complex that also can be reconfigured readily. Three MAINPROC statements are required. One statement defines a single-image main and includes all the CPUIDs; the other two statements each define the partitions of the processor complex with their corresponding CPUIDs. Defining a processor complex this way, as opposed to the previous example, requires that an additional main be reserved. This configuration is not appropriate for complexes where the number of mains reserved is a concern.

Using this definition, the single-image main can be active simultaneously with *one* of the two partitioned mains, or the two partitioned mains can be active, replacing the single-image main.

Each main represented by a MAINPROC statement should have a unique set of CTC connections. If the processor complex is partitioned, alternate CTC connections are lost.

The figure illustrates the possible ways to run JES3 on a 3084 processor complex that is defined as one single-image main and two partitioned mains. SYX is the global main.



The following MAINPROC and DEVICE statements are required:

```
MAINPROC,NAME=SYQ,SYSTEM=JES3,CPUID=(077777,177777,277777,377777),...
MAINPROC,NAME=SYA,SYSTEM=JES3,CPUID=(077777,277777),...
MAINPROC,NAME=SYB,SYSTEM=JES3,CPUID=(177777,377777),...
MAINPROC,NAME=SYX,SYSTEM=JES3,CPUID=(099999),...

DEVICE,DTYPE=SYSMAIN,JNAME=SYQ,                                          X
JUNIT=(NONE,SYQ,,,1C6,SYA,,ON,1C7,SYB,,ON,(22B,30F),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYA,                                          X
JUNIT=(NONE,SYA,,,1C6,SYQ,,ON,NONE,SYB,,,(C04,EF2),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYB,                                          X
JUNIT=(NONE,SYB,,,1C7,SYQ,,ON,NONE,SYA,,,(D04,8B0),SYX,,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYX,                                          X
JUNIT=(NONE,SYX,,,(32B,40F),SYQ,,ON,(E04,FF2),SYA,,ON,(F04,9B0),SYB,,ON)
```

Determining Storage Capacity

Because JES3 supports the 31-bit addressing capability of MVS/Extended Architecture, much virtual storage constraint imposed by the JES3 private area is alleviated. Common area storage constraints are also relieved because most of the JES3 common area code and control blocks reside in the extended common area above the 16-megabyte line. However, because all constraints are not relieved with 31-bit addressing capability (for example, C/I space or the JES3 parameter-imposed job limit), you may still wish to consider the tuning guidelines provided in this chapter.

Determining the Size of the JES3 Buffers

You may specify between 1952 and 4084 bytes as the size of the JES3 buffers on the BUFSIZE parameter of the BUFFER initialization statement. The BUFSIZE parameter value also sets the size of the spool records. Buffers in CSA or the auxiliary address space reside above 16 megabytes. When selecting the buffer size, select a size that will optimize the transmission of records. Consider the following factors:

- JES3 writes some control blocks as single record files. If the buffer size is significantly larger than the size of a single record file, buffer space will be wasted.
- Some spool data is blocked when written to the JES3 buffers. Therefore, a larger buffer size may reduce the number of I/O operations.
- All MVS processors read and write directly to the spool data sets for SYSIN and SYSOUT. Therefore, a larger buffer size will reduce contention for spool data sets.
- The largest spool-resident control block must fit within the buffer.
- A smaller buffer size allows JES3 to define more buffers and therefore serve more users.

- The GMS checkpoint facility limits the number of execution resource tables (EXRESC) that are checkpointed per processor to the number of entries that can be accommodated by the buffer size that you specify on the BUFSIZE= keyword parameter of the BUFFER initialization statement. Use the following formula to determine the minimum buffer size required to enable checkpointing the status of all groups defined:

  ```
  BUFSIZE=(10 x NUMBER OF EXRESC'S) + 255 + 176
  ```

  Where 10 is the size of each EXRESC entry to be checkpointed as mapped by macro IATYGMS, NUMBER is the amount of EXRESC keywords that specify the the same main name, 255 is the space reserved for CLASS entries and 176 is the space reserved for the fixed section of the checkpoint record mapped by macro IATYGMS.

CAUTION

You can change the buffer size only during a cold start.

Determining the Size of the JES3 Buffer Pool

After selecting the size of the JES3 buffers, you can specify the size of the JES3 buffer pool in the JES3 global address space, JES3 local address spaces, and C/I FSS address spaces. To specify the size of the buffer pool, use the PAGES parameter on the BUFFER initialization statement.

Use the following algorithm to determine the size of the JES3 buffer pool in the JES3 global address space:

```
PAGES=(a+b+c)/(4096/(BUFSIZE+12))
```

where:

a	is 9 times the number of converter/interpreter subtasks
b	is 3 times the number of active readers
c	is 20 times the number of other JES3 functions expected to be concurrently active
BUFSIZE	is the length of each JES3 buffer

Try to ensure that the buffer pool size is large enough for all functions that you expect to be active concurrently. The buffer pool size may be changed during a warm start.

To determine the size of the JSAM buffer pool in the CI FSS address space, start with half the size of the JES3 buffer pool in the JES3 global address space, or allow one JSAM buffer for every 8 to 14 JCL statements that the C/I FSS address space can process concurrently. Refer to the MAXASST parameter value on the appropriate FSSDEF statement for the maximum number of JCL statements allowed. Use performance measurement tools to tune the size of the buffer pool to a smaller value, if necessary.

Reducing the Amount of Common Service Area Used by JES3

Because JES3 supports 31-bit addressing, reducing the amount of common service area (CSA) used by JES3 may not be necessary. If you are concerned that your extended CSA (CSA residing above 16 megabytes) is constrained, you should consider allocating some of the staging areas and user spool access method (USAM) buffers in the JES3 auxiliary address space. There are several advantages to doing this:

- It reduces the amount of CSA used by JES3. Thus, more CSA will be available for application programs and for other subsystems.
- It makes it practical to have more staging areas. Having more staging areas reduces the probability that an application program will terminate abnormally because it needs a staging area when none are available.
- Having more staging areas can mean that an application program, such as IMS, can continue to execute even though a system failure prevents the application program from communicating with the JES3 address space. With this kind of failure, the application program can fill the staging areas but JES3 cannot empty them.

  When the application program fills a specified percentage of the staging areas (you specify the percentage via the STXTNT parameter on the MAINPROC statement), JES3 puts the application program into the wait state. With a large staging area pool, however, there may be enough time for you to correct the problem and prevent the application program from going into the wait state.

To specify how many buffers JES3 is to allocate and how JES3 is to distribute the buffers between CSA and the JES3 auxiliary address space, use the MAINPROC statement. For staging area buffers, specify the information by means of the STXTNT parameter. For USAM buffers, use the PRTPAGE parameter.

Determining How Many Buffers to Allocate in the JES3 Auxiliary Address Space

Before deciding how many user spool access method (USAM) data buffers and subsystem request buffers (staging areas) to allocate in CSA and how many to allocate in the JES3 auxiliary address space, you should understand how your decision might affect JES3 performance. Knowing this will help you obtain a distribution of buffers between CSA and the JES3 auxiliary address space that is best for your installation.

By allocating some of these buffers in the JES3 auxiliary address space, you can reduce the amount of CSA that JES3 uses. This means that more CSA will be available for use by other address spaces. Thus, your system may be able to do more work because you may be able to start more address spaces.

On the other hand, it takes more time for JES3 to move data between a user address space and the JES3 auxiliary address space than it takes to move the same amount of data between a user address space and CSA storage. It takes longer because JES3 uses the MVCS instruction to move data between address spaces and the MVC instruction to move data between an address space and CSA: more

preparation is required before using the MVCS instruction than is required before using the MVC instruction.

How much more time it takes depends on whether your processors execute cross-memory instructions or whether software simulates their execution. It takes more time for software to simulate the execution of cross-memory instructions than it does for hardware to execute them.

JES3 uses the buffers in the JES3 auxiliary address space only when all of the buffers in CSA are in use. For example, if JES3 needs to move data into a USAM buffer and all of the USAM buffers in CSA storage are in use, JES3 uses a USAM buffer in JES3 auxiliary storage. Therefore, the number of buffers you allocate in CSA storage and how often they are used determine how often JES3 must use buffers in the JES3 auxiliary address space.

If your installation experiences heavy console message traffic, then you should consider allocating your secondary staging areas in the JES3 auxiliary address space. During a console buffer shortage, JES3 will not process write-to-operator (WTO) requests. These queued requests can rapidly deplete CSA, thus causing unnecessary 6FB abnormal terminations.

In summary, there are several things to consider before allocating USAM buffers or staging areas in the JES3 auxiliary address space:

- The amount of CSA needed
- The amount of CSA now used
- Whether software simulates the execution of cross-memory instructions or whether hardware executes them
- How often JES3 might use the buffers in the JES3 auxiliary address space

Several factors increase the probability that JES3 will use buffers that are allocated in the JES3 auxiliary address space:

- Decreasing the number of buffers in CSA
- Increasing the number of active address spaces
- An increase in the amount of spool data produced by user address spaces

Reducing Page Fixing/Freeing for PBUFs

Each time JES3 does an I/O operation involving a page of a USAM protected buffer (PBUF), JES3 fixes the page into storage. After completing the I/O operation, JES3 frees the page. You can eliminate this fix/free pair and improve JES3 performance by fixing PBUF pages during JES3 initialization.

For information about how to fix PBUF pages, see the description of the FIXPAGE parameter on the MAINPROC initialization statement in Chapter 12, "Initialization Statement Reference."

Using the Writer Output Multitasking Facility

When the global processor is a multiprocessor, the writer output multitasking facility enables JES3 to do more work in parallel. When enabled (turned on), the writer output multitasking facility provides an additional task under which JES3 can do its work. This task, called the JES3 auxiliary task, lets JES3 off-load work that would otherwise be done under the JES3 primary task. JES3 off-loads that part of the output writer's work that actually prints or punches the output. The JES3 primary task can then execute in parallel with the JES3 auxiliary task and other JES3 subtasks.

If the global processor is a multiprocessor, you should turn on the multitasking facility. To turn the facility on through the initialization stream, specify MT=ON on the OPTIONS initialization statement. Or you can turn it on while JES3 is running by issuing the command *MODIFY,MT=ON.

If the global processor is a uniprocessor, you should turn off the multitasking facility. On a uniprocessor, with the multitasking facility turned on, MVS must do additional task switching for JES3. Also, it may take the output writers longer to print or punch their output because JES3 changes its work sequence. To turn off the multitasking facility during initialization, specify MT=OFF on the OPTIONS initialization statement. To turn it off while JES3 is running issue the command *MODIFY,MT=OFF.

Restart Considerations

After you restart the global processor, the initialization stream last read determines whether JES3 turns the multitasking facility on or off. If this initialization stream contained an OPTIONS statement that specified MT=ON, JES3 turns on the multitasking facility. Otherwise, JES3 turns the facility off.

DSI Considerations

If you use DSI to switch the JES3 global function to a new processor, the initialization stream last read determines the status of the multitasking facility on the new global processor. If this initialization stream contained an OPTIONS statement that specified MT=ON, JES3 turns on the multitasking facility on the new global processor. Otherwise, JES3 turns the facility off.

To turn on the multitasking facility after using DSI, use the command *MODIFY,MT=ON; to turn it off, use the command *MODIFY,MT=OFF.

To determine whether the multitasking facility is on or off, issue the command *INQUIRY,MT.

Chapter 8. JES3 Remote Job Processing

Remote job processing allows users at locations remote from a JES3 installation to submit jobs to MVS through JES3. JES3 reads jobs submitted from remote work stations, prepares them for execution, and, after they execute, sends their output back to the work station.

JES3 supports remote job processing that uses:

- Binary synchronous communication (BSC) protocols
- Systems network architecture (SNA) protocols

Binary Synchronous Communication Remote Job Processing

JES3 BSC supports the remote terminal processing (RTP) programs. These programs are available for systems that operate in BSC mode.

JES3 BSC RJP also supports BSC RJP work stations that have MULTI-LEAVING. MULTI-LEAVING is the fully synchronized two-directional transmission of a variable number of data streams between the work station and a processor.

Jobs submitted from BSC RJP terminals follow the same programming conventions as those established for jobs submitted locally. If standard job routing is used, the JES3 system automatically replaces conventional printing and punching with output terminal transmission. Output from remotely submitted jobs may be returned to any terminal specified by the submitter or may be processed locally. The standard operator console is supported as either a full- or limited-function remote operator console. Only one console is supported on each remote work station.

For BSC RJP, background utilities can be used between the central site and a BSC RJP work station or between two BSC RJP work stations (with certain exceptions). The input unit on a nonprogrammable terminal must be used only for batch processing of jobs, but the output units are available for callable dynamic support programs (DSPs). The same applies to readers defined as automatic readers on the /*SIGNON card. The DSPs that can be called are card-to-card (CC), card-to-printer (CP), card-to-tape (CT), card reader (CR), tape-to-card (TC), and tape-to-printer (TP). Tape devices are not supported on remote work stations; therefore, DSPs using tape must have the tape unit on the central system. Only the callable CC and CP DSPs may have both input and output remote device support. All other callable DSPs are restricted to use of one remote unit.

Data Security Considerations

If a remote operator using a BSC RJP dial-up line hangs up the phone, turns off the modem, or is disconnected for any reason before signing off, a data security problem may arise. Under certain conditions, disconnecting without signing off allows another user to connect to the same port and receive data intended for the previous user. To ensure data security, use dedicated BSC RJP lines or use SNA RJP. To minimize data security problems when using BSC RJP dial-up lines, remote operators must sign off before disconnecting.

Data Compression

Data transmitted to or from programmable terminals may first be compressed into blocked transmission groups, with all strings of three or more (up to 31) duplicate characters reduced to two-byte control groups. Duplicate blanks are reduced to one character. This compressed format improves transmission speed by not transmitting redundant information. Print records have the first character reserved for carriage control. Punch records are 80-character EBCDIC card images for punching in data mode 1. Compression is specified by coding the CS parameter on the RJPTERM statement.

Operator Communications

The task of initiating and canceling BSC RJP is the responsibility of the operator at the central location. The extent to which a console operator at a work station location can enter commands is dependent upon JES3 initialization parameters specified for that work station.

The following commands are used to initiate and cancel BSC RJP: *CALL,RJP; *START,RJP; *RESTART,RJP; and *CANCEL,RJP.

Operator Commands Accepted from BSC RJP Work Stations

Several sets of JES3 commands are available to operators of JES3 BSC RJP terminals with console typewriters. The remote operator uses these commands the same as the operator at the central JES3 facility. The specific command set available is determined by the console's level of authority, as defined in the LEVEL= parameter of the JES3 CONSOLE initialization statement. "Authorizing JES3 Commands" contains information about authority levels for remote consoles.

Debugging Facilities

To help you debug BSC RJP errors, JES3 provides statistics about line errors, a capability to dump specific data during channel-end processing, and a capability to trace BSC RJP activity.

BSC RJP Line Error Statistics

Each time a line error is encountered, an entry is made in a line statistics area. This statistical area may be displayed on the calling console, using the *INQUIRY,T,L=lname,STAT command.

BSC RJP Line Snap Facility

The BSC RJP snap facility consists of the RJP modify feature used in combination with the RJPSNPS DSP. This facility allows the user to generate snap dumps of the line DCT, the current IOB, and the transmission data areas during channel end processing.

The SNAP trace facility is invoked separately from the RJPSNPS DSP, but is required for channel-end snapping. Tracing is started by entering the command *MODIFY,T,L=nnnnnnnn,SNAPON, where nnnnnnnn is the appropriate line name. Tracing is stopped by entering the command *MODIFY,T,L=nnnnnnnn,SNAPOFF, where nnnnnnnn is the appropriate line name.

The RJPSNPS DSP, when active, spools the channel-end snap data to a multirecord file on the JES3 queue volumes and, when terminated, causes the data file to be closed and scheduled for print service. This facility is also described in *MVS/Extended Architecture JES3 Diagnosis.*

BSC RJP MLOG Trace Facility

The BSC RJP event trace is always active when BSC RJP is active. The trace can be displayed on the MLG class of consoles by using the *MODIFY,T,L=lname,TRCEON command.

The MLG class of consoles must be able to print the console messages generated by BSC RJP during event tracing at the same rate that they are being generated. This may not always be the case if a high-speed line is being traced or if more than one line is being traced. Enabling the event trace for more than one line should be avoided. This facility is also described in *MVS/Extended Architecture JES3 Diagnosis.*

Initialization Statements that Affect BSC RJP

Parameters specified on the following JES3 initialization statements affect JES3 BSC RJP execution and operation.

CONSOLE. Each terminal that requires real or simulated console support should be specified with a CONSOLE initialization statement.

DEVICE. Each terminal device that requires special definition must be defined by a DEVICE statement. An example of such a device is a remote 3211 Printer.

RJPLINE. Each line to be used by BSC RJP must be defined during JES3 initialization.

RJPTERM. Each terminal to be used by BSC RJP must be defined during JES3 initialization.

Generating Remote Terminal Processing Programs

If your installation supports programmable work stations that use binary synchronous communication (BSC), you must generate remote terminal processing (RTP) programs for these work stations. You can do this after you install JES3.

The RTP programs that you can generate are the:

- System/360 and System/370 BSC RTP programs
- System/360 Model 20 BSC RTP program including the 2922 RTP program
- 1130 RTP program
- 1130 loader program
- System/3 RTP program

Before you can generate the RTP programs, however, you must install product EML1102 and apply the program temporary fix (PTF) against EML1102 that fixes APAR OZ43643. The PTF that originally fixed this APAR was UZ29226. If that PTF has been incorporated into another PTF, use RETAIN to find the new PTF number.

The programs that you need in order to generate the RTP programs are:

Program	Member
System/360 and M20 BSC Remote Program	HRTP360
1130 Loader Program	HRTPLOAD
1130 Remote Program	HRTP1130
System/3 Remote Program	HRTPSYS3
RMTGEN Standard Options List	HRTPOPTS

You receive these programs with product EML1102.

The EML1102 product also contains the following utility programs:

- REMOTGEN: Acts as a monitor linking to other remote terminal utility programs and to the assembler during an RMT generation.

- GENRMT: Reads the card input stream during RMT generation for the RTP program identification, selects the appropriate standard option list for the RTP program to be generated, prints the parameter default values, and updates the source modules with the changes read from the RMT parameters.

- LETRRIP: An 1130 post-processor that creates an 1130 object-deck image on the SYSPUNCH data set.

- SYS3CNVT: A System/3 post-processor that produces a System/3 object-deck image on the SYSPUNCH data set.

These programs can aid you when you generate the RTP programs.

After installing product EML1102, use the following procedure to generate the RTP programs:

- Code the options deck for each remote work station for which an RTP program will be generated. See Appendix A, "RMT Option Statements" for a description of the statements that you can include in the options deck.
- Execute the following job:

```
//JES3JOB  JOB .....
//RMTGEN   EXEC  RMTGEN
//OPTIONS  DD    *
           .
           .
           .
/*
```

- Save the listings and the self-loading object decks that are produced.

Before generating the RTP programs, ensure that the device characteristics and buffer sizes specified in the options deck are compatible with the device characteristics and buffer sizes specified in the initialization stream.

For information on how to load the object deck into the work station and how to operate the work station, see *Operator's Library: OS/VS2 Remote Terminals (JES3).*

Systems Network Architecture Remote Job Processing

Systems network architecture (SNA) remote job processing (RJP) permits the input and output of jobs to and from SNA work stations. SNA RJP uses the SNA logical unit (LUTYPE 1) interface to support the IBM 3770 Data Communication System and the IBM 3790 Communication System. Before reading this section, you should become familiar with the concepts and terminology found in *Systems Network Architecture General Information.*

SNA RJP Implementation of SNA Concepts

SNA divides communication system functions into a set of well-defined layers called:

- Application layer (APPL)
- Transmission subsystem layer (TS)
- Function management layer (FM)

All of these layers must exist at both ends of a common path. The application layer performs the data-oriented processing, but is not involved in the protocol or procedures for controlling a communication line or routing data units through the network. In the host, the following JES3 services serve as the application layer:

- Input service
- Console service
- Output service

The transmission subsystem layer routes and moves data units between origins and destinations. This layer does not examine, use, or change the contents of the data units. In the host processor, the virtual telecommunication access method (VTAM) provides this layer.

VTAM enables application programs to transfer data to and from terminals that are a part of a telecommunication network. For more information about VTAM, see *Introduction to VTAM*.

The function management layer (FM) provides control of the data flow between application layers and transfers data presented to the network. The data flow control protocol support defined by SNA is implemented through a set of encoded requests called data flow control (DFC) requests. These requests are used to handle data units and structures such as chains of related data units. They are also used to manipulate the state conditions, such as "send" or "receive" state, that are defined by the SNA-supplied data flow control protocol.

To enable data to flow between two application layers, SNA defines three types of addressable elements called network addressable units (NAUs). The types of NAUs are:

- System services control point (SSCP): This is the control function for a SNA network. VTAM provides the SSCP function.

- Physical unit (PU): Each node in the network whose existence has been defined to the SSCP has a PU. The SSCP establishes a session with each PU in the network as part of the bring-up process.

- Logical unit (LU): The LU is the port through which an application layer accesses the network. An LU supports at least two concurrent sessions; one with the SSCP and one with another LU. Each LU must first establish a session with the SSCP before it can enter a session with another LU.

Figure 8-1 shows the three SNA communication layers and the communication paths between the layers. Figure 8-2 shows the interface between JES3 and VTAM and the internal JES3 interfaces required to support SNA RJP.

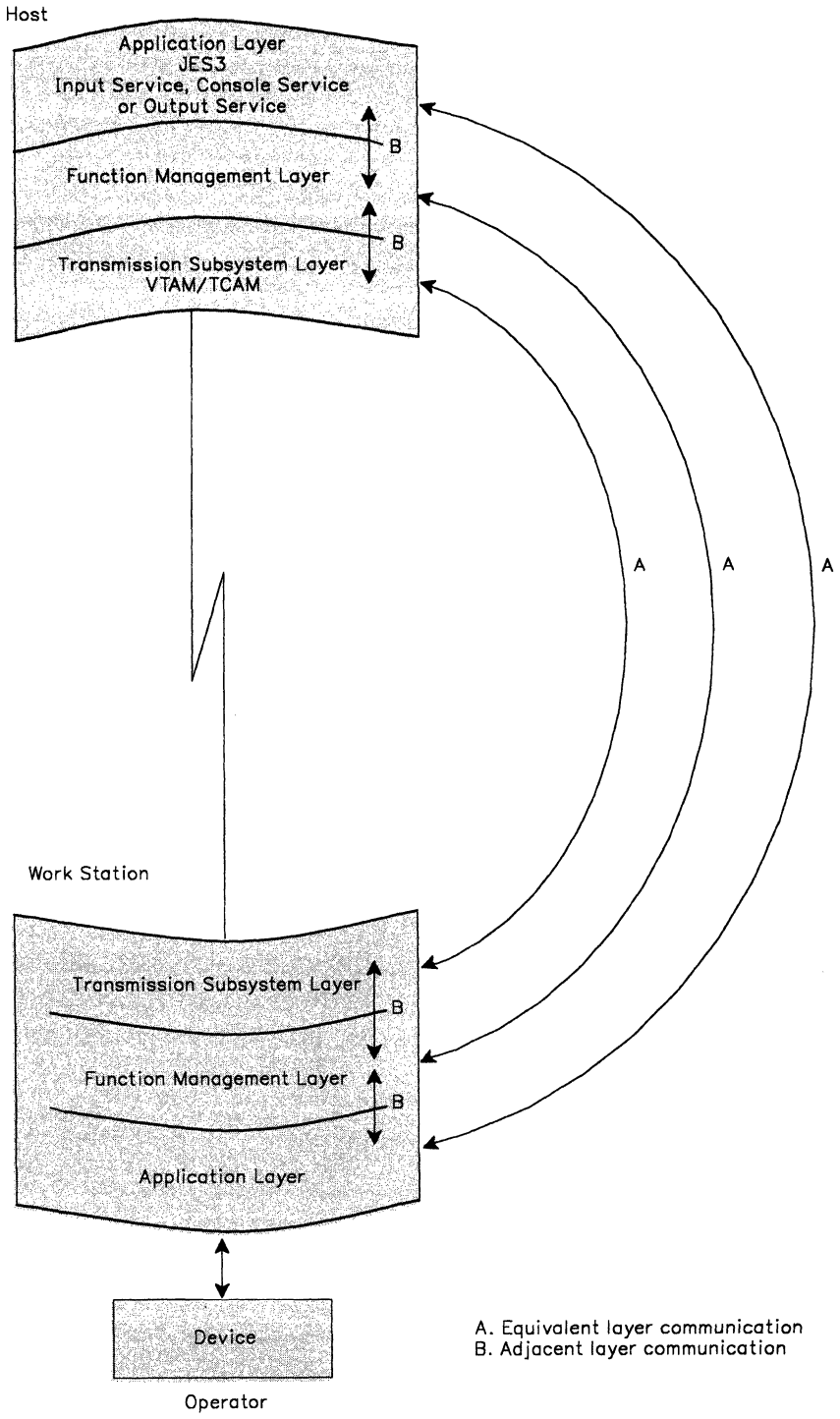


Figure 8-1. Overview of SNA Environment for JES3

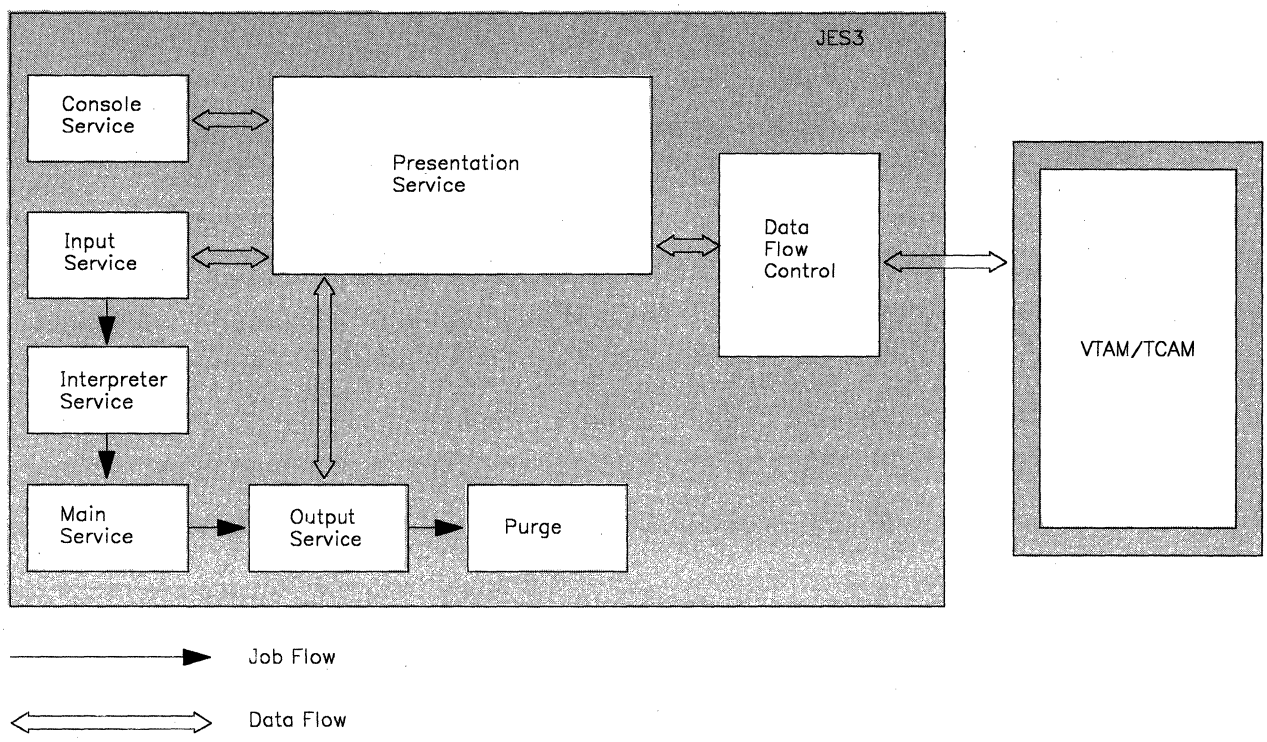


Figure 8-2. JES3-VTAM Interface

Function Management Presentation Services (FMPS)

The function management presentation services (FMPS) component of SNA RJP provides the DFC interface to JES3 readers, writers, and console message processors. FMPS is composed of three functional areas: (1) function management (FM) inbound, (2) function management (FM) outbound, and (3) presentation services (PS) routines. FM inbound and outbound routines provide a logical record interface to JES3 readers, writers, and console message processors and an RU interface to DFC. The PS routines perform data character code translation (EBCDIC to ASCII and ASCII to EBCDIC as applicable to the SNA work station), compression, and compaction.

FM Inbound

FM inbound routines receive request units (RUs) -- an RU is the SNA unit of transmitted information -- transmitted from SNA work stations.

If inbound RUs contain data from a work station card reader or console, the PS routines perform decompression, decompaction, and character code translation (ASCII/EBCDIC) as required. SNA character string (SCS) processing decomposes the RU into logical records for the card reader DSP or into a console command message for the SNA RJP DSP to issue the INTERCOM macro.

Figure 8-3 shows the console and card reader control function characters that FM inbound routines support.

Function	Destination	Function Character
New Line (NL)	console	X'15'
Carriage Return (CR)	console	X'0D'
Forms Feed (FF)	console	X'0C'
Interchange Record Separator (IRS)	console	X'1E'
	card reader	X'1E'
Transparency (TRN)	console	X'35'
	card reader	X'35'

Figure 8-3. SCS Function Characters Supported by FM Inbound Routines

Card images smaller than the card record size specified in the FM header for the input file are padded to the right with blanks. Card images larger than the card record size are deblocked into cards of the specified size. Card images can span RU boundaries if the spanning option is selected in the work station LU session BIND parameters.

FM Outbound

The FM outbound routines are invoked via the IATXLRPT macro. The JES3 writer (module IATOSSN) uses this macro to send print/punch logical records. Module IATCNRM also uses this macro to send console messages to a SNA work station LU.

The FM outbound routines interface with DFC to determine if logical records (lines of print, card images) are packed into the RU. SCS processing is performed on the data. If character code translation is needed, a presentation services routine is called. See Figure 8-4 for supported characters.

Function	Source	Function Character
New Line (NL)	console	X'15'
Transparency (TRN) Interchange Record Separator (IRS)	reader	X'35' X'1E'
New Line (NL) Carriage Return (CR) Forms Feed (FF) Select (SEL) Transparency (TRN) Set Vertical Format (SVF)	printer	X'15' X'0D' X'0C' X'04' X'35' X'2BC2'

Figure 8-4. SCS Function Characters Supported by FM Outbound Routines

Presentation Service

Presentation service performs data compression and data compaction, converts logical records into RUs and converts RUs into logical records.

Compression and Compaction: Compression removes strings of repeated characters, thereby reducing the amount of data that must be sent over communication lines to the remote work station. Deleted characters are replaced with a one- or two-byte character string, the first of which is called a string control byte (SCB). If a string of two or more repeated blanks is removed, a special SCB with a count indicating the number of removed blanks is inserted. The count can indicate one to sixty-three. Strings of three or more repeated nonblanks are replaced by a two-byte character string. The first byte is an SCB indicating a repetition of a nonblank with a character count. The second byte is the value to be replicated. Since SCBs can look like a data byte, noncompressed character strings must also be preceded by an SCB. This allows the receiver to locate SCBs and carry out decompression SCBs. They indicate a noncompressed string, a string of blanks, or a string of nonblanks. Compression is very useful if the original data contains large amounts of repeated characters.

Compaction is similar to compression in that it is an attempt to reduce the number of characters sent over communication lines. To use compaction, you must first define one or more compaction tables. A compaction table defines the set of characters which can be included in a compacted string. Both master characters, which can be represented using a 4-bit code, and nonmaster characters, which can be represented using an 8-bit code are included. The compacted character string begins with a compaction SCB.

A single file can be both compressed and compacted. However, each character in the file is represented in either a noncompressed, compressed, or compacted character string.

Compaction tables are defined by using the COMPACT initialization statement. The SYSOUT statement can be used to specify that certain SYSOUT classes are to be compacted and which table is to be used. The user can also request compaction and specify the name of a compaction table in the JCL via the //*FORMAT statement.

Compaction can reduce the number of bytes sent out by up to 50%; however, the outbound data would have to be completely composed of master characters to reach the full 50% reduction.

If at the time the session is established, the work station requests that ASCII be used rather than EBCDIC, then PS in JES3 translates all data to/from ASCII from/to EBCDIC. If ASCII is used, compression, compaction, and certain SCS control codes are not used.

Conversion of Logical Records and RUs: The final function of presentation services is the conversion of logical records to and from RUs. RUs are the unit of data that flow over the session. They are always of the fixed size that was passed to JES3 at LOGON. Since logical records are variable in nature, especially after they have been compressed and compacted, they must be packed into fixed length RUs before they can be sent. RUs sent by JES3 can contain several logical records; in fact, they can contain pieces of a logical record. For

inbound streams, PS deblocks logical records out of RUs. It can get part of a logical record from one RU and the rest from the next.

Data Flow Control

Data Flow Control (DFC) transmits RUs to VTAM. Through the use of the CHNSIZE parameter on the SYSOUT statement or JCL statements, you can request that DFC chain RUs before they are transmitted.

Chaining RUs: The user can elect to send an entire SYSOUT data set as one chain. This causes the best performance since only one response need ever be sent by the receiving logical unit. If, however, an error occurs, JES3 notifies the remote operator and allows him to enter a *RESTART command for the device to which the stream was directed. To ensure that the remote device receives all of the data, the remote operator should issue a *RESTART command for the device to have the entire data set resent. Buffering prevents the operator from accurately determining how many lines or pages were sent prior to the error.

A second option allows JES3 to start a new chain every time a skip to channel one is found in the output. Each user-defined page becomes a chain. If an error occurs, JES3 notifies the operator and, unless directed by the operator to the contrary, resends the chain (page) in error. This is a desirable option if you are printing on special forms since the forms cannot become misaligned because of an error.

The final user error recovery option allows the user to specify the actual number of logical records to go into each chain. This can be any number from 1 to 254. The user specifies which of the three options is desired either in the SYSOUT initialization statement, DEVICE statement, or on the //*FORMAT statement in the job's JCL.

JES3 to VTAM Interface

To communicate with VTAM, JES3 uses the macros and exits that VTAM provides. JES3 uses the RECORD mode of the VTAM application program interface (API).

VTAM Considerations

VTAM directs transmission of data between JES3 in the global processor and the batch work stations in the telecommunications network. The JES3 SNA RJP user must be familiar with the VTAM requirements for work station definition. This section provides a brief description of those requirements. For more information about these requirements, see *Advanced Communications Function for VTAM Version 3 Programming*.

Application Definition. An application definition statement (APPL) in SYS1.VTAMLST that defines JES3 as an application is required. The name can be selected by the installation, but it must agree with the name specified on the COMMDEFN initialization statement. The passwords must also match.

Physical Unit Definition. The work station physical unit (PU) must be defined to VTAM using the PU macro. The DISCNT parameter on the PU macro should specify YES for work stations connected via switched lines.

Logical Unit Definition. The work station LUs must be defined to VTAM via the LU macro. The LU names are the same LU names that may be specified on the RJPWS initialization statements.

LU parameters that can be used to define the LOGON command syntax are LOGTAB, SSCPFM, and USSTAB. LU parameters that affect session characteristics are PACING and VPACING (the defaults are acceptable to JES3), BATCH (BATCH=NO should be specified), BUFLIM (this number should be large), and MODETAB.

Initialization Statements that Affect SNA RJP

Parameters on the following initialization statements affect JES3 SNA RJP execution and operation.

COMMDEFN. The user communication subsystem interface (VTAM) is specified with a COMMDEFN initialization statement.

COMPACT. A compaction table is defined with the COMPACT initialization statement.

CONSOLE. Each terminal that requires real or simulated console support should be specified with a CONSOLE initialization statement.

DEVICE. Each terminal device that requires special definition must be defined by a DEVICE statement. An example of such a device is a remote 3211 Printer.

OUTSERV. Default values and standards for printers and punches are defined on the OUTSERV initialization statement.

RJPWS. The characteristics of each SNA RJP work station must be defined during JES3 initialization.

SYSOUT. Class characteristics are specified on the SYSOUT initialization statement for output classes requiring other than print processing.

Basic Exchange Support

Basic exchange allows a user to transmit print output from the host to a work station and then within the work station to a diskette rather than to a hard-copy device. Basic exchange requires the diskette to be set up in exchange format and the data length to be 1 to 128 character per record. Basic exchange is specified by using the SELECT=BEn and LEN parameters on the DEVICE statement. The //*FORMAT PR statement should be specified to indicate the printer device (name as specified on a DEVICE statement) to be used as a basic exchange diskette.

With basic exchange, any of the 256 EBCDIC characters can be transmitted as data. Transmitted data cannot be compressed or compacted.

Exchange Support

Exchange allows a user to transmit print output from the host to the work station. It then allows the work station to place the output on a diskette rather than on a hard-copy device. To specify exchange, use the SELECT = EXn parameter on the DEVICE initialization statement. The printer device to be used as an exchange diskette should be specified on the //*FORMAT PR statement.

The physical diskette must be formatted normally in order for the work station to place output on the diskette. For further information on the local diskette function, see *Operating Procedures Guide: IBM 3774 and 3775 Communications Terminals.*

Output sent to the diskette is in the same format that it would be in if sent to a hard-copy device; carriage control characters, SCS characters, and other control information is in the diskette record. For proper operation of exchange support, only one DEVICE statement per physical diskette device should be included in the initialization stream.

Exchange and Basic Exchange Initialization Considerations

- For exchange and basic exchange devices, the HEADER and BURST parameters on the DEVICE statement are ignored. No header or burst information is recorded on the diskette.
- For basic exchange devices, many work station printers can be defined, using DEVICE statements, as basic exchange diskettes even though there are less physical diskettes than there are basic exchange DEVICE statements. However, the LEN parameter on each DEVICE statement must be different for proper operation.
- The DEVICE statement restriction of no more than 15 printer devices per work station includes printers defined as exchange devices.
- The PR parameter on the RJPWS statement must specify a value that includes all physical printers and all defined exchange and basic exchange diskettes.

Chapter 9. JES3 Networking

A network is a combination of interconnected equipment and programs used for moving information between points where it may be generated, processed, stored, and used. Networking provides an installation with the ability to:

- Send a job to another node for execution
- Receive a job from another node either to execute or to send to yet another node for execution
- Send or receive SYSOUT data produced from a job
- Reroute jobs or SYSOUT data to another node

To make your JES3 complex part of a network job entry (NJE) environment, you must define the network via JES3 initialization statements. Some things that you can define are:

- Nodes in the network and their characteristics
- The protocol used for communicating with remote nodes
- Lines that connect remote nodes to your node
- A logical console for the network
- A class for network messages
- A routing table for commands received from the network

To monitor data sent to or from your node, you can use user exit routines. These exit routines, which you must write, allow you to monitor:

- Files sent to your node by means of the TSO/E TRANSMIT or CMS SENDFILE command
- Commands sent to your node from remote nodes
- Header records that accompany jobs or SYSOUT data received at or sent from your node

Networking Protocols

A networking protocol is the means by which a complex can participate in a job entry network. Protocols are rules that direct the logical structure, formats, and operational sequences for transmitting information through networks and controlling the network configuration and operation. The two protocols that JES3 can use to establish a networking environment are:

- Binary synchronous communication (BSC)
- Systems network architecture (SNA)

BSC protocol allows data flow between nodes over physical links, which are typically channel-to-channel adapters, or leased or dial-up telephone lines.

SNA protocol provides a networking capability for JES3 that works in combination with MVS/Bulk Data Transfer (MVS/BDT) Version 2. Networking is established between nodes through MVS/BDT "sessions." Sessions can be established between channel-to-channel adapters, telephone lines, microwave links, or by satellite, and are controlled by VTAM.

A JES3 complex can use BSC or SNA protocol, or both. A user submitting an NJE job is not aware of whether JES3 is using BSC or SNA. To define the type of protocol that JES3 will use, code the TYPE= parameter on the NJERMT initialization statement.

Converting Networking Protocols

It is possible to convert nodes from one networking protocol to the other. In other words, nodes that are currently using BSC protocol can be converted to SNA, and vice versa. The following scenario describes converting an existing BSC node to SNA. Converting back to BSC can be accomplished by reversing the procedure.

Before converting to SNA, you must install:

- MVS/BDT Version 2
- ACF/VTAM Version 1 with the Multisystem Networking Feature, ACF/VTAM Version 2 Release 1, or ACF/VTAM Version 3 Release 1
- TSO/E Release 2 (if networking jobs are to be submitted through TSO)

Because SNA protocol uses MVS/BDT to transfer data between nodes, ensure that node names are defined exactly the same in both the JES3 and MVS/BDT initialization streams. To define a node to MVS/BDT, use the MVS/BDT SYSID and BDTNODE statements. For detailed information on defining MVS/BDT initialization statements, see *MVS/Bulk Data Transfer Facility: Initialization and Network Definition.*

Defining networking nodes to JES3 requires coding the NJERMT initialization statement. (See Figure 9-1 later in this chapter to determine what statement parameters you will need.) A JES3 SYSID initialization statement is also required if you have more than one MVS/BDT facility in your JES3 complex.

Once the nodes are defined in both initialization streams, you can warm start JES3 and MVS/BDT. The operator can then migrate any jobs that remain waiting for a BSC line by using the NJEROUT DSP. Jobs are transmitted when the MVS/BDT session is activated. See *MVS/Bulk Data Transfer Facility: Operator's Guide* for descriptions of operator commands to control MVS/BDT sessions.

When the SNA job entry network is stable, you can remove the physical BSC links from the complex, if desired.

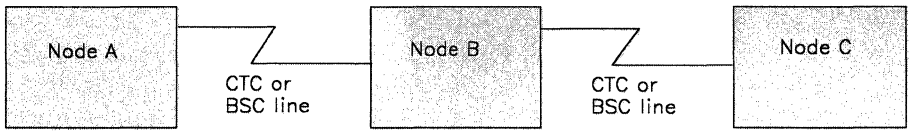
Types of Nodes

In JES3 terminology, a node is called either a home node or a remote node. What you call a particular node depends on your point of reference.

The view from any given node is that that node is the home node and other nodes are remote nodes. Thus, you always view your node as the home node and other nodes as remote nodes. On the other hand, a system programmer at another node views that node as the home node and other nodes (including yours) as remote nodes.

There are two kinds of remote nodes: directly-connected and indirectly-connected. A directly-connected remote node is *adjacent* to your node, and is connected through direct BSC or SNA links. An indirectly-connected remote is *not adjacent* to your node, but is connected to your node through one or more other nodes.

In the following diagram, A and B are directly-connected nodes as are B and C. Nodes A and C are indirectly-connected via node B.



You must define the home node and all remote nodes with which the home node will communicate. To define a node, use the NJERMT initialization statement.

Note: When defining an indirectly-connected remote node, you may wish to include parameters that would allow this node to become directly-connected at some future time. By doing so during initialization, you need only use the *F,NJE command to change the communication path from indirect to direct.

Figure 9-1 identifies required and optional parameters for the home node and for each type of remote node. If you code the other parameters, those that are *not* indicated as required or optional for the type of node you are defining, JES3 will ignore them.

NJERMT Parameter	Type of Node		
	Home Node	Directly-connected Remote Node	Indirectly-connected Remote Node
AUTO		O (BSC)	
BDTID	O (SNA)		
BFSIZ		O (BSC)	
CTC		O (BSC)	
EXPWD		O (BSC)	
EXSIG		O (BSC)	
HOME	(See Note 1)	(See Note 2)	(See Note 2)
LINE		O (BSC)	
MAXLINE	(See Note 3)	(See Note 4)	(See Note 3)
NAME	R (BSC,SNA)	R (BSC,SNA)	R (BSC,SNA)
NJEPR	O (BSC)		
NJEPU	O (BSC)		
PATH		(See Note 5)	R (BSC,SNA)
PRTDEF	O (BSC,SNA)		
PUNDEF	O (BSC,SNA)		
PWD		O (BSC)	
RDLY		O (BSC)	
SIG		O (BSC)	
STREAM		O (BSC)	
TYPE		O (BSC,SNA)	

Notes:
1. Specify HOME=YES (BSC,SNA).
2. Specify HOME=NO (BSC,SNA).
3. Specify MAXLINE=0 (BSC)
4. Specify MAXLINE=1, 2, or 3 (BSC).
5. Do not specify for directly-connected nodes.

Legend:
R - Required parameter
O - Optional parameter

Figure 9-1. Parameter Requirements for the NJERMT Statement

Defining the Home Node

The first step in defining the home node is to name it and to specify it as the home node. To name the node, code the NAME= parameter. To specify the node as the home node, code HOME=YES.

If this node is to be part of an NJE network that uses the SNA protocol, you can also code the BDTID parameter. This parameter identifies which MVS/BDT subsystem in your JES3 complex is processing SNA networking transactions. The name specified on this parameter must match the name specified on the MVS/BDT SYSID statement.

If the node uses the BSC protocol, you can also specify the number of logical printers or punches associated with this node. You can specify from 1-99 logical printers, from 1-99 logical punches, or both. The number of logical printers determines how many jobs can concurrently spool network print data to a networking writer. The number of logical punches determines how many jobs can concurrently spool network punch data to a networking writer.

To request logical printers, code the NJEPR parameter; for logical punches, code the NJEPU parameter. If you omit the NJEPR parameter, JES3 assumes 3 logical printers. If you omit the NJEPU parameter, JES3 assumes 3 logical punches.

When defining the home node, you can also specify default SYSOUT classes for print and punch output received from other nodes. If the characteristics of the SYSOUT class associated with received output does not match the characteristics of the SYSOUT class defined for the home node, the default class that you specify is used. For example, if SYSOUT class C is defined as a print class at the home node, and punched output, designated as SYSOUT class C, is received from a remote node, the value that you specified in the PUNDEF parameter is used.

To assign a default print class, code the PRTDEF parameter; to assign a default punch class, code the PUNDEF parameter. If you omit the PRTDEF or PUNDEF parameter, JES3 assumes SYSOUT class A for print output and SYSOUT class B for punch output.

Defining a Remote Node

To define a remote node to your home node, name the node and specify it as a remote node. Use the NAME= parameter to name the node. To specify the node as a remote node, omit the HOME= parameter or code HOME=NO.

Because nodes using BSC or SNA protocol can coexist in the same JES3 network, you must specify the networking protocol that JES3 is to use for this node. Specify TYPE=BSC or TYPE=SNA on an NJERMT statement that defines a directly-connected node. TYPE=BSC is the default value.

For those nodes using the BSC protocol, you can optionally specify on the NJERMT statement for the remote node:

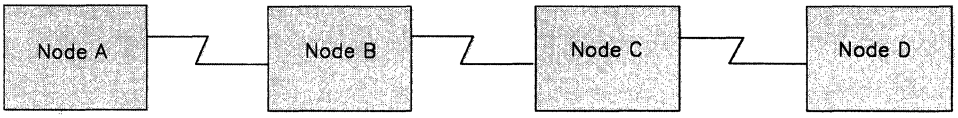
- A communications path to each indirectly-connected node (PATH=)
- The size of the buffer to be used for communication with a directly-connected remote node (BFSIZ=)
- Passwords to be used for communication with a directly-connected remote node (SIG=, PWD=, EXSIG=, and EXPWD=)
- The use of channel-to-channel adapters or leased or dialup lines (CTC=)
- The maximum number of lines that an operator can start (MAXLINE=)

For remote nodes using the SNA protocol, NAME= and TYPE= are the only required parameters. JES3 ignores any other parameters specified. For information relating to the use of each networking protocol, read the sections in this chapter entitled, "BSC Considerations" and "SNA Considerations."

Specifying a Communications Path for Indirectly-Connected Nodes

Before your node can communicate with an indirectly-connected remote node, you must define the first node in the path to the remote node. To do this, code the NAME= and the PATH= parameters on the NJERMT statement that defines the remote node.

To illustrate, consider the following sample network.



For nodes A and D to communicate, the system programmer at each node must code the following NJERMT statements:

At Node A:

```
NJERMT,NAME=A,HOME=YES,...
NJERMT,NAME=B,...
NJERMT,NAME=D,PATH=B,...
```

At Node B:

```
NJERMT,NAME=B,HOME=YES,...
NJERMT,NAME=A,...
NJERMT,NAME=C,...
NJERMT,NAME=D,PATH=C,...
```

At Node C:

```
NJERMT,NAME=C,HOME=YES,...
NJERMT,NAME=B,...
NJERMT,NAME=D,...
NJERMT,NAME=A,PATH=B,...
```

At Node D:

```
NJERMT,NAME=D,HOME=YES,...
NJERMT,NAME=C,...
NJERMT,NAME=A,PATH=C,...
```

BSC Considerations

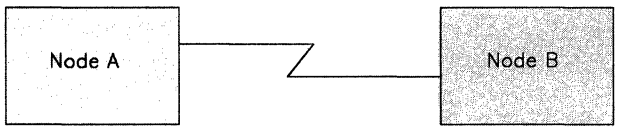
The following topics apply to defining nodes that are using the BSC networking protocol.

Defining the Buffer Size

You can specify the size of the buffer your node is to use for communicating with a directly-connected remote node. The maximum buffer size you can specify is the size of the spool buffers (specified on the BUFFER initialization statement) minus 44. To specify the buffer size, use the BFSIZ= parameter on the NJERMT statement that defines the remote node. If you omit the BFSIZ parameter, JES3 uses a buffer size of 400 bytes.

The system programmer at the remote node must specify the same buffer size that you specify. If he does not do this, the nodes will be unable to communicate. To specify the buffer size, the system programmer at the remote node must use the BFSIZ= parameter on the NJERMT statement that he codes to define your node.

The following example shows two directly-connected nodes, A and B, and the NJERMT statements that define the nodes and the buffer sizes:



NJERMT,NAME=B,BFSIZ=512,... NJERMT,NAME=A,BFSIZ=512,...

Specifying Passwords

You can require that a directly-connected remote node provide a password to establish communications with your node. By requiring a password, you make it more difficult for an unauthorized person to start the line and gain access to the network. In addition, by requiring passwords for nodes that are connected by dial-up lines, you make it more difficult for a node that is not part of the network, but has access to the line, to gain access to the network.

To specify a password for any directly-connected remote node, use the EXPWD= parameter. The password specified by this parameter allows the remote node to start the line. Code the EXPWD= parameter on the NJERMT statement that defines the remote node.

For a directly connected remote node connected via a dial-up line, also use the EXSIG= parameter to specify a second password. This password identifies the specific node that started the line (with a dial-up line, several nodes can have the capability to start the line).

The system programmer at the remote node provides the required passwords by coding them on the NJERMT statement that he codes to define your node. However, he must use a different set of parameters than the parameters you used to specify the passwords. He must use:

- PWD= to specify the password that you specified with EXPWD=
- SIG= to specify the password that you specified with EXSIG=

When a remote node tries to establish communications with your node, JES3 checks for only those passwords that you have specified the remote node must provide. For example, if you specify only EXPWD=, JES3 checks for only that password. If the remote node does not provide the correct passwords, JES3 does not allow the node to communicate with your node.

The system programmer at a directly-connected remote node can require that you provide passwords in order to communicate with the remote node. To specify the passwords, use the PWD= and SIG= parameters on the NJERMT statement that you code to define the remote node.

If you do not provide the correct passwords, your node and the remote node will be unable to communicate. The operator at your node will receive no messages to indicate that anything is wrong. The operator at the remote node will receive a message indicating that your node sent an incorrect password.

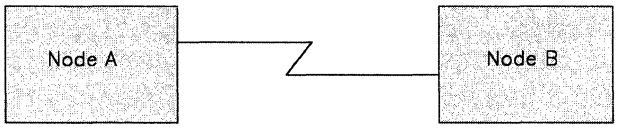
Note: If a node that is not expecting passwords receives them, the node ignores the passwords and does not issue a message.

The following examples show two ways to use passwords and how to specify passwords on the NJERMT statement.

Example 1:

Node A requires no passwords from Node B. Node B requires that Node A provide passwords P1 and P2.

```
NJERMT,NAME=B,PWD=P1,SIG=P2,...
```

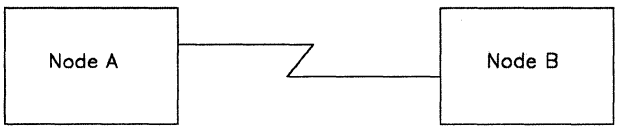



```
NJERMT,NAME=A,EXPWD=P1,EXSIG=P2,...
```

Example 2:

Node A requires that Node B provide two passwords, P3 and P4. Node B requires that Node A provide one password, P1.

NJERMT,NAME=B,PWD=P1,EXPWD=P3,Exsig=P4,...



NJERMT,NAME=A,PWD=P3,SIG=P4,EXPWD=P1...

Defining BSC Communication Lines

You must name and define each communication line that connects your BSC node to a directly-connected remote node. To name and define a line, code a DEVICE initialization statement (you must code one for each line). You must also include information about the line on the NJERMT statement that defines the remote node that is connected to the line.

DEVICE Statement

On the DEVICE statement for a line, you must code DTYPE=NJELINE. This parameter indicates that the line is part of the JES3 network.

To name the line, code the JNAME= parameter. If an operator uses the start command to start the line, this is the line name the operator must specify.

You must also code the JUNIT parameter to specify:

- The line address
- The name of the global processor to which the line is attached and the names of any local processors that can assume the role of the global processor
- The message class to be used for messages about this line

The default DGROUP for a network device is NETWORK.

NJERMT Statement

On the NJERMT statement that defines the remote node, you can optionally:

- Identify the line to the remote node
- Specify the type of line
- Specify the maximum number of lines that an operator can start for communicating with a remote node
- Specify whether JES3 is to transmit one data stream at a time or is to interleave the transmission of two data streams over the line

- Specify whether JES3 is to automatically restart the line if the remote node interrupts data transmission

To identify the line to the remote node, specify the name of the line--you named the line with the JNAME parameter on the DEVICE statement--as the subparameter of the LINE= parameter. If you do not code the LINE= parameter, the operator must specify the line name when he starts the line. When starting the line, the operator can override the line name that you have specified with the LINE= parameter.

To specify the type of line that connects your node to the remote node, use the CTC= parameter. If the nodes are connected by a channel-to-channel adapter, code CTC=YES. If the nodes are connected by a leased line or by a dial-up BSC line, omit the CTC= parameter or code CTC=NO.

To specify the maximum number of lines that an operator can start for communicating with a remote node, code the MAXLINE= parameter. You can specify from 0-3 lines. If you omit this parameter, the operator will be able to start one line.

When JES3 transmits networking jobs to another node, it can transmit one job after another or it can interleave the transmission of one job with the transmission of another job. You specify how JES3 is to transmit networking jobs by coding the STREAM= parameter on the NJERMT statement that defines the remote node.

When transmitting one job after another, JES3 transmits one complete job followed by another complete job. JES3 continues this pattern until it has transmitted all of the jobs. The position of the jobs in a JES3 queue determines if JES3 transmits a job stream or transmits a SYSOUT stream.

When JES3 interleaves the transmission of jobs, JES3 transmits part of a job stream, followed by part of a SYSOUT stream, followed by another part of the job stream, and so forth. JES3 continues this pattern of transmission until all jobs are transmitted.

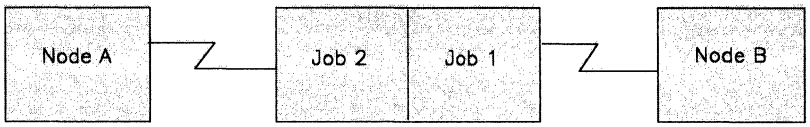
If you want JES3 to transmit one complete networking job at a time, code STREAM=1. If you want JES3 to interleave the transmission of networking jobs, code STREAM=2. These are the only valid subparameter values for the STREAM= parameter.

You should consider using interleaving only when the SYSOUT streams are large compared to the size of the job streams. Without interleaving, a large SYSOUT stream will tie up the line and could delay the transmission of job streams. From a performance point of view, however, it takes longer for JES3 to transmit an interleaved job stream than it does for JES3 to transmit the same job stream one job at a time.

In the following examples, Node A transmits two jobs to Node B. In Example 1, Node A transmits one complete job at a time. In Example 2, Node A interleaves the transmission of Job 1 with the transmission of Job 2.

Example 1:

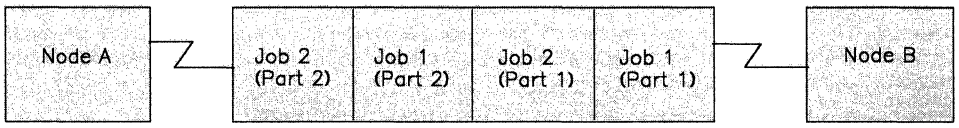
NJERMT,NAME=B,STREAM=1,...



job stream

Example 2:

NJERMT,NAME=B,STREAM=2,...



job stream

Note: The size of each part that a node transmits depends on the value specified for the BFSIZ parameter. The number of parts into which JES3 Networking divides a job depends on the value specified for the BFSIZ parameter as well as the size of the job.

It is possible, because of problems, for either node to interrupt data transmission. If you want JES3 to automatically restart the line after such an interruption, code AUTO=YES. If you do not code AUTO=YES and one of the nodes interrupts transmission, the operator will have to restart the line. If you do not want JES3 to automatically restart the line, omit the AUTO= parameter or code AUTO=NO.

When you code AUTO=YES, you can also specify a time delay value. This value specifies how long JES3 is to wait before it restarts the line. The delay gives the remote node time to terminate the line and to reinitialize it. The amount of delay you specify depends on the system installed at the remote node. For a remote node with JES3, specify a delay of at least two minutes. For a remote node with either JES2, VM/RSCS, or VSE/POWER, specify a delay of at least three minutes. If you specify a delay that is too short, your node will be unable to sign on to the network after the remote node has interrupted transmission.

To specify the delay, code the RDLY= parameter. The subparameter specifies the time delay in minutes and can range from 0-99. If you omit this parameter, JES3 uses a delay of five minutes.

Logical Senders: How JES3 Names Them

To vary a node online or offline, you must specify the names of a logical senders on the JES3 *VARY command. Use the JES3 inquiry command, *I NJE, to list the names of the logical senders. To help identify the names, you may wish to understand how JES3 names logical senders.

A logical sender is a type of logical device that JES3 uses to communicate with a directly-connected node. JES3 creates and names one or two logical senders for each line that is connected to a directly-connected node provided the NJERMT statement for that node specifies MAXLINE=1, 2, or 3.

The MAXLINE parameter and the STREAM parameter on the NJERMT statement determine how many logical senders JES3 creates for a node. If MAXLINE=1, 2, or 3 and STREAM=1, JES3 creates one logical sender per line. If MAXLINE=1, 2, or 3, and STREAM=2, JES3 creates two logical senders per line. (If MAXLINE=0, JES3 creates no logical senders.)

To create a logical sender name, JES3 starts with a 6-character base name. The form of the base name is XYYYYY, where X is an alphanumeric character and YYYYY is a number between 00001 and 99999. JES3 verifies that the base name is unique by checking the base name against a list of existing base names. If the new base name is unique, JES3 adds it to the list. If the name is not unique, JES3 makes it unique by changing one or more characters of the name.

Figure 9-2 shows how JES3 completes the logical sender name by adding a two-byte suffix to the base name. The content of the suffix depends on the number of logical senders that JES3 created for the node.

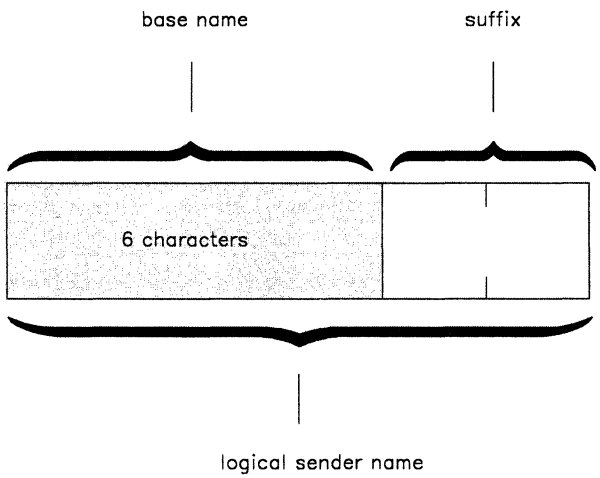


Figure 9-2. How JES3 Creates a Logical Sender Name

Figure 9-3 shows, for all valid combinations of the MAXLINE and STREAM parameters, the number of logical senders JES3 will create and the suffixes JES3 will append to the base name.

		MAXLINE=			
		0	1	2	3
STREAM=1	number of logical senders created	0	1	2	3
	suffix used	none	SN	S1 S2	S1 S2 S3
STREAM=2	number of logical senders created	0	2	4	6
	suffix used (See note)	none	JB OP	J1 O1 J2 O2	J1 O1 J2 O2 J3 O3

Note: JES3 appends a J suffix (J1, J2, or J3) to logical sender names created for lines that will transmit job streams. JES3 appends an O suffix (O1, O2, or O3) to logical sender names created for lines that will transmit SYSOUT streams.

Figure 9-3. The Number of Logical Senders Created and Suffixes Used for Valid Combinations of the MAXLINE and STREAM Parameters

SNA Considerations

The following topics apply to defining nodes that are using the SNA networking protocol.

Changing Node Definitions

Because SNA protocol uses MVS/BDT to transfer data between nodes, node names must be defined in both the JES3 and MVS/BDT initialization streams.

If a node name is changed, added, or removed from one of the initialization streams, the same change should be made to the other stream. If a change is made to only one initialization stream, jobs directed to the affected node will be held and JES3 or MVS/BDT will issue a message.

How Restarts Affect Networking Jobs

Because JES3 and MVS/BDT are separate address spaces, the failure of one system does not impact processing in the other. If your installation is using the SNA networking protocol, networking jobs or SYSOUT that are being sent to another node should not be lost if either system fails. JES3 and MVS/BDT automatically take certain actions according to the type of restart that is required. Generally, if JES3 fails, MVS/BDT must wait to access SNA/NJE work; if MVS/BDT fails, JES3 will not send SNA/NJE work.

JES3 Hot or Warm Start: If JES3 requires a hot or warm start, networking jobs or SYSOUT that were sent to MVS/BDT, but not acknowledged as queued, are automatically resent. This is called *synchronization* of the queues, and, depending on the workload at the time of the restart, might require some time to accomplish. Other JES3 processing continues.

JES3 Cold Start: All jobs on the JES3 spool are lost if a JES3 cold start is required. During the connect processing between JES3 and MVS/BDT, MVS/BDT detects that the jobs are lost, and consequently purges all networking jobs from its work queue.

MVS/BDT Hot or Warm Start: When an MVS/BDT hot or warm start is performed, JES3 automatically resends all NJE transactions that were not acknowledged as queued by MVS/BDT. This synchronization can also be time-consuming.

MVS/BDT Cold Start: If an MVS/BDT cold start is required, the MVS/BDT work queue is erased. To synchronize the queues, JES3 resends all NJE transactions.

Rerouting Jobs and SYSOUT

Job and SYSOUT streams can be rerouted to other nodes within your network. The home node, as well as directly-connected and indirectly-connected remote nodes, can receive rerouted networking streams. Job and SYSOUT streams, however, cannot be rerouted *from* the home node using this facility.[1]

The networking protocol that a node is using does not affect the rerouting of job or SYSOUT streams. Job or SYSOUT streams that are currently destined to a node that uses BSC protocol can be rerouted to a node using SNA protocol, or the reverse. Rerouting is especially useful if you are converting from one protocol to the other, since some jobs may have been queued for one resource or the other during the conversion period.

To begin the rerouting process, the operator must call the NJEROUT DSP. The *START and *RESTART commands can then be used to reroute specific jobs.

Jobs rerouted from one node to another can still fail during execution, particularly if system resources or installation defaults differ. It is important to prepare procedures *in advance* for handling problems that can arise as a result of rerouting.

---

1 If a node is using the BSC protocol, SYSOUT streams can be rerouted using the *F,U,Q=WTR,ND= command.

Networking Job Numbers

Networking jobs are assigned a job identification number by JES3 when they are submitted at a given JES3 node. If possible, the same number is reassigned if the job requires execution at another node. If output must be returned to the node where the job was submitted, JES3 again attempts to reassign the original job number whenever possible. JES3 assigns another job number only if the original number is not available.

Defining a Network Message Class

You can define the message class to which JES3 is to send network messages. To do this, code the message class on the CLASS parameter on the NJECONS statement. If you do not specify a message class, JES3 sends all network messages to class S12.

Any JES3 console whose associated CONSOLE statement specifies the network message class will receive network messages.

Defining a Network Console

You must use a CONSOLE statement to define one logical console for the network. On this statement, code JNAME=NETWORK and TYPE=NJE. These are the only parameters you should code.

Although this CONSOLE statement does not define a real console, it does cause JES3 to create an entry in the JES3 console status table. JES3 Networking needs this entry.

Defining a Command Routing Table

Each time a remote node sends a command to your node and expects a response, JES3 must keep track of which console gets the response. To do this, JES3 creates an entry in a table called the network command table. You must define the number of entries in this table.

To define the number of entries in the network command table, code the SIZE= parameter on the NJECONS statement. The subparameter for the SIZE= parameter can be a value in the range 1-4095. If you omit this parameter, JES3 creates a network command table with 32 entries.

The value that you specify for the SIZE= parameter depends on the number of commands you expect to receive and how frequently you expect to receive them. The more commands you receive or the faster you receive them, the larger the table should be. Initially, start with the default value and change it as experience dictates. A slow response time can indicate that the default value is too small.

Monitoring the Job Entry Network with User Exits

Several user exits enable you to monitor job streams, SYSOUT streams, or commands that are sent over the network. The routines that you write for these exits can:

- Supplement or replace the validity checking that JES3 does for commands received at your node
- Inspect or change the header records that accompany job or SYSOUT streams sent to or from your node
- Collect accounting information about job or SYSOUT streams received at your node

Figure 9-4 lists each user exit, states when JES3 passes control to the exit, and states what the exit routine can do.

User Exit	Receives Control:	The exit routine can:
IATUX35	After your node receives commands from another node.	• Validity check the command. • Request that JES3 bypass validity checking of the command.
IATUX36	After your node receives a job or SYSOUT stream from another node.	Collect accounting information from the job header and fill in the job management record (JMR).
IATUX37	After your node receives a SYSOUT stream from another node.	Inspect or change the data set header.
IATUX38	After your node receives a SYSOUT stream from another node.	Change the SYSOUT class.
IATUX39	Before your node sends a SYSOUT stream (that was created at your node) to another node.	Inspect or change the data set header.
IATUX40	Before your node sends a job stream to another node.	Inspect or change the job header.
IATUX42	After your node receives a file that was sent by means of the TSO TRANSMIT or CMS SENDFILE command.	To find out what IATUX42 can do, see the topic "Monitoring Files Sent via TSO/E TRANSMIT or CMS SENDFILE" in this chapter.
IATUX43	Before your node sends a SYSOUT stream to another node.	Inspect or change the job header.

Figure 9-4. Network User Exit Summary

Figure 9-5, Figure 9-6, and Figure 9-7 group the user exits according to the type of events that cause the exit routines to receive control.

- Figure 9-5 lists the user exits that receive control when your node sends or receives a networking job.
- Figure 9-6 lists the user exits that receive control when your node sends or receives a SYSOUT stream.
- Figure 9-7 lists the user exit that receives control when your node receives a command from another node.

When a figure lists more than one user exit, they are listed in the order they receive control.

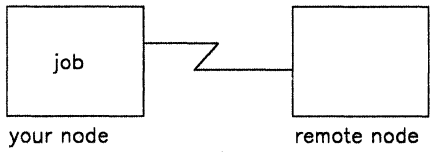
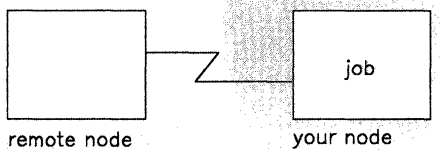
When	JES3 at your node invokes user exit
A job that was submitted at your node is to be sent to a remote node.  	IATUX40
your node receives a job from a remote node and your node is to execute the job.  	IATUX36

Figure 9-5. Job Related User Exits

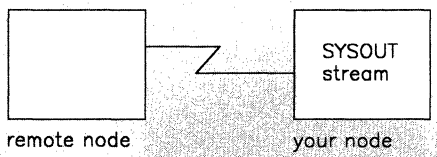
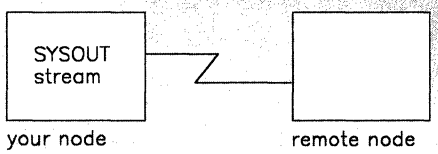
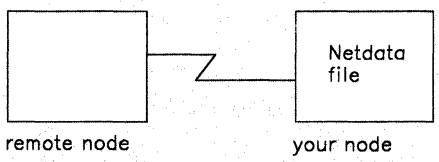
When	JES3 at your node invokes user exit
your node receives a SYSOUT stream that it is to process.  	IATUX36 IATUX37 IATUX38
your node sends a SYSOUT stream to a remote node.  	IATUX39 IATUX43
your node receives a file that was transmitted from another node by means of the TSO/E TRANSMIT or CMS SENDFILE command.  	IATUX36 IATUX37 IATUX38 IATUX42

Figure 9-6. SYSOUT Related User Exits

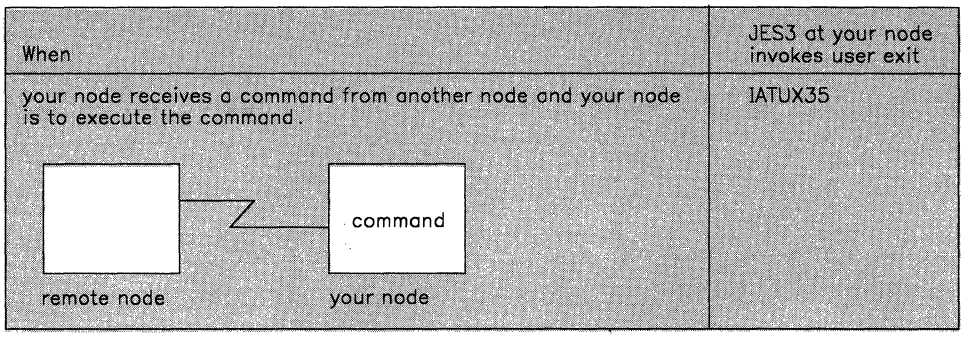


Figure 9-7. Command Related User Exits

Monitoring Files Sent via TSO/E TRANSMIT or CMS SENDFILE

A user at another node can send a file to a user at your node by using the TSO/E TRANSMIT or CMS SENDFILE command. (The TSO/E TRANSMIT command is part of the TSO/E interactive data transmission facility, described in *TSO/E User's Guide*. The facility is part of Program Product 5665-285.) To receive the file, the user at your node (this user is called the receiver) must issue the TSO/E RECEIVE command.

When an incoming file arrives, JES3 does not notify the receiver. The receiver finds that there is a file only after issuing the TSO/E RECEIVE command.

To notify the receiver that an incoming file has arrived or to monitor incoming files, write an exit routine for user exit IATUX42. This exit routine can:

- Check the validity of a file's control information
- Accept a file and notify the receiver that the file has arrived
- Delete a file
- Notify the sender of a file that the file has been deleted

To find out how to code IATUX42, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*

Deleting Files from the Spool

Files sent to a user by means of the TSO/E TRANSMIT or CMS SENDFILE command are written to the spool. They stay there until the user issues a RECEIVE command. At that time, JES3 frees the spool space occupied by the files.

It is possible for one or more of these files to remain on the spool indefinitely. This will happen when files are sent to a user who does not issue the RECEIVE command frequently--the user could be on vacation, for example. It will also happen if files are sent to a user whose user identification is not known to your system. Depending on the size and number of these files they could tie up considerable spool space.

One way to prevent such files from tying up spool space is to write an exit routine for user exit IATUX42. The exit routine can:

- Delete files that are sent to a TSO user id not known to your system
- Delete files that are sent to a user who will not issue the RECEIVE command for some time

Chapter 10. JES3 Start-Up and Initialization

Each time JES3 starts, initialization occurs. During initialization, MVS executes the JES3 cataloged procedure. This causes MVS to allocate the data sets required by JES3. Depending on the type of start specified, JES3 will process the JES3 initialization stream thus initializing itself.

You can tailor JES3 by modifying the JES3 cataloged procedure, by modifying the JES3 initialization stream, or by using an alternate JES3 initialization stream.

Starting JES3

There are eight ways to start JES3:

- Hot start
- Hot start with analysis
- Warm start
- Warm start with analysis
- Warm start to replace a spool data set
- Warm start with analysis to replace a spool data set
- Cold start
- Local start

Local start is the only type of start that you can use to start or to restart a local processor. All other types of starts can be used to start or to restart the global processor.

When you should use each type of start depends on why you need to start or restart JES3. The rest of this topic discusses the purpose of each type of start, what it does, and when you should use it. For instructions on how to perform each type of start, see the *MVS/Extended Architecture Operations: JES3 Commands.*

Hot Start

Use a hot start to start JES3 on the global processor:

- After an orderly shutdown
- After a JES3 failure on the global processor from which JES3 cannot automatically recover

- After an MVS failure that causes termination of all functions in the global processor
- To replace one of your checkpoint data sets

JES3 does not read the initialization stream during a hot start. Instead, JES3 initializes itself by using the parameter values established at the last cold start or warm start. In addition, certain changes made by the operator after the last restart remain in effect. For a list of these changes, see *MVS/Extended Architecture Operations: JES3 Commands.*

If you perform an IPL before the hot start, JES3 restarts jobs that were executing on the global processor and are eligible for restart. If a job is not eligible for restart, JES3 processes it based on the job's failure options. For an explanation of the failure options, see the FAILURE parameter on the CLASS initialization statement. JES3 also restarts all FSS address spaces that were executing on the global processor and reschedules all jobs that were active in the FSS address space at the time of the IPL.

If you do not perform an IPL before the hot start, all jobs and FSS address spaces that were executing before the hot start continue to execute after the hot start.

After a hot start, JES3 continues to process jobs that were in the job queue before the hot start. Internal job numbering resumes with the next available job number.

Jobs and FSS address spaces executing on local processors before the hot start continue to execute after the hot start.

Any changes to the definition of an FSS address space brought about by using the *MODIFY operator command before the hot start remain in effect across a hot start with or without an IPL.

Overflow partition relationships, including any changes brought about by using the *MODIFY operator command before the hot start, also remain in effect across a hot start.

For information on using a hot start to replace a checkpoint data set, see "Recovering from Permanent Errors on the Checkpoint Data Set" in Chapter 11, "JES3 Recovery."

Hot Start with Analysis

Use hot start with analysis after a hot start fails and/or you suspect problems with the JES3 job queue.

Hot start with analysis performs the same functions as hot start. In addition, JES3:

- Removes jobs that have job control blocks that contain invalid data from the job queue
- Records the job control blocks that are associated with the removed jobs (they are recorded in the data set defined by the DD statement //JES3SNAP)

You are not required to IPL or to restart the local processors, although you may optionally do so.

Warm Start

Use a warm start to restart JES3 on the global processor:

- After either type of hot start fails
- After an abend of the global processor because of a software or hardware failure
- When you want to change the initialization stream
- Following a system generation

During a warm start, JES3 reads and processes the initialization stream, saves jobs that are in the job queue, and terminates active jobs according to the job's failure options.

After a warm start, you must IPL and then restart all local processors. All FSS address spaces terminate at IPL time and are restarted by JES3 after IPL. JES3 then reschedules all jobs that were active in the FSS address space at the time of warm start. Any changes to the definition of an FSS address space brought about by using the *MODIFY operator command before the warm start are lost. JES3 continues to process jobs that were in the job queue at the time of the warm start.

Although you can change the initialization stream during a warm start, you should use caution when changing the following statements or parameters:

- BUFFER statement - Do not change the BUFSIZE parameter.
- CLASS statement
  - NAME parameter: If you delete a class name for which jobs are scheduled, JES3 will not schedule the jobs for execution.
  - GROUP parameter: If you delete a group name for which jobs are scheduled, JES3 will not schedule the jobs for execution.
- Do not change the sequence of CLASS statements. You can, however, add statements to or delete statements from the end of the sequence.
- CONSOLE statement - Do not change the sequence of CONSOLE statements. You can, however, add statements to or delete statements from the end of the sequence.
- DEVICE statement - Omitting one statement might cause a subgeneric split or loss of a device. Omitting all statements for a device causes loss of the device as a JES3-managed device.
- FORMAT statement - You can replace FORMAT statements with TRACK statements for formatted spool data sets.

- GROUP statement
  - NAME parameter: If you delete a group name for which jobs are scheduled, JES3 will not schedule the jobs for execution.
  - DEVPOOL parameter: If the devopt subparameter on a GROUP statement specifies GROUP, and you delete a device that is allocated to a job in that group, the job abnormally terminates.
- Do not change the sequence of GROUP statements. You can, however, add statements to or delete statements from the end of the sequence.
- MAINPROC statement - Do not change the sequence of MAINPROC statements. You can, however, add statements to or delete statements from the end of the sequence.
- OPTIONS statement - Do not change the SE parameter.
- SETNAME statement - The loss of an esoteric or generic name caused by omitting this statement will cause jobs that use the named device to terminate abnormally.
- TRACK statement
  - STT or STTL parameter: If the STT or STTL parameter was specified on a FORMAT statement that is being replaced by a TRACK statement, the STT or STTL parameter must also be specified on the TRACK statement.
  - You can replace FORMAT statements with TRACK statements if the corresponding spool data set is already formatted.

If you do not change internal job numbering (via the JOBNO parameter on the OPTIONS statement) during the warm start, JES3 uses the next available job number

Warm Start with Analysis

Use warm start with analysis when JES3 terminates abnormally and you suspect problems with the JES3 job queue, or when you change the level of JES3 and want to verify the integrity of the JES3 job queue across the change.

Warm start with analysis performs the same functions as warm start. In addition JES3:

- Removes jobs that have control blocks that contain invalid data from the job queue
- Records the job control blocks that are associated with the removed jobs (they are recorded in EBCDIC format in the data set defined by the statement //JES3SNAP)

After performing a warm start with analysis, you must perform an IPL and then a restart (local start) on all local processors.

Warm Start to Replace a Spool Data Set

Use warm start to replace a spool data set when you want to replace a spool data set. This type of start performs the same as a warm start in addition to allowing you to replace a spool data set.

After performing a warm start to replace a spool data set, you must perform an IPL and then a restart (local start) on all local processors.

For information on how to use this type of start to replace a spool data set, see "Replacing a Spool Data Set" in Chapter 4, "Defining and Managing Spool Data Sets."

Warm Start with Analysis to Replace a Spool Data Set

Use warm start with analysis to replace a spool data set when you suspect problems with the JES3 job queue and you want to replace a spool data set.

This type of start performs the same as warm start. In addition, you can replace a spool data set and JES3:

- Removes jobs that have control blocks that contain invalid data from the job queue
- Records the job control blocks that are associated with the removed jobs (they are recorded in EBCDIC format in the data set defined by the statement //JES3SNAP)

After performing a warm start with analysis to replace a spool data set, you must perform an IPL and then a restart (local start) on all local processors.

Cold Start

Use a cold start to start JES3 on the global processor:

- When all types of warm starts are unsuccessful
- If you have:
  - Changed the BUFSIZE parameter on the BUFFER initialization statement
  - Changed the SE parameter on the OPTIONS statement
  - Changed the STT or STTL parameter on a TRACK or FORMAT statement and want the change to become effective immediately for existing spool data sets
  - Changed the sequence of CLASS, GROUP, or MAINPROC statements. (You can add statements to the end of the sequence or delete statements from the end of the sequence without doing a cold start.)

During a cold start, JES3 reads and processes the initialization stream. JES3 also removes all jobs (jobs that were active and jobs that were in the job queue at the time of the cold start) from the system. Thus, you must resubmit all jobs after a cold start.

All FSS address spaces terminate at IPL time. Any changes to the definition of an address space brought about by using the *MODIFY operator command before the cold start are lost. To restart a C/I FSS address space after a cold start, you must have specified START=YES on its FSSDEF statement or you must issue the *F,F,FSSNAME=fssname,ST=YES operator command.

If JES3 is executing properly before the cold start, you can save jobs by using the dump job (DJ) facility. This facility is described in *MVS/Extended Architecture JES3 Diagnosis.*

Figure 10-1 summarizes the seven types of starts you can use to start the global processor. This figure uses the following mnemonics to identify each type of start:

- Hot start (H)
- Hot start with analysis (HA)
- Cold start (C)
- Warm start (W)
- Warm start with analysis (WA)
- Warm start to replace a spool data set (WR)
- Warm start with analysis to replace a spool data set (WAR)

Figure 10-1 uses the following legend:

N - no
Y - yes
O - optional
R - required

Type of start	Characteristics						
	Retains job queue	Reads initialization stream	Can replace spool data set	Spool data lost	IPL global	IPL locals	Job disposition (See notes)
H	Y	N	N	N	O	O	1,2
HA	Y	N	N	N	O	O	1,2,5
C	N	Y	Y	Y	R	R	3
W	Y	Y	N	N	R	R	4
WA	Y	Y	N	N	R	R	4.5
WR	Y	Y	Y	N	R	R	4
WAR	Y	Y	Y	N	R	R	4,5

Notes:

1. If you do not perform an IPL on the global processor, jobs that were executing at the time of the hot start continue to execute.
2. If you perform an IPL of the global processor, JES3 restarts jobs that are eligible for restart and were executing at the time of the hot start. JES3 processes jobs that are not eligible for restart according to the job's failure options.
3. All jobs must be resubmitted.
4. JES3 restarts jobs that are eligible for restart. JES3 processes jobs that are not eligible for restart according to the job's failure options.
5. JES3 removes jobs that could cause a restart failure from the job queue and records control blocks that are associated with the job.

Figure 10-1. Characteristics of Global Processor Starts

Local Start

Use a local start to start JES3 on a local processor:

- After a cold start or any type of warm start on the global processor
- After JES3 fails on the local processor
- After a processor complex has been partitioned

After a cold start or any type of warm start, initialization of the global processor must be complete before you can start or restart a local processor.

JES3 does not read the initialization stream to initialize a local processor. Instead, the global processor provides the initialization values that it read from the initialization stream during the last cold start or warm start.

Before using a local start on a partition of a processor complex, make sure that the main representing that partition is stopped. If the main is not inactive, spool damage could result.

If you do not perform an IPL on the local processor, jobs and FSS address spaces that were active at the time of the local start will continue to execute. If you do perform an IPL, JES3 processes jobs that were active according to the job's failure options. You must also restart any dynamic support programs that were running on a local main prior to an IPL. At the time the local processor is connected to the global, JES3 on the global processor restarts FSS address spaces that were active on the local processor before the local start.

For a local start with or without an IPL, changes to the definition of an FSS address space brought about by using the *MODIFY operator command remain in effect. The local start does not affect FSS address spaces executing on other processors.

Initializing JES3

You can tailor JES3 during initialization by:

- Modifying the JES3 cataloged start procedure
- Modifying the JES3 initialization stream
- Creating and using alternate initialization streams
- Coding a program for user exit IATUX15 (For a description of this user exit, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*)

Modifying the JES3 Cataloged Start Procedure

The JES3 cataloged start procedure contains the DD statements needed to allocate the data sets required by JES3. IBM provides a basic cataloged start procedure with JES3. During system generation, MVS stores this procedure in a SYS1.PROCLIB member. If the PRISUB parameter is coded on the SYSGEN SCHEDULR macro, MVS uses the name specified in the PRISUB parameter to name the member. Otherwise, MVS names the member JES3.

You can change this procedure by using the TSO EDIT command or the utility program IEBUPDTE. Before you can make changes, however, JES3 initialization must be completed. Changes do not take effect until you restart JES3.

You should not code a REGION= keyword on the EXEC statement of the start procedure. Doing so would affect performance by preventing JES3 from using the entire JES3 private area.

Figure 10-2 shows a sample of the JES3 procedure. This sample contains all of the required DD statements.

Figure 10-3 explains the purpose of each DD statement in the procedure.

```
//IEFPROC   EXEC  PGM=IATINTK,DPRTY=(15,15)
//STEPLIB   DD    DISP=SHR,DSN=SYS1.JES3LIB
//CHKPNT    DD    DISP=OLD,DSN=SYS1.JES3CKPT
//CHKPNT2   DD    DISP=OLD,DSN=SYS1.JS3CKPT2
//JES3JCT   DD    DISP=OLD,DSN=dsn
//spool1    DD    DISP=OLD,DSN=dsn
     .
     .
     .
//spoolnn   DD    DISP=OLD,DSN=dsn
//JES3OUT   DD    UNIT=00E
//JES3SNAP  DD    UNIT=AFF=JES3OUT
//JESABEND  DD    UNIT=AFF=JES3OUT
//SYSABEND  DD    UNIT=AFF=JES3OUT
//IATPLBST  DD    DISP=SHR,DSN=SYS1.PROCLIB
     .
     .
     .
//IATPLBnn
//JES3IN    DD    DISP=SHR,DSN=SYS1.PARMLIB(JES3IN00)
```

If the disk reader facility (DR) is desired, specify:

```
//JES3DRDS    DD    DISP=SHR,DSN=dsn
```

If MSS support is desired, specify:

```
//JSMSSTAB  DD    DISP=SHR,DSN=dsn
```

Figure 10-2. Sample JES3 Cataloged Start Procedure

If you introduce an error while changing the procedure, JES3 cannot be restarted. In this case, you must use another system (for example, the starter system) to change the procedure.

Statement	Notes	Description
//IEFPROC	R	Specifies the name of the JES3 job step task, load module IATINTK.
//STEPLIB	O	Defines the JES3 module library. If used, this library must contain at least the following modules: IATINTK, IATINGL, IATINSV, IATGRSQ, IATGRTX, IATSSDQ, IATSSVT, and IATUX15. This library must not contain any JES3 modules that reside in LPA. If JES3LIB statements are included, they override this statement.

//CHKPNT
&/or
//CHKPNT2
	R,S	Defines the JES3 checkpoint data set(s).
		At least one of the two checkpoint data sets must be allocated and cataloged prior to JES3 operation.
		Each checkpoint data set must be allocated as a single extent which begins and ends on a cylinder boundary. The size of each data set should be at least 2 cylinders on a 3330, 3340, 3350, 3375 or 3380 volume. See Chapter 6, "Defining and Managing JES3 Resources" for further allocation information.

//JES3JCT	R,D,S	Defines the JES3 job control table (JCT) data set. This data set must be allocated and cataloged prior to JES3 operation. The data set must be large enough to accommodate the maximum number of JCT records to be allocated concurrently during normal system operation.

Figure 10-3 (Part 1 of 2). Description of the Statements in the JES3 Cataloged Start Procedure

Statement	Notes	Description

//spool1
.
.
.
//spoolnn
	R,D,S	Defines the spool data sets. The installation selects the ddnames and data set names for these statements. The ddname for this statement must be the same ddname specified on the BADTRACK, FORMAT, or TRACK initialization statements. Spool data sets must be allocated and cataloged prior to JES3 operation. (These data sets may be any size; however, a minimum of 100 cylinders is recommended.)

//JES3OUT	R,D	Defines the data set upon which the JES3 initialization stream and initialization error messages are printed. This data set is deallocated after initialization is completed.
//JES3SNAP	O,D	Defines the data set used if JES3 produces a dump during warm start with analysis or hot start with analysis.
//JESABEND	O,D	Defines the data set used for a JES3 formatted dump. If omitted, a formatted dump of the JES3 control blocks will not be produced.

//SYSABEND
or
//SYSUDUMP
	O,D	Defines the data set for JES3 system dumps.

//IATPLBST
.
.
.
//IATPLBnn
	R,D	Defines the installation's standard procedure library. If concatenated data sets are defined, the JES3 procedure library update facility cannot be invoked for any of these data sets.
		
		If the JES3 procedure library update facility is used, the IATPLBST procedure library must be defined by a DYNALLOC statement and not by a DD statement.

//JES3IN	R	Defines the data set containing the JES3 initialization stream. This data set must be a blocked or unblocked partitioned data set. When JES3 is started, the operator can specify that the initialization stream is to be read from a card reader. The default initialization stream is read from SYS1.PARMLIB (member JES3IN00).
//JES3DRDS	O,D	Defines the partitioned data sets containing input for the JES3 disk reader facility. The maximum block size for this data set is 3200. Concatenated data sets may be used.
//JSMSSTAB	O,D	Defines the data set created by the MSS Table Build Program (IATSATAB). The data set must be allocated and cataloged prior to JES3 initialization.

Notes:

D - May be dynamically allocated.
O - Optional; include only if the indicated function is to be used.
R - Required statement.
S - Must be on a device that is shared by the global and all local processors.

Figure 10-3 (Part 2 of 2). Description of the Statements in the JES3 Cataloged Start Procedure

The following considerations and restrictions apply to DD statements in the JES3 cataloged start procedure.

- Do not code FREE=CLOSE on any DD statement.
- Do not code a ddname of JS3Dnnnn (nnnn is 4 decimal digits) on a DD statement. These ddnames define data sets dynamically allocated by the JES3LIB initialization statement.

- If the JESABEND or SYSABEND DD statements specify a printer that is also defined on a DEVICE statement, interleaved output at the printer can occur as a result of the ABDUMP task and the JES3 task writing concurrently.

- Do not use the ddname END on a spool DD statement.

- DD statements added to the JES3 cataloged start procedure:

  - Must refer to data sets that are cataloged in the master catalog or that are specified by volume serial number

  - Must not specify the * or DATA parameters

Modifying or Creating a JES3 Initialization Stream

The JES3 initialization stream defines the JES3 system configuration and the job processing options. During system generation, MVS creates a basic initialization stream and stores it in SYS1.PARMLIB member JES3IN00. To tailor JES3, you can modify the basic stream or create an alternate stream.

To modify the basic stream or to create an alternate stream, use TSO edit or a utility program such as IEBUPDTE. An alternate stream can be punched in cards or stored in a data set. If you store the alternate stream as a single member of a partitioned data set (PDS), you must name the member JES3INxx. If you store the alternate stream as a concatenated set of PDS members, you must name the first member JES3INxx. In both cases, xx must be two alphameric characters.

The following statements must be included in all initialization streams:

- DEVICE
- ENDINISH
- ENDJSAM
- FORMAT or TRACK
- MAINPROC

When modifying or creating an initialization stream, you must be aware of related initialization statement parameters. With related parameters, the value you code for one parameter influences the value you code for another parameter. Figure 10-4 identifies related parameters.

Primary Statement	Related Statement	
	Name	Related Parameters[1]
ACCOUNT	none	
BADTRACK	FORMAT	DDNAME
	TRACK	DDNAME (See "Formatting Spool Data Sets" in Chapter 4, "Defining and Managing Spool Data Sets.")
BUFFER	FORMAT	(See "Formatting Spool Data Sets" in Chapter 4, "Defining and Managing Spool Data Sets.")
	TRACK	(See "Formatting Spool Data Sets" in Chapter 4, "Defining and Managing Spool Data Sets.")
	OPTIONS	SE(BUFSIZE)
	CONSTD	FLAG(TKLIM)
CIPARM	STANDARDS	PRTY(PARM)
CLASS	STANDARDS	FAILURE(FAILURE) JOBSTEP(JOBSTEP) PRTY(PRTY)
	GROUP	NAME(GROUP) DEF(GROUP)
	SELECT	JOBMIX, CHOICE(IORATE)
	MAINPROC	NAME(MDEPTH, MLIMIT, SYSTEM)
COMMDEFN	none	
COMPACT	none	
CONSOLE	DEVICE	JNAME(JNAME) DTYPE(DEST)
	NJECONS	CLASS(DEST)
	RJPTERM	N(JNAME)
	MAINPROC	NAME(UNIT, MAIN) MDEST(DEST)
	PFK	N(PFK, SP)
	RJWS	
CONSTD	BUFFER	TKLM(FLAG, CONSBUF)
	CONSOLE	DEST(HARDCOPY)
DEADLINE	none	
DEVICE	BUFFER	BUFSIZE(RECORDS)
	CONSOLE	DEST(JUNIT)
	FSSDEF	FSSNAME
	MAINPROC	NAME(JUNIT, XUNIT)
	NJERMT	LINE(DTYPE, JNAME)
	OUTSERV	CARDSTOCK CARRIAGE, CHARS, FLASH, FORMS, MODIFY, WS=L, STACKER, TRAIN(FORMS, CARRIAGE, CHARS, FLASH, FORMS, MODIFY, LINELIM, STACKER, TRAIN)
	SETNAME	XTYPE(XTYPE)
	SETPARAM	MDSLOG(XUNIT) ADDRSORT(DEST)
DYNALDSN	none	
DYNALLOC	none	
ENDINISH	none	
ENDJSAM	none	
FORMAT	BUFFER	BUFSIZE

[1]The names in parentheses are the related parameters of the primary statement.

Figure 10-4 (Part 1 of 3). Related Initialization Statements

Primary Statement	Related Statement	
	Name	Related Parameters[1]
FSSDEF	DEVICE	FSSNAME
GROUP	MAINPROC	NAME(EXRESC) JOBCLASS(NAME)
	SETNAME	POOLNAME(DEVPOOL)
	CLASS	GROUP(NAME)
	SELECT	GROUP(NAME)
HWSNAME	SETNAME	NAMES(TYPE)
	STANDARDS	SETUP(TYPE)
INTDEBUG	none	
JES3LIB	none	
MAINPROC	CONSOLE	DEST(MDEST) UNIT(NAME)
	SELECT	NAME(SELECT)
MSGROUTE	MAINPROC	NAME(MAIN)
	CONSOLE	DEST(routcd)
	CONSTD	HARDCOPY(J)
NJECONS	CONSOLE	DEST(CLASS)
NJERMT	DEVICE	DTYPE(LINE) JNAME(LINE)
	SYSID	NAME(BDTID)
OPTIONS	BUFFER	BUFSIZE(JOBNO)
OUTSERV	DEVICE	see DEVICE
	SYSOUT	CHARS(CARRIAGE) THRESHLD
PKF	CONSTD	EDIT(M)
	CONSOLE	PFK(N),SP(N)
PROC	none	
RESCTLBK	none	
RESDSN	none	
RJPLINE	RJPTERM	N(T)
RJPTERM	DEVICE	JNAME(C)
RJPWS	none	
SELECT	CLASS	IORATE(CHOICE)
	GROUP	CLASS(DPRTY) NAME(GROUP)
	MAINPROC	SELECT(NAME)
SETACC	MAINPROC	NAME(VOL)
	DEVICE	XTYPE(VOL)

[1]The names in parentheses are the related parameters of the primary statement.

Figure 10-4 (Part 2 of 3). Related Initialization Statements

Primary Statement	Related Statement	
	Name	Related Parameters[1]
SETNAME	DEVICE GROUP HWSNAME	XTYPE(XTYPE) DEVPOOL(POOLNAMS) TYPE(NAMES)
SETPARAM	CONSOLE DEVICE HWSNAME	DEST(DAFETCH, MDSLOG, TAFETCH) DEST(ADDRSORT) TYPE (MSS)
SETRES	none	
SPART	CLASS FORMAT MAINPROC SYSOUT TRACK	SPART(NAME) SPART(NAME) SPART(NAME) SPART(NAME) SPART(NAME)
STANDARDS	CIPARM HWSNAME	PARM(PRTY) TYPE (SETUP)
SYSID	NJERMT	BDTID(NAME)
SYSOUT	OUTSERV DEVICE	CARRIAGE(CHAR) THRESHLD JNAME(DEST) DGROUP(DEST)
TRACK	BUFFER	BUFSIZE

[1]The names in parentheses are the related parameters of the primary statement.

Figure 10-4 (Part 3 of 3). Related Initialization Statements

Correcting an Invalid Subparameter

If you restart JES3 using an initialization stream that has an invalid subparameter for any of the parameters listed in Figure 10-5, JES3 uses the parameter's default (individual parameter descriptions in Chapter 12, "Initialization Statement Reference" define valid subparameters). By using the parameter's default, JES3 can continue to run. Thus, you have a running system that you can use to correct the initialization stream.

To warn you about an invalid subparameter, JES3 issues message IAT3255. This message identifies the affected parameter, the invalid subparameter, and the default value. If the initialization stream is on DASD or on magnetic tape, use the TSO EDIT command or a utility program such as IEBUPDTE to correct the invalid subparameter.

If you do not want JES3 to continue to run with the subparameter default, restart JES3 using the corrected initialization stream.

Statement	Parameter	Default
BUFFER	BUFSIZE	2036
	FD	128
	GRPSZ	135
	MINBUF	16
	PAGES	64,16,32
	SPLIM	10,25
CLASS	IORATE	MED
	JOURNAL	NO
	SPART	default spool partition
	TRKGRPS	1,2
COMMDEFN	LU	255
COMPACT	DEFAULT	N
CONSOLE (Non-RJP)	ALTCON	1st active JES3 console in the initialization stream
	DEPTH	50
	LEVEL	15
	TIME	2
CONSOLE (RJP)	DEPTH	50
	LEVEL	0
DEVICE (Define I/O device)	BURST	YES
	CARRIAGE	YES,STANDARD
	CB	N
	CHARS	YES,STANDARD
	CKPNT	200(punch),10000(3800),2000(3211),1000(others)
	CKPNTPG	200
	CKPNTSEC	200 pages
	FLASH	YES,STANDARD
	FORMS	YES,STANDARD
	FSSNAME	JNAME specification
	HEADER	YES
	MODE	With the FSSNAME parameter: FSS Otherwise: COMP
	MODIFY	YES,STANDARD
	NPRO	STANDARD
	PAGELIM	0+
	PM	With MODE=FSS: (LINE,PAGE) With MODE=COMP: LINE
	RECLIM	0+
	STACKER	YES,STANDARD
	TRAIN	YES,STANDARD
	XLATE	YES
	WC	STANDARD
	WS	STANDARD
FSSDEF	DSPCNT	2,1
	MAXASST	0
	START	YES
	TERM	NO
GROUP	BAR	16

Figure 10-5 (Part 1 of 2). Invalid Subparameter Default Selections

Statement	Parameter	Default
MAINPROC	STXTNT	primext: 50 secdext: 10 noext: 10 %min: 90
	SPART	default spool partition
	TRKGRPS	1,2
	USRPAGE	2
OPTIONS	DUMP	JES
	MT	OFF
	WANTDUMP	YES
OUTSERV	CB	N
	NPRO	90
	OUTLIM	16777215
	STACKER	C
PFK	E	YES
RESCTLBK	FCT	0
	RQ	0
RJPLINE	I (1-character line interface)	A
	I (mode of operation)	Y
	S	2400
RJPTERM	0	DC3
RJPWS	C	R
	PL	2
	PR	1
	PU	0
	RD	1
SELECT	CHOICE	FFIT
	MAGEL	14
	MAGER	0
	SAGEL	14
	SAGER	0
	SBAR	16
	SDEPTH	255
SETPARAM	ADDRSORT	YES
	ALLOCATE	AUTO
	DSN	0
	FETCH	YES
	REMOUNT	1
SPART	DEF	NO
	GRPSZ	135
	INIT	NO
	OVRFL	YES
	SPLIM	(10,25)
STANDARDS	BYTES	1500,WARNING
	CARDS	2,WARNING
	CICNT	(2,1)
	JESMSG	NOTSO
	INTPMID	01
	INTPROC	ST
	LINES	1,WARNING
	MAXJOBST	0
	MAXASST	0
	PRTY	0
	SETUP	JOB
SYSOUT	SPART	default spool partition
	TRKGRPS	1,2

Figure 10-5 (Part 2 of 2). Invalid Subparameter Default Selections

Organizing the Initialization Stream

Statements must be placed into the initialization stream in a specific order. Figure 10-6 shows how to order the statements.

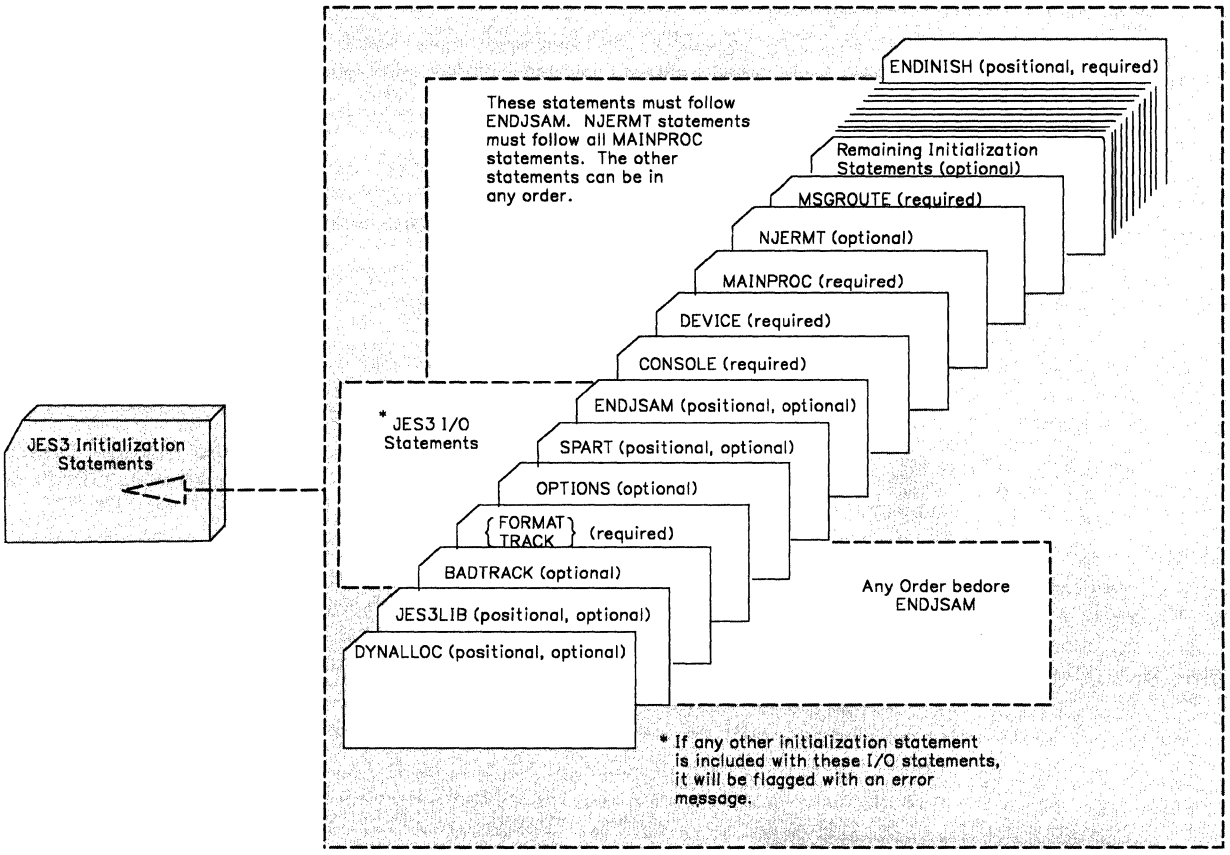


Figure 10-6. Structure of the JES3 Initialization Stream

Testing Your Initialization Stream

You can use the JES3 initialization stream checker utility to test your JES3 initialization statements before you perform a warm or cold start of JES3. The initialization stream checker detects most syntax errors and some logical errors in the initialization stream. You can execute this utility as a batch job or as a TSO job using a command list (CLIST).

How to Execute the Initialization Stream Checker

You need to perform two steps to execute the initialization stream checker:

1. Step 1 gathers data for the initialization stream checker.
2. Step 2 executes the initialization stream checker.

Step 1 obtains MVS configuration data that the initialization stream checker uses to detect logical errors in the DEVICE, HWSNAME, RJPLINE, and SETNAME initialization statements. If you omit this step, the initialization stream checker performs only syntax checking. The following sections contain sample JCL for obtaining configuration data.

Step 2 executes the initialization stream checker. The initialization stream checker examines all initialization statements for correct syntax, except the DYNALLOC and JES3LIB statements and creates the JES3 intermediate initialization tables.

The initialization stream checker detects logical errors in the DEVICE, HWSNAME, RJPLINE, and SETNAME statements by comparing the JES3 initialization data with the configuration data that you obtain in step 1. The initialization stream checker can detect the following types of logical errors:

- subgeneric splits; for example, devices defined to JES3 as belonging to one subgeneric group and defined to MVS as belonging to a different subgeneric group. See Chapter 6, *Grouping I/O Devices*, for information about defining subgeneric groups.
- writer burster-trimmer-stacker mismatches for the 3800 printer.
- missing or incorrect parameters required to define hardware.

How to Execute Step 1

If you want the initialization stream checker to check for logical errors in your initialization stream, you must first obtain the MVS configuration data by executing the MVS configuration program. Otherwise, you can omit this step.

MVS places the data in a file that you specify on the JES3= keyword in the MVS control program JCL procedure. You must place the data in a member of a partitioned data set to execute step 2. Use the same main names that you specify in your JES3 initialization stream when naming the members of a partitioned data set.

Step 1: Sample JCL for Executing the MVS Configuration Program

```
//MVSCP       JOB 'ACCTINFO','NAME',MSGLEVEL=(1,1),
//                MSGCLASS=R,REGION=4096K
//TEST        EXEC PROC=MVSCP,TYPE='IODEF'
//                JES3='INSTALL.JES3(SY1)'
//
```

See *MVS/Extended Architecture: MVS Configuration Program Guide and Reference* for a description of the MVS configuration program.

How to Execute Step 2

When you have obtained the configuration data for each processor in your complex, submit the JCL that executes the initialization stream checker. Your JCL should contain the following data sets:

DDNAME	PURPOSE
JES3IN	for the data set that contains your JES3 initialization stream. You can use a partitioned data set if you specify a fully-qualified name.
JES3OUT	for the data set to receive the initialization stream checker's output. The output consists of syntax error messages, logical error messages, warning messages, and a copy of the initialization statements.
JES3LIB	for the data set(s) containing the JES3 load module library.
STG1CODE	for the partitioned data set created as a result of the MVS configuration program if you want the initialization stream checker to test for logical errors, otherwise specify a dummy data set.

IATPLBxx	for the data set containing the corresponding procedure library (or libraries), if you include PROC initialization statements in the initialization stream you are testing. If you do not include PROC initialization statements, you can allocate a dummy data set to avoid a warning message.
	
	The variable ***xx*** must match the parameter you specify on the LIB= keyword of the PROC initialization statement. For example, if you code the PROC initialization statements as:
	
	`PROC,LIB=ST,...`
	`PROC,LIB=ST,...`
	`PROC,LIB=S1,...`
	`PROC,LIB=T1,...`
	
	then you should allocate the following data sets for the initialization stream checker:
	
	`//IATPLBST  DD  DSN=...`
	`//IATPLBS1  DD  DSN=...`
	`//IATPLBT1  DD  DSN=...`

JSMSSTAB	for the data set containing the table created by the JES3 MSS table build routine, if you specify the MSS= keyword on the SETPARAM initialization statement.
JESABEND	an optional dummy data set that you can define to avoid a warning message.

Figure 10-7. Data sets Required to Execute the Initialization Stream Checker

Step 2: Sample JCL for Executing the Initialization Stream Checker

```
//INITCHK    JOB 'ACCTINFO','NAME',MSGLEVEL=(1,1),
//                MSGCLASS=R,...
//IATUTIS    EXEC PGM=IATUTIS,PARM='P=1F1R'
//STEPLIB    DD DSN=SYS1.JES3PLIB,DISP=SHR
//JES3LIB    DD DSN=SYS1.JES3PLIB,DISP=SHR
//             DD DSN=SYS1.JES3LIB,DISP=SHR
//JESABEND   DD DUMMY
//JES3IN     DD DSN=INIT.PARMLIB(JES3IN00),DISP=SHR
//JES3OUT    DD SYSOUT=*
//STG1CODE   DD DSN=INSTALL.JES3,DISP=SHR
//IATPLBST   DD DSN=SYS1.PROCLIB,DISP=SHR
//
```

Sample TSO CLIST for Executing the Initialization Stream Checker

```
ALLOC F(JES3LIB) DA('SYS1.JES3PLIB' 'SYS1.JES3LIB') SHR
ALLOC F(JESABEND) DUMMY
ALLOC F(JES3IN) DA('INIT.PARMLIB(JES3IN00)') SHR
ALLOC F(JES3OUT) DA(*)
ALLOC F(STG1CODE) DA('INSTALL.JES3') SHR
ALLOC F(IATPLBST) DA('SYS1.PROCLIB') SHR
CALL 'SYS1.JES3LIB(IATUTIS)' 'P=1F1R'
```

Note: If you have written your own exit routine for IATUX15 (which permits you to scan and modify initialization statements), the initialization stream checker invokes this routine after it performs syntax checking. If IATUX15 requires the P= parameter, specify that parameter on either the EXEC statement of the JCL procedure or the CALL command of the TSO CLIST that executes the initialization stream checker.

Storage Requirements

The initialization stream checker program does not require space in the JES3 address space, common service area (CSA), or pageable link pack area (PLPA). However, the initialization stream checker does require auxiliary storage space to contain the load module. The load module is the size of the JES3 nucleus (IATNUC) plus approximately 8K bytes of storage.

If you want the initialization stream checker to check for logical errors, you must allocate an additional 2048 bytes of auxiliary storage space for the partitioned data set that contains the output from the MVS configuration program.

When the initialization stream checker executes in your user address space, it occupies approximately 540K bytes of virtual storage.

Abends

If your initialization stream contains critical syntax errors, the initialization stream checker does not check for logical errors and terminates with a JES3 DM code of U001. See *MVS/Extended Architecture JES3 Diagnosis* for a description of JES3 DM codes. Correct the critical errors as described in JES3OUT, and resubmit the initialization stream checker.

Chapter 11. JES3 Recovery

Recovery procedures minimize system reinitialization time that may result from hardware and software failures. The recovery procedures discussed in this chapter are:

- JES3 and C/I Functional Subsystem Failsoft
- Alternate CPU Recovery
- Reconfiguring a Processor Complex
- Checkpoint/Restart
- Restarting JES3 After a Failure
- JES3 Checkpoint Data Set(s)
- Dynamic System Interchange
- Recovering from CTC Failures
- BSC RJP Recovery
- Recovering From Output Writer Functional Subsystem Failures

Recovering from spool I/O errors is discussed in "Recovering From Spool I/O Errors," in Chapter 4, "Defining and Managing Spool Data Sets."

Recovering from failures of C/I functional subsystem address spaces is discussed in "Recovering from C/I FSS Address Space Failures," in Chapter 3, "Defining and Managing C/I Service."

JES3 and C/I Functional Subsystem Failsoft

JES3 failsoft provides recovery facilities to avoid JES3 restarts whenever possible. JES3 failsoft in C/I functional subsystem (FSS) address spaces provides the same facilities to avoid C/I FSS abnormal termination whenever possible.

- For jobs, the process of restarting is determined by installation- and programmer-supplied restart parameters.

- For failing functions or DSPs, the system recovery facility and the JES3 JESTAE and failsoft facilities allow the function or DSP to recover, if possible, or be terminated. The failure is recorded in the SYS1.LOGREC data set. The values specified for the DUMP parameter and the WANTDUMP parameter on the OPTIONS initialization statement determine when and where JES3 takes a dump. When a critical function cannot recover, JES3 or the C/I FSS address space terminates.

JES3 retains the Failsoft Logout across a JES3 restart. You can retrieve the Failsoft Logout using the MVS D R command. This applies only to the JES3

address space (not for C/I or Output Service FSSs. JES3 retains only the most recent logout on subsequent failures.

Job Recovery

Jobs active on processors at the time of system failure are restarted according to installation- and programmer-supplied restart parameters. The action that JES3 takes for jobs affected by the failure depends upon the options specified on the FAILURE parameter. The user can specify the FAILURE parameter on the //*MAIN JES3 control statement. You (the system programmer) can specify it on the CLASS or STANDARDS initialization statements.

The order of overrides for the FAILURE parameter are: //*MAIN overrides CLASS or STANDARDS; CLASS overrides STANDARDS.

Valid options for the FAILURE parameter and the action JES3 takes for each affected job are:

- **CANCEL:** Print any job output that is in a SYSOUT class that is specified as TYPE=PRINT. After printing the output, JES3 cancels the job.
- **HOLD:** Place the job into the hold queue.
- **PRINT:** Print any job output that is in a SYSOUT class that is specified as TYPE=PRINT. Then place the job into the hold queue.
- **RESTART:** Restart the job from the first job step. The job will be restarted on the processor on which it was active.

Whenever a processor fails in an MVS system, the MVS checkpoint/restart facility (or warm start facility, if applicable) is invoked before the failure options are examined:

- The checkpoint/restart facility is used to record information about a job at programmer-designated checkpoints so that, if necessary, the job can be restarted at one of these checkpoints or at the beginning of a job step. Restarts can take place immediately (initiated by the console operator) or be deferred until the job is resubmitted.
- Any job with a journal data set will attempt warm start. The warm-start facility will ensure cleanup of any scratch VIO data sets for the job.

Job Journal Data Set Usage

The job journal is a sequential data set that resides on a spool volume of JES3. Unique to MVS, its function is to contain a set of selected job-related control blocks that are critical to automatic restart processing.

The job journal is necessary because MVS maintains its scheduler control blocks in the scheduler work area (SWA) in pageable storage, rather than on a job queue on external storage. When a job or the system fails, there is a resultant loss of the address space that contains the SWA and its job control blocks. Because it preserves up-to-date copies of certain critical control blocks, the job journal makes it possible to reconstruct the SWA. SWA control blocks will be

reconstructed to their state just prior to the failing step for automatic step restart. For automatic checkpoint restart they will be reconstructed as they appeared at the most recently issued CHKPT macro in the job step. This capability is available for the following kinds of restart:

- Automatic step restart
- Automatic checkpoint restart
- System restart (including completion of job or step termination)

Therefore, if a job does not have job journaling, automatic restarts cannot be used.

Job journaling is provided to a job in JES3 in one of three ways:

1. The job class of the job has requested journaling (JOURNAL=YES on the CLASS initialization statement).

2. The job has a //*MAIN statement with JOURNAL=YES overriding the job class table.

3. The job's JCL has either RESTART= on the JOB statement or RD= on the JOB or EXEC statement.

After a system failure and JES3 restart of the failing main processor, those jobs in execution that had requested job journaling will be MVS system restarted (or warm started). If a job is eligible for MVS automatic restart, the system will issue message IEF225D asking if the job should restart. If the job is not eligible for restart, or the operator indicates that restart should not be attempted, any scratch or VIO data sets the job had allocated will be deleted and the job will be terminated. Therefore it may be desirable for certain classes of jobs which make a significant use of scratch and/or VIO data sets to request job journaling.

Function or DSP Recovery

When JES3 or a C/I FSS address space abnormally terminates, the JES3 ESTAE recovery processing routines in the terminating address space are given control. These routines examine the function control table active at the time of termination to determine which function or DSP has failed.

JES3 uses two levels of ESTAE recovery processing. The lower level ESTAE receives control whenever JES3 abnormally terminates. The higher level ESTAE is entered only if ESTAE percolation occurs, either because of a failure in the lower level ESTAE or because the lower level ESTAE returned to the control program indicating that termination continue. Once the higher level ESTAE is given control, JES3 will be terminated.

The JES3 ESTAE retry routines pass control to the JESTAE exit routine of the failing function or DSP. JESTAE then diagnoses the error, performs pre-termination processing, and informs JES3 whether the failing function or DSP can recover or has to be terminated (quiesced). All other functions or DSPs remain in execution.

If a function or DSP has to be terminated, system resources are returned, all units listed in the function control table for the failing function or DSP are returned,

and the function or DSP is placed in a permanent (nondispatchable) AWAIT state.

There are certain functions that are critical to JES3 operation:

- Console service
- Spool space allocation (in a JES3 local address space or a C/I FSS address space)
- Initialization
- Job segment scheduler (in the JES3 global address space)
- FSSDRVR (in a C/I FSS address space)
- JES3 lower level ESTAE

If one of these functions fails, the ESTAE routine abnormally terminates JES3. It is then the responsibility of the operator to restart JES3.

For a JES3 local address space or a C/I FSS address space, the spool space allocation function is critical. If that function fails, the address space terminates and the operator must restart it.

Alternate CPU Recovery

Alternate CPU recovery (ACR) provides a tightly-coupled multiprocessing system with the ability to recover system operation on the operational processor after one processor fails. ACR recovers as much work from the failing processor as possible, and terminates work it cannot recover.

If JES3 was active on the failing processor, JES3 analyzes the function active at the time of the hardware failure (for example, the device path might be analyzed):

- If a critical JES3 function was active and cannot recover, JES3 is terminated. The operator must restart JES3.
- If a noncritical JES3 function was active and cannot recover, the function is either terminated or quiesced.

If the global processor is a tightly-coupled multiprocessor and a failure occurs on one of the processors, ACR attempts recovery on the other processor:

- If the CTC adapters to the local processors are attached to the failing processor, the local processors will continue until they require global service. Restarting of these local processors might require reconfiguration or switching to alternate CTC adapters, and either restarting of the global system or invoking dynamic system interchange (DSI) to get back a CTC connection.
- If the CTC adapters to the local processors are not attached to the failing processor, jobs on these local processors will continue executing.

Reconfiguring a Processor Complex

Reconfiguring a processor complex without restarting JES3 is possible if you initially define your JES3 complex to permit partitioning. During initialization, MAINPROC statements should be included for all configurations of mains that your complex might choose to use. A processor complex can be reconfigured from a single-image main into two partitioned mains; or the reverse, two partitioned mains into a single-image main. A single-image main also can have one side partitioned off, while still maintaining its single image name. The partitioned side can then be IPLed as a separate main. See "Defining Mains" in Chapter 7, "Defining and Managing JES3 Mains and Storage" for illustrated examples of processor complex configurations.

Reconfiguration requires careful planning of your hardware environment. For information about planning an I/O configuration that supports reconfiguring and for instructions on the reconfiguration process, see *MVS/Extended Architecture Planning: Recovery and Reconfiguration.*

With the appropriate main definitions, you can use partitioning and reconfiguration to:

- remove a failed partition
- perform scheduled preventive maintenance on a partition
- reinstate a previously inactive partition
- provide a backup main for another system in the JES3 complex or a system outside the JES3 complex

Adding or removing a JES3 main affects the job selection environment. When a processor complex is running in single-image mode, more processing power and storage resources are available than when running in partitioned mode. Consider the resources available for each main that will be active in the complex, and define your job selection modes appropriately. You may need to dynamically modify the selection modes for a main by using the *F,G,main,S,selectname command.

With the //*MAIN SYSTEM= facility, users can specify the mains on which they want their jobs to execute. Also, users can specify the job class or group (which may be associated with a certain main) in which they want their jobs to be processed. Jobs directed to a main that is removed will not execute. Similarly, jobs requiring resources only attached to a partitioned-off main will not run. If the main is not going to be returned to the complex, these jobs should be restarted in the SETUP phase.

Checkpoint/Restart

The checkpoint/restart technique provided under MVS is supported by JES3.

Checkpoint/restart is a technique for recording information about a job at programmer-designated checkpoints so that, if necessary, the job can be restarted at one of these checkpoints or at the beginning of a job step.

A checkpoint is taken when a user program issues the CHKPT macro instruction. (Refer to *MVS/Extended Architecture Data Management Macro Instructions* for more detail on the CHKPT macro.) This macro causes the contents of the program's virtual-storage area and certain system control information to be written as a series of records in a checkpoint data set. These records can then be retrieved from the data set if the job terminates abnormally or produces erroneous output, and the job can be restarted. Restart can take place immediately (initiated by the operator at the console) or be deferred until the job is resubmitted. In either case, the time-consuming alternative of rerunning an entire job is eliminated. Refer to *MVS/Extended Architecture Checkpoint/Restart* for further detail.

Restarting A Job

There are three types of restarts:

1. *Step restart:* from the beginning of a step.

2. *Checkpoint restart:* from a checkpoint within a job step. Checkpoints are established in a job step by coding the CHKPT macro for each checkpoint desired. This macro writes the contents of the program's virtual storage area and specific system control information, as a series of records, to a data set. These records can be retrieved from the data set if the job terminates abnormally or produces erroneous output, and the job can be restarted. Restart can take place immediately (initiated by the operator) or be deferred until the job is resubmitted.

3. *System failure restart:* by specifying the FAILURE=RESTART parameter on the //*MAIN control statement. In the event that the job cannot complete execution because of a system failure and the job is not eligible for automatic restart, JES3 will automatically reschedule the job from the beginning. See *MVS/Extended Architecture JCL User's Guide* for more information on the //*MAIN statement.

Automatic Restart: To use automatic restart, code the RD (restart definition) parameter on the JOB or EXEC control statement. JES3 creates a job journal for any job specifying the RD parameter. A job journal is established to hold restart information for each program in execution.

When a system failure occurs or a job step abnormally terminates and RD=R is specified on the JOB or EXEC statement, MVS attempts to restart the job. If checkpoints are taken, an automatic restart is attempted at the last checkpoint regardless of the RD parameter. When a job step abnormally terminates or a system failure occurs while the job is executing, and the installation has not implemented job journaling, these jobs are ineligible for automatic restart.

Deferred Restart: To use deferred restart, the RESTART parameter on the JOB statement must be specified. This parameter causes the job to restart at the beginning of the specified step of checkpoint. The SYSCHK DD statement is required when a job is submitted for deferred checkpoint restart. This statement must immediately follow the JOBLIB DD statement.

Refer to *MVS/Extended Architecture Checkpoint/Restart* for information about how to plan for and use checkpoint/restart.

Operator Restart Considerations

A job may abnormally terminate as a result of a hardware, programming, or system error. Such an error can occur any time during execution and could cost the loss of valuable machine time. The checkpoint/restart feature of the system is provided to allow a restart of an abnormally ended job either at the beginning of a job or at a checkpoint within a step. The programmer determines whether an automatic restart or a deferred restart is to be performed.

Automatic Restart

If the programmer provides for an automatic restart and the job abnormally ends, message IEF255D is issued asking if the indicated job should restart. The message may indicate the checkpoint id, thus allowing you to prevent repeated restarts at the same checkpoint or job step. When requested to authorize an automatic restart, the operator should reply YES, HOLD, or NO.

- Reply YES if the restart is to be performed at a specific checkpoint or job step for the first time. If a step restart is to occur and the step to be restarted used a card input data set that was not part of the SYSIN stream, you must return all cards read by the job step before it ended abnormally to the appropriate card readers. If a checkpoint restart is to occur, follow the programmer's instructions for replacing the input cards.

- Reply HOLD to defer the restart; for example, to permit another job to be run first. Enter the *MODIFY command with the release operand when you are ready to restart the job. Also, if desired, you may cancel the job. However, canceling the job may result in unrecoverable paging space or the failure of certain data sets to be deleted if virtual I/O is being used.

- Reply NO if no restart is to be performed. When you reply NO, and the programmer wants a restart to be performed, the job must be resubmitted for a deferred restart.

When V=R is specified, the restart may be delayed by the system waiting for the allocation of storage. If another job is using the required storage, you will not receive a message--only a delay. Enter a DISPLAY A command to see if a system task or other job is using the storage required by a job with a V=R region. You may stop or cancel the conflicting task or job. The system may ask you to mount data volumes other than those required at the beginning of the job. The job's I/O will be set up by JES3 for the first job step, not the step being restarted. Canceling a job in a dependent network will prevent successor jobs from executing if they are dependent upon successful completion of the canceled job.

Note: Any operator commands in the input stream of the job step being restarted will not be executed.

Deferred Restart

If the programmer provides for a deferred restart and the job abnormally ends, the job must be resubmitted to have this restart performed. To restart the job, the programmer must provide a restart stream for submission to the system through the system input reader. The JCL statements to be included are described in detail in the publication *MVS/Extended Architecture JCL User's Guide.*

The device configuration of the system at the time of restart need not be the same as it was when the job abnormally ended. However, enough devices must be available to satisfy the needs of the job step being restarted. The system under which a step restart is run need not be the same as it was for the job's original execution. However, a checkpoint/restart should be run under the original system unless the alternate system can meet the following restrictions:

- The release number is the same.
- The link pack area modules in use at the checkpoint must reside in the same storage locations.
- Jobs specifying V=R require an area of storage identical to the original area.

If the required storage is not available, the system will cancel the restart and you will receive message IEF2091 which states that virtual storage is unavailable for the job.

If the required storage is not available, it is for one of the following reasons:

- The link pack area expands into the required storage. This may occur if an initial program loading (IPL) has been performed after the original execution of the job and prior to the restart. If this does occur, contact the system programmer for a respecification of the system parameters and repeat initial program loading using the new values.
- The system queue area expands into the required storage. When this occurs, contact the system programmer for a respecification parameter and repeat initial program loading using the new SQA value.

When a job restarts correctly, you will receive two messages: IHJ0061 and IHJ0081. If, for V=R jobs, these messages do not appear, enter the DISPLAY A command to see if a system task or other job is using the required storage. You can then stop or cancel the conflicting job.

The system may ask that you mount volumes other than those required at the beginning of the job. The job's I/O will be initially set up by JES3 for the first job step, not the step being restarted. In addition, any card input data sets that have been used by the failing job step must again be made available to the system.

Restart of JES3-controlled jobs may be accompanied by messages IAT2006 and/or IAT2575. Refer to *MVS/Extended Architecture Message Library: JES3 Messages* for response to the messages.

Restarting JES3 After a Failure

After an MVS failure or a JES3 failure, you must restart JES3. After an MVS failure, you must also perform an MVS IPL. When restarting JES3, use the type of JES3 start that causes the least amount of disruption to your system.

Restarting the Global Processor

After an MVS failure on the global processor, you must perform an MVS IPL before restarting JES3. For a JES3 failure including abnormal termination of the JES3 address space, you need not perform the IPL, just restart JES3.

To restart JES3, if you do not suspect problems with the job queue, perform a hot start. If, however, you suspect problems with the job queue, perform a hot start with analysis.

If either type of hot start fails, perform a warm start. If you suspect problems with the job queue, perform a warm start with analysis. You should also perform a warm start with analysis, after an equipment failure causes JES3 to terminate.

If a permanently damaged spool data set or spool device causes a JES3 failure, you can reallocate the spool data set on the same device or on a different device. After reallocating the spool data set, you must perform a warm start to replace a spool data set.

If you also suspect problems with the JES3 job queue, perform a warm start with analysis to replace a spool data set. For an explanation of how to replace a spool data set, see "Recovering From Spool I/O Errors."

If you cannot restart JES3 with any type of hot start or warm start, perform an MVS IPL and then a cold start.

After any type of warm start or a cold start, you must perform an MVS IPL and then a local start on each local processor.

For additional information about each type of start and to determine the disposition of jobs after a restart, see "Starting JES3" in Chapter 10, "JES3 Start-Up and Initialization." For information about the sequence of commands you must specify to restart JES3, see the *MVS/Extended Architecture Operations: JES3 Commands.*

Assigning Global Processor Functions to a Local Processor

If you cannot restart the global processor, assign the functions of the global processor to a local processor. This local processor then becomes the global processor. For information on how to assign the global processor functions to a local processor, see "Dynamic System Interchange" in this chapter.

Restarting a Local Processor

You must restart a local processor after an MVS failure or a JES3 failure on the local processor. You must restart all local processors after performing a cold start or any type of warm start on the global processor.

After an MVS failure on a local processor, perform an MVS IPL and then a local start. After a JES3 failure, you need not perform the IPL, just the local start.

After an IPL on a local processor, JES3 processes jobs that were previously executing on the local processor according to their failure options.

If JES3 cannot be restarted on a local processor, logically remove the processor from the complex. To do this, the operator must issue the command *S,main,FLUSH.

JES3 Checkpoint Data Set(s)

The JES3 checkpoint data set(s), allocated using the JES3 cataloged procedure, provides the capability to warm start or hot start the JES3 system with minimum or no loss of system information.

The JES3 checkpoint facility writes job-related control block information to the JES3 checkpoint data set(s) at appropriate points in time during system processing; that is, as information changes in the system. This control block information is restored to the system after performing a hot or warm start. All other information is lost.

The JES3 checkpoint data set contains the information required to initialize either a global or local JES3 processor. This information consists of the following data areas:

- **JES3 complex status record (IATYCSR)**, containing the last known status of each processor in the JES3 complex.
- **Initialization dynamic allocation checkpoint record (IATYS99)**, identifying the data sets that must be dynamically allocated during JES3 initialization.
- **Spool volume checkpoint record (IATYVOL), spool partition checkpoint record (IATYSPR), BADTRACK checkpoint record (IATYBTR), partition track allocation table checkpoint record (IATYPTC)**, containing the initialization data required for accessing the JES3 spool.

- **Initialization checkpoint record (IATYICP)**, containing other initialization data and the spool record addresses of multirecord files that contain the remaining initialization data.

- **Checkpoint data area (IATYCKP)**, containing the spool record addresses of single record files and multirecord files that checkpoint the status of individual functions within JES3. These individual functions and the files related to them (whose addresses are contained in IATYCKP) are described below.

  - *Main Device Scheduler (MDS):* the MDS volume unavailable table, which contains the volume serial numbers of volumes unavailable to MDS processing, and the data areas indicating the online/offline status of real devices eligible for setup and of MSS virtual units.

  - *Output Service:* the job data accounting block (IATYJDA or JDAB), job data set control block (IATYJDS), and output scheduling element (IATYOSE) data areas, which contain the checkpoint data for the output service driver module.

  - *Deadline Scheduling:* the deadline scheduling queue data areas.

  - *JESNEWS:* the JESNEWS data set.

  - *TSONEWS:* the TSONEWS data set.

  - *RJPNEWS:* the RJPNEWS data set.

  - *Generalized Main Scheduling:* data areas containing information about GMS selection modes, execution resources and various GMS parameters.

  - *Device Fencing:* the device fencing data areas.

  - *Dependent Job Control (DJC):* the checkpointed net control block (IATYNCK), which contain entries for each DJC network in the complex.

  - *Functional Subsystems:* the FSS/FSA table checkpoint (IATYFCK) data area, which checkpoints functional subsystem and functional subsystem applications information.

IATYCKP also contains the range of job numbers assigned in the system and the JCT priority hold flags.

The JES3 checkpoint area is allocated to either one unique data set or two duplicate data sets. You can cause information to be checkpointed in the IATYCKP control block by issuing the JESCKPNT macro.

Recovering from Permanent Errors on the JES3 Checkpoint Data Sets

If a permanent I/O error occurs on one of the JES3 checkpoint data sets, your recovery options depend on whether or not you have allocated one or both checkpoint data sets. If you have allocated only one checkpoint data set and it develops a permanent I/O error, you must perform a cold start. If you have allocated both checkpoint data sets, then you may replace the data set having the I/O error over a hot or warm start.

If you replace one checkpoint data set with a new checkpoint data set during a warm or hot start, JES3 copies the checkpoint records it finds on the older checkpoint data set over to the new checkpoint data set. When the hot or warm start is finished, JES3 has two complete checkpoint data sets once again.

When you replace a checkpoint data set, be sure that the checkpoint data set you are *not* replacing contains a complete copy of all active checkpoint records. If you are not sure whether that data set is complete, use the MVS display command to see if there are any messages indicating problems with it. If it has any problems, you may have to perform a cold start.

Recovering from a Checkpoint Data Set Out-of-Space Condition

If either checkpoint data set runs out of space, the data set must be replaced. Recalculate the amount of space the checkpoint data set needs and allocate a new checkpoint data set that is larger than the old one. See "Determining the Size and Placement of the Checkpoint Data Set(s)" in Chapter 6, "Defining and Managing JES3 Resources" for the method to calculate checkpoint data set size.

Dynamic System Interchange

Dynamic system interchange is a process by which the JES3 global function can be assigned to a JES3 local processor which then becomes the new JES3 global processor. DSI can be used when:

- The global processor is not active.
- The installation wants a local processor assigned as the global processor.

If the global processor is not active, the operator can invoke DSI to keep the complex running. Once DSI is complete, JES3 on the old global processor can be reinitialized as a local processor without an intervening IPL, once it becomes available for reinitialization.

If the global processor is active but the installation requires that another processor be assigned as the global processor, the operator can invoke DSI. (This procedure could be used for such reasons as scheduled preventive maintenance or for alternate processor utilization.)

The basic distinction in the two uses for DSI is the way the global processor is disabled.

Disabling the Old Global

When your global is inactive and you need to perform a DSI, you should disable the global by performing a system reset. A system reset causes MVS and JES3 to terminate. All jobs that were executing on the global are lost; JES3 will reschedule them.

All FSS address spaces on the global are also lost. You must restart all FSS address spaces that were executing at the time of the system reset. (For instructions on how to restart FSS address spaces, see *MVS/Extended Architecture Operations: JES3 Commands.*)

If your global is active but you want another main to become the global, you must disable the old global by entering a *CALL,DSI and *START,DSI command on the MCS console attached to the old global. Before entering the *CALL,DSI command, you must complete all reconfiguration tasks that require JES3, such as stopping RJP to disable communication lines.

If you disable the global using a *CALL,DSI command, then any output writer FSSs that were active on the old global remain active when the new global attempts to connect to the old global. However, if the old global fails as a result of an IPL or system reset, all output writer FSSs that were active on the old global terminate.

Once you complete the DSI, you can reinitialize the old global as a local main, without an intervening IPL.

Starting a Local Main as a Global

DSI is started by entering the *CALL,DSI command on the master console of the local main that you want to make the new global. All FSSs that were executing on local mains at the time of the DSI, including the local that is to become the new global, continue processing during and after the DSI.

If you disable the global using a *CALL,DSI command, then any output writer FSSs that were active on the old global remain active when the new global attempts to connect to the old global. However, if the old global fails as a result of an IPL or system reset, all output writer FSSs that were active on the old global terminate.

If a failure occurs during DSI, you must perform a warm start.

Defining Dynamic System Interchange Procedures

During DSI, messages are issued calling for review of the installation-defined local and global processor DSI procedures. When defining these procedures, you should take the following restrictions and recommendations into consideration:

- The old global processor must be disabled. If it is not, job spool damage may occur and a cold start may be required. Your procedures should indicate the way in which the global processor is to be disabled.

- DSPs that were called from a console on the old global processor and issue input commands to that console should be canceled if the calling console will not be valid on the new global processor.

- Those global devices (devices defined via the JUNIT parameter on the DEVICE statement) that the user requires on the new global processor must be switched from the old global to the new global processor (if they are not already shared). Your procedure should indicate the way in which devices are to be switched to the new global processor.

- Functions using devices that cannot be shared with all processors or switched to the new global processor cannot continue after DSI. For example, if the old global processor and the new global processor are different processor models, they may not support the same set of devices. These functions should be specified in your procedure as nontransferable.

- When you perform a DSI, JES3 rebuilds the MSGROUTE and CONSOLE routing tables. This means that unless you symmetrically defined your processors, messages from called DSPs may appear on different MCS consoles on the new global.

- After a DSI, FSS address spaces continue operating on the same processor as before. FSS address spaces defined to operate on a specific processor depending on which processor is the global processor (that is, specifying paired system names on the SYSTEM parameter of the FSSDEF statement) change location, if necessary, the next time the FSS is restarted.

  Changes to the definition of an FSS address space brought about by using the *MODIFY command before the DSI remain in effect across the DSI.

- Jobs queued as a result of the SYSTEM=JGLOBAL or SYSTEM=JLOCAL parameters on //*MAIN statements prior to DSI are not requeued to the new global or local processor after the DSI. The jobs remain queued on the processor on which they were previously queued. If that processor becomes available, the jobs can execute.

- Each nonfailing local processor should be connected to the new global processor before a subsequent DSI is performed. If the local processor is not connected to the new global processor before another DSI is attempted, an IPL must be performed on the local processor before connecting it to the next new global processor. Your procedure should include the method for connecting eligible local processors to the new global processor.

- When SNA RJP is active on the old global processor, some VTAM operations must be performed before starting SNA RJP on the new global processor. These operations vary according to the level of VTAM installed, and the configuration. They are:

  1. For VTAM Level 3 or ACF/VTAM, when the new global does not have VTAM already started:

     a. Start VTAM

     b. Vary the application definition (which contains the JES3 application) online to VTAM.

     c. Vary the required network online to VTAM.

     d. Issue the *CALL,SNARJP command.

  2. In ACF/VTAM, when VTAM is active on the new global processor in a domain:

     a. Vary the cross-domain resources definition (which defines JES3 as a cross-domain resource in the domain with the old global processor) offline in all domains,

     b. Vary online the application definition (in the domain with the new global processor) that defines SNA RJP to VTAM,

     c. Vary online all new cross-domain resource definitions for all domains defining SNA RJP as being in the domain with the new global processor.

     d. Issue the *CALL,SNARJP command.

- After DSI completes, determine the status of the writer output multitasking facility by issuing the command *INQUIRY,MT. Then:

  - If the new global processor is a multiprocessor and the multitasking facility is off, turn it on by issuing the command *MODIFY,MT=ON.

  - If the new global processor is a uniprocessor and the multitasking facility is on, turn it off by issuing the command *MODIFY,MT=OFF.

Recovering from CTC Failures

There are several ways to recover from a CTC failure. The method you use depends on whether you have defined an alternate CTC address. (You define an alternate address using the same DEVICE initialization statement that defines the primary address.)

If you have defined an alternate CTC address and a failure occurs, JES3 issues messages IAT2070, IAT2071, IAT2075, and IAT2076:

- IAT2070 identifies the failing CTC address.
- IAT2071 identifies the alternate CTC address.
- IAT2075 reminds you that you must do the hardware switching needed to make the logical CTC paths available.
- IAT2076 is a WTOR message that enables you to attempt to reestablish communication using the primary address, to switch to the alternate address, or to stop using the CTC.

If you think the failure is intermittent or is temporary, you should attempt to reestablish communication using the primary address. You do this by replying REDRIVE to message IAT2076. If this fails or you think the failure is permanent, switch to the alternate address by replying SWITCH to message IAT2076.

If the error persists on both the primary and alternate addresses, reply NO to message IAT2076. After you reply NO, JES3 can no longer communicate with the local processor that is attached to the CTC adapter.

If you have defined only a primary CTC address and a CTC failure occurs, JES3 issues messages IAT2070 and IAT2072:

- IAT2070 identifies the failing CTC address.
- IAT2072 is a WTOR message that enables you to attempt to reestablish communication using the primary address or to stop communication over that CTC.

To attempt to reestablish communication, reply REDRIVE to message IAT2072.

If the error persists, reply NO to message IAT2072. After you reply NO, JES3 can no longer communicate with the local processor that is attached to the CTC adapter.

BSC RJP Recovery

The BSC remote job processing (RJP) facility attempts to automatically recover from errors and suspended operations that might normally require system restarts. If failures occur, BSC RJP permits analysis and selective termination of specific functions or lines rather than the entire BSC RJP function.

If a system failure occurs while the BSC RJP function is processing, the BSC RJP JESTAE exit routine initializes its retry registers and indicates to the abend function that BSC RJP is to be reinstated. The retry routine that is given control issues a message explaining the reason for the abend, and then attempts to recover from the error condition:

- If the error is associated with a line I/O event or timer event, the corresponding line is canceled immediately.
- If the error occurred during line starting, the line is varied offline and cannot be started until the operator varies the line online again.
- If the error occurred during line canceling, the line is lost to the JES3 system; BSC RJP continues to service the rest of the lines.
- If the error occurred while processing an operator message, the message is ignored.

If an error occurs during remote terminal access method (RTAM) processing, a message is issued explaining the reason for the abend, and the corresponding line is canceled immediately.

Recovering from Output Writer FSS Failures

JES3 keeps track of each data set sent to an output writer functional subsystem (FSS) until it receives notification that the data set has been printed. If an output writer FSS fails, JES3 makes all the data sets that were being processed by the FSS available for rescheduling.

An output writer FSS may be lost in either of the following situations.

- The output writer FSS fails, in which case JES3 also fails the writer driver in the JES3 global address space that sends work to the FSS.
- The writer driver in the JES3 global address space fails, in which case the output writer FSS associated with it also fails.

The output writer FSS address space may also be lost during an IPL or canceled by an operator.

In all of the above cases, JES3 recovers the work the output writer FSS was processing and reschedules it. You can, in effect, restart an output writer FSS by using operator commands that start a writer DSP for a page mode device. The writer DSP restarts the output writer FSS.

If you perform a hot start of the JES3 global address space, the output writer FSSs continue running in their own address spaces unless you specified the TERM=YES parameter on the FSSDEF initialization statement. (The TERM=YES parameter specifies that the FSS being defined be terminated if JES3 terminates due to an *RETURN or *DUMP operator command.) If the output writer FSS runs out of work before the JES3 global address space restarts, the output writer FSS remains idle until the restart. After the hot start, JES3 restarts the writer DSPs, both hot writers and dynamic writers, that were associated with output writer FSSs active prior to the hot start. The writer DSPs reestablish contact with the output writer FSSs and work continues as before the hot start. For information about the effect of a dynamic system interchange on an output writer FSS see "Dynamic System Interchange."

Recovering an IBM 3480 Tape Drive for a Stand-Alone Dump

You may need to recover an IBM 3480 tape drive when you want to use it for a stand-alone dump. The stand-alone dump program (SADMP) is an MVS service aid that you can use to dump the contents of storage from a system that has failed. For more information about the SADMP service aid, see *MVS/Extended Architecture System Programming Library: MVS Service Aids.* You need to recover an IBM 3480 tape drive when all of the following conditions exist:

- The global has failed.
- You want the IBM 3480 to receive the stand-alone dump from the global.
- The IBM 3480 is assigned to more than one main (multi-system assigned).

JES3-managed IBM 3480s are always multi-system assigned. A multi-system assigned IBM 3480 belongs to one or more local mains in your complex, as well as to the JES3 global. You must issue operator commands to give the global sole possession of the IBM 3480 tape drive.

You can determine if an IBM 3480 is assigned to a local main by entering the MVS DISPLAY U,,,ddd, where **ddd** specifies the device number of the IBM 3480. You must issue this command from a locally-attached MCS console.

Once you have determined which JES3 local mains are assigned to the IBM 3480, use the VARYL DSP to unassign the IBM 3480 from each JES3 local. For example, if you need to take a stand-alone dump on a device that is multi-system assigned at address 560, you would enter the following commands:

```
*X,VARYL
*S,VARYL,560,OFF
*C,VARYL
```

If a local processor has failed and you want to use the IBM 3480 for a stand-alone dump for that processor, enter a *VARY,ddd,offline command (where ddd specified the device number) from the JES3 global.

Local connect processing resynchronizes the JES3 and MVS device tables when you have completed running the stand-alone dump program and you have restarted the global.

Part II: Initialization Reference

Part II includes the following chapters:

- Chapter 12, "Initialization Statement Reference"
- Appendix A, "RMT Option Statements"
- Appendix B, "Remote Terminal Bootstrap (RTPBOOT)"

Chapter 12. Initialization Statement Reference

This chapter includes the rules for coding initialization statements, an explanation of the notation used in the format descriptions for each statement, and a description of each JES3 initialization statement and its parameters. For information on creating or modifying an initialization stream, correcting invalid parameters, organizing the initialization stream, and testing the initialization stream, see Chapter 10, "JES3 Start-Up and Initialization."

Coding Rules for Initialization Statements

You must observe the following rules when coding initialization statements:

- Code the statement name and parameters in columns 1-71. Column 72 must be blank if the statement is complete; column 72 may be blank or non-blank if the statement is continued. JES3 ignores columns 73-80.
- Use commas to separate the statement name from the first parameter and to separate one parameter from another.
- If you code a keyword parameter more than once, JES3 usually uses the last value coded. In some cases, however, JES3 treats a duplicate keyword as a continuation of a previously coded keyword of the same name. In other cases, when JES3 encounters a duplicate keyword, it issues a diagnostic message.
- If you wish to continue an initialization statement (individual statement descriptions specify whether a statement can be continued):
  - The last item you code in columns 1-71 must be a complete parameter (a parameter and all of its subparameters) followed by a comma or a subparameter followed by a comma.
  - If one or more blanks follows a comma, that statement is considered continued, and any data beyond the blank(s) is treated as a comment. If a comma appears in column 71 following a statement parameter, that statement is considered continued.
  - Column 72 optionally may contain any nonblank character if the statement is being continued. Use of a nonblank character in a statement that is *not* to be continued results in an error and message IAT3253 is issued.
  - On the continuation statement you may start coding in any column. The statement can complete on this line or be continued.

- You can place any number of comment statements (indicated by an asterisk in column 1) anywhere in the initialization stream. Comment statements cannot be continued; however, multiple comment statements can be included consecutively.

- Initialization statements can contain embedded comments provided that at least one blank separates the last parameter and the comment. For initialization statements that are continued, at least one blank must follow the continuation comma (with the exception of a comma in column 71).

- You should not rely on parameter default values to be sufficient for your environment. Test each default value used, and be alert to performance problems that may arise as your environment changes.

Examples: The following are examples of valid initialization statement continuation:

```
CLASS,NAME=ABC,DEF=YES,                                          X
MDEPTH=(SY1,10)

CLASS,NAME=ABC,GROUP=TESTGP,SYSTEM=(SY1,
      SY2)

CLASS,NAME=ABC,GROUP=TESTGP,SYSTEM=(SY1,SY2),                    X
DEF=YES

CLASS,NAME=ABC,DEF=YES,     This is a comment
      MDEPTH=(SY1,10)

CLASS,NAME=ABC,DEF=YES,     This is a comment
*This is a comment line
      MDEPTH=(SY1,           This is a comment                   X
*This is a comment line
                 10)
```

Notation for Initialization Statement Format Descriptions

The format descriptions of the JES3 initialization statements in this section use the notation described below:

- You must code the following exactly as they appear in the format descriptions:
  - comma ,
  - equal sign =
  - numbers
  - parentheses ()
  - period .
  - plus sign +
  - question mark ?
  - uppercase letters and words
- Lowercase letters and words represent variables for which you must substitute a value.
- Braces {} group related items. You must code one and only one of the items grouped within braces. Do not code the braces.
- Brackets [] group optional items. You can code none, one, some, or all of the items grouped within brackets. Do not code the brackets.
- An ellipsis ... (three consecutive periods) indicates that you can code the preceding item more than once. If the ellipsis follows a group of items that are enclosed in parentheses or brackets, you can code the entire group more than once
- An underlined subparameter is the default value. If you do not specify a value for a subparameter, JES3 will use the default.

Figure 12-1 summarizes the functions of each of the JES3 initialization statements. The individual statement descriptions follow Figure 12-1.

Statement	Function
ACCOUNT	Specifies default DSP accounting information.
BADTRACK	Identifies a defective track on the spool volume and prevents JES3 from using it for subsequent allocations from spool.
BUFFER[4]	Establishes the JES3 buffer pool and the size of JES3 DASD records.
CIPARM	Specifies variable converter/interpreter parameter default values.
CLASS	Specifies the characteristics of the JES3 job class.
COMMDEFN	Identifies the VTAM/JES3 interface.
comment	Provides the facility for comments in the initialization stream.
COMPACT	Identifies a compaction table.
CONSOLE	Specifies the characteristics of an operator console and assigns message classes to it.
CONSTD	Establishes console-related installation standards for line-editing, buffering, and message logging.
DEADLINE	Establishes deadline scheduling algorithms.
DEVICE[1,2]	Specifies the characteristics of each device defined to JES3.
DYNALDSN	Identifies protection requirements for dynamically allocated data sets or JES3 permanently-resident devices.
DYNALLOC	Dynamically allocates a data set or a device.
ENDINISH[1]	Identifies the end of the initialization stream.
ENDJSAM[1]	Identifies the end of the JES3 I/O statements which precede the bulk of the initialization stream.
FSSDEF	Defines the characteristics of a functional subsystem.
FORMAT[3]	Requests formatting for the JES3 spool data set.
GROUP	Specifies the characteristics of JES3 job class groups.
HWSNAME	Identifies groups of I/O units for high watermark setup.
INTDEBUG	Provides the facility to generate a dump associated with an initialization stream error message.
JES3LIB	Dynamically allocates and concatenates one or more JES3 STEPLIB data sets.
MAINPROC[2,5]	Identifies and specifies the characteristics of each processor in the JES3 complex.
MSGROUTE	Assigns MVS routing codes for each processor.
NJECONS	Defines the message class to which JES3 Networking is to send network messages.
NJERMT	Defines a node in a network of nodes.

[1]Required.
[2]Required for local device.
[3]Unformatted spool data sets need a FORMAT statement; formatted spool data sets need either a FORMAT statement or a TRACK statement.
[4]Can only be changed across a restart.
[5]Can not change the order of MAINPROC statements across a cold start.

Figure 12-1 (Part 1 of 2). JES3 Initialization Statements and Their Functions

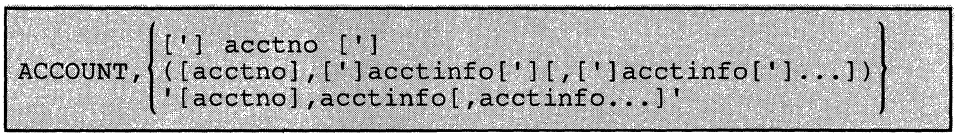
Statement	Function
OPTIONS	Specifies certain options for the JES3 system.
OUTSERV	Specifies output service defaults and standards.
PFK	Associates an operator command with a program function key or selector pen.
PROC	Identifies frequently used procedures.
RESCTLBK	Specifies that main storage be preallocated for the high-usage function control table (FCT).
RESDSN	Specifies the names of frequently used permanently resident data sets.
RJPLINE	Specifies the characteristics of a single BSC line that the JES3 global processor will use for remote job processing.
RJPTERM	Specifies the characteristics of a remote terminal for BSC remote job processing.
RJPWS	Specifies the characteristics of a remote work station for SNA RJP.
SELECT	Specifies the scheduling parameters for job selection modes.
SETACC	Defines nonshared permanently resident direct access volumes prior to processor initialization.
SETNAME	Specifies groups of devices within device types.
SETPARAM	Specifies main device scheduler (MDS) parameters for allocating I/O devices.
SETRES	Specifies which direct access volumes are to be made JES3 mounted at processor initialization.
SPART	Defines and names a spool partition.
STANDARDS	Specifies standard default values and job options for the JES3 complex.
SYSID	Defines the default MVS/bulk data transfer (MVS/BDT) node for the JES3 complex.
SYSOUT	Specifies the characteristics of a SYSOUT class.
TRACK[1]	Identifies a formatted volume that is allocated for the spool data set.
[1]Unformatted spool data sets need a FORMAT statement; formatted data sets need either a FORMAT statement or a TRACK statement.	

Figure 12-1 (Part 2 of 2). JES3 Initialization Statements and Their Functions

Note: Initialization statement keywords are interdependent. See Figure 10-4 for a chart of related JES3 initialization statements.

ACCOUNT (Job Accounting)

The ACCOUNT initialization statement defines default job accounting information. This default applies if an operator does not include an ACCT parameter with the JES3 *CALL command invoking a DSP. You can code an exit routine for user exit IATUX27 to inspect the accounting information. For more information on user exit IATUX27, see *MVS/Extended Architecture System Programming Library: JES3 User Modifications and Macros.*



[']acctno[']
Specifies an account number only. All characters except blanks are valid. If this account number contains any special characters, enclose the entire account number in apostrophes. If the special character is an apostrophe, specify it as two consecutive apostrophes. The maximum length of this parameter, including apostrophes, is 42 characters.

([acctno],[']acctinfo['],[']acctinfo[']...])
Specifies that accounting information consists of either the account number, with one or more items of additional information, or only the additional accounting information. All parameters within the parentheses are considered separate fields. The maximum length for accounting information, including parentheses, commas, and apostrophes, is 42 characters. All characters except blanks are valid. Enclose all special characters except hyphens in apostrophes. If the special character is an apostrophe, specify it as two consecutive apostrophes. **Note:** *If acctno is not specified, a comma must follow the left parenthesis.*

'[acctno],acctinfo[,acctinfo...]'
Specified if accounting information consists of either the account number, with one or more items of additional accounting information, or only the additional accounting information. All information within the apostrophes is to be considered one field.

All characters except blanks are valid. If the special character is an apostrophe, specify it as two consecutive apostrophes. The maximum length for accounting information (including apostrophes and commas) is 42 characters.

Statement Default: (0,0,0,0)

Examples: The following two examples specify only the account number.

1. An account number without special characters:

   ACCOUNT,12A75

2. An account number with two special characters--a period (.) and an apostrophe ('):

   ACCOUNT,'12A.75''20'

The following example specifies an account number and two items of additional accounting information. All three parameters are considered separate fields.

```
ACCOUNT,(12A75,DEPT-D58,706)
```

The following example shows the same parameters as those specified above; however, all information between the apostrophes is considered to be one field.

```
ACCOUNT,'12A75,DEPT-D58,706'
```

The following two examples show special characters in the account number parameter and in the first parameter of additional information.

```
ACCOUNT,('B.93','DEPT/99',BLDG002)
ACCOUNT,'B.93,DEPT/99,BLDG002'
```

The following examples do not supply an account number: however, the examples include other accounting information.

```
ACCOUNT,(,'H.J.WELLSCOMPANY',NYC,55-33,1,1,3)
ACCOUNT,',H.J.WELLSCOMPANY,NYC,55-33,1,1,3'
```

Override: The ACCT parameter on a JES3 *CALL command overrides information specified on the ACCOUNT initialization statement.

BADTRACK (Bypass Defective Tracks)

The BADTRACK statement identifies defective tracks on a spool volume. When a message indicates that a defective track is discovered, JES3 dynamically adds an entry to the BADTRACK table identifying the defective track. If possible, add a BADTRACK statement to your initialization stream at that time so that JES3 keeps a record of the defective track across a warm or cold start. If you cannot add the BADTRACK statement immediately, use the *INQUIRY,Q,BT command to display the necessary information and add the BADTRACK statement before the next warm or cold start.

Specify only one track on a statement. The maximum number of BADTRACK statements that you can specify is 32,768. You cannot use a BADTRACK statement to identify bad tracks in the single track table (STT).

```
BADTRACK,DDNAME=ddname
         ,CYL=nnnn
         ,TRK=nnnn
```

DDNAME =ddname
Specifies the name of the DD statement or matches the ddname on the DYNALLOC statement that defines the spool volume with the bad track.

CYL =nnnn
Specifies a 4-digit hexadecimal value identifying the cylinder that contains the defective track.

TRK =nnnn
Specifies a 4-digit hexadecimal value identifying the defective track.

Statement Default: None. If the initialization stream contains neither BADTRACK nor FORMAT statements and defective tracks existed on a spool volume before the most recent warm or cold start, I/O errors will occur when JES3 attempts to use the track group containing those tracks.

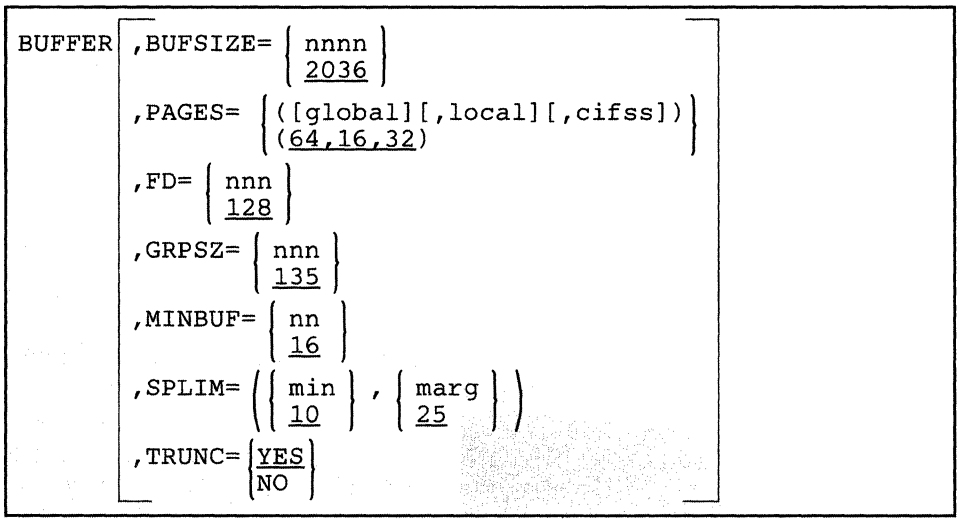
Associated Statement/Parameter: Do not substitute a FORMAT statement for a BADTRACK statement, although it also causes JES3 to identify defective tracks.

Example: In the following example, assume tracks 1 and 13 are bad on cylinder 10 of SPOOL2. If you warm start or cold start JES3 and a FORMAT initialization statement is not in the initialization stream, BADTRACK statements must be in the stream to prevent JES3 from using these tracks. If you do not include these BADTRACK statements, tracks 1 and 13 are assigned for I/O activity and data on them may be lost.

```
BADTRACK,DDNAME=SPOOL2,CYL=000A,TRK=0001
BADTRACK,DDNAME=SPOOL2,CYL=000A,TRK=000D
```

BUFFER (JES3 Spool Work Buffers)

The BUFFER statement defines the size of the JES3 buffer pool and the length of JES3 buffers and spool data set records. The number of buffers is computed at initialization and is based on the number of pages and buffer size specified on the PAGES and BUFSIZE parameters of this statement. For more information, see "Processors and Storage" in Chapter 7, "Defining and Managing JES3 Mains and Storage."



BUFSIZE = { nnnn / 2036 }

Specifies the length, in bytes, of the data portion of each JES3 buffer and of each record in the spool data sets. The value must be a multiple of four in the range from 1952 to 4084. If virtual storage in the JES3 global address space is not constrained, the recommended value is 4084. Otherwise, the recommended value is 2036. Buffers do not span page boundaries; therefore, specifying 2036 or less provides two buffers per page. Specifying a larger value provides one buffer per page.

Note that these recommended values optimize virtual storage usage but do not optimize spool usage. For further guidelines in helping you define this parameter, see "Determining the Size of the JES3 Buffers" in Chapter 7, "Defining and Managing JES3 Mains and Storage."

If you change the BUFSIZE parameter, you must cold start JES3 and reformat all of the spool data sets. For a discussion about how to format the spool data sets, see "Formatting Spool Data Sets" in Chapter 4, "Defining and Managing Spool Data Sets."

Parameter Default: 2036

PAGES = { ([global][,local][,cifss]) / (64,16,32) }

Specifies the size of the JES3 buffer pool in the JES3 global address space, all JES3 local address spaces, and all C/I functional subsystem (FSS) address spaces, respectively. Each value must be a number from 16 to 1024

that indicates the number of pages to be assigned to the buffer pool for that type of address space.

For guidelines to help you select a value for this parameter, see "Determining the Size of the JES3 Buffer Pool" in Chapter 7, "Defining and Managing JES3 Mains and Storage."

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: (64,16,32)

Performance Considerations: In selecting the size of the buffer pool, make sure the size is large enough for all functions to be active at the same time; the buffer pool is not expandable after initialization. Also, it is recommended that you make the buffer pool larger than the value you calculate. The larger buffer pool will better accommodate bursts of I/O activity.

FD = { **nnn** / **128** }

Indicates the number of multi-record file (MRF), file directory block (FDB) data sets and single record files (SRFs) that may be open at one time. The FD parameter defines the number of entries in the JES3 file directory; one 40-byte entry exists for each open data set. For single record files (SRF), file directory entries are added when certain conditions arise (such as no JSAM buffers) and for all WTRCHAIN requests. The maximum value that you can specify is 999.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 128

GRPSZ = { **nnn** / **135** }

Specifies the number of spool records in each track group. (The BUFSIZE parameter determines the size of each spool record.) The number must not be greater than 999. JES3 rounds the specified value up to the nearest number of records in a whole physical track for the selected spool device type. For guidelines to help you select a value for this parameter, see "Determining the Size of a Track Group" in Chapter 4, "Defining and Managing Spool Data Sets."

Parameter Default: 135

MINBUF = { **nn** / **16** }

Specifies the value that JES3 is to use to calculate the acceptable minimum number of free JSAM buffers. nn may be any decimal integer between 8 and 64, inclusive.

To calculate the acceptable minimum number of free JSAM buffers, JES3 divides nn into the total number of JSAM buffers. The quotient defines the acceptable minimum number. JES3 ignores the remainder.

When the number of free JSAM buffers is equal to or less than the acceptable minimum number, JES3 turns on the minbuf flag (specified by the FLAG parameter of the CONSTD statement) in all console messages. This warns the operator of the minimum buffer condition. If the operator does not take corrective action, stop some converter interpreters, for example, JES3 may go into the wait state.

If the minbuf flag appears on console messages frequently, increase the number of JSAM buffers during the next warm start or cold start. To increase the number of buffers, increase the value of the PAGES parameter on the BUFFER statement.

If the minimum buffer condition occurs for a C/I FSS address space, JES3 issues message IAT1101.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 16

SPLIM =({ **min** / **10** } , { **marg** / **25** })

Specifies the minimum and marginal percentages of spool space still available in active spool partitions. An active spool partition is one containing at least one spool data set. If the minimal or marginal percentages of spool space are reached, indicating that a spool partition is nearly full, JES3 issues action messages to the operator.

Define the character to be used by console service to warn the operator of minimum and marginal spool space conditions by using the FLAG parameter of the CONSTD statement.

min

Specifies the percentage of total spool space in an active spool partition which, when that percentage is all that is still available, defines a minimum spool space condition. For example, if *min* specifies 10, a minimum spool space condition exists when 10% or less of the spool space in an active spool partition is still available.

The percentage of spool space defining this condition may be between 0 and 99. It must, however, be smaller than or equal to the percentage defining a marginal spool space condition (see the next subparameter).

When a spool partition reaches a minimum spool space condition, JES3 issues a message stating that this condition has occurred. The message alerts the operator to inquire whether the spool partition automatically overflows into another partition. If the spool partition does overflow, no operator action is required. Otherwise, the operator can use operator commands to take appropriate actions. For information on actions to take, see "Balancing the Workload Across Partitions" in Chapter 4, "Defining and Managing Spool Data Sets."

If a minimum spool space condition arises on the default spool partition, JES3 suspends all SYSOUT buffer processing. JES3 does

not resume SYSOUT buffer processing until enough spool space is freed to reach a marginal spool space condition.

Always specify a minimum spool space percentage for the default spool partition so that enough spool space remains to perform a warm start. Otherwise, if JES3 requires a warm start and not enough spool space is available, you must perform a cold start.

marg
Specifies the percentage of total spool space in an active spool partition which, when that percentage is all that is still available, defines a marginal spool space condition. For example, if *marg* specifies 20, a marginal spool space condition exists when 20% or less of the spool space in an active spool partition is still available.

The percentage of spool space defining this condition may be between 0 and 99.

When a spool partition reaches a marginal spool space condition, JES3 issues a message stating that this condition has occurred. The message alerts the operator to inquire whether the spool partition automatically overflows into another partition. If the spool partition does overflow, the operator need not take any action. Otherwise, the operator can use operator commands to take appropriate actions. For information on actions to take, see "Balancing the Workload Across Partitions" in Chapter 4, "Defining and Managing Spool Data Sets."

When a marginal spool space condition arises, job selection is suspended.

If you specify an integer greater than the maximum allowable, JES3 uses the parameter default. If you specify a negative integer or a non-numeric character, JES3 issues message IAT3245 and initialization terminates.

Parameter Default: (10,25)

TRUNC = { **YES** / **NO** }

Specifies whether or not trailing blanks are to be truncated from all SYSOUT data that is produced in the complex.

YES
Indicates that trailing blanks are to be truncated from all SYSOUT data.

NO
Indicates that trailing blanks are not to be truncated from all SYSOUT data.

Parameter Default: YES

Statement Default: The statement default is as follows:

```
BUFFER,BUFSIZE=2036,PAGES=(64,16,32),FD=128,GRPSZ=135,     X
MINBUF=16,SPLIM=(10,25),TRUNC=YES
```

Associated Statement/Parameter: If you change the BUFSIZE parameter and do not reformat the spool data sets before the next cold start, include FORMAT statements in the initialization stream during the next cold start.

The FLAG parameter on the CONSTD statement sets the character used by console service to warn the operator when a spool partition reaches a minimum or marginal spool space condition. The SPLIM parameter of this statement defines the minimum and marginal spool space conditions.

The SDEPTH parameter on the CLASS statement specifies the number of jobs that can be set up simultaneously. Since the JSTs for every job set up are read into JSAM buffers, the SDEPTH value may impact the number of JSAM buffers that you specify.

Overrides: The GRPSZ parameter specification on the SPART initialization statement overrides the GRPSZ parameter specification on the BUFFER initialization statement.

The SPLIM parameter specifications on the SPART initialization statement override the SPLIM parameter specifications on the BUFFER statement.

If specified, the TRUNC parameter on the SYSOUT statement overrides the TRUNC parameter on the BUFFER statement for that SYSOUT class.

Example: In the following example, the BUFFER statement specifies that:

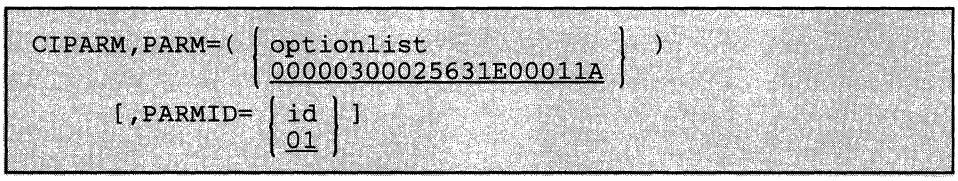
- a minimal spool space condition is reached when only 3% of the total spool space in an active spool partition is still available
- a marginal spool space condition is reached when only 5% of the total spool space in an active spool partition is still available
- the JES3 buffer pool size for the JES3 global processor is 52 pages
- the size of a track group is 12 spool records
- the size of a JES3 buffer and of a spool record is 1952 bytes

```
BUFFER,SPLIM=(3,5),PAGES=52,GRPSZ=12,BUFSIZE=1952
```

All other parameters and subparameters use default values.

CIPARM (Converter/Interpreter Parameters)

The CIPARM statement specifies the options to be used by the MVS converter/interpreter (C/I). These options are used as system defaults applied to certain JCL statement parameters and other options for jobs scheduled on any processor. The PARMID parameter assigns an identifier to the option list. This parameter is used to distinguish option lists if you include more than one CIPARM statement in the initialization stream.



PARM =(optionlist
00000300025631E00011A)

Specifies an option list of 21 EBCDIC characters. The option list has 10 fields that you must code in the format shown. Two fields have a fixed value always set by JES3: positions 13 and 15-18. Six fields are set at initialization: positions 2-3, 4-9, 10-12, 19, 20, and 21. JES3 restricts one field (position 1) to certain values. If you specify any value incorrectly, the default value *for that field* is used. You may substitute values only for the underscored items in the following format:

bppttttttcccrlaaaaefh

b

Specifies whether an account number and/or programmer name is required for this job, and whether the scheduler work area is to be located above 16-megabytes when the job executes.

The combination of options is as follows:

b	SWA Above 16 Mg	Acct. Num. Required	Prog. Name. Required
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
7	Yes	Yes	Yes

No values other than 0 to 7 are valid. Default value is 0.

pp

Specifies the default job priority. This field is ignored by JES3 because the default priority comes from the job specified by the PRTY parameter on the STANDARDS statement. The range of values is from 00 to 14; the default value is 00.

tttttt

Specifies the maximum time that each step may execute. When a step exceeds this limit, JES3 cancels the step. The first 4 characters indicate minutes; the last 2 characters indicate seconds. JES3 assigns this time limit to the step when the EXEC statement does not specify a time limit. The value specified must be less than 144000; the default value is 30 minutes (003000).

ccc

Specifies the job-step region default. Specify 3 numeric characters as the number of 1024-byte blocks assigned to each job step. (Do not specify 000.) JES3 assigns this region size to a step when the JOB or EXEC statement does not specify the region size and does specify ADDRSPC=VIRT for the step. The default value is 256.

Note: You must specify this value if you are defining a C/I FSS. The default does not provide adequate region size.

r

Specifies the command disposition. This field is always set to 3 by JES3, indicating that the MVS converter is to ignore commands read from the input stream.

l

Specifies how the BLP parameter of the DD statements is to be processed:

0 Specifies that the BLP parameter is to be ignored; the LABEL parameter is processed as if NL was specified.

1 Specifies that the BLP parameter is to be processed as intended. The default is 1.

aaaa

Specifies the MCS command authority. JES3 always sets this field to E000. This indicates that operator commands in Groups 1, 2, and 3 are to be executed.

e

Specifies the job MSGLEVEL default.

0 indicates that only the JOB statement is to be written as output.

1 indicates that all input JCL statements (including in-stream procedures) are to be recorded in the system message data set. The default is 1.

2 indicates that only input JCL statements are to be written.

f

Specifies the allocation MSGLEVEL default.

0 indicates that no allocation/termination messages are recorded in the system message data set unless the job terminates abnormally.

1 indicates that all allocation/termination messages are recorded in the system message data set. The default is 1.

h

Specifies the MSGCLASS default. The default class is A.

The option list must be 21 characters long in the fixed format described above. The installation may specify only those values defined by b, ttttt, ccc, l, e, f, and h. JES3 values are assumed for the other fields.

Parameter Default: (00000300025631E00011A)

PARMID = { **id** / **01** }

Specifies a 2-byte identifier associated with this option list. This parameter provides the facility to have a variety of C/I option lists. The operator may select the option list to be used by specifying the identifier on the *CALL, CR, DR, or TR command.

Parameter Default: 01

Statement Default: The statement default is as follows:

```
CIPARM,PARM=(00000300025631E00011A),PARMID=01
```

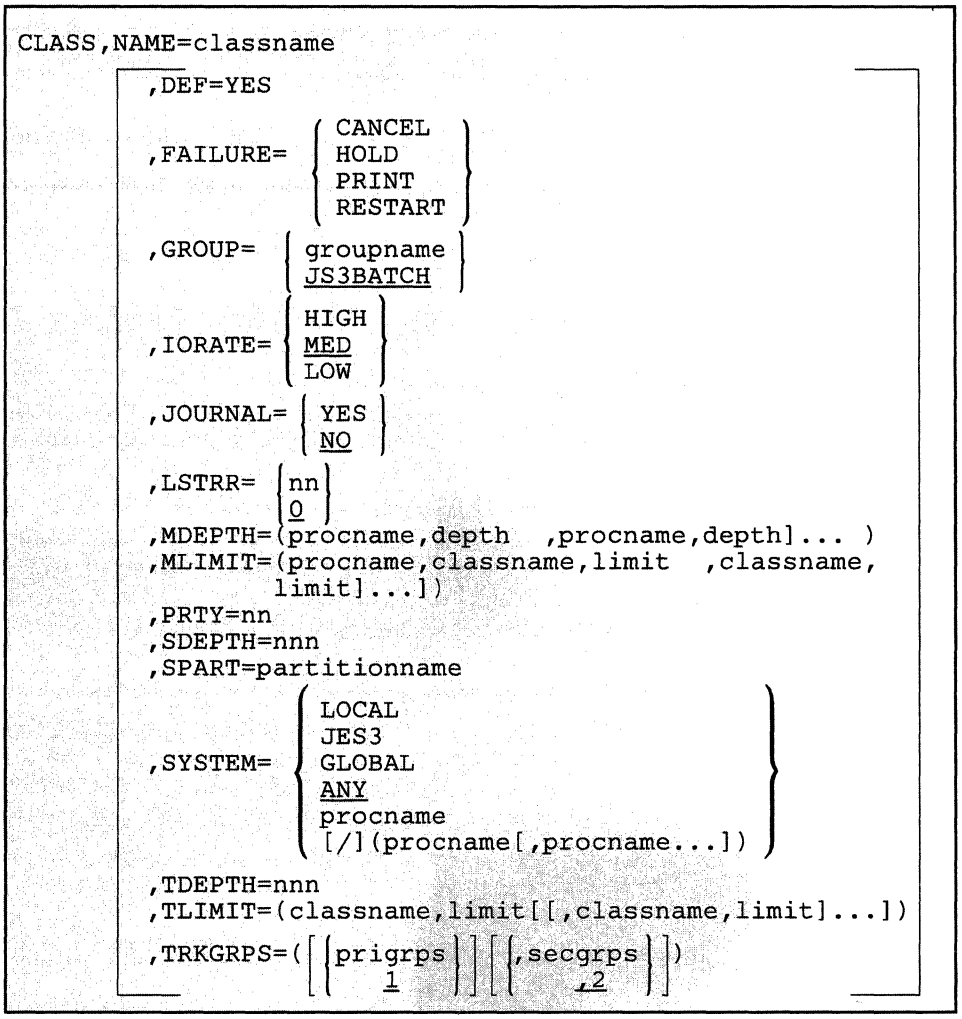
Associated Statement/Parameter: The PRTY parameter on the STANDARDS initialization statement overrides any value specified in columns 2 and 3 of the option list.

Example: In the following example, a C/I parameter list identified as 10, specifies a default region size as 60K, a default label processing option which ignores the bypass label processing parameter, and a default message class of B.

```
CIPARM,PARM=(00000003006030E00011B),PARMID=10
```

CLASS (JES3 Job Class Definition)

The CLASS initialization statement defines the characteristics of the JES3 job classes. A CLASS statement must define each class that can appear on a //*MAIN control statement. You can define up to 255 classes.



NAME =classname

Specifies the 1- to 8-character name of the job class. This name corresponds to the CLASS parameter on the //*MAIN control statement.

DEF =YES

Identifies this class as the default job class. Specify DEF=YES on one CLASS statement only. If you specify DEF=YES on multiple statements, the last such statement in the initialization stream defines the default class.

When you omit the CLASS parameter from a job's //*MAIN statement or the CLASS= keyword on the JOB statement, JES3 assigns the default job class to that job.

Parameter Default: If you omit DEF=YES from all CLASS statements in the initialization stream, JES3 defines JS3BATCH as the default class.

Note: A job class that is not defined to JES3 cannot be specified on the //*MAIN statement. If it is specified, JES3 cancels the job.

FAILURE = { **CANCEL** | **HOLD** | **PRINT** | **RESTART** }

Specifies the job recovery option that JES3 is to use when a system failure affects jobs in this class.

CANCEL
Print any job output that is in a SYSOUT class that is specified as TYPE=PRINT. After printing the output cancel the job.

HOLD
Place the job into the hold queue.

PRINT
Print any job output that is in a SYSOUT class that is specified as TYPE=PRINT. Then place the job into the hold queue.

RESTART
Restart the job from the first step. The job will be restarted on the processor on which it was active.

GROUP = { **groupname** | **JS3BATCH** }

Specifies the name of a job class group to which this job class is to be assigned. This parameter must match the NAME parameter on a GROUP statement.

Parameter Default: If the GROUP parameter is omitted, JES3 assigns this class to the default group (JS3BATCH).

IORATE = { **HIGH** | **MED** | **LOW** }

Specifies the default I/O rate for the jobs in this class. JES3 attempts to balance the mixture of jobs in execution based on the values that you have specified for the JOBMIX and CHOICE parameters on the appropriate SELECT statements. This parameter can be overridden for individual jobs by the IORATE parameter on the //*MAIN statement. For more information about how the IORATE parameter relates to job selection, see the description of the SELECT statement in this chapter and in "Job Selection and Scheduling" in Chapter 2, "JES3 Job Management."

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: MED

JOURNAL = { **YES** | **NO** }

Specifies whether jobs in this class are to have a journal data set.

YES

Indicates that jobs in this class are to have a journal data set. A job that does not have a journal data set is ineligible for automatic restart after a system failure.

NO

Indicates that there is to be no journal data set. If a system failure occurs affecting a job in this class, JES3 takes the action specified in the FAILURE option on this statement.

This parameter may be overridden by individual jobs with the JOURNAL parameter on the //*MAIN JES3 control statement.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: NO

LSTRR =nn

Specifies the logical storage reduction rate (0 to 99). If a job's logical region size is not specified in the LREGION parameter on the //*MAIN control statement, JES3 uses this parameter as a percentage of the region size to determine the job's logical region. If the LSTRR parameter is specified as 0, logical storage processing is effectively disabled for jobs not specifying LREGION.

If specified, the LREGION parameter overrides logical storage processing.

For more information about defining logical storage, see "Defining Logical Storage for MVS Processors" in Chapter 2, "JES3 Job Management."

Coding Consideration: See the LSTOR parameter on the SELECT initialization statement. If LSTOR=0 is specified on the SELECT statement, logical storage *scheduling* is disabled.

Parameter Default: 0

MDEPTH =(procname,depth[[,procname,depth]...])

Determines the maximum number of jobs in this class that can be run on the indicated processor at any one time. You may repeat the procname and depth subparameters in pairs as is necessary.

procname

Specifies the name of a processor. The processor name must match the NAME parameter on a MAINPROC statement.

depth

Specifies a decimal number from 0 to 255. This parameter indicates the maximum number of jobs of this class that JES3 can schedule to run concurrently on the named processor.

MLIMIT =(procname,classname,limit[[,classname,limit]...])
Determines the maximum number of jobs of other job classes that can run on the indicated processor and still let JES3 schedule jobs in this class. When any of the limits are exceeded, JES3 does not schedule any more jobs in the class defined by this CLASS statement on that processor. JES3 only schedules jobs of this class when the number of jobs running from the other named classes is equal to or less than the assigned limit.

This parameter may be repeated to describe job class limitations on each processor. The classname and limit parameters should be repeated, in pairs, to indicate the job limits of each class by processor.

procname
Specifies the name of a processor as specified in the NAME parameter of a MAINPROC statement.

classname
Specifies the name of another job class. This must match the NAME parameter on another CLASS statement.

limit
Specifies a number from 0 to 255. This value indicates the number of jobs of the class.

PRTY =nn
Specifies the JES3 job priority (0 to 14) to be assigned to each job in this class. The PRTY parameter may be overridden on a job basis by the PRTY parameter on the JOB statement.

Parameter Default: The PRTY parameter on the STANDARDS statement.

SDEPTH =nnn
Specifies the maximum number of jobs in this class requiring MDS operator mounts that can be set up at one time. The value of nnn is a number from 0 to 255.

When JES3 counts the number of jobs that are set up, JES3 considers a job to be set up from the time the job enters allocation until the time the job's devices are deallocated.

Note: When deferred mounting is specified in the JCL (UNIT=(TAPE,,DEFER)), JES3 bypasses normal setup processing. Thus, JES3 does not include a job using only deferred volume mounting in setup depth counts.

SPART =partitionname
Specifies the spool partition that JES3 is to use for jobs in this job class. The partition name must match the partition name specified on one of the SPART statements. To accept the default spool partition, omit this parameter.

A partition name specified on a //*MAIN JES3 control statement or on a SYSOUT initialization statement can override this parameter. For a discussion of the order of overrides, see "Determining the Order of Spool

Partition Overrides" and "How the User Can Request a Spool Partition" in Chapter 4, "Defining and Managing Spool Data Sets."

If you specify a partition name that has not been defined on an SPART statement, JES3 uses the default partition.

SYSTEM = { **LOCAL** | **JES3** | **GLOBAL** | **ANY** | **procname** | **[/](procname[,procname...])** }

Defines the processor name(s) or type of system to be used for jobs in this class.

LOCAL
Indicates that jobs in this class are to run on a JES3 local processor only.

JES3
Indicates that jobs in this class are to be run on the global processor or any local processor that can satisfy the job requirement.

GLOBAL
Indicates that jobs in this class are to run on the JES3 global processor only.

ANY
Indicates that jobs in this class can run on any processor (global or local) that will satisfy the job requirement.

procname
Indicates, when specified without a preceding slash (/), the only processor(s) to be eligible for running jobs in this class. If the "/" precedes the procname(s), this parameter indicates the processors to be excluded from consideration for jobs in this class. The procname must match the NAME parameter on a MAINPROC statement.

Parameter Default: ANY

TDEPTH =nnn
Specifies the maximum number of jobs of this class that can be scheduled into the total JES3 complex at one time. The value of nnn is a number from 0 to 255.

TLIMIT =(classname,limit[[,classname,limit ...)
Specifies the maximum number of jobs of other classes that can be scheduled into the total JES3 complex and still allow jobs in this job class to be scheduled. If any of the limits are exceeded, more jobs in this class will not be scheduled. Jobs of this class are scheduled only when the number of jobs running from other classes is equal to or less than the assigned limit.

classname
Specifies the name of another job class.

limit
Specifies the number of jobs of the other class. The maximum value is 255.

The classname and limit parameters may be repeated, in pairs, to indicate the job limit for each class.

TRKGRPS = ({ **prigrps** / **1** } { **,secgrps** / **,2** })

Specifies the number of track groups (as defined by the GRPSZ parameter on the BUFFER or SPART statement) JES3 is to allocate to jobs within this class. For guidelines on how to determine the appropriate value for the TRKGRPS parameter for your installation, see "Determining Track Group Allocation Sizes" in Chapter 4, "Defining and Managing Spool Data Sets."

prigrps
Specifies the number of track groups to be initially allocated to jobs in this class. The specified value may be 1 through 9.

secgrps
Specifies the number of track groups to be allocated to jobs in this class subsequent to their primary allocation. JES3 allocates the specified amount of spool space after the job uses up its initial allocation, and again (for an unlimited number of times) when the job uses up each secondary allocation and requests more spool space. The specified value may be 1 through 9.

Parameter Default: (1,2)

Associated Statement/Parameter: The FAILURE option on the STANDARDS statement is the default value when the CLASS statement does not include the FAILURE parameter. The NAME parameter on the MAINPROC statement is referred to in the MDEPTH, MLIMIT, and SYSTEM parameters on the CLASS statement. The NAME parameter on the GROUP statement is used on the GROUP parameter of the CLASS statement.

The IORATE parameter on the CLASS statement affects the JOBMIX and CHOICE parameters on the SELECT statement.

If you do not specify the PRTY parameter on the CLASS statement, JES3 uses the PRTY parameter on the STANDARDS statement as the default.

The TLIMIT and MLIMIT parameters refer to the NAME parameter on other CLASS statements.

The GRPSZ parameter of the BUFFER or SPART statement relates to the TRKGRPS parameter of the CLASS statement. (For information about the relationship, see "Determining Track Group Allocation Sizes" in Chapter 4, "Defining and Managing Spool Data Sets.")

Example: In the following example, the job class ABC is defined. This class is associated with job-class group TESTGP. Jobs in this class are to run on processors SYS1 and SYS2. Two other job classes, NOP and QRS, are to have job limits of 0 on processor SYS1 and 1 on processor SYS2.

```
CLASS,NAME=ABC,GROUP=TESTGP,SYSTEM=(SYS1,SYS2),
MLIMIT=(SYS1,NOP,0,QRS,0),
MLIMIT=(SYS2,NOP,1,QRS,1)
```

An ABC class job in this example would only be scheduled on SYS1 when no NOP or QRS jobs are running on SYS1. An ABC job would only be scheduled on SYS2 when one or less NOP and QRS jobs are running.

Override: For individual jobs, the //*MAIN JES3 control statement and the EXEC JCL statement can be used to override specific parameters on the CLASS statement. For further information, see specific parameter descriptions.

The specification of the TRKGRPS parameter on the SYSOUT initialization statement and //*MAIN JES3 control statement overrides the specification on the CLASS statement. If you do not specify the TRKGRPS parameter on the CLASS statement or any of the statements that override the CLASS statement, JES3 uses the specification given (or defaulted to) on the MAINPROC statement.

The operator can override parameters specified on the CLASS statement by using the command

```
*F,G,main, {CLASS|GROUP|SELECT}.
```

COMMDEFN (Communication Subsystem Interface Definition Records)

The COMMDEFN statement identifies the optional user communication subsystem interface (VTAM) parameters.

```
COMMDEFN [,APPLID= { applname }
                   { JES3     }
         ,P=password
         ,LU= { nnnn }
              { 255  } ]
```

APPLID = { **applname** / **JES3** }

Indicates the application name as specified to VTAM in a VTAM generation application definition statement (APPL). This parameter is used in establishing the application program interface (API) at OPEN ACB time.

Parameter Default: JES3

P = password

Specifies the 1- to 8-character password which is used by JES3 when issuing an OPEN ACB macro. This parameter must be the same as the password specified in the PRTCT parameter of the VTAM APPL application definition statement.

Parameter Default: Null character string which indicates no password.

LU = { **nnnn** / **255** }

Specifies the limit for the number of active sessions between SNA RJP and VTAM allowed at any one time. This number must not exceed 4095.

If you specify an integer greater than the maximum allowable, JES3 uses the parameter default. If you specify a negative integer or a non-numeric character, JES3 issues message IAT3245 and initialization terminates.

Parameter Default: 255

Statement Default: The statement default is as follows:

```
COMMDEFN,APPLID=JES3,LU=255
```

* (Comment Statement)

The comment statement allows you to include comments in a JES3 initialization stream. Comment statements begin in column one with an asterisk. They may be dispersed freely anywhere within the initialization stream. Comment statements cannot be continued; however, multiple comment statements can be included consecutively.

Comments can also be embedded within initialization statements in the initialization stream. If a comment is embedded within an initialization statement (whether the statement is complete on one line or continued), at least one blank must precede the beginning of the comment. No asterisk is required for embedded comments.

```
*comment
```

comment
Indicates, in columns 2 through 71, (after the asterisk) the comment to be included with the initialization statements. The text may be any length up to 70 characters; to continue comment text, a new comment statement must be specified.

Example 1: The following is an example of a comment statement.

```
*THIS IS A COMMENT STATEMENT AND REQUIRES AN ASTERISK.
```

Example 2: The following comment exceeds 70 characters and requires two comment statements.

```
*AS MANY COMMENT STATEMENTS AS ARE DESIRED MAY BE PLACED IN
*THE JES3 INITIALIZATION STREAM.
```

Example 3: The following is an example of a comment that is embedded within an initialization statement.

```
CLASS,NAME=ABC,DEF=YES       This is an embedded comment
```

Example 4: The following comments are included within a continued initialization statement.

```
CLASS,NAME=ABC,DEF=YES,      *Comment
     MDEPTH=(SY1,10)          *Comment
```

Example 5: The following example contains embedded comments, as well as comment statements.

```
CLASS,NAME=ABC,DEF=YES,    This is a comment
*THIS IS A COMMENT STATEMENT
      MDEPTH=(SY1,          This is a comment          X
*THIS IS A COMMENT STATEMENT
                  10)
```

COMPACT (Compaction Table Definition)

The COMPACT statement defines a compaction table to JES3. The compaction table is a set of characters which can be transmitted as a compacted character string.

```
COMPACT,NAME=comptab,MCCT=nn,CHAR=(a[a],...a[a])
        [,DEFAULT= {Y}]
                   {N}
```

NAME =comptab
Specifies the 1- to 8-character (alphanumeric) name of the compaction table. NAME=NO cannot be specified.

MCCT =nn
Specifies the number of master compaction characters in the CHAR parameter. This number must be an integer from 3 to 15.

CHAR =(a[a],...a[a])
Specifies the master and nonmaster compaction characters.

A single 'a' is any EBCDIC character except a comma or a blank. An 'aa' pair is the hexadecimal representation of any 8-bit character. The master compaction characters are the first nn characters in this field. The number of master compaction characters is specified in the MCCT parameter. The master compaction characters are transmitted in a compacted character string using a 4-bit representation. The remaining characters are nonmaster compaction characters which are transmitted in a compacted character string using a 8-bit representation.

The number of master characters specified determines the number of nonmaster characters. The following table lists the number of each allowed.

Master	Nonmaster
3	244
4	236
5	226
6	214
7	200
8	184
9	166
10	146
11	124
12	100
13	74
14	46
15	16

If the list is too long, or if any character is repeated, the COMPACT statement is ignored and an error message is issued.

Note: The compaction algorithm uses each byte of the encoded stream to represent either (1) a single character or (2) two compacted characters. The coded stream always consists of an exact number of bytes. Master

characters that stand alone are not compacted to 4 bits, but are coded as a full byte. SCS characters should be defined in this table also.

DEFAULT = { **Y** / **N** }

Specifies whether or not this is the default compaction table.

Y

Specifies that this is the default compaction table. This parameter is overridden by the COMPACT parameter on the RJPWS statement, the SYSOUT statement, or //*FORMAT control statement.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: N

Coding Considerations: The efficiency of the compaction algorithm is dependent upon correct specification and selection of a compaction table. For example, numeric data files should be transmitted using a table which specifies numeric master characters. Uppercase and lowercase text should be transmitted with a table specifying the most commonly used lowercase letters as master characters.

The following characters cannot be specified as a single character; their hexadecimal representation must be used:

character	**HEX**
/	61
=	7B
,	6B
blank	40
(	4D
)	5D

The SCS characters used as delimiters for data should be included in the compaction character set for better utilization of the compaction function. Including these characters avoids switching in and out of compaction mode on each line of output. The characters to be included are:

- NL (new line) - x'15'
- CR (carriage return) - x'0D'
- FF (forms feed) - x'0C'
- TRN (transparency) - x'35'
- IRS (Inner Record Separator) - x'1E'

Example: The following example defines a compaction table named MCCT3, having 3 master compaction characters.

```
COMPACT,NAME=MCCT3,CHAR=(A,B,C,                                    x
01,02,03,04,05,06,07,08,09,0A,0B,0C,0D,0E,0F)
```

CONSOLE (MVS/BDT)

CONSOLE for MVS/BDT Console Support

This CONSOLE statement defines a console for MVS/Bulk Data Transfer (5665-302) console support. If the JES3 complex includes one or more MVS/BDT facilities, you must include this statement in the JES3 initialization stream. Do not code this statement as the first CONSOLE statement in your JES3 initialization stream.

```
CONSOLE,JNAME=BDTDUMMY,TYPE=RJP
```

JNAME =BDTDUMMY
Specifies a dummy MVS/BDT console.

TYPE =RJP
Specifies that the console being defined is an RJP console.

CONSOLE for JES3 Operator Consoles (JES3-managed)

Use the CONSOLE initialization statement to define each JES3-managed or JES3-only console. If you are defining a remote job processing (RJP) console, see "CONSOLE for RJP Operator Consoles" on page 12-38. If you are defining an MCS-managed console with logical associations, see "CONSOLE for JES3 Operator Consoles (MCS-managed)" on page 12-36.

```
CONSOLE,JNAME= { name     }
               { NETWORK  }
      ,TYPE=   { NJE                  }
               { devicetype           }
               { (devicetype,model)   }
    ,DEST=     { destclass                          }
               { (destclass,destclass[,...])        }
    ,UNIT=(procname, { ddd    }   [,...])
                     { DUMMY  }
                     { NONE   }
    ,MAIN=     { procname }
               { *        }
    ,ALTCON=name
    ,DEPTH=    { 50 }
               { nn }
    ,LL=nnn
    ,TIME=     { 2  }
               { nn }
    ,LEVEL=    { 15 }
               { nn }
    ,PFK=tblname
    ,SP=tblname
    ,IOWAIT=   { seconds }
               { 30      }
    ,3290=     { YES }
               { NO  }
```

JNAME = { name / NETWORK }

Specifies the console name. For a JES3 networking logical console, specify NETWORK. For all other consoles, select a name for the ***name*** variable. The name you specify must match the name you specify on the JNAME= keyword of the DEVICE statement for this console.

TYPE = { NJE / devicetype / (devicetype,model) }

Specifies the type or the type and model number of a device that you are defining; or, in the case of the NJE value, specifies how you want to use this device.

NJE
Specifies a device as a logical console for the job entry network.

devicetype
(devicetype,model)
Specifies a device as a JES3 console. To define a device as a JES3 console, code the device type or the device type and model number as shown in figure 1. You must attach any device that you define as a JES3 console to the global. If you omit the model number for certain devices, JES3 uses a default model number. The column headed **Model Number Defaults** shows which model number JES3 uses as a default for these devices. You must code the parentheses () if you specify both a device type and model number.

Device	Model Number Defaults	Device	Model Number Defaults
1403		3279,3A	
2740		3279,3B	
3203		3284	JES3 assumes model 3
3211		3284,1	
3277		3284,2	
3278		3284,3	
3278,2		3286	JES3 assumes model 1
3278,2A		3286,1	
3278,3		3286,2	
3278,4		3288	JES3 assumes model 2
3278,5		3288,2	
3279	JES3 assumes model 2A	3289	JES3 assumes model 1
3279,2A		3289,1	
3279,2B		3289,2	
		3290	

Figure 12-2. I/O Devices Supported as JES3 Consoles

Coding Notes:

To define one of the following consoles, code the TYPE= keyword as shown:

- 3290 Information Panel: For each logical console being defined to JES3, code a separate CONSOLE statement. Specify the TYPE= keyword as 3278 or 3279 (depending on the model on whose screen size the 3290 was customized).
- 3287: Specify all models of the 3287 as TYPE=(3284,3).
- 308x service support console: Specify TYPE=(3278,2).

DEST = { destclass | (destclass,destclass[,...]) }
Specifies one or more classes of JES3 messages that you want sent to this JES3 console. You can specify the following values for the ***destclass*** parameter. They represent the 95 JES3 destination classes plus three general categories:

ALL - Console receives operator broadcast messages. Consoles that you define to receive this class display messages sent by the *MESSAGE command with the destination specified as 'ALL'.

ERR - Console receives messages concerning equipment failure as well as JES3 failsoft messages.

JES - Console receives general information messages pertaining to the operation of the global.

LOG - Console receives messages concerning active jobs, such as jobs started or jobs ended.

MLG - Console receives all input and output messages, except security messages (SEC message class), provided that you specify MLG on the HARDCOPY= keyword of the of the CONSTD statement. This is in addition to the original source consoles or destination consoles which also receive the messages. If you are logging route code 9 (security messages) and/or 11 (write-to-programmer messages), You should not not make the master log available to unauthorized persons

SEC - Console receives security messages. Only consoles that you define to receive SEC class messages receive user-determined security messages. The SEC destination class is for user applications only.

TAP - Console receives messages which apply to the JES3 background utilities with JES3-controlled tape requirements, such as tape-to-print or card-to-tape.

TP - Console receives messages which apply to RJP teleprocessing equipment.

UR - Console receives messages which apply to the operation of JES3-controlled unit record equipment.

{ **DALL** | **D1 to D22** }

Console receives messages which apply to a user-defined console configuration. If you specify DALL, the console receives all messages in this group, or you can specify specific classes in this group by specifying one or a subset from D1 to D22.

{ **MALL** | **M1 to M32** }

Console receives messages that are unique to a JES3 local or JES3 global main. The destination class is associated with a main defined by the MDEST= keyword of the MAINPROC statement. If you specify MALL, the console receives all messages in this group, or you can specify specific classes in this group by specifying one or a subset from M1 to M22.

{ **SALL** | **S1 to S32** }

Console receives messages which pertain to JES3 device setup. If you specify SALL, the console receives all messages in this group, or you can specify specific classes in this group by specifying one or a subset from S1 to S22. You can associate setup messages regarding a specific device by defining the destination class on the JUNIT= or XUNIT= keywords of the DEVICE statement.

NONE - Console receives no messages.

OUTPUT - Console receives messages in all message classes except MLG.

TOTAL- Console receives messages from all 95 message classes including MLG.

Note that a device is allocated as a console if you specify any valid destination class (those for 95 message classes plus OUTPUT and TOTAL). If you specify DEST=NONE, JES3 allocates a device as a console only if you specify DTYPE=CNSxxxx on the DEVICE statement for this device. Otherwise, JES3 does not allocate the device as a console, but makes it available as a JES3 device, such as a printer.

UNIT =(procname, { **ddd** / **DUMMY** / **NONE** } **[,...])**

The UNIT keyword identifies all mains with which this console can communicate. At MVS initialization and at JES3 initialization, you can define JES3 consoles so that communication with local mains is possible.

procname

Identifies the name of a main associated with this console. This name must match the NAME= keyword on a MAINPROC statement.

{ **ddd** / **DUMMY** / **NONE** }

Required for each procname to indicate how or if you specified the console in the CONSOLxx member of SYS1.PARMLIB of that system.

ddd

If the named main is the global, ***ddd*** is the actual device number of this console as specified in the CONSOLxx member of SYS1.PARMLIB. This console can communicate directly with MVS. If the named main is a local, ***ddd*** is the device number of a console to which a JES3-managed or MCS-managed console attached to the global is associated. If you have previously used a device number on another statement for a logical association, JES3 replaces that device number with the DUMMY parameter.

DUMMY

Specifies that the next available subsystem-allocatable console should be used instead of a real console device. Refer to Chapter 5, "Defining Consoles and Message Routing" on page 5-1 for information about subsystem-allocatable consoles.

NONE

The NONE parameter specifies that the console was not identified to the system in the CONSOLxx member of SYS1.PARMLIB during MVS initialization and, therefore, this console cannot communicate with MVS.

MAIN = { **procname** / * }

Specifies that you can send MVS command to a processor without using the *SEND command. The procname must match the NAME= keyword you specify on a MAINPROC statement. You can specify the MAIN= keyword for any JES3 console. You can also enter JES3 commands at the console; you must prefix JES3 commands with an asterisk (*) or one of the synonym characters that you define on SYN= keyword of the CONSTD initialization statement. MAIN=* specifies the global main. You must code the UNIT= keyword in conjunction with this keyword.

ALTCON =name

Specifies the alternate console to which JES3 switches if this JES3 console goes out of ready status and reaches the message limit specified on the DEPTH= keyword, or has an unrecoverable error. To prevent automatic switching to an alternate console (such as is needed if special paper is used on this console), specify the same name for this keyword as that for the JNAME= keyword.

Parameter Default: The alternate console is the first active, locally-attached JES3-managed console (other than itself) defined in the initialization stream. See "Defining Alternate Consoles" on page 5-18 for a description of alternate consoles.

DEPTH = { **nn** / **50** }

Specifies the maximum number of messages that buffers associated with a JES3 console can maintain. You can specify any value from 1 to 512. When a console reaches its maximum, subsequent messages of priority 8 or less issued to that console wait. Use the *INQUIRY,O,D command to monitor consoles that are frequently queued to depth. Use the *INQUIRY,C,C command display the status of console buffers. Setting a low value can cause messages to wait. Setting a high value can flood the console with messages, thus causing performance problems and a storage shortage. If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 50

LL =nnn

Specifies the longest line length you want displayed on this JES3 console. If the actual line exceeds this length, the console breaks the line at a comma or a blank and continues on the next line. JES3 ignores this parameter for 3284, 3286, 3288, and 3289 consoles.

Parameter Default:

- 79 for all 3277s, 3278s, and 3279s.
- 80 for any other graphic console except a 3270 type device. The actual number of characters that can be printed per line is the specified value minus the number of device control characters.
- 120 for all hard-copy consoles.

TIME = { **nnn** / **2** }

Specifies the delay, in seconds, between displays of output messages on a graphic device. Acceptable values are .25, .50, or any other one-digit or two-digit integer from 0 to 20 inclusive. If you set the delay too high, a large backlog of messages occurs which can exceed the limit specified on the DEPTH keyword, causing the system to wait (See also the description of the DEPTH= keyword). This keyword is valid only for IBM 3277, IBM 3278, and IBM 3279 consoles. If you specify an invalid value, JES3 uses the parameter default.

If you specify TYPE=3277, 3278, or 3279, you can also activate a page mode facility by specifying TIME=0. When a 3277, 3278, or 3279 is in page mode, messages are simultaneously displayed until they fill the screen. This page of messages is displayed for 15 seconds, then automatically changed to display another page of messages.

Parameter Default: 2

LEVEL = { **nn** / **15** }

Specifies the JES3 authority level for JES3 or MCS-managed consoles. The authority level determines the commands that you can enter at this console. For an explanation of authority levels, see "Authorizing JES3 Commands" on page 5-14. To change authority levels, replace the JES3-supplied exit routine, IATUX18, with your own exit routine. If you specify an invalid value, JES3 uses the parameter default.

Parameter Default: 15

PFK =tblname

Valid for 3277, 3278, and 3279 JES3 consoles only. The PFK keyword indicates that this console has program function key support. The tblname must have been specified on the N= keyword of a PFK statement.

Parameter Default: The program function key is treated as a CANCEL key.

SP =tblname

Valid for 3277, 3278, and 3279 JES3 consoles only. The SP= keyword specifies that this console has selector pen (light pen) support. You must also specify the ***tblname*** parameter on the N= keyword of a PFK statement.

Parameter Default: The light pen is treated as a CANCEL key.

IOWAIT = { **seconds** / **30** }

Specifies, in seconds, the amount of time you want to allow to send a message to the console. If the console has a buffer, this value specifies the amount of time you want to allow to write the buffer to the console. (JES3 measures the time from the EXCP instruction that sends the message or writes the buffer until the console issues a device-end interrupt.)

If the I/O operation takes more time than you allow, JES3 issues message IAT7110 (Lost I/O) and tries to switch to the alternate console.

Specifying a value of 0 results in no time limit. If you do not want to place a time limit on the I/O operation, specify IOWAIT=0. Maximum value allowed is 30 seconds.

Parameter Default: 30.

3290 = { **YES** / NO }

Specifies that a 3277, 3278, or 3279 value on the TYPE= keyword represents one logical session of a 3290 console (a 3290 console can have from 1 to 4 logical sessions concurrently.) This keyword is valid only on console statements for these consoles.

Parameter Default: NO.

Associated Statement/Parameter: The JNAME= keyword keyword must match the JNAME= keyword on this console's DEVICE statement. The PFK= and SP= keywords must be the same as the N= keyword on a PFK statement.

Coding Considerations: When defining a logical network console, code only the JNAME= and TYPE= keywords.

Example 1: This example describes an:

- IBM 3277 display station that displays all output messages
- IBM 2740 communications terminal that is for device mounting, tape, unit record, and teleprocessing
- IBM 2740 communications terminal that is for monitoring all user-defined setup messages

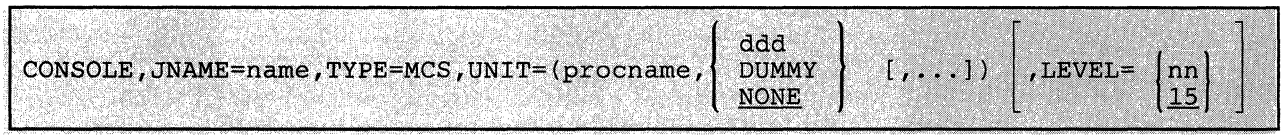
```
CONSOLE,JNAME=CN1,UNIT=(SY1,041,SY2,041),TYPE=3277,     X
DEST=OUTPUT
CONSOLE,JNAME=CN4,UNIT=(SY1,0BA,SY2,NONE),TYPE=2740,    X
DEST=(TP,TAP,UR)
CONSOLE,JNAME=CN5,UNIT=(SY1,0BB,SY2,DUMMY),TYPE=2740,   X
DEST=SALL
```

Override: You can use the JES3 *VARY command to change a device defined as console to a printer. You can use the JES3 *MODIFY command to override the DEST, LEVEL, PFK, or SP values. You can use the JES3 *SWITCH command by itself or in conjunction with the JES3 *DISABLE command to reroute output from one console to another.

CONSOLE (MCS-managed)

CONSOLE for JES3 Operator Consoles (MCS-managed)

Use this form of the CONSOLE statement to define an MCS-managed console with logical associations. If you are defining a remote job processing (RJP) console, see "CONSOLE for RJP Operator Consoles" on page 12-38. If you are defining a JES3-managed or JES3-only console, see "CONSOLE for JES3 Operator Consoles (JES3-managed)" on page 12-29.



JNAME = name
Specifies the console name.

TYPE = MCS
Specifies that you want MCS to manage this console. This definition allows you to associate this console with MCS consoles on local processors. A console association permits you to send commands to MVS on another processor from an MCS console attached to the global and receive responses to those commands. See "Establishing Logical Associations" on page 5-10 for information about logical associations.

UNIT =(procname, { ddd / DUMMY / NONE } [,...])
The UNIT= keyword identifies all mains with which this console can communicate. At MVS initialization and at JES3 initialization, you can define JES3 and MCS consoles so that communication with local mains is possible.

procname
Identifies the name of a main associated with this console. This name must match the NAME= keyword on a MAINPROC statement.

{ ddd / DUMMY / NONE }
Required for each procname to indicate how or if you specified the console in the CONSOLxx member of SYS1.PARMLIB of that system.

ddd
If the named main is the global, ***ddd*** is the actual device number of this console as specified in the CONSOLxx member of SYS1.PARMLIB. This console can communicate directly with MVS. If the named main is a local, ***ddd*** is the device number of a console to which this console is associated. If you have previously used a device number on another statement for a logical association, JES3 replaces that device number with the DUMMY parameter.

DUMMY
Specifies that the next available subsystem-allocatable console should be used instead of a real console device. Refer to Chapter 5, "Defining Consoles and Message Routing" on page 5-1 for information about subsystem-allocatable consoles.

NONE
The NONE parameter specifies that the console was not identified to the system in the CONSOLxx member of SYS1.PARMLIB during MVS initialization and, therefore, this console cannot communicate with MVS.

LEVEL = { **nn** / **15** }

Specifies the JES3 authority level for this console which determines the JES3 commands that you can enter from this console. For an explanation of authority levels, see "Authorizing JES3 Commands" on page 5-14. To change authority levels, replace the JES3-supplied exit routine, IATUX18, with your own exit routine. If you omit this keyword or specify an invalid value, JES3 uses the parameter default.

Parameter Default: 15

Examples:

Example 1: The following example shows the use of the CONSOLE statement is a three main complex (SY1 is the global). The UNIT= keyword allows you to send MVS commands to processors SY2 and SY3. Refer to "Establishing Logical Associations" on page 5-10 for information about defining logical associations:

```
CONSOLE,JNAME=MCS1,TYPE=MCS,UNIT=(SY1,3E1,SY2,3E1,SY3,3E1)
```

Example 2: The following example shows the use of the CONSOLE statement in a three main complex to allow an automation product, such as the IBM NetView Program Product, to send commands each local main. You must always specify DUMMY on the UNIT= keyword for the global main when defining a console for automation. See "Automating Message Processing" on page 5-37 for information about using automation packages:

```
CONSOLE,JNAME=MCSAUTO1,TYPE=MCS,UNIT=(SY1,DUMMY,SY2,DUMMY,SY3,DUMMY)
```

CONSOLE (RJP)

CONSOLE for RJP Operator Consoles

This CONSOLE statement is required to define each SNA RJP console and is optional for BSC RJP consoles. This statement is used to assign JES3 destination classes to that console. Only one console can be specified on each CONSOLE statement.

```
                                                          ┌ ,DEPTH=┌ nn ┐ ┐
                                                          │        └ 50 ┘ │
CONSOLE,JNAME=name,TYPE=RJP,DEST= ┌ destclass           │ ,LL=   ┌ nnn ┐ │
                                  └ (destclass,destclass[,...]) ┘        └ 120 ┘ │
                                                          │ ,LEVEL=┌ 0  ┐ │
                                                          └        └ nn ┘ ┘
```

JNAME =name

Specifies the name of a BSC RJP terminal or SNA RJP work station. If the name specified is the name of a BSC RJP terminal, it must match the name specified in the N parameter on a RJPTERM statement. If the name specified is a SNA RJP work station, it must match the name specified in the N parameter on a RJPWS statement.

TYPE =RJP

Specifies that the console being defined is an RJP console.

DEST = { **destclass** | **(destclass,destclass[,...])** }

Specifies one or a series of destination classes which represent message classes you want sent to the specified RJP console. JES3 assigns to the primary console those destination classes which you do not assign to any JES3 console. any of these is not assigned to a console, the associated messages are directed to the console defined by the first CONSOLE statement in the initialization stream.

See the DEST= keyword of the CONSOLE (Non-RJP) initialization statement for a complete list and description of JES3 destination classes.

DEPTH = { **nn** | **50** }

Specifies the maximum number of messages that buffers associated with a JES3 console can maintain. The DEPTH may be from 1 to 512. When a console reaches its maximum, subsequent messages of priority 8 or less issued to that console wait. Use the *INQUIRY,O,D command to monitor consoles that are frequently queued to depth. Use the *INQUIRY,C,C command display the status of console buffers.

Setting the DEPTH value too low can cause messages to wait. Setting DEPTH too high can allow too many messages to reach the console, thus causing performance problems and a storage shortage.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 50

LL =nnn

Specifies the longest line length you want displayed on this JES3 console. If the actual line exceeds this length, the console breaks the line at a comma or a blank and continues on the next line.

Parameter Default: The default depends on the device type:

- 79 for all 3277s, 3278s, and 3279s.
- 80 for any other graphic console except a 3270 type device. The actual number of characters that can be printed per line is the specified value minus the number of device control characters.
- 120 for all hard-copy consoles.

JES3 ignores this parameter for 3284, 3286, 3288, and 3289 consoles.

LEVEL = { **nn** / **15** }

Specifies the JES3 authority level for JES3 consoles. The authority level determines the commands that you can enter at this console. For an explanation of authority levels, see "Authorizing Console Commands" in Chapter 6, "Defining and Managing JES3 Resources."

To change authority levels, replace the JES3-supplied exit routine, IATUX18, with your own exit routine.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 15

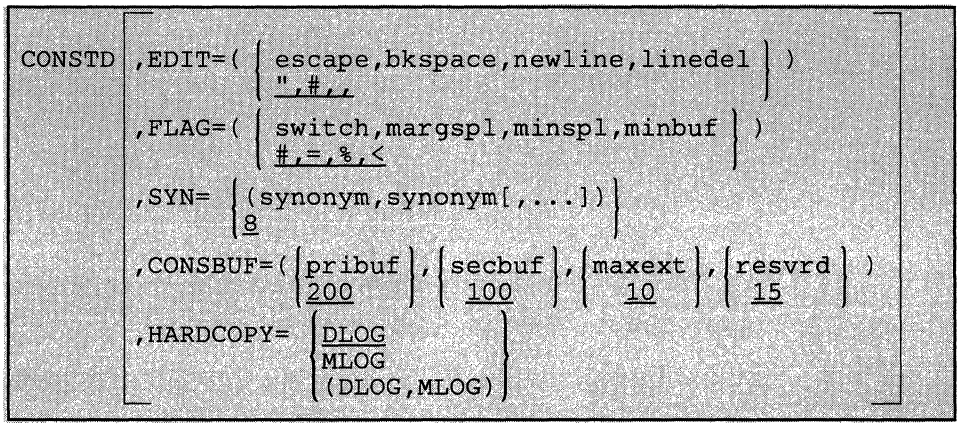
Associated Statement/Parameter: The JNAME parameter must match the N parameter on a RJPTERM statement or the N parameter on a RJPWS statement.

Examples:

```
CONSOLE,TYPE=RJP,JNAME=WS000,DEST=NONE
CONSOLE,JNAME=S360A,TYPE=RJP,DEST=NONE,DEPTH=100,   X
LL=80,LEVEL=15
```

CONSTD (Console Service Standards)

Use the CONSTD statement to define standards for your console configuration. These standards include special characters to be used in editing and flagging commands and messages processed by JES3 console service, console buffer pool allocation, action message retention, and system log facility routine. Certain special characters are used by the syntax scanner as control characters. These characters are ( ) , = / ″ and the blank. If any of these special characters are used, each must be immediately preceded by the escape character which is the quote (″).



EDIT =({ escape,bkspace,newline,linedel })
{ ",#,, }

Specifies the special characters to be used in the logical editing of console input. (Logical line editing is described in *MVS/Extended Architecture Operations: JES3 Commands.*) Any character specified or assumed for this parameter must not be specified for the SYN parameter.

escape
Specifies the logical escape character.

bkspace
Specifies the character to be used as the backspace key.

newline
Specifies the character you want to use as a command delimiter (also called new-line or end-of-line key). Use the delimiter to separate commands when stacking multiple commands on the command line of an operator console. Ensure that the delimiter you specify matches the MVS command delimiter. Using different characters may prevent MVS or JES3 from properly separating stacked commands. IBM does not provide a default character.

linedel
Specifies the logical line delete character. There is no default.

Parameter Default: (″,#,,)

Note: If you use any of the special characters (( ) , = / ″ and the blank) as subparameters, you must immediately precede each with the escape character which is the quote (″). Do not code a comma between the ″ and the special character. For example, if you specify the default in the EDIT parameter, you must code

```
EDIT=("",#,,)
```

FLAG =({ **switch,margspl,minspl,minbuf** / **#,=,%,<** } **)**

Specifies the time-stamp suffixes to be used to alert the operator to unusual JES3 conditions.

switch
Specifies the character to be used to flag messages that were initially routed to one console but were switched to another.

margspl
Specifies the character to be used to warn the operator that a marginal spool space condition exists. (See the SPLIM parameter of the BUFFER statement.)

minspl
Specifies the character to be used to warn the operator that a minimal spool space condition exists. (See the SPLIM parameter of the BUFFER statement.)

minbuf
Specifies the character to be used to warn the operator that a minimal buffer condition exists. (See the MINBUF parameter of the BUFFER statement.)

Parameter Default: (#,=,%,<)

Note: If you use any of the special characters (( ) , = / ″ and the blank) as subparameters, you must immediately precede each with the escape character which is the quote (″). Do not code a comma between the ″ and the special character. For example, if you specify the default in the FLAG parameter, you must code

```
FLAG=(#,"=,%,<)
```

SYN = { **(synonym,synonym[,...])** / **8** }

Specifies a set of synonyms for the standard asterisk (*) used to prefix all JES3 commands. Up to six synonyms may be defined. If you define a synonym using this keyword, you can continue to use 8 as a command prefix. Any character specified or assumed for this parameter must not be specified for the EDIT parameter.

Parameter Default: 8

CONSBUF =([pribuf / 200] , [secbuf / 100] , [maxext / 10] , [resvrd / 15])

Specifies the number of preallocated console message buffers that are formatted for the JES3 complex.

pribuf

The pribuf subparameter specifies the number of buffers in the primary extent of the console text buffer pool for the JES3 complex. Values that are not multiples of 20 will be rounded to the next highest multiple of 20. A percentage of the primary buffers will be allocated as reserved.

secbuf

The secbuf subparameter specifies the number of buffers to be dynamically obtained when the primary extent is exhausted. Values should be specified as multiples of 20; values that are not multiples of 20 are rounded to the next highest multiple of 20.

maxext

The maxext subparameter specifies the maximum number of times the console text buffer pool can be extended. Specifying zero indicates that no limit is placed on the number of extensions; the buffer pool can be expanded until storage is exhausted.

resvrd

The resvrd subparameter specifies a percentage of console text buffers in the buffer pool's primary extent that are to be reserved. The maximum value is 30 percent. The number of non-reserved buffers left in the primary extent is rounded down to the nearest multiple of 8.

Coding Notes:

To help cope with heavy console message loads, CONSBUF parameter values may be adjusted. When selecting values for the pribuf, secbuf, maxext, and resvrd subparameters, you should understand how these subparameters affect each other, and in turn how they impact JES3 performance.

Remember that a percentage of the number of buffers specified as the pribuf value is allocated as reserved. You should specify a number large enough to accommodate your primary extent needs, plus the reserve number. For example, if you code CONSBUF=(100,50,10,15), 15% of the 100 primary buffers are reserved. Since non-reserved buffers are rounded down to the nearest multiple of 8, the primary extent would have 80 non-reserved buffers and 20 reserved buffers.

When available storage is low or fragmented, tuning the secbuf and maxext subparameters can increase chances of getting storage. As an example, suppose that you have used the default, CONSBUF=(200,100,10,15). To create a secondary extent, JES3 must obtain five contiguous pages of storage. If console message traffic is very heavy, this may not be possible. By changing the CONSBUF parameter to (200,20,50,15), the same amount of space is obtained, but in smaller segments.

Initially, you may wish to use the CONSBUF parameter defaults, and then adjust these values according to your installation's needs.

Parameter Default: (200,100,10,15)

HARDCOPY = { **DLOG** | **MLOG** | **(DLOG,MLOG)** }

Specifies how you want the system log maintained. You must specify both DLOG and MLOG if you want to direct an *FREE command to an MLOG device that has failed or run out of paper. Refer to "Defining the System Log" on page 5-19 for information about defining and using the system log.

DLOG
Specifies that you want the system log written to the spool data set for subsequent printing by output service.

MLOG
Specifies that you want the system log written to consoles with the MLG destination. If there is no DLOG, at least one such console must be a hard-copy device.

Parameter Default: DLOG

Statement Default:

```
CONSTD,EDIT=(",#,,),FLAG=(#,=,%,<),SYN=8,                    X
CONSBUF=(200,100,10,15),HARDCOPY=DLOG
```

DEADLINE (Deadline Type Definition)

The DEADLINE statement defines a deadline type for job scheduling; each type determines how the priority of a job is increased so the job is scheduled within a specified time limit. A deadline type is associated with a specific job by indicating the type on the //*MAIN control statement. Up to 36 DEADLINE statements can be specified.

```
DEADLINE,type=(prtyset,timeset[,prtychng,timincr])
```

type

Specifies a single-character identifier from A to Z or 0 to 9. To apply this deadline definition to a job, the type must be indicated in the DEADLINE parameter on the job's //*MAIN control statement.

prtyset

Indicates the initial change in priority. This parameter is a priority level (1-14) to be set, or an incremental value to be added to a job's priority. An incremental value must be preceded by a plus sign (+). If the prtyset value is less than or equal to the job's priority, the job priority is not modified.

timeset

Specifies the time in minutes, hours, or TOD clock time, when a job priority is to be set to prtyset. This parameter represents the lead time for a job, that is, the amount of time required for a job to start and finish by the deadline associated in the DEADLINE parameter of the //*MAIN control statement.

To indicate time in minutes, specify the letter M immediately following the time; such as 1M for 1 minute. To indicate time in hours, specify the letter H immediately following the time; such as 1H for 1 hour. The range of acceptable values is from 0 to 1440 minutes *or* hours (depending on whether M or H is specified.)

To indicate TOD clock time, specify the time in hours *and* minutes (for example, 1045 indicates 10 hours and 45 minutes). The range of acceptable values is from 0 to 1440.

prtychng

Specifies either a subsequent priority level change overriding prtyset or subsequent increments to be added to prtyset. If you indicate a new priority level, it will be the priority of a job until its completion. If you indicate a priority increment, it will be added periodically until a job either completes or reaches the highest priority level (14). An incremental value must be preceded by a plus sign (+). When prtychng is specified, timincr must also be specified.

timincr

Specifies the amount of time in minutes, hours, or TOD clock time before the prtyset priority is to be updated by prtychng. To indicate time in minutes, specify the letter M following the time; such as 1M for 1 minute. To indicate the time in hours, specify the letter H following the time; such as 1H for 1 hour.

To indicate TOD clock time, specify the time in hours and minutes (for example, 1045 indicates 10 hours and 45 minutes).

Example: The following example shows how a deadline type is applied to a job. Also shown are the //*MAIN and DEADLINE statements which specify the job deadline and deadline type. The //*MAIN statement indicates that the deadline is 8 a.m. March 21, 1975 and the deadline type is A. The DEADLINE statement defines type A, which causes a job priority to be set to 10 one hour prior to the job's deadline and causes the priority to be incremented by 1 every 30 minutes until the job is complete.

```
DEADLINE,A=(10,1H,+1,30M)
//*MAIN DEADLINE=(800,A,032175)
```

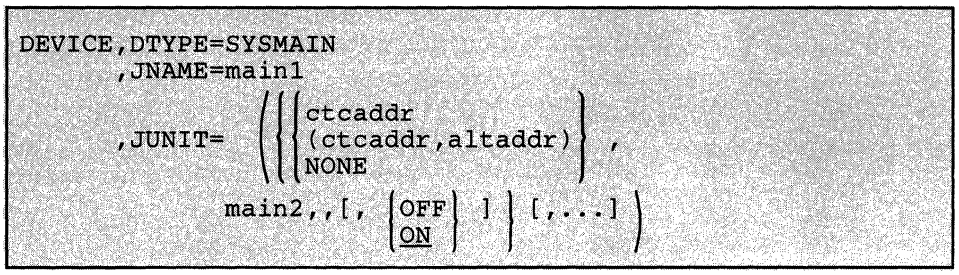
Event	Time	Priority
Job submitted	0400	PRTY = 5
	0700	Set PRTY = 10
	0730	Increment + 1 to 11
Deadline	0800	Increment + 1 to 12
	0830	Increment + 1 to 13
Job complete	0840	

Override: The *CANCEL,DEADLINE,PURGE operator command stops deadline scheduling being applied to jobs in the system.

DEVICE (Define Processor CTC Connections)

This form of the DEVICE statement (there are two other forms of the DEVICE statement) defines the CTC connections between mains in a JES3 complex. You must code one of these DEVICE statements for each main on every processor in the complex.

For more information on defining mains, see "Defining Mains" in Chapter 7, "Defining and Managing JES3 Mains and Storage."



DTYPE =SYSMAIN
Specifies that this statement defines the CTC connections for a main.

JNAME =main1
Specifies the name of a main whose CTC connections are being defined. main1 must match the main name as defined on a MAINPROC statement.

JUNIT =({ { ctcaddr / (ctcaddr,altaddr) / NONE } ,main2,,[, { OFF / ON }] } [,...]
Specifies the CTC addresses that a main (main2) uses to communicate with the main whose CTC connections are being defined (main1). Repeat this group of subparameters for each main in the JES3 complex.

Each main communicating with the global processor must use a unique CTC address on the global processor.

ctcaddr
Specifies the primary CTC address that main2 must use to communicate with main1.

(ctcaddr,altaddr)
Specifies the primary and the secondary (alternate) CTC addresses that main2 must use to communicate with the main1.

NONE
Specifies that the main2 does not communicate with main1 via a CTC connection.

main2

If you specified an address--either ctcaddr or (ctcaddr,altaddr)--in the previous subparameter, main2 must be the name of the main that will communicate with main1 through the specified CTC address.

If you specified NONE as the previous subparameter, main2 must be the name of a main that will not communicate with main1 through a CTC address.

,

Specifies the absence of the destclass subparameter. This subparameter, which may be coded on other forms of the DEVICE statement, may not be coded on this form of the statement.

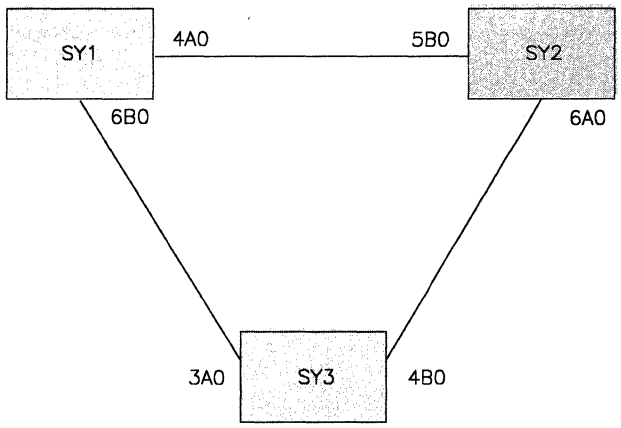
{ **OFF** }
{ **ON** }

Specifies the initial status of the CTC connection as viewed by main1. OFF specifies off-line; ON specifies on-line.

Coding Notes: For the main named in the JNAME= parameter, specify JUNIT=(NONE,...). This tells JES3 that the main named in the JNAME= parameter does not communicate with itself.

Example 1: This example shows three mains, SY1, SY2, and SY3 interconnected via CTC adapters. In the drawing the CTC adapter addresses appear at the end of the lines that connect the mains.

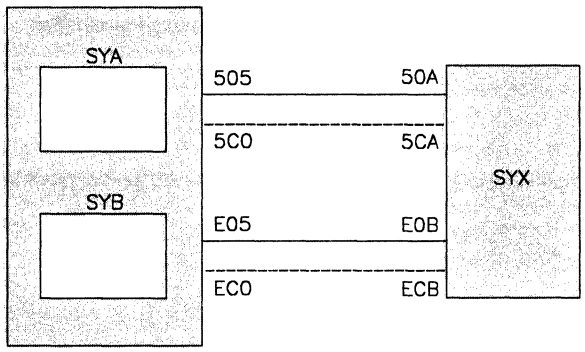
A sample initialization stream follows the drawing. This stream contains MAINPROC statements that define each main and DEVICE statements that define the main's CTC connections. The stream also contains comment statements that explain each DEVICE statement.



```
          .
          .
          .
MAINPROC,NAME=SY1,SYSTEM=JES3,...
MAINPROC,NAME=SY2,SYSTEM=JES3,...
MAINPROC,NAME=SY3,SYSTEM=JES3,...
          .
          .
          .
DEVICE,DTYPE=SYSMAIN,JNAME=SY1,JUNIT=(NONE,SY1,,,          X
5B0,SY2,,,3A0,SY3)
*
*
* The previous DEVICE statement defines the CTC
* addresses that SY2 and SY3 must use to communicate
* with SY1.  The first group of JUNIT= subparameters:
*
*                     JUNIT=(NONE,SY1,,,...)
*
* specifies that SY1 does not communicate with itself.
* The next group of subparameters:
*
*                     JUNIT=(.....5B0,SY2,,,....)
*
* specifies that SY2 uses address 5B0 to communicate
* with SY1.  The last group of subparameters:
*
*                     JUNIT=(.........3A0,SY3)
*
* specifies that SY3 uses address 3A0 to communicate
* with SY1.
*
*
DEVICE,DTYPE=SYSMAIN,JNAME=SY2,JUNIT=(4A0,SY1,,,          X
NONE,SY2,,,4B0,SY3,,OFF)
*
*
* The previous DEVICE statement defines the CTC
* addresses that SY1 and SY3 must use to communicate
* with SY2.  The OFF subparameter specified for SY3
* specifies that SY3 is initially off-line to SY2.
*
DEVICE,DTYPE=SYSMAIN,JNAME=SY3,JUNIT=(6B0,SY1,,,          X
6A0,SY2,,OFF,NONE,SY3)
*
*
* The previous DEVICE statement defines the CTC
* addresses that SY1 and SY2 must use to communicate
* with SY3.
*
```

Example 2: In the following example, a 3084 processor complex is defined as two single-image mains, SYA and SYB. SYX is the global processor. A unique *set* of CTC connections should exist between the global main and each local main.

In the illustration, SYA is active.



The following MAINPROC and DEVICE statements are required:

```
          .
          .
          .
MAINPROC,NAME=SYX,SYSTEM=JES3,CPUID=(011111,211111),...
MAINPROC,NAME=SYA,SYSTEM=JES3,CPUID=(099999,199999,299999,399999),...
MAINPROC,NAME=SYB,SYSTEM=JES3,CPUID=(009999,199999,299999,399999),...
          .
          .
          .
DEVICE,DTYPE=SYSMAIN,JNAME=SYX,                              X
JUNIT=(NONE,SYX,,,(505,5C0),SYA,,,(E05,EC0),SYB)
DEVICE,DTYPE=SYSMAIN,JNAME=SYA,                              X
JUNIT=(NONE,SYA,,,NONE,SYB,,,(50A,5CA),SYX)
DEVICE,DTYPE=SYSMAIN,JNAME=SYB,                              X
JUNIT=(NONE,SYB,,,NONE,SYA,,,(E0B,ECB),SYX)
```

DEVICE (Network)

DEVICE (Network BSC line or CTC Connection)

This form of the DEVICE statement defines a BSC line or a CTC connection that connects your node to another node in a network. You must code a DEVICE statement for each such line or connection.

For more information on defining a line or connection, see "Defining Communication Lines" in Chapter 9, "JES3 Networking."

```
DEVICE,DTYPE=NJELINE,JNAME=linename,JUNIT=(address,main,TP [, {OFF} ]  [,...])
                                                             {ON }
```

DTYPE =NJELINE
Specifies that this statement defines a BSC line or a CTC connection for a network of nodes.

JNAME =linename
Specifies a name of a BSC line or CTC connection. The variable ***linename*** must match the name you specify on the LINE= keyword of the JES3 NJERMT initialization statement that defines your node.

JUNIT = (address,main,TP [, {OFF}] [,...])
{ON }
Specifies information about the line or CTC connection. You must code the following parameters for the global. You must repeat these parameters for each local main that has access to the line or connection and could become the global.

address
Specifies the address of the BSC line or CTC connection.

main
Specifies the name of a main that has access to the line. The variable ***main*** must match the name of the main that you define on a MAINPROC statement.

TP
Specifies the destination class that is to receive messages about this line or connection.

{OFF}
{ON }
Specifies the initial online/offline status of the line or connection as viewed by the named main.

Coding Notes: Although you cannot code the DGROUP= keyword, JES3 assigns all network lines or connections to the device group named NETWORK.

Example: The following example defines a BSC line named LINE1.

```
DEVICE,DTYPE=NJELINE,JNAME=LINE1,JUNIT=(207,SY1,TP)
```

DEVICE (Define I/O Devices)

This form of the DEVICE statement defines a device that JES3 can use:

- to satisfy its own functions (JES3 device).
- to satisfy the needs of a job (execution device).
- as a JES3 device or as an execution device (shared device).

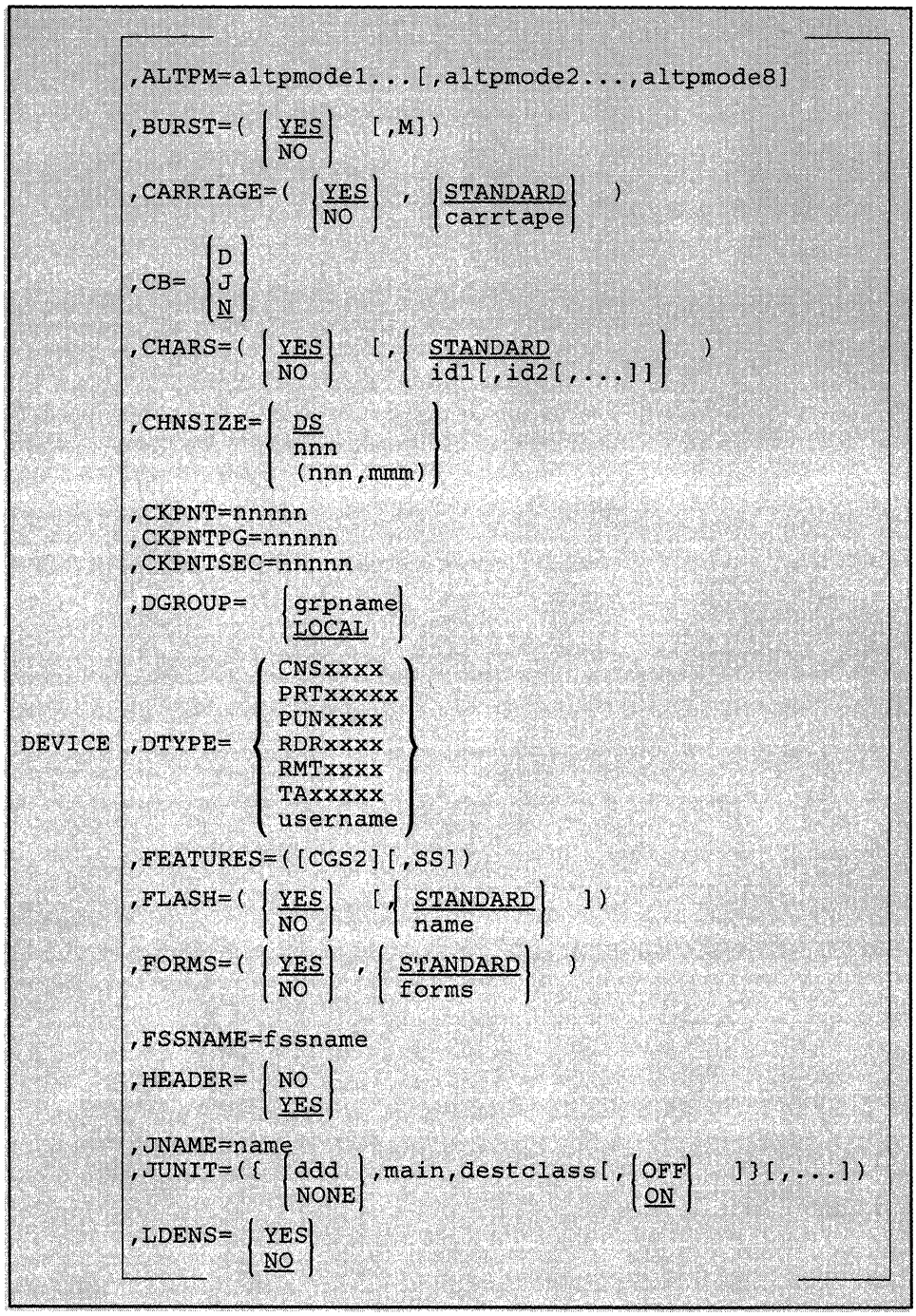
Because JES3 varies assignable devices and all execution devices online or offline to MVS as well as to JES3 at initialization, satisfy all conditions necessary for the MVS VARY command to execute properly before initializing JES3.

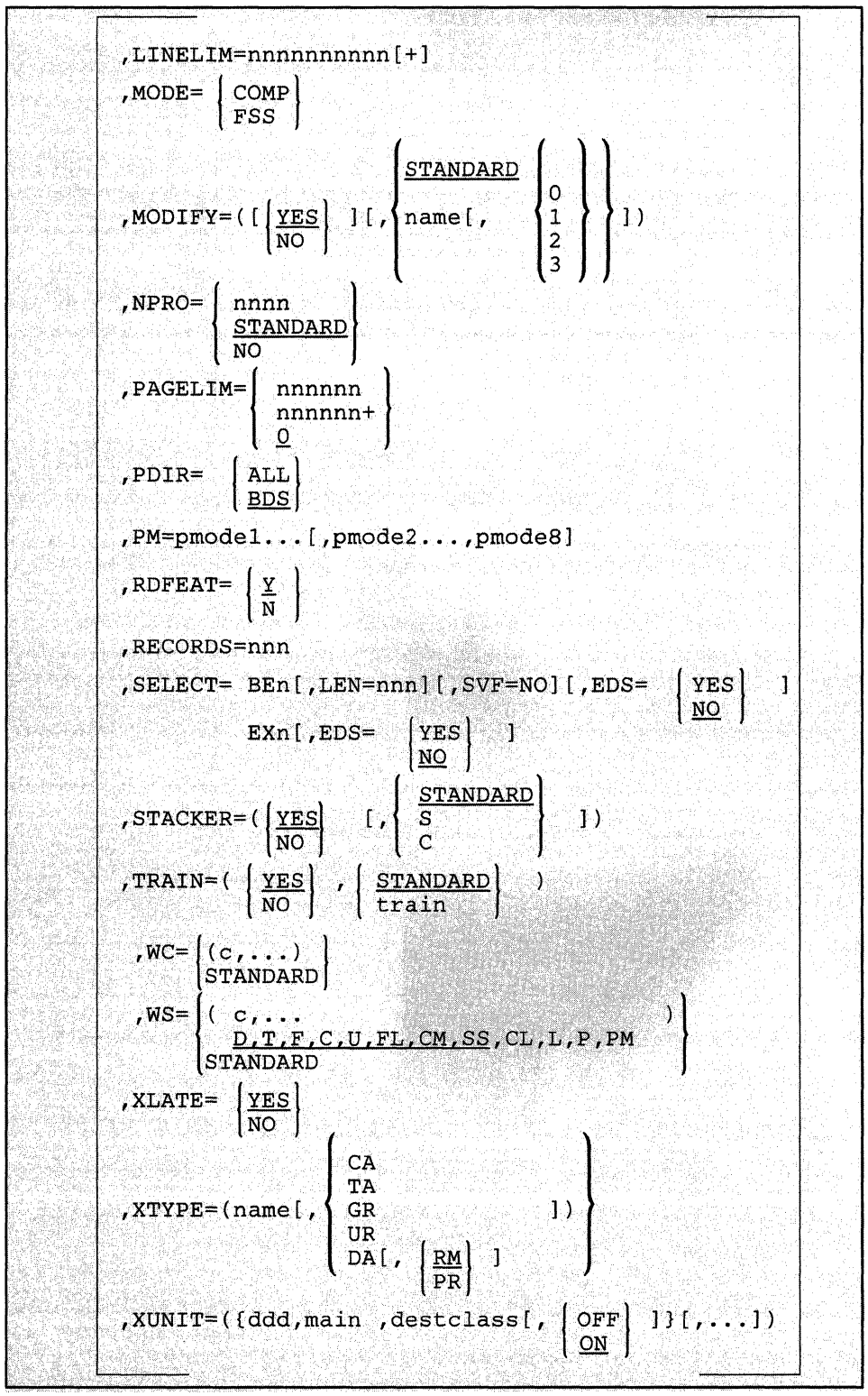
If you want two or more mains to share a device, use the XUNIT parameter to define each main. These devices are called ***shared devices***.

If you want to run a channel-attached device, such as the IBM 3800 model 3 printer, under the control of an output writer functional subsystem, you must define the device a shared device. For a non- channel attached device operating under the control of an FSS, such as the IBM 3820 page printer, you must not define the device as shared.

Except for RJP devices, you must code one DEVICE statement for each I/O device that you want JES3 to use.

For more information on defining I/O devices, see "Defining I/O Devices to JES3" in the chapter "Defining and Managing JES3 Resources."




,LINELIM=nnnnnnnnnn[+]
,MODE= { COMP / FSS }
,MODIFY=([{ YES / NO }][, { STANDARD / name[, { 0 / 1 / 2 / 3 }] }])
,NPRO= { nnnn / STANDARD / NO }
,PAGELIM= { nnnnnn / nnnnnn+ / 0 }
,PDIR= { ALL / BDS }
,PM=pmode1...[,pmode2...,pmode8]
,RDFEAT= { Y / N }
,RECORDS=nnn
,SELECT= BEn[,LEN=nnn][,SVF=NO][,EDS= { YES / NO }]
EXn[,EDS= { YES / NO }]
,STACKER=({ YES / NO } [, { STANDARD / S / C }])
,TRAIN=({ YES / NO } , { STANDARD / train })
,WC= { (c,...) / STANDARD }
,WS= { (c,... D,T,F,C,U,FL,CM,SS,CL,L,P,PM) / STANDARD }
,XLATE= { YES / NO }
,XTYPE=(name[, { CA / TA / GR / UR / DA[, { RM / PR }] }])
,XUNIT=({ddd,main ,destclass[, { OFF / ON }]}[,...])

Keyword	All Devices	All Printers	All Punches	3800-1	3800-3 3800-6	3820	4245 4248	3525	SNA RJP
ALTPM=		valid	valid	valid	valid	valid	valid	valid	
BURST=		valid	valid	valid	valid	valid	valid	valid	
CARRIAGE=		valid		valid	valid	valid	valid		
CB=				valid			valid		
CHARS=				valid	valid	valid			
CHNSIZE=									valid
CKPNT=		valid	valid	valid	valid	valid	valid	valid	
CKPNTPG=					valid	valid			
CKPNTSEC=					valid	valid			
DGROUP=	valid	valid	valid	valid	valid	valid	valid	valid	valid
DTYPE=	valid	valid	valid	valid	valid	valid	valid	valid	valid
FEATURES=				valid					
FLASH=				valid					
FORMS=		valid	valid	valid	valid	valid	valid	valid	
FSSNAME=					valid	valid			
HEADER=		valid	valid	valid	valid	valid	valid	valid	
JNAME=	valid	valid	valid	valid	valid	valid	valid	valid	valid
JUNIT=	valid	valid	valid	valid	valid	valid	valid	valid	valid
LDENS=									valid
LINELIM=		valid	valid	valid	valid	valid	valid	valid	
MODE=					valid	valid			
MODIFY=				valid					
NPRO=					valid				
PAGELIM=		valid	valid	valid	valid	valid	valid	valid	
PDIR=									valid
PM=		valid	valid	valid	valid	valid	valid	valid	
RDFEAT=								valid	
RECORDS=		valid	valid	valid	valid	valid	valid	valid	
SELECT=									valid
STACKER=				valid					
TRAIN=							valid		
WC=		valid	valid	valid	valid	valid	valid	valid	
WS=		valid	valid	valid	valid	valid	valid	valid	
XLATE=		valid	valid	valid	valid	valid	valid	valid	
XTYPE=	valid	valid	valid	valid	valid	valid	valid	valid	valid
XUNIT=	valid	valid	valid	valid	valid	valid	valid	valid	valid

Figure 12-3. Valid Keywords by Device for the DEVICE Initialization Statement

The following chart shows required and optional parameters for defining a JES3 device, execution device, or shared device. For each device type, use only the required or optional parameters. R indicates a required parameter; O indicates an optional parameter.

DEVICE statement parameters	Device type		
	JES3	execution	shared
DTYPE=	R (See notes 1 and 2)		R (See notes 1 and 2)
DGROUP=	O		O
JNAME=	R		R
JUNIT=	R		R
XTYPE=		R	R
XUNIT=		R	R
printer or punch parameters (See note 3)	O	O	O

Notes:

1. *If you code DTYPE=PRTxxxxx, or PUNxxxx, or RDRxxxx, or TAxxxxx, and the JUNIT= parameter specifies a device number, you must code the XTYPE= and XUNIT= parameters.*

2. *JES3 allows you to define a printer (1403, 3203, or 3211) as a console. You do this by coding a CONSOLE statement and a DEVICE statement for the printer.*

   - *If you define a printer as a console and intend to use it only as a console, code DTYPE=CNSxxxx on the DEVICE statement. If you intend to use the printer as both a printer and a console, code DTYPE=PRTxxxxx.*
   - *If the CONSOLE statement contains a valid destination class, JES3 treats the printer as a console.*
   - *If the CONSOLE statement specifies DEST=NONE and the DEVICE statement specifies DTYPE=PRTxxxxx, JES3 treats the printer as a printer and not as a console.*
   - *The operator can use the commands *F,V or *V to change the function of a printer from a console to a printer or from a printer to a console.*

3. *If you define a printer as a JES3 device and that printer is also defined on a JES3ABEND, JES3SNAP, or SYSABEND DD statement, the printer can contain interspersed output. This can happen when the JES3 task and the ABDUMP task write concurrently to the printer.*

Figure 12-4 lists all the I/O devices that JES3 supports. For each device type name, the figure shows whether you can specify that device type in the variable field of a DTYPE subparameter.

Generic Device Type	Can this device type name be specified in the variable field of DTYPE=?						
(model)	CNSxxxx?	PRTxxxxx?	PUNxxxx?	RDRxxxx?	RMTxxxx?	TAxxxxx?	Notes:
1403	Yes	Yes	No	No	Yes	No	
2250(3)	Yes	No	No	No	No	No	
2265	No	No	No	No	No	No	
2305(2)	No	No	No	No	No	No	
2501	No	No	No	Yes	No	No	
2540(1,2)	No	No	Yes	Yes	Yes	No	
2671	No	No	No	No	No	No	
2740	Yes	No	No	No	No	No	
2955	No	No	No	No	No	No	
3060	Yes	No	No	No	No	No	
3180(1)	Yes	No	No	No	No	No	Specify as the 3278 that is being emulated
3203	Yes	Yes	No	No	Yes	No	
3211	Yes	Yes	No	No	Yes	No	
3277(1,2)	Yes	No	No	No	No	No	
3278 (2,2A,3,4,5)	Yes	No	No	No	No	No	
3279(2,2A,2B, 2C,3A,3B)	Yes	No	No	No	No	No	
3284(1,2)	Yes	No	No	No	No	No	
3286(1,2)	Yes	No	No	No	No	No	
3288(2)	Yes	No	No	No	No	No	
3289(1,2)	Yes	No	No	No	No	No	
3290	Yes	No	No	No	No	No	
3330	No	No	No	No	No	No	
3330-1	No	No	No	No	No	No	
3330V	No	No	No	No	No	No	
3340	No	No	No	No	No	No	
3350	No	No	No	No	No	No	
3375	No	No	No	No	No	No	
3380	No	No	No	No	No	No	
3400 (2,3,4,5,6,8)	No	No	No	No	No	Yes	Specify 73400 for 7-track and 93400 for 9-track
3422	No	No	No	No	No	Yes	Supported as 3400 Tape
3430	No	No	No	No	No	Yes	Supported as 3400 Tape
3480	No	No	No	No	No	Yes	Specify 03480
3505	No	No	No	Yes	No	No	
3525	No	No	Yes, Note 1	Yes, Note 2	No	No	1. Specify 3525I for a 3525 with the two-line print feature. Specify 3525M for a 3525 with the multiline print feature. 2. 3525 is valid for RDRxxxx only if the card reader feature is installed.
3800 (1,2,3,6,8)	No	Yes	No	No	No	No	For a 3800 model 3 or 6, specify 38003. For a 3800 model 8, specify 38008. For all other models, specify 3800.
3820	No	Yes	No	No	No	No	
3880(11)	No	No	No	No	No	No	For 3880 model 11, specify 3350.
3890	No	No	No	No	No	No	
5450	Yes	No	No	No	No	No	
6870	No	No	No	No	No	No	

Figure 12-4. I/O Generic Device Type Names Supported in a JES3 Complex

ALTPM = altpmode1[,altpmode2...,altpmode8]

Specifies that JES3 can schedule data sets with this process mode to this device when an operator uses a command to change the device operating mode from what you originally defined on the MODE= keyword of the DEVICE statement. In other words this value is an alternate to that defined on the PM= keyword. You can specify 1 to 8 alphanumeric characters for the alternate process mode. You can define a maximum of eight alternate process modes for a device.

A combined total of 255 process modes and alternate process modes can exist in one JES3 complex at one time.

For more information about this keyword, see "Defining Process Modes" in the chapter "Defining and Managing JES3 Resources."

Parameter Default: If the printer is an IBM 3800 model 3 with MODE=FSS specified (or defaulted to) on this statement, the default is LINE. If the printer is an IBM 3800 model 3 with MODE=COMP specified (or defaulted to) on this statement or a line mode printer, the default is (LINE,PAGE).

BURST =({ YES / NO } [,M])

Specifies whether or not you want burst (trailer) pages printed or punched. You can specify this keyword with printer and punch devices only.

YES

Indicates that you want this device to print or punch burst pages at the end of each job.

NO

Indicates that you do not want this device to print or punch burst pages.

M

Specifies that you want JES3 to mark the edges of the burst page, or three blank pages following a job's output for ease of separation. You can modify the mark forms specification after initialization using a *CALL, *START, or *RESTART,WTR command. See *MVS/Extended Architecture Operations: JES3 Commands* for information about these commands.

Parameter Default: YES

Coding Consideration: If you specify the SELECT parameter, JES3 ignores the BURST parameter. If you specify an invalid subparameter, JES3 uses the parameter default.

CARRIAGE =({ YES / NO } , { STANDARD / carrtape })

Specifies the carriage tape or forms control buffer (FCB) associated with this device. You can specify this keyword for printers only.

NO

Specifies that the carriage tape or FCB on this printer cannot be changed during writer execution.

YES
Specifies that the carriage tape or FCB (3211, 3203, or 3800 only) can be changed during writer execution as required.

STANDARD
Specifies the standard carriage tape that you define on the OUTSERV initialization statement.

carrtape
Specifies the name (1 to 8 characters) of the carriage tape that is initially mounted on the printer or the name (1 to 4 characters) of the FCB that is to be initially loaded on the printer. If the device is a 3211, 3203, or 3800 printer, a module must be included in SYS1.IMAGELIB, having the name FCB2xxxx or FCB3xxxx, where xxxx is the 1- to 4-character name of the FCB.

For more information about SYS1.IMAGELIB, see *MVS/Extended Architecture System Programming Library: Data Management.*

Parameter Default: (YES,STANDARD)

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

CB = { **D** / **J** / **N** }

Specifies when you want JES3 to clear the device's buffer. You can specify this keyword for the IBM 3800, IBM 4245, and IBM 4248 printers only.

D
Specifies that you want the data set option used. This causes output service to clear the buffer at the end of each data set and pause shortly at the end of each data set.

J
Specifies that you want the job option used. This causes output service to clear the buffer at the end of each job and pause shortly at the end of each job.

N
Specifies that you do not want the device's buffer cleared unless required by a function. Maximum performance is obtained with CB=N. Examples of functions that require a clear printer command are:

- loading new characters
- requesting operator setup
- waiting for work

When the device is in manual mode (M specified on *X, *R, or *S and M= is specified in message IAT8562), a clear printer command is issued (to clear the buffer) before each data set and CB= has no effect.

Parameter Default: N

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

CHARS =({ **YES** / **NO** } [, { **STANDARD** / **id1,...** }])

Specifies the image(s) to be set up on the 3800 printer and whether or not they are allowed to be changed.

YES
Allows changing of images.

NO
Specifies that no image changing is allowed.

STANDARD
Specifies that the name of the image to be set up is defined on the OUTSERV initialization statement.

id1
Specifies a 1- to 4-character name of an image to be set up. Up to four names can be specified.

Parameter Default: (YES,STANDARD)

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

CHNSIZE = { **DS** / **nnn** / **nnn,mmm** }

Specifies the size of the RU chain to be transmitted to this SNA work station.

DS
Specifies that the entire data set is to be sent as a single chain.

nnn
Specifies the number of pages that the chain is to contain. Control characters in the data (skip to channel 1 for printers or eject for punches) delimit the page size. If you specify CHNSIZE=nnn but the data set has no carriage control characters, the entire data set is sent as a single chain.

The value for nnn can be any number from 1-255.

(nnn,mmm)
Specifies the number of pages (nnn) in the chain and the number of logical records (mmm) in each page. This form of the subparameter allows you to transmit, as a multiple chain, a data set that contains no control characters.

The value for nnn and for mmm can be any number from 1-255.

Notes:

1. *If you specify DS, JES3 takes output checkpoints as specified by the CKPNT parameter on this statement.*
2. *If you specify nnn or (nnn,mmm), JES3 takes an output checkpoint before each chain is transmitted.*
3. *CHNSIZE specified on a SYSOUT initialization statement or on a //*FORMAT statement overrides this CHNSIZE specification.*
4. *Sending the entire data set as a single chain gives the best performance. If, however, the receiving device requires an operator intervention or there is a transmission error, JES3 recovery will cause the entire data set to be resent.*

CKPNT =nnnn

Specifies that a checkpoint is to be taken after the specified number of records. The number specified must be between 99 and 32,767, inclusive.

Parameter Default: For a punch, 200 is the default. For a printer, the following chart shows the default for each printer type.

Printer Type	Default
3800	CKPNT=10000
3211	CKPNT=2000
Other line mode printers	CKPNT=1000
FSS-supported printers	Invalid

CKPNTPG =nnnnn

Specifies that a checkpoint is to be taken after the specified number of pages. The number specified must be between 1 and 32,767, inclusive. If this parameter is specified for any printer other than one operating in FSS mode (such as a 3800 model 3), the parameter is ignored. If the 3800 model 3 is running in compatibility mode, the checkpoint value is taken from the CKPNT parameter.

Parameter Default: If neither this parameter nor the CKPNTSEC parameter is coded, the default is CKPNTPG=200.

CKPNTSEC =nnnnn

Specifies that a checkpoint is to be taken after the specified number of seconds. The number specified must be between 1 and 32,767, inclusive. If this parameter is specified for any printer other than one operating in FSS mode (such as a 3800 model 3), the parameter is ignored. If the 3800 model 3 is running in compatibility mode, the checkpoint value is taken from the CKPNT parameter.

Parameter Default: If neither this parameter nor the CKPNTPG parameter is coded, the default is CKPNTPG=200.

DGROUP = { grpname / LOCAL }

Specifies the 1 to 8 alphameric-character name used to combine devices by physical location. When input is received from a device in a particular DGROUP, JES3 makes an attempt to send the associated output to the

same DGROUP location. This parameter is ignored when defining remote devices.

Parameter Default: LOCAL

DTYPE = { **CNSxxxx** | **PRTxxxxx** | **PUNxxxx** | **RDRxxxx** | **RMTxxxx** | **TAxxxxx** | **username** }

Specified for a JES3 device to indicate the device type. This parameter must precede the JUNIT and SELECT parameters on a DEVICE statement. See Figure 12-4 for the device type names that can be specified in the variable fields (xxxx) of the subparameters.

CNSxxxx
Identifies a console.

PRTxxxxx
Identifies a locally-attached printer.

PUNxxxx
Identifies a locally-attached punch.

RDRxxxx
Identifies a locally-attached reader.

RMTxxxx
Identifies a remote terminal (described by an RJPTERM or RJPWS statement.) For SNA RJP printers, specify RMTPRINT. For SNA RJP punches, specify RMTPUNCH.

TAxxxxx
Identifies a tape device.

username
Indicates a device type which is associated with a user DSP.

FEATURES =([CGS2][,SS])
Specifies the features of a 3800 printer.

CGS2
Specifies that the additional character generation storage feature is installed on the printer.

SS
Specifies that the burster-trimmer-stacker is attached.

FLASH =([YES / NO] [, [STANDARD / name]])

Specifies the forms overlay frame associated with this device. You can specify this keyword for an IBM 3800 printer only.

YES
Indicates that forms flashing is allowed to change.

NO
Indicates that forms flashing must not change.

STANDARD
Indicates that the corresponding ID in the OUTSERV initialization statement will be used in the mounting of the forms overlay frame.

name
Indicates that the 4-character name specified will be used in the mounting of the forms overlay frame.

Parameter Default: (YES,STANDARD)

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

FORMS =([YES / NO] , [STANDARD / forms])

Specified for printer and punch devices only. This parameter describes the forms associated with this device.

YES
Indicates that the forms on this printer or punch can be changed during writer execution as required.

NO
Indicates that the forms on this printer or punch cannot be changed during writer execution.

STANDARD
Indicates the standard forms defined on the OUTSERV initialization statement are to be used.

forms
Indicates the name (1 to 8 characters) of the forms that are initially mounted on this printer.

If you specify the name of the initial forms, you must first explicitly specify either YES or NO.

Parameter Default: (YES,STANDARD)

FSSNAME =fssname
Specifies the installation-defined name of the functional subsystem (FSS) under which a 3820 or 3800 model 3 printer will operate. The name must match the name of an output writer FSS defined on a FSSDEF initialization statement.

Parameter Default: The value specified for the JNAME parameter on this DEVICE statement.

HEADER = { **YES** / **NO** }

Specified for printer and punch devices only.

NO
Indicates that this printer or punch is not to print or punch block header pages for a job or its data sets.

YES
Indicates that this printer or punch is to print or punch block header pages for each job and each data set.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: YES

Coding Consideration: If you specify the the SELECT= keyword, JES3 ignores the HEADER= keyword. If you specify an invalid subparameter, JES3 uses the parameter default.

JNAME =name
Specifies the 1 to 8-character name of a JES3 device. This name should be unique to the device and must not be the same as used for the DGROUP parameter name. JNAME is used by the operator to refer to a JES3 device. For RJP devices, the JNAME must include (1) the work station name, which is assigned by the user, and must be exactly 5 characters in length, (2) followed by PRn for a printer, PUn for a punch, and RDn for a reader. The 'n' can be any number from and including x'1' to x'F'.

JUNIT = { ({ **ddd** / **NONE** } **,main[,destclass][** { **,OFF** / **,ON** } **],...**) }

Specifies:

- The number of the device on that main (specify NONE if you do not want to attach the device the specified main.)
- The name of a main to which the device is attached
- A destination class for messages about the device
- Whether the device is initially on-line or off-line, or not identified to the system during MVS initialization.

You must code the following group of subparameters for each processor to which the device is attached.

ddd or NONE
Specifies the device number. This must be the address to which the device is attached on the processor named *main*. Specify NONE if you do not want to attach the device to the specified main. If a device is not channel-attached (for example, the 3820 printer), do not code ddd; instead, use a comma as a position holder.

main
Specifies the name of the global processor or the name of a local processor. The device must be attached to this processor at the address specified by *ddd*.

destclass
Specifies the console destination class for messages concerning this device.

OFF
Specifies that the initial JES3 status of this device (as a JES3 device) is offline to JES3. For an assignable device, JES3 also varies the device offline to MVS.

ON
Specifies that the initial JES3 vary status of this device (as a JES3 device) is online to JES3. For an assignable device, JES3 also varies the device online to MVS.

Coding Considerations: For the 3820 printer, specifying JUNIT=(ddd... is not valid. Instead, you must use a comma as a placeholder when specifying the JUNIT= parameter. This keyword cannot precede the DTYPE= keyword.

LDENS = { **YES** / **NO** }

Specifies whether or not the set line density command is to be sent with the set vertical format (SVF) record to the work station. If the SVF=NO subparameter of the SELECT parameter is specified, this parameter is ignored.

YES
Specifies that the set line density command is to be sent with the SVF record. The work station printer will print the number of lines per inch specified in the FCB image in SYS1.IMAGELIB. The number of lines will be either 6 or 8 per inch.

NO
Specifies that the set line density command is not to be sent with the SVF record. The work station printer will print 6 lines per inch.

Note: Before specifying LDENS=YES, consult the software component description for the receiving unit to determine whether or not the software supports the set line density command. If the software does not support this command and the command is sent, the results are unpredictable.

Associated Statement/Parameter: The CARR, CHARS, FLASH, FORMS, MODIFY, STACKER, AND TRAIN parameters on the SYSOUT statement may override values specified on the DEVICE statement.

LINELIM =nnnnnnnnnn[+]
Specifies line limits for printer or punch devices only. (LINELIM is associated with the WS=L parameter on the *CALL,WTR, *START, or *RESTART operator commands or the DEVICE or OUTSERV

initialization statement.) The number specified must be between 0 and 2,147,483,647, inclusive.

nnnnnnnnnn
Indicates that a data set must have this number of output lines or less to be selected for output processing.

nnnnnnnnnn+
Indicates that a data set must have this number of output lines or more to be selected for output processing.

MODE = { **COMP** / **FSS** }

Specifies the mode of an FSS-controlled device.

COMP
Specifies that JES3 is to consider the printer as a line mode printer compatible with other 3800 printers. JES3 creates an FSS table entry for the printer but the printer runs under the control of a writer DSP in the JES3 global address space. This parameter is only valid for an IBM 3800 model 3 printer.

FSS
Specifies that JES3 is to consider the printer as one that runs under the control of an output writer FSS in its own address space.

For guidelines on choosing a value for this parameter, see "Running a Page Mode Printer" in the chapter "Defining and Managing JES3 Resources."

Parameter Default: If you specify the FSSNAME parameter on this statement, the default value for the MODE parameter is FSS. Otherwise, the default value is COMP.

Coding Consideration: If the device is an IBM 3820 printer, JES3 assumes MODE=FSS, whether explicitly coded or not.

MODIFY =({ **YES** / **NO** } **,** { **STANDARD** / **name**[**,** { **0** / **1** / **2** / **3** }] } **)**

Specified for the 3800 printer only. This parameter specifies the name of the copy modification module to be used as the table reference character for this device.

YES
Specifies that copy modification module changes can be made.

NO
Specifies that copy modification module changes cannot be made.

STANDARD
Specifies that the copy modification module to be used is defined by the OUTSERV initialization statement.

name
Specifies the 1- to 4-character name of the copy modification module.

0-3
Specifies the table reference character to be used with the copy modification module.

Parameter Default: (YES,STANDARD)

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

NPRO = { **nnnn** | **STANDARD** | **NO** }

Specifies the non-process run-out interval for a 3800 model 3 printer running in FSS mode.

nnnn
Specifies the number of seconds the printer will wait for more data before forcing out the already-printed pages. The value specified must be between 0 and 9999, inclusive.

STANDARD
Specifies that the run-out interval will be taken from the NPRO parameter on the OUTSERV statement.

NO
Specifies that the run-out interval is not to be used for this printer.

Parameter Default: STANDARD

PAGELIM = { **nnnnnn[+]** | **0** }

Specifies page limits for output processing on a 3820 or 3800 model 3 printer. (PAGELIM is associated with the WS=L parameter on the *CALL,WTR *START, or *RESTART operator commands or the DEVICE or OUTSERV initialization statement.)

nnnnnn
Indicates that a data set must have this number of output pages or less to be selected for output processing on a 3820 or 3800 model 3 printer. The value specified must be an integer between 0 and 999999, inclusive.

nnnnnn+
Indicates that a data set must have this number of output pages or more to be selected for output processing on a 3820 or 3800 model 3 printer. The value specified must be an integer between 0 and 999999, inclusive.

PDIR = { ALL / BDS }

Specifies at what point(s) a peripheral data set information record (PDIR) is to be sent with a job.

ALL
Specifies that a PDIR is to precede every data set in a job.

BDS
Specifies that a PDIR is to be sent at the beginning of the job, when JES3 sends the begin destination select (BDS) command. If you specify this parameter but the setup characteristics or copy count change between data sets, JES3 sends a PDIR between those data sets.

PM = pmode1[,pmode2...,pmode8]

Specifies that data sets with this process mode may be scheduled to this device when the device is in the operating mode that is specified on the MODE= keyword. The process mode is 1 to 8 alphanumeric characters. A maximum of eight process modes may be defined for a particular device.

For more information about this parameter, see "Defining Process Modes" in the chapter "Defining and Managing JES3 Resources."

Parameter Default:

- LINE for all line mode printers.
- (LINE,PAGE) for all printers that must run under control of an output writer FSS or a 3800 model 3 printer that you are running in FSS mode (MODE=FSS on the DEVICE statement).
- LINE for a 3800 model 3 printer that you are running in compatibility mode (MODE=COMP on the DEVICE statement).

RDFEAT = { Y / N }

Indicates whether the 3525 card punch has the card read feature.

Y
indicates that the 3525 has the card read feature.

N
indicates that the 3525 does not have the card read feature.

Parameter Default: Y

Coding Consideration: If you use this parameter, code it after coding the DTYPE= parameter.

RECORDS =nnn

Specified for printer or punch devices only.

nnn

Indicates the number of records (from 1 to 255) to be read from the spool data set at one time by JES3 output service. The record referred to is a JES3 spool record whose size is specified by the BUFSIZE parameter of the BUFFER initialization statement.

If a RECORDS parameter is specified on more than one DEVICE initialization statement, JES3 selects the largest value to determine the number of records JES3 writes to the spool sequentially.

Parameter Default: For a 3800, RECORDS=n, where

```
n=(16K + (BUFSIZE/4))/BUFSIZE
```

For a 3211, RECORDS=2.

For all other devices, RECORDS=1.

SELECT = { **BEn[,LEN=nnn][,SVF=NO][,EDS=** { **NO** } **YES]** }
{ **EXn[,EDS=** { **YES** } **]** }
{ **NO** }

Specifies that this statement defines a basic exchange device or an exchange device. The device must be attached to the SNA work station that is named in the first five characters of the JNAME parameter. The last three characters of the JNAME parameter must specify a printer. If you code the SELECT parameter, code it after coding the DTYPE parameter.

BEn

A basic exchange device is attached to the work station. Valid values for n are 1 to 9.

EXn

An exchange device is attached to the work station. Valid values for n are 1 to 9.

LEN =nnn

Specifies the record length to be used for the basic exchange device. Valid values are 1 to 128.

SVF =NO

Specifies that the set vertical format record (FCB load) is not to be sent to the work station. Specify this parameter if the data will be rerouted to a basic exchange device.

EDS = { **YES** }
{ **NO** }

For data sets on the device's diskette, specifies whether JES3 is to separate data sets with unlike print requirements (this means unlike carriage control characters, forms control buffers, or forms).

Coding Considerations: If the SELECT parameter is specified, the HEADER and BURST parameters on this statement are ignored and no header, or burst is used.

When specifying the SELECT=BEn parameter for two devices, the SELECT parameter can be the same on the two DEVICE statements for those devices. However, the LEN parameters cannot be the same because unpredictable results can occur: For example, the following is the correct way to indicate this in the initialization stream:

```
DEVICE,DTYPE=RMTPRINT,JNAME=WS000PR1,SELECT=BE2,LEN=1
DEVICE,DTYPE=RMTPRINT,JNAME=WS000PR1,SELECT=BE2,LEN=2
```

When SELECT=EXn is specified, you cannot define more exchange devices than there are physical disks available to support them.

If you specify the EDS parameter for a device that is not a SNA RJP device or for a device that uses Peripheral Data Set Information Record (PDIR) support, JES3 ignores the parameter.

STACKER =({ YES } [, { STANDARD }])
{ NO } **{ S }**
{ C }

Specified for 3800 printer only. This parameter describes the output stacking of the device. If the SS subparameter of the FEATURES parameter is not specified, STACKER=(NO,C) is forced.

YES
Indicates that changing of output stacking is permitted.

NO
Indicates that changing of output stacking is not permitted.

STANDARD
Specifies that stacking is to be as specified on the OUTSERV initialization statement.

S
Specifies that output is to be placed in the sheet stacker, where offset stacking is performed.

C
Specifies that output is to be placed in the continuous forms stacker.

Parameter Default: (YES,STANDARD)

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

TRAIN =({ YES / NO } , { STANDARD / train })

Specified for printer devices or the 3525/3525M/3525I punch with print feature only.

YES
Indicates that the initial print train or print band on this device can be changed during writer execution as required. JES3 does not provide UCS buffer loading to remote printers. However, the user may change trains or bands on a remote printer but must provide the UCS buffer loader as applicable to the type of work station.

NO
Indicates that the print train or print band on this printer cannot be changed during writer execution.

STANDARD
Indicates that the print train or print band specified on the OUTSERV initialization statement will be used. Do not specify STANDARD if TRAIN=ANY is specified on the OUTSERV initialization statement.

train
Indicates the name of the print train or print band that is mounted on this device at JES3 initialization. This field can be any valid name as specified in SYS1.IMAGELIB. For more information about SYS1.IMAGELIB, see *MVS/Extended Architecture System Programming Library: Data Management.*

If a user-created print train or band image is specified in this subparameter and XLATE=YES is specified on the same DEVICE statement, the user-created print train or band image must be included in SYS1.IMAGELIB and the user must supply a translate table member named IATWxx in SYS1.JES3LIB. The variable xx is the name of the print train or print band as specified in the TRAIN parameter of this statement or of the OUTSERV statement.

Parameter Default: (YES,STANDARD)

Coding Consideration: If you specify an invalid subparameter, JES3 uses the parameter default.

WC = { (c,...) / STANDARD }

(c,...)
Specifies writer classes. This parameter indicates SYSOUT classes in the order they are processed by the output service writers for this device. Data sets for a class not specified will not be selected. The value of c specifies a valid SYSOUT class (A-Z, 0-9) that appears as a SYSOUT parameter on a DD statement. If a list of classes is specified, separate each class by a comma.

STANDARD
Specifies that the values on the WC= parameter of the OUTSERV initialization statement will be assumed.

Parameter Default: STANDARD

WS = { **(c,...)** / **D,T,F,C,U,FL,CM,SS,CL,L,P,PM** / **STANDARD** }

(c,...) D,T,F,C,U,FL,CM,SS,CL,L,P,PM
Specifies the writer selection criteria. The value of c indicates the items JES3 output service checks, in order of importance, when selecting a data set for output processing on this device. (These WS parameter values remain in effect over a hot start.)

Specify the selections in order of importance and separate specifications with a comma.

Selection Character	Meaning
C	Carriage tape or FCB
CL	SYSOUT class
CM	Copy modification
D	Data set destination
F	Forms requested
FL	Flash
L	Limit scheduling (line, page or record)
SS	Stacker
P	Data set priority
PM	Processing mode
T	Specific device type requested
U	Train image UCS

STANDARD
Specifies that the values on the WS= parameter of the OUTSERV initialization statement will be assumed.

CAUTION

JES3 does not take any action (such as issue mount messages or load FCB and UCS) if you omit printer and punch setup characteristics on this keyword parameter. For example, a job that requires special forms may print on the wrong form if you omit forms as a selection criterion.

Parameter Default: STANDARD

XLATE = { **YES** / **NO** }

Specified for printer or punch devices only, to indicate whether or not unprintable characters in output lines are to be translated to blanks.

YES
Indicates that unprintable characters are to be translated to blanks.

NO
Indicates that unprintable characters are not to be translated to blanks.

Parameter Default: YES

Coding Considerations: This parameter is invalid for 3211 compatible devices such as the IBM 4245, and IBM 4248 devices. If you specify an invalid subparameter, JES3 uses the parameter default.

XTYPE =(name[, { **TA** / **CA** / **TA** / **GR** / **UR** / **DA[,** { **RM** / **PR** } **]** } **])**

Specifies the characteristics of the JES3-managed or jointly managed device as it is used by jobs in execution. It must precede the XUNIT parameter, which is required if XTYPE is specified.

name
Specifies a 1- to 8-character name that defines a device that can be referenced. It should match the name specified in the XTYPE parameter on a SETNAME initialization statement.

CA
Specifies that the device is cartridge tape.

TA
Specifies that the device is reel tape.

GR
Specifies that the device is graphic.

DA
Specifies that the device is direct access.

UR
Specifies that the device is unit record.

RM
Specifies that the device will contain removable volumes whose mounting is to be controlled by MDS.

PR
Specifies that the device will contain MVS permanently resident volumes.

Coding Considerations: Devices within a specific XTYPE should have compatible characteristics. For the 3820 printer, XTYPE= is not a valid parameter.

XUNIT =({ ddd,main[,destclass][, { OFF / ON }] } [,...])

Specifies the characteristics of a device attached to one or more processors. If a device is shared between two or more processors, all four subparameters are specified for each processor to which the device is attached. If a device is shared between channels of the same processor, the SYSGEN primary address should be the only subparameter indicated.

ddd
Specifies the device number of the device attached to the processor.

main
Specifies the processor to which the device is attached. This name must be the same as that defined in the NAME parameter on the MAINPROC initialization statement.

destclass
Specifies the console to which all device-related JES3 and MVS messages will be routed. If you do not specify destclass, the SETUP keyword on the STANDARDS statement determines the default value that JES3 uses:

- If you specify SETUP=NONE, JES3 uses console destination S1.
- If you specify any other value for SETUP or omit SETUP, JES3 uses the value specified for the MDSLOG keyword on the SETPARAM statement.

ON and OFF specify whether the device is to be online or offline (as an execution device) to JES3 and to MVS when it is initialized. Specifying OFF does not affect the MVS online or offline status of devices defined to MVS as permanently resident.

Coding Consideration: The XTYPE= keyword must precede the XUNIT= keyword if you specify both. Do not specify the XUNIT= keyword for an IBM 3820 printer.

Example 1: In the following example, a JES3 3211 printer, PR2, is defined on the global processor, SY2. If this device is to be used as both a printer and as a backup console, then a CONSOLE initialization statement referring to this device must also be specified.

```
DEVICE,DTYPE=PRT3211,JNAME=PR2,JUNIT=(004,SY2),XTYPE=(PRT,UR),      X
XUNIT=(004,SY2,,OFF)
```

Example 2: In the following examples, both 1403 and 3211 printers are defined when the SY2 processor is eligible to be the global processor.

```
DEVICE,DTYPE=PRT1403,JNAME=PR1,JUNIT=(00E,SY1,UR,,10E,SY2,UR),      X
XTYPE=(PRT,UR),XUNIT=(00E,SY1,,OFF,10E,SY2,,OFF)

DEVICE,DTYPE=PRT3211,JNAME=PR2,                                     X
JUNIT=(004,SY1,UR,ON,104,SY2,UR,OFF),                               X
XTYPE=(PRT,UR),XUNIT=(004,SY1,,OFF,104,SY2,,OFF)
```

Example 3: IN the following example a JES3 console, CN1, is defined. Also a CONSOLE initialization statement is required for this console.

```
DEVICE,DTYPE=CNS3277,JNAME=CN1,JUNIT=(01F,SY1)
```

Example 4: In the following example, a 3211 printer, PR3, is defined in a special device group. It may also be used by jobs in execution on SY1.

```
DEVICE,DTYPE=PRT3211,DGROUP=UPSTAIRS,JNAME=PR3,                          X
JUNIT=(00F,SY1,D1,ON),XTYPE=(PRNTR,UT),XUNIT=(00F,SY1,S10,OFF)
```

Example 5: In the following example, a 9-track tape is shared between a global processor, SY1, and a local processor. It is initially offline to JES3 and to the SY2 local processor. The console message class for JES3 on the global processor is S7, for the global processor it is S9, and for the SY2 local processor it is S6. The default group name (DGROUP) of LOCAL is assumed.

```
DEVICE,DTYPE=TA92400,JNAME=T90,JUNIT=(180,SY1,S7,OFF),                   X
XTYPE=(TAPE9,TA),XUNIT=(180,SY1,S9,,280,SY2,S6,OFF)
```

Example 6: In the following example, a direct-access device is shared between two processors SY1 and SY2. It is to be available for volume mounting required by jobs (RM specified in the XTYPE parameter).

```
DEVICE,XTYPE=(DISK,DA,RM),XUNIT=(130,SY1,S4,,230,SY2,S5)
```

Example 7: In the following example, a 3800 model 3 printer is defined as shared among 3 processors, SY1 through SY3. The printer is initially offline to all 3 processors. The printer will run in FSS mode under the control of an output writer FSS named WTRFSS1.

```
DEVICE,DTYPE=PRT38003,JNAME=PRT803,                                      X
JUNIT=(803,SY1,S1,OFF,803,SY2,S2,OFF,803,SY3,S3,OFF),                    X
XTYPE=(PRT383,UR),                                                       X
XUNIT=(803,SY1,S1,OFF,803,SY2,S2,OFF,803,SY3,S3,OFF),HEADER=NO,          X
CHARS=(YES,STANDARD),FLASH=(YES,STANDARD),STACKER=(YES,STANDARD),        X
BURST=(YES),FSSNAME=WTRFSS1,MODE=FSS,PM=PAGE,PAGELIM=3+,                 X
LINELIM=0+,NPRO=150
```

Example 8: In the following example, a 3820 printer is connected through VTAM to SY1. The printer must run in FSS mode (which is explicitly coded) and is under the control of an output writer FSS named WTRFSS2. Both *page* and *line* mode data can be routed to this device.

```
DEVICE DTYPE=PRT3820,JNAME=PRTABC1,JUNIT=(,SY1,S1,OFF),MODE=FSS,         X
WS=(D),DGROUP=TESTGRP,PM=(LINE,PAGE),FSSNAME=WTRFSS2
```

DYNALDSN (Integrity Requirements for Dynamically Allocated Data Sets)

The DYNALDSN statement is used to specify which data sets on permanently resident or reserved DASD volumes require data set integrity protection when the data set is dynamically allocated.

```
DYNALDSN { ,PROTECT=(dsn[,dsn]...) }
         { ,BYPASS=(dsn[,dsn]...)  }
```

PROTECT =(dsn[,dsn]...)
Specifies that the named data sets require full MDS data set integrity protection.

BYPASS =(dsn[,dsn]...)
Specifies that the named data sets require no data set integrity protection.

Statement Default: DYNALDSN,PROTECT=(*)

Coding Consideration: Two special characters (? and *) can be used in the dsn subparameter.

*
Indicates that the remainder of the data set name to the right of the asterisk is ignored. All data set names that match up to the * are protected or bypassed as specified.

?
Indicates that the remainder of the current qualifier to the right of the question mark is ignored. Qualifiers are separated by a period (.), by the beginning of a data set name, or by the end of the data set name. For example, in the data set name ABC.D.EFGH, the qualifiers are 'ABC', 'D', and 'EFGH'.

Examples: (1) If ABC.D.EFGH is specified in the PROTECT or BYPASS parameter, the matching dsn must be ABC.D.EFGH. (2) If AB* is specified, the matching dsn must be any data set whose name begins with AB. (3) If A?.D.EFGH is specified, the matching dsn must be a data set name with 3 qualifiers. The name must begin with A. The second qualifier must be D and the third qualifier must be EFGH. (4) If ?.?.? is specified, then any data set name with three qualifiers is a matching dsn. (5) If * is specified, then any data set name is a match.

Example:

```
DYNALDSN,BYPASS=(?.?.LIST)
DYNALDSN,PROTECT=(JOES.*,SAMS.*)
DYNALDSN,BYPASS=(?.?.LOAD,?.?.ASSEMBLE)
DYNALDSN,PROTECT=(*)
```

This example causes integrity protection to be bypassed for all data sets (a) which have exactly three qualifiers and whose third qualifier is 'LIST' or (b) whose third qualifier is 'LOAD' or 'ASSEMBLE', except when the first qualifier is 'JOES' or 'SAMS'. Integrity protection is provided for all other data sets.

DYNALLOC (Dynamically Allocate Data Sets and Devices)

The DYNALLOC statement specifies a data set or a device that is to be dynamically allocated to JES3 during initialization. The DYNALLOC statement allows you to allocate a data set or device without changing the JES3 cataloged procedure.

The DYNALLOC statement must begin in column 1; it *cannot* be continued, nor can comments be embedded.

Notes:

1. *If a dynamically allocated data set or device is unaccessible to a processor in the JES3 complex, operator intervention will be required during JES3 initialization.*

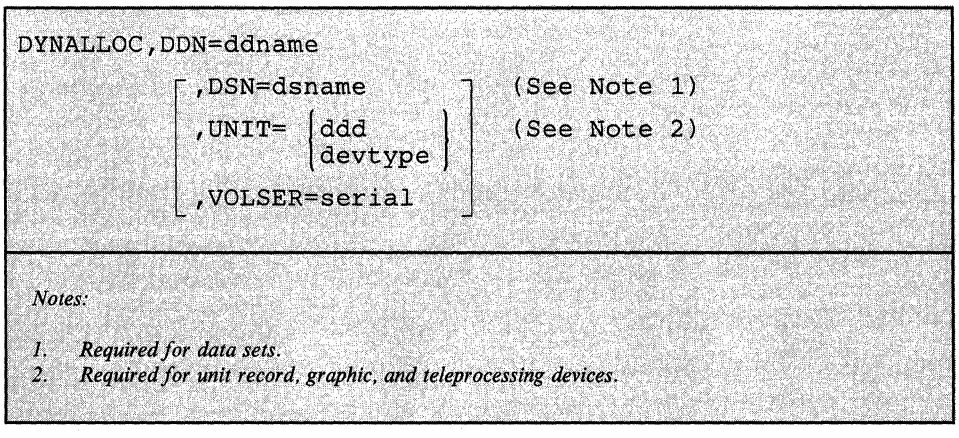
2. *You cannot dynamically allocate dummy data sets or data sets defined by the ddnames:*

   - *CHKPNT*
   - *JES3IN*
   - *JES3LIB*
   - *STEPLIB*

3. *For C/I FSS address spaces you can dynamically allocate only data sets with the ddname, IATPLBxx. Any other required data sets must be allocated by including a DD statement in the C/I FSS start procedure.*

DYNALLOC statements, if used, must be placed first in the JES3 initialization stream. All parameters pertaining to a DYNALLOC statement must be defined on one statement; continuation statements are not permitted.

The maximum number of DYNALLOC initialization statements is 1635 minus the number of JES3LIB initialization statements in your initialization stream and minus the number of DD statements in the JES3 start procedure.



DDN =ddname
Specifies the 1- to 8-character ddname associated with this data set or device.

Up to 16 input data sets can be concatenated by specifying the same ddname on successive DYNALLOC statements. During the concatenation, JES3 generates temporary ddnames of the form JS3Dxxxx, where xxxx is a 4-digit sequence number. The JS3Dxxxx ddnames cannot be used in the JES3 procedure. If a procedure library data set in a concatenation cannot be found, JES3 removes the data set from the concatenation and issues a message.

Note: You cannot concatenate spool data sets.

If a required resource (such as a dsname, unit, or volume) is unavailable, JES3 will wait for it to become available. If a required unit is offline, the operator will be asked to bring it online. An unavailable ddname is regarded as an error condition and will produce an error message.

DSN =dsname
Required to specify the 1- to 44-character name of a standard label data set. If all other parameters are included, and are specified at their maximum lengths, then the dsname is limited to 17 characters including periods and parentheses. Generation data group sets must use actual version and generation numbers.

UNIT = { ddd / devtype }
For dynamic device allocation, specifies the 3-character device number; for dynamic data set allocation, specifies the 1- to 8-character device type. This parameter is not required for a cataloged data set, but should be included for data sets that are not cataloged.

VOLSER =serial
Specifies the serial number of the volume that contains the data set. This parameter is not required for a cataloged data set, but should be included for data sets that are not cataloged.

Examples: In the following example, a JES3 spool data set, SYS1.SPOOL2, which resides on the 3330 volume SPOOL2, is allocated to JES3:

In the following example, an alternate procedure library is defined and two cataloged data sets are concatenated:

ENDINISH (End of Initialization Stream)

The ENDINISH statement is required to identify the end of the initialization statements in the initialization stream.

```
ENDINISH
```

ENDJSAM (End of JES3 I/O Statements)

The ENDJSAM initialization statement indicates the end of the JES3 spool initialization statements. ENDJSAM must immediately follow the last JES3 spool statement.

```
ENDJSAM
```

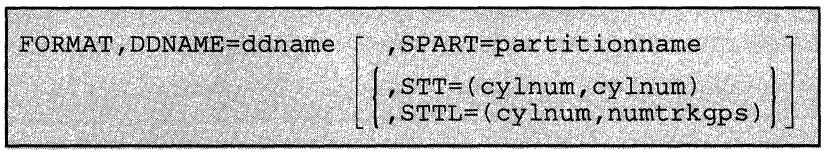
Associated Statement/Parameter: The spool initialization statements, if used, which must precede the ENDJSAM statement are: BADTRACK, BUFFER, FORMAT, TRACK, OPTIONS, and SPART. Any DYNALLOC and JES3LIB statements must precede the spool initialization statements.

FORMAT (Format Spool Data Set)

The FORMAT statement specifies that a data set residing on a direct-access spool volume is to be formatted during initialization. This statement is only specified when either an unformatted volume is introduced into a JES3 system or the BUFSIZE parameter has been changed on the BUFFER statement included in this initialization stream.

The FORMAT statement must precede the ENDJSAM statement in the initialization stream. The maximum number of FORMAT statements is 1024.

After initialization completes, replace FORMAT statements in the initialization stream with TRACK statements. If you do not do this, when you initialize JES3 in the future JES3 will issue a warning message each time it encounters a FORMAT statement for a formatted spool data set. Initialization will continue, however.



DDNAME =ddname
Specifies the ddname of the DD statement or the ddname on a DYNALLOC statement that defines the spool data set to be formatted. The ddname cannot be JES3JCT.

SPART =partitionname
Specifies that the spool data set defined by this statement is a member of the named spool partition. The partition name must match a partition name specified on an SPART statement.

To specify that the data set is a member of the default partition, omit this parameter. If you do not specify at least one spool data set as a member of the default spool partition, JES3 terminates its processing during initialization.

STT =(cylnum,cylnum)
Specifies the range of cylinders to be allocated to the single track table (STT). This range must be within the extent allocated to the data set. The value of *cylnum* specifies an absolute cylinder number. (Absolute cylinder numbers are device-dependent; the component description for the device describes the numbering scheme.)

The range indicated by *(cylnum,cylnum)* can be one cylinder (for example: 24,24) or several cylinders (for example: 24,28) and can be in ascending or descending order. JES3 allocates to the STT only those track groups that fall completely within the indicated range of cylinders. The value for *cylnum* cannot be 0.

For fixed head devices, allocate cylinders under the fixed heads for better performance. For other devices, allocate cylinders in the center of the data set.

If you omit either the STT or STTL keyword, or you specify an invalid range, JES3 allocates the centermost 2 track groups of each spool data set in the default partition as the initial STT allocation.

Note: If you change this parameter and want the change to go into effect for existing spool data sets, you must perform a cold start.

STTL =(cylnum,numtrkgps)

Specifies the location and number of track groups to allocate to the single track table (STT). These track groups must be within the extent allocated to the data set. The value for *cylnum* specifies an absolute cylinder number indicating the beginning cylinder number of the STT allocation in this extent. (Absolute cylinder numbers are device-dependent; the component description for the device describes the numbering scheme.) The value for *numtrkgps* specifies the number of track groups to allocate to this extent, beginning with the first track group that is located completely in cylinder *cylnum*. The maximum number of track groups that may be allocated to the STT is 9999.

For fixed head devices, allocate cylinders under the fixed heads for better performance. For other devices, allocate cylinders in the center of the data set.

If you omit either the STT= or STTL= keyword, or you specify an invalid range, JES3 allocates the centermost 2 track groups of each spool data set in the default partition as the initial STT allocation.

Note: If you change this parameter and want the change to go into effect for existing spool data sets, you must perform a cold start.

Example: In the following example, the direct-access spool volume defined by the ddname SPOOL1 is formatted and assigned membership in spool partition PART1. Cylinders 30 and 31 are allocated to the STT.

```
FORMAT,DDNAME=SPOOL1,STT=(30,31),SPART=PART1
```

The corresponding DD statement in the JES3 start procedure is:

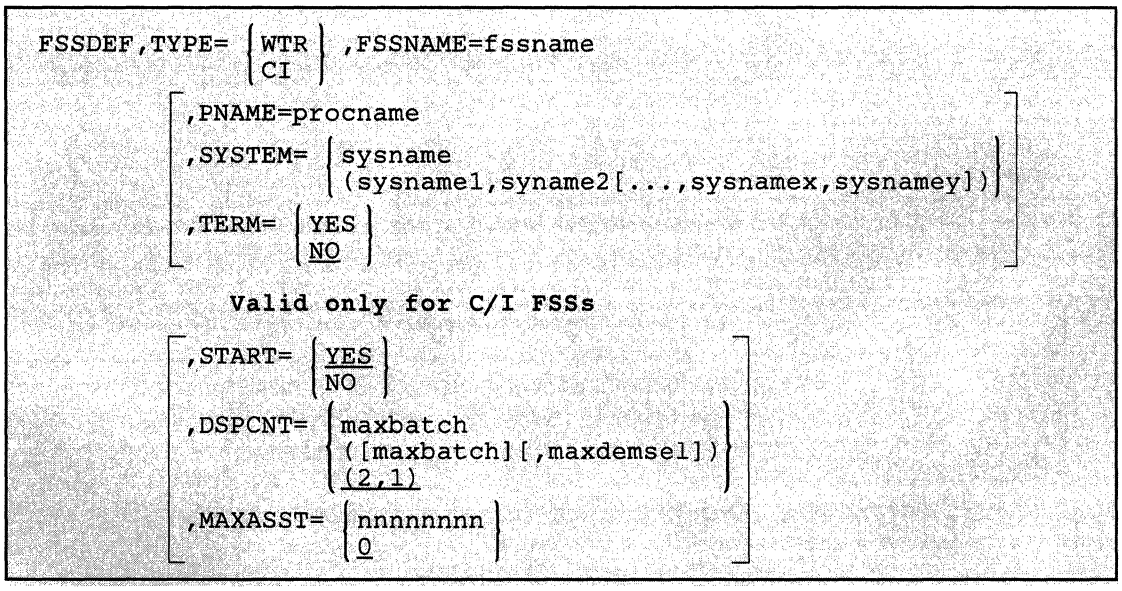
```
//SPOOL1 DD DSN=JES3.QUES2,DISP=OLD,UNIT=3330,          X
VOL=SER=MVSRW1
```

FSSDEF (Functional Subsystem Definition)

The FSSDEF statement defines the characteristics of a functional subsystem (FSS) which operates in its own address space. Use a FSSDEF statement for either of the following:

- To define every C/I FSS the installation desires.
- To control the characteristics of an output writer FSS. However, if you do not specify the FSSNAME parameter on the DEVICE initialization statement or do not provide a FSSDEF statement, JES3 creates its own output writer FSS definition for any printer that can run under a FSS.

For guidelines on how to choose values for the FSSDEF statement parameters, see "Configuring C/I Processing" in Chapter 3, "Defining and Managing C/I Service" for C/I FSSs and "Defining an Output Writer FSS" in Chapter 6, "Defining and Managing JES3 Resources" for output writer FSSs.



TYPE = { WTR / CI }

Specifies what kind of FSS is being defined.

WTR
Specifies that an output writer FSS is being defined.

CI
Specifies that a JES3 converter/interpreter (C/I) FSS is being defined.

FSSNAME = fssname

Specifies the installation-defined name of the FSS. The first character of the name must be an alphabetic character; the remaining characters must be alphameric or national characters. The name may be 1 to 8 characters.

PNAME =procname

Specifies a member of the procedure library for started task jobs, which contains a cataloged procedure for starting the FSS. The member must be in the procedure library defined by the STCPROC parameter of the STANDARDS statement, or in procedure library IATPLBST, if the STCPROC parameter is omitted.

Parameter Default: For a C/I FSS, the default member name is JES3CI. For an output writer FSS, the default member name is APSWPROC (3800-3) or APSWPROS (3820).

SYSTEM = { **sysname** | **(sysname1,sysname2[...,sysnamex,sysnamey])** }

Specifies the JES3 processor on which the FSS is to operate. The name(s) must be the same as specified on the NAME parameter of the MAINPROC statement for the processor.

If you specify pairs of processors (the (sysname1,sysname2) form), the first processor name specifies a potential global processor. The second processor name specifies the processor on which the FSS is to operate when the first processor is actually the global processor. If a dynamic system interchange (DSI) takes place, the FSS will be relocated according to the specifications of this parameter the next time the FSS is started. If the actual global processor is not the processor specified by the first processor name in any pair, the FSS is not eligible to run.

For example, suppose this parameter specifies SYSTEM=(SY1,SY2,SY2,SY3). When JES3 or the operator starts the FSS, the global processor is SY1 and JES3 establishes the FSS address space on SY2. Later, a DSI takes place and SY2 becomes the global processor. The FSS address space continues running on SY2 until, for whatever reason, the FSS terminates. When JES3 or the operator restarts the FSS, JES3 establishes the FSS address space on SY3.

You must specify either a single system name or the system names must be paired. Otherwise, the default value(s) will be used. You may specify up to 8 pairs of system names.

Parameter Default:

For a C/I FSS, the default value is the name of the global processor.

For an output writer FSS, JES3 finds the first printer defined in the initialization stream which is assigned to this FSS. If that printer is defined as attached to only one processor, that processor is the default. If the printer is defined as attached to more than one processor including the global processor, the global processor is the default. Otherwise, the default is the processor specified on the first MAINPROC statement in the initialization stream to which the printer is defined as attached.

TERM = { **YES** | **NO** }

Specifies whether or not the FSS is to be terminated if the JES3 global address space terminates as the result of an *RETURN or *DUMP operator command.

Parameter Default: NO

START = { **YES** | **NO** }

Specifies whether or not JES3 should start the FSS automatically when the processor on which the FSS is to run is connected to the global processor. This parameter applies only to C/I FSSs. If specified for an output writer FSS, this parameter is ignored.

Parameter Default: YES

DSPCNT = { **maxbatch** | **([maxbatch][,maxdemsel])** | **(2,1)** }

Specifies the maximum number of CI DSPs that can operate in the C/I FSS address space at any time. The first subparameter (*maxbatch*) specifies the maximum number of CI DSPs that process batch jobs. The second subparameter (*maxdemsel*) specifies the maximum number of CI DSPs that process demand select jobs (that is, started tasks and TSO LOGONs).

The sum of the two subparameters cannot exceed 256. CI DSPs defined to process batch jobs cannot be used to process demand select jobs, and vice versa. This parameter applies only to C/I FSSs. If specified for an output writer FSS, this parameter is ignored.

Parameter Default: (2,1)

MAXASST = { **nnnnnnnn** | **0** }

Specifies the maximum number of JCL statements that may be processed concurrently by all CI DSPs in the C/I FSS address space. The value must be an integer between 0 and 99999999, inclusive. A value of 0 means no JCL statement limit applies; JES3 does not check how many JCL statements are being processed.

For guidelines on choosing a value for this parameter, see "Managing the Scheduler Work Area," in Chapter 3, "Defining and Managing C/I Service."

This parameter applies only to C/I FSSs. If specified for an output writer FSS, this parameter is ignored.

Parameter Default: 0

Example 1: The following example defines a C/I FSS named CIFSS1 that runs on processor SY3 when processor SY1 is the JES3 global processor. The maximum number of batch CI DSPs that can operate in the C/I FSS address space at any one time is 4, while the maximum number of demand select CI DSPs is 5. The C/I FSS will not be started automatically by JES3, nor will it terminate if JES3 terminates as the result of an *RETURN or *DUMP command.

```
FSSDEF,TYPE=CI,FSSNAME=CIFSS1,SYSTEM=(SY1,SY3),DSPCNT=(4,5),START=NO,    X
TERM=NO
```

Example 2: The following example defines an output writer FSS named WTRFSS1 that runs on processor SY1. The procedure in the procedure library for starting the output writer FSS has a member name of WTR3800. The output writer FSS will terminate if JES3 terminates as the result of an *RETURN or *DUMP command.

```
FSSDEF,TYPE=WTR,FSSNAME=WTRFSS1,SYSTEM=SY1,PNAME=WTR3800,TERM=YES
```

GROUP (Job-Class Group Definition)

The GROUP statement defines the characteristics of a JES3 job-class group. A GROUP statement must define each job class group (except for the default group, JS3BATCH) named on a CLASS initialization statement. A maximum of 255 groups can be defined.

```
GROUP,NAME=groupname
     ,EXRESC=(procname,initcnt
              [,storsize    (See Note)  ]
              [,alopt                   ]
              [,unalopt                 ]
              [,storopt     (See Note)  ]
              [,devopt,devname,devcount ]
              [,devname,devcount ...]
          [ ,BAR= {PRTY}                ]
                  {nn  }
                  {16  }
          [ ,DEF=YES                    ]
          [ ,DEVPOOL=(devopt,devname,devcount
                     [,devname,devcount]...)
          [ ,JSPAN= {nnnnn}             ]
                    {ALL  }
```

Note: The positional subparameters storsize and storopt are no longer supported. They have been retained so that initialization streams will not have to be changed.

NAME =groupname

Specifies the 1- to 8-character name of a job class group. A job class is assigned to this job-class group by placing this group name on its CLASS statement. The NAME parameter must be the first parameter on the GROUP statement. **Note:** *The first character of this name must be alphabetic (MVS restriction).*

EXRESC =(procname,initcnt

```
[,storsize          (See Note)                          ]
[,alopt                                                 ]
[,unalopt                                               ]
[,storopt           (See Note)                          ]
[,devopt, devname, devcount  ,devname,devcount ]...
```

Defines the execution resources, such as initiators, processors, and devices, which are to be assigned to this job class group. Jobs in this group use devices assigned to the group to satisfy requests for mountable volumes. An EXRESC parameter must be specified for each processor on which this group may be scheduled.

Note: The positional subparameters storsize and storopt are no longer supported. They have been retained so that initialization streams will not have to be changed.

procname

Identifies the processor associated with the execution resources. The processor name must match the NAME keyword on a MAINPROC statement.

initcnt

The initcnt value indicates the dedicated initiator count. This subparameter defines the number of initiators to be assigned exclusively to this group. When scheduling jobs from this group, only the dedicated initiators will be used; therefore, this count defines the maximum number of jobs of this group that can be concurrently scheduled to the specified processor. Dedicated initiators that become idle will not be used for scheduling any other job class group. The value of initcnt must be less than the value of the MAXUSER= parameter in the IEASYSxx member of SYS1.PARMLIB that will be used for the IPL. The MAXUSER= parameter is discussed in *MVS/Extended Architecture System Programming Library: Initialization and Tuning*.

The initiator identifier is the same as the group name. Each initiator started for MVS processors is identified by the GROUP name. For example, if the dedicated initiator count for GROUP=ABC is 2, then two initiators would be started with identifier ABC. The dedicated initiator count for the default group JS3BATCH is 2.

Note: The initnum count parameter on the SELECT initialization statement overrides the initcnt parameter.

storsize

JES3 no longer supports this subparameter. It has been retained so that initialization streams will not have to be changed.

If you plan to code any of the following positional subparameters, code a comma (,) to represent this subparameter.

alopt

Determines when the execution resources are to be allocated to the group. Four options are available: DEMAND, DYNAMIC, IPL, or MANUAL.

DEMAND indicates that initiators are to be allocated to satisfy the requirements of selectable jobs in this group up to the number specified by initcnt. DEMAND is the default allocation option.

DYNAMIC indicates that the named execution resources are to be allocated when the first job of this class is eligible for scheduling on a processor.

IPL indicates that the execution resources are to be allocated whenever IPL occurs on the processor associated with this group. If subsequent reallocation is necessary, it must be done manually.

MANUAL specifies that the execution resources are to be allocated whenever the operator enters the command *F,G,main,G,grp,ON to turn on the group.

Subparameter Default: The default allocation option is DEMAND.

unalopt

Determines when the execution resources are to be released from the group. Three options are available: DEMAND, DYNAMIC, or MANUAL.

DEMAND indicates that the initiators are to be deallocated when the number of selectable jobs for this group is less than the number of allocated initiators minus the number of initiators that are currently active. DEMAND is the default deallocation option.

DYNAMIC indicates that all execution resources are to be released when no selectable jobs exist for this group.

MANUAL indicates that all execution resources are not to be released until the operator disables the job class group by issuing the command *F,G,main,G,grp,OFF. MANUAL is the deallocation option for the default group JS3BATCH.

Subparameter Default: For all groups except the defaulted groupname JS3BATCH, the default deallocation option is DEMAND.

storopt

JES3 no longer supports this subparameter. It has been retained so that initialization streams will not have to be changed.

If you plan to code any of the following positional subparameters, code a comma (,) to represent this subparameter.

devopt

Indicates those devices on the specific processor (procname) which are dedicated for use by jobs in this group for volume mounting (this is device fencing). This parameter specifies how jobs in this group may allocate the specified devices. The options are ANY and GROUP.

ANY specifies that jobs are to be able to allocate any devices that are accessible from the processor. Note that allocation will attempt to get dedicated (fenced) devices prior to using devices outside the fence.

GROUP indicates that jobs in this group may allocate requests requiring volume mounting only from devices dedicated to the group on the processor. This allows device fencing exclusively because no nondedicated devices are to be used for volume mounting. GROUP is the default device allocation option.

devname,devcount ,devname,devcount

Identifies the device(s) to be dedicated. The subparameters *devname* and *devcount* may be repeated, in pairs as is necessary.

devname

Indicates the unit address of the device or the name of the device as specified in the NAMES= parameter on the SETNAME statement. The *devname* subparameter should have *devcount* immediately following it.

For MSS dedication, one or more SDGs may be specified. The form is SDGxx, where xx is the staging drive group number (00-27). When specified, *devcount* must be omitted and 01 is assumed.

devcount
Specifies how many of the indicated devices are to be dedicated; *devcount* must be omitted for MSS staging drive group dedication. If a device number is specified for *devname*, the associated *devcount* specified must be 1.

If devices are dedicated to a group on a processor using the EXRESC parameter, the DEVPOOL parameter cannot be used. (Specification of dedicated devices in the EXRESC parameter allows device fencing for groups by processor.)

BAR = { **PRTY** / **nn** / **16** }

Specifies a job priority barrier. All jobs in this group that have a priority equal to or above this barrier must be scheduled before any attempt is made to schedule jobs below this barrier.

PRTY
Specifies that each priority level is to be treated as a barrier. Each job in a priority level must be scheduled before any job in any lower priority will be scheduled.

nn
Specifies a job priority number from 0 to 15.

16
No barrier is imposed on jobs in this group.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 16

Note: When a value other than 16 is specified, the operator should be informed of its impact. For example, a held job equal to or above the barrier which is 'awaiting selection of main' will prevent any job below the barrier from being selected.

DEF =YES
Indicates that this is the default group to be assigned to all job classes that have no GROUP parameter on the CLASS statement. Only one GROUP statement should specify the DEF=YES parameter; if you specify DEF=YES on multiple statements, the last such statement in the initialization stream defines the default class.

Parameter Default: JS3BATCH is assigned as the default group if DEF=YES is not specified on any GROUP statement in an initialization stream.

DEVPOOL =(devopt,devname,devcount ,devname,devcount ...)

Specifies that a pool of devices is to be dedicated to jobs in this group for volume mounting on all processors for which this group may be enabled. If the DEVPOOL parameter is specified, devices may not be dedicated in the EXRESC parameter.

The POOLNAMS parameter on the SETNAME statement, with appropriate XTYPE definitions on the DEVICE statement, may be used to define unique sets of devices with the sharability (physical attachment) desired for this group.

devopt

Specifies how JES3 is to allocate devices to jobs in this group.

ANY

Indicates that jobs in this group can use dedicated devices as well as nondedicated devices. JES3 always tries to allocate dedicated devices first. The default is ANY.

GROUP

Indicates that only devices dedicated to this group can be used for volume mounting.

devname

Identifies the devices to be dedicated. For MSS, one or more SDGs may be specified in the form SDGxx, where xx is the staging drive group number. When specified, *devcount* must be omitted and one device is assumed.

devcount

Indicates how many of the named devices are necessary. This subparameter is omitted for MSS staging drive group dedication.

JSPAN = { **nnnnn** | **ALL** }

Specifies the number of jobs in this group to be examined in selecting a job to be scheduled.

nnnnn

Indicates a decimal number from 1 to 32767.

ALL

Indicates that all the jobs in the group are to be examined.

Parameter Default: ALL

Examples: The following example shows how to dedicate three initiators to a job class group, ABC, on a dynamic basis (whenever jobs of that class exist in the system) on local processor SY1.

The following example defines a group named UPDATE. Jobs in this group can execute on two processors, SY1 or SY2. For processor SY1, two initiators and five devices named TAPE9 are dedicated. For processor SY2, three initiators, three devices named TAPE9, two MSS storage drive groups, and two devices named DISK are dedicated.

```
GROUP,NAME=UPDATE,                                          X
EXRESC=(SY1,2,,DYNAMIC,DYNAMIC,,GROUP,TAPE9,5),             X
EXRESC=(SY2,3,,DYNAMIC,DYNAMIC,,GROUP,TAPE9,3,              X
SDG00,SDG04,DISK,2)
```

In the following example, a pool of six TAPE12 devices is fenced for use (in conjunction with nondedicated devices) by jobs in the PROC job class group on local processors SY1 and SY2. Three initiators on both processors are dedicated to this group. (Note that the name TAPES12 defines a group of tape devices accessible from local processors SY1 and SY2.)

```
GROUP,NAME=PROC,EXRESC=(SY1,3),EXRESC=(SY2,3),              X
DEVPOOL=(,TAPES12,6)
```

See the description of the SELECT statement for an example showing how the GROUP statement relates to the SELECT statement.

HWSNAME (High Watermark Setup Names)

The HWSNAME statement is used to:

- Define, to JES3, all names by which a given device type can be referenced to enable high watermark setup (HWS) processing.
- Identify the characteristics of each device name for the specific JES3 complex. This statement can be used to define which device names are subsets of other device names. In general, the fewer the number of alternate names, the more restrictive the device name being defined. This ensures that initial allocation for devices that are reused from step to step is the most restrictive device. It also ensures that attempts to override passed or cataloged unit names are processed correctly. Non-HWS users are encouraged to supply HWSNAME information to take advantage of this function.

The device names and alternate names specified on the HWSNAME statement must be defined as JES3 supported-names on the SETNAME initialization statement or, for MSS devices, included as input to the MSS table build routine (routine IATSATAB) as JES3-managed unit names.

Use continuation statements if more alternate names are needed than can be accommodated by one statement.

```
HWSNAME,TYPE=(devname[,altnam]...)
```

TYPE =(devname[,altnam]...)
Specifies the name(s) of a device type that is valid for high watermark setup.

devname
Specifies any user-supplied or IBM-supplied group name (1 to 8 characters) associated with the specified unit name(s). The devnam identifies a device type valid for high watermark setup.

altnam
Specifies a list of valid user-supplied or IBM-supplied device names. These are alternate units to be used in device selection. The order of these names is the order in which allocation is attempted; when a device is selected, no search for a later alternate is made.

Special care must be taken when specifying alternate names. The alternate device must be compatible, for MVS allocation purposes, with the device specified by the devname subparameter. Thus, you may not specify a 2314 DASD as an alternate name for a 3330 DASD, and you may not specify 3330 as an alternate name for 2314. Similarly, you may not specify a 2400-series tape drive as an alternate for a 3400-series tape drive. You may, however, specify a 3400-series tape drive as an alternate for a 2400-series tape drive.

The following figure shows which tape drives MVS considers acceptable alternates to a request for a specific tape drive.

If you request a:	MVS may allocate a:						
	2400	2400-3	2400-4	3400-3	3400-4	3400-5	3400-6
2400 (800 BPI)	X		X		X		
2400-3 (1600 BPI)		X	X	X	X		X
2400-4 (800/1600 BPI)			X		X		
3400-3 (1600 BPI)				X	X		X
3400-4 (800/1600 BPI)					X		
3400-5 (6250 BPI)						X	X
3400-6 (1600/6250 BPI)							X

Notes:

1. MVS/XA does not support the 2400, 2400-3, or the 2400-4 device.
2. MVS does not consider any tape drives acceptable alternatives to a request for a 3480 tape drive.

Figure 12-5. Tape Drive Device Types Eligible for Allocation

If your JES3 complex includes dual-density tape drives, your initialization stream must include certain HWSNAME statements to achieve the best use of high-watermark setup. These statements are necessary because of the way MVS catalogs data sets that you requested to be created on dual-density tape drives. MVS determines which single-density tape drive (of the same series as the dual-density tape drive) uses the same density as the data set. MVS then catalogs the data set as requiring that single-density tape drive.

For example, if you create a data set on a 3400-6 dual-density tape drive at 6250 BPI, MVS catalogs the data set as requiring a 3400-5 tape drive (single density, 6250 BPI). MVS catalogs data sets this way regardless of whether or not your installation includes the single-density tape drive.

This cataloging method aids the device allocation process, because a request for a single-density device has more acceptable alternates for allocation than a request for a dual-density device. (See Figure 12-5.) Thus, in the above example, MVS may allocate either a 3400-5 or a 3400-6 for the data set. If the data set had been cataloged as requiring a 3400-6, MVS could allocate only a 3400-6.

Note: One special case exists in which MVS catalogs a data set as requiring a dual-density device. Because there is no single-density tape drive in the 3400-series with a density of 800 BPI, a data set created on a 3400-4 at 800 BPI is cataloged as requiring a 3400-4.

To use high-watermark setup, you must let JES3 know which device types are valid for HWS. If you have dual-density tape drives in your complex, you must include HWSNAME statements for all single-density tape drives that appear in the catalog as required devices for data sets. Otherwise, JES3 considers requests for those devices, whether through data set catalog entries or otherwise, ineligible for HWS.

Refer again to the example in which MVS catalogs a data set as requiring a 3400-5 tape drive when the data set was created on a 3400-6 tape drive at 6250 BPI. If you do not have any 3400-5 devices in your complex, MVS allocates a 3400-6 device for that data set. (See Figure 12-5.) If you want jobs using that cataloged data set to be eligible for HWS, you must tell JES3 that a request for a 3400-5 is a valid HWS request and that it is equivalent to a request for a 3400-6. Code your HWSNAME statements so that the 3400-6 and 3400-5 appear as alternates to each other:

```
HWSNAME,TYPE=(3400-6,3400-5)
HWSNAME,TYPE=(3400-5,3400-6)
```

See Figure 12-6 for sample HWSNAME statement specifications for various configurations of 3400-series tape drives. The figure shows only the recommended generic device type specifications; you may add any esoteric names appropriate for your installation. Note that when your complex includes a single-density tape drive, the allocation of the single-density device cannot satisfy a request for the dual-density device. Therefore, the two device types are not equivalent. You must not specify the single-density device as an alternate to the dual-density device when the complex includes the single-density device.

Use the sample HWSNAME statements for 3400-series tape drives as a guideline for writing your HWSNAME statements for 2400-series tape drives.

For this configuration:			
3400-6 (1600/6250 BPI	3400-5 (6250 BPI)	3400-3 (1600 BPI)	Include the following statements in your initialization stream
X	X	X	HWSNAME,TYPE=(3400-6) HWSNAME,TYPE=(3400-5,3400-6) HWSNAME,TYPE=(3400-3,3400-6)
X	X		HWSNAME,TYPE=(3400-6,3400-3) HWSNAME,TYPE=(3400-5,3400-3,3400-6) HWSNAME,TYPE=(3400-3,3400-6)
X		X	HWSNAME,TYPE=(3400-6,3400-5) HWSNAME,TYPE=(3400-5,3400-6) HWSNAME,TYPE=(3400-3,3400-5,3400-6)
X			HWSNAME,TYPE=(3400-6,3400-5,3400-3) HWSNAME,TYPE=(3400-5,3400-3,3400-6) HWSNAME,TYPE=(3400-3,3400-5,3400-6)
	X	X	HWSNAME,TYPE=(3400-5) HWSNAME,TYPE=(3400-3)

Figure 12-6. HWSNAME Statements for 3400-series Tape Drive Configurations

For more details on how MVS allocates and catalogs data sets, see *MVS/Extended Architecture System Logic Library: Volume 2.*

Associated Statement/Parameter: The SETUP parameter on the STANDARDS statement specifies whether or not high watermark setup is to be used. (This statement may be overridden by the //*MAIN JES3 control statement.)

Every device name specified on the HWSNAME statement must be specified on the applicable SETNAMES statement(s).

Example 1: The following statements define the relationship among the names supplied for a user's tape drives. Assume the user's JES3 complex includes 3400-4, 2400-4, and 2400 tape drives. The esoteric (user-defined) name TAPE encompasses all tape drives in the complex, while the esoteric name DUALDEN encompasses only dual-density tape drives. (Statement numbers to the left of each statement are for the purpose of discussion only.)

```
1.  HWSNAME,TYPE=(TAPE,DUALDEN,3400-4,3400-3,2400-4,2400-3,2400)
2.  HWSNAME,TYPE=(DUALDEN,3400-4,3400-3,2400-4,2400-3,TAPE)
3.  HWSNAME,TYPE=(2400,3400-4,3400-3,2400-4,2400-3,DUALDEN,TAPE)
4.  HWSNAME,TYPE=(2400-4,2400-3,3400-4,3400-3,DUALDEN,TAPE)
5.  HWSNAME,TYPE=(3400-4,3400-3,DUALDEN,TAPE)
6.  HWSNAME,TYPE=(2400-3,2400-4,3400-4,3400-3,DUALDEN,TAPE)
7.  HWSNAME,TYPE=(3400-3,3400-4,DUALDEN,TAPE)
```

Statement 1 defines all the device names that can be used to satisfy a request for UNIT=TAPE. Note that both generic and esoteric alternate names are used. Statement 2 defines all the names that can be used to allocate a UNIT=DUALDEN request. Note that 2400 is not listed because it is not a dual-density device and there are 2400s in the complex. On the other hand, 2400-3 and 3400-3 do appear as alternates because there are no devices of those types in the complex, so a request for one always results in the allocation of a dual-density device. TAPE also appears as an alternate because TAPE could be used to satisfy a DUALDEN request.

In statement 3, all device names are shown as valid alternates for 2400. This is in keeping with allocation rules that allow a single-density request to be satisfied by certain dual-density devices. Statements 4 and 5 define the alternate names for the respective dual-density generic names. Note that 3400-4 is an alternate for 2400-4, but that the reverse is not true. This is because a 2400-4 cannot be used to satisfy a request for a 3400-4.

Statements 6 and 7 define single-density device names that are entered in the catalog for data sets created on 2400-4 or 3400-4 dual-density devices at 1600 BPI.

Assuming the HWSNAME input shown, the following JCL results in one device, a 2400-4, being allocated for the entire job.

```
//TST JOB
//STPA EXEC PGM=IEFBR14
//DDA1 DD DSN=A,UNIT=2400,VOL=SER=V1,DISP=OLD
//STPB EXEC PGM=IEFBR14
//DDB1 DD DSN=B,UNIT=DUALDEN,VOL=SER=V2,DISP=OLD
//STPC EXEC PGM=IEFBR14
//DDC1 DD DSN=C,UNIT=2400-4,VOL=SER=V3,DISP=OLD
```

According to the HWSNAME entries, a 2400-4 can be used to satisfy all unit requests in this job. That is, 2400-4 is an acceptable alternative to the DDA1's request for a 2400 unit (HWSNAME card 3) and DDB1's request for a DUALDEN unit (HWSNAME statement 2). One 2400-4 unit, therefore, will be assigned for DDA1 and this will be propagated to the remainder of the requests.

By contrast, if you want to completely separate the single-density and dual-density drives, the following could be provided.

```
HWSNAME,TYPE=(TAPE,DUALDEN,3400-4,3400-3,2400-4,2400-3)
HWSNAME,TYPE=(DUALDEN,3400-4,3400-3,2400-4,2400-3,TAPE)
HWSNAME,TYPE=(3400-4,3400-3,DUALDEN,TAPE)
HWSNAME,TYPE=(2400-4,2400-3,3400-4,3400-3,DUALDEN,TAPE)
HWSNAME,TYPE=(TAPSING,2400)
HWSNAME,TYPE=(2400,TAPSING)
```

With this HWSNAME input, the same JCL would result in two tape drives being assigned: one 2400 drive, to satisfy the DDA1 request, and the 2400-4 drive to be used for both the DDB1 and DDC1 requests. Note that the single-density tape drives defined as alternates to the dual-density tape drives do not violate the decision to separate dual- and single-density devices. As in the previous set of HWSNAME statements, these single-density tape drives are equivalent to the dual-density tape drives because a request for one of them will always result in the allocation of a dual-density device.

Example 2: Assume that an installation has 3400-5 and 3400-6 tape drives. The 3400-5 single density tape drive should not be defined as an alternate for the 3400-6 dual density tape drive.

For example, consider the following JCL:

```
//STEPA EXEC PGM=X
//DD1 DD UNIT=3400-5,VOL=SER=VOL001,DISP=OLD
//DD2 DD DSN=ABC,UNIT=AFF=DD1,DISP=OLD
```

Assume that data set ABC is cataloged as residing on volume VOL003 and requiring a device type of 3400-6.

In a complex where both the 3400-5 and 3400-6 devices exist and are JES3-managed, the following HWSNAME statements should be used:

```
HWSNAME,TYPE=(3400-5,3400-6)
HWSNAME,TYPE=(3400-6)
```

This indicates that a 3400-6 is an acceptable alternate for a 3400-5, but the 3400-5 is not an acceptable alternative for a 3400-6. The UNIT affinity in the example, therefore, is negated because the referenced 3400-5 is not a valid subset of the cataloged unit (3400-6). Two devices would be required under these circumstances.

If the JES3 complex contained only 3400-6 tape drives, then the volume requested by DD1 would have to be mounted on a 3400-6 and the unit affinity on DD2 would be valid. However, the HWSNAME information would not allow JES3 to honor the affinity and two devices (3400-6) would still be required. But by modifying the HWSNAME statements as follows, the user could enable JES3 to allocate devices more efficiently:

```
HWSNAME,TYPE=(3400-5,3400-6)
HWSNAME,TYPE=(3400-6,3400-5)
```

Because the installation contains only dual density devices (3400-6), any device request can be satisfied by any available unit.

INTDEBUG (Initialization Debugging Facility)

The INTDEBUG statement allows you to specify error message text and an index value. If the specified message text is issued the number of times indicated by the index value, JES3 issues a DM005 JES3 completion code and takes a storage dump.

This initialization debugging facility monitors all occurrences of a specified message up to the occurrence where a dump is being requested. Only one message can be monitored at a time. Following the dump, the processor must be reinitialized.

The INTDEBUG statement must start in column one; it *cannot* be continued, nor can comments be embedded. Only one INTDEBUG statement is active at a time; if more are specified, only the last statement is used. An INTDEBUG statement must be placed in an initialization deck so that it precedes the point where the error message occurs in the JES3OUT listing. (Exception: INTDEBUG must not precede a DYNALLOC or JES3LIB statement). The INTDEBUG statement can not be continued.

```
INTDEBUG,index,message$$
```

index
Required to identify which occurrence of a message should produce a dump. The index must be a decimal number from 1 to 9 indicated in column 10 of this card. To determine the value of the index, count the number of times a message occurs after the INTDEBUG statement, up to and including the occurrence that is to cause the dump.

message
Required to indicate text to compare against generated error messages; the text is terminated by $$. The message text may be variable length, in columns 12-69, with $$ indicating the end. The $$ terminator may be placed as far as columns 70 and 71. If the terminator is omitted, the message text is assumed to be in columns 11-71.

Examples: In the example which follows, a dump is needed to analyze why, when error message IAT3602 occurs for the third time, it contains meaningless data. This statement is placed so that problem message IAT3602 is the third IAT3602 message after this statement.

```
INTDEBUG,3,IAT3602 MAIN$$
```

In the example which follows, error message IAT3251 is being issued for an IOB parameter on a BUFFER statement. To get a dump associated with this parameter error, this statement is placed in front of the BUFFER statement:

```
INTDEBUG,1,IAT3251 BAD KEYWORD,(IOB),SCAN ENDED$$
```

JES3LIB (JES3 Library Dynamic Allocation)

Use this statement to dynamically allocate a private JES3 step library. The concatenation identified by these statements completely replaces any STEPLIB library defined in the JES3 and JES3CI procedures in SYS1.PROCLIB.

The JES3LIB statement must start in column 1; it *cannot* be continued, nor can comments be embedded.

The JES3LIB statements, if included, must follow the DYNALLOC statements, if any, in an initialization stream. Up to 16 JES3LIB statements may be specified. When more than one JES3LIB statement is included, the specified data sets are dynamically concatenated in the order in which the statements occur. The rules for these concatenated data sets are the same as those for MVS concatenated data sets.

All parameters pertaining to a JES3LIB statement must be defined on the statement. Continuation statements are not permitted.

The ddname JES3LIB is used to dynamically allocate the data set on the first JES3LIB statement. Data sets on subsequent statements are allocated with ddnames of the form JS3Dxxxx, where xxxx is a sequence number. JS3Dxxxx ddnames cannot be used in the JES3 procedure.

The following modules must be in either (1) the JES3 STEPLIB data set defined in the JES3 start procedure and the C/I FSS start procedure, or in (2) one of the LINKLIST data sets defined during the MVS initial program load (IPL) procedure:

IATABM	IATGRTX	IATINTK	IATSIBD
IATABN0	IATINFA	IATINTKF	IATSNLS
IATBDCI	IATINGL	IATISCB	IATSSDQ
IATGROP	IATINGS	IATNUC	IATSSVT
IATGRSQ	IATINSV	IATNUCF	IATUX15

The following modules must not be in any library named in the STEPLIB DD statement or on a JES3LIB statement. You must request that MVS load these modules into the link pack area during MVS initialization. To find out how to do this, see *MVS/Extended Architecture System Programming Library: Initialization and Tuning*.

IATABIP	IATIIII	IATSIDO	IATSIWO
IATDMBS	IATOSDI	IATSIDR	IATSI34
IATDMDK	IATSIAD	IATSIEM	IATSSCM
IATDMDM	IATSIBS	IATSIJS	IATSSDI
IATDMDS	IATSICA	IATSIMS	IATSSRE
IATDMEB	IATSICC	IATSIOP	IATUX26
IATDMFR	IATSICD	IATSIOR	IATUX32
IATDMIT	IATSICF	IATSIST	IATUX57
IATDMUB	IATSICN	IATSIVL	

```
JES3LIB,DSN=dsname
        [,VOLSER=volserial]
        [,UNIT=cccc       ]
```

DSN =dsname
Specifies the 1- to 44-character data set name of the step library. If you specify the other parameters (VOLSER= and UNIT=) at their maximum lengths, dsname is limited to 31 characters. For generation data group data sets, actual version and generation numbers must be used.

VOLSER =volserial
Specified to indicate the 1- to 6-character volume serial number associated with the STEPLIB data set. This parameter is not required if the data set is cataloged.

UNIT =cccc
Specified to indicate the unit where the volume resides. This parameter is not required if the data set is cataloged. The unit specification is defined the same way as it is on a DD statement.

Statement Default: If no JES3LIB statements are specified, the STEPLIB defined in the JES3 start procedure, if any, will be used.

Example: In the following example, three concatenated STEPLIB data sets would replace the SYS1.JES3LIB STEPLIB distributed in the JES3 procedure in SYS1.PROCLIB.

```
JES3LIB,DSN=JES3.TEAM3,VOLSER=JOBLIB,UNIT=3330
JES3LIB,DSN=JES3.JES3MOD,VOLSER=JOBLIB,UNIT=3330
JES3LIB,DSN=JES3.JES3LIB,VOLSER=JOBLIB,UNIT=3330
```

The following example shows the JES3LIB statements if the concatenated data sets indicated above were cataloged:

```
JES3LIB,DSN=JES3.TEAM3
JES3LIB,DSN=JES3.JES3MOD
JES3LIB,DSN=JES3.JES3LIB
```

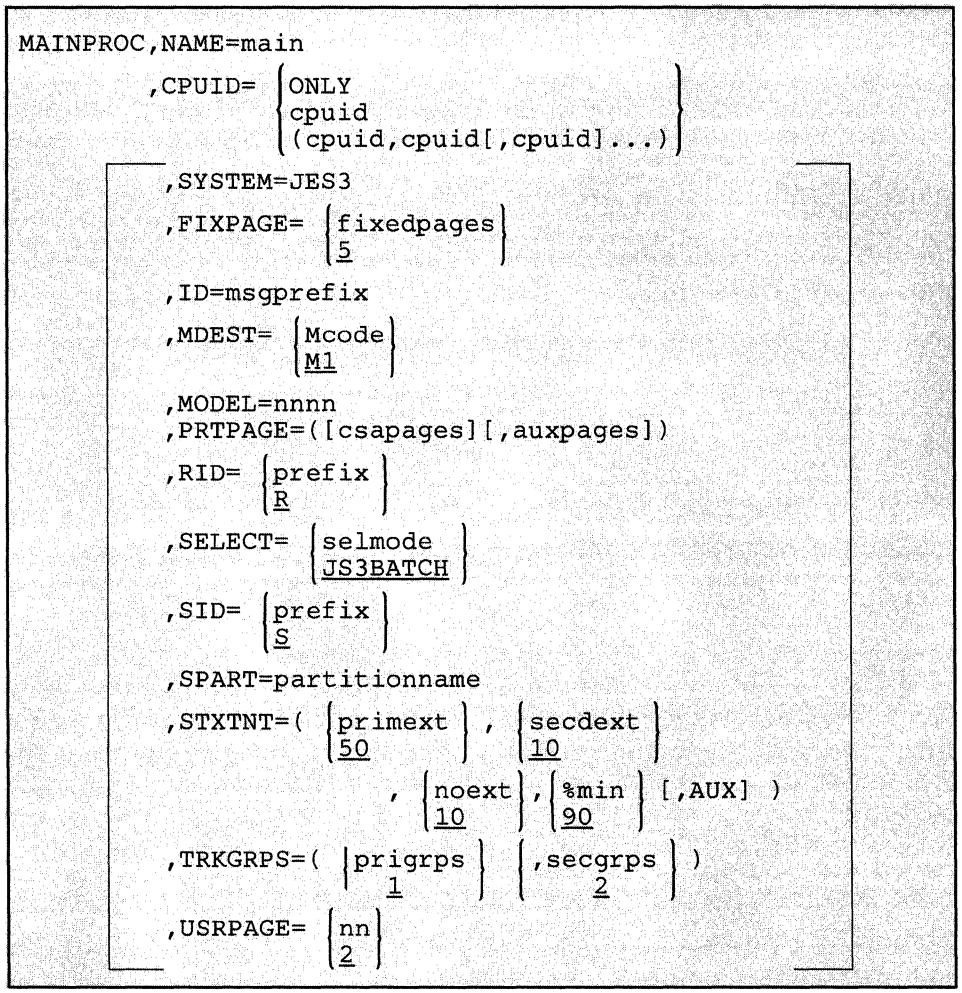
MAINPROC (Define a JES3 Processor)

This statement defines a uniprocessor (such as a 3083) or a multiprocessor (such as a 3081 or 3084) as a JES3 main. The initialization stream must include one MAINPROC statement for each main that you wish to define to JES3.

To define a uniprocessor, code the processor CPUID on one MAINPROC statement.

To define a multiprocessor, code the processor CPUIDs on one MAINPROC statement.

To define a processor complex (partitionable multiprocessor), you can code the same processor CPUIDs on multiple MAINPROC statements. For more information, see "Defining Mains" in Chapter 7, "Defining and Managing JES3 Mains and Storage."



NAME =main

Specifies the name (1 to 8 characters) of a JES3 main. The name should match the purpose of the main, or at least the hardware it is using. This name you specify is used in operator commands and in the CLASS, CONSOLE, DEVICE, GROUP, MSGROUTE, and SETACC initialization statements to refer to the main. It must be the first or second parameter on the statement. Do not use names or abbreviations of operator commands, such a 'CONTROL', or message destination classes as names.

CPUID = { **ONLY** | **cpuid** | **(cpuid,cpuid[,cpuid]...)** }

Identifies one or more specific processors or the CPUs of a processor complex. If you specify *cpuid* or *(cpuid,cpuid ,cpuid ...)*, *cpuid* must be the 6-digit processor identification number that the Store CPU ID (STIDP) instruction returns.

ONLY

Supports the global-only testing environment. Limit the use of this subparameter because locals cannot be added without doing a cold start. If the global is being run on a multiprocessor, do not specify CPUID=ONLY.

cpuid

Specifies the CPUID of a uniprocessor. If you intend to use a multiprocessor as uniprocessors:

- On one MAINPROC statement specify the CPUID of one CPU.
- On another MAINPROC statement specify the CPUID of the other CPU.

(cpuid,cpuid[,cpuid]...)

Specifies the CPUIDs of a multiprocessor. If the multiprocessor is partitionable, the CPUIDs can be included on more than one MAINPROC statement.

If you have MSS connections to any mains in a partitionable processor complex, then you must include a dummy CPUID. This number must be unique, and should not actually exist on the complex. This dummy CPUID should follow standard naming conventions for CPUIDs, and should be used as input to the JES3 MSS Table Build program. For more information about defining the MSS, see Chapter 6, "Defining and Managing JES3 Resources" on page 6-1.

SYSTEM =JES3

Specifies that JES3 is running on this main. You do not need to code this parameter.

FIXPAGE = { **fixedpages** / **5** }

Specifies the number of pages of USAM protected data buffers (PBUFs) that JES3 is to fix in storage during JES3 initialization.

- If fixedpages is less than csapages (specified on the PRTPAGE parameter), JES3 fixes the specified number of pages in CSA only.
- If fixedpages is greater than csapages, JES3 fixes all PBUF pages in CSA. If you specify auxpages on the PRTPAGE= parameter, JES3 determines the remaining number of pages to be fixed (fixedpages - csapages) and fixes that number of pages in the JES3 auxiliary address space.

Parameter Default: 5

ID =msgprefix

Indicates a prefix (1 to 8 characters) to be added to every message sent to or received from this main and logged on the MDEST console (determined by the MDEST parameter). This prefix precedes an R= or S= prefix appearing with a message. If you specify any combination of the ID, RID, or SID parameters, the last such parameter in the initialization stream defines the default prefix.

Parameter Default: If you do not specify the ID parameter, the specified values or the default values for the RID and SID parameters are used.

MDEST = { **Mcode** / **M1** }

Specifies the destination class for messages that are about this processor. Mcode is a 2- or 3-character code selected from the range M1 to M32. This destination class is associated with a specific console by the DEST parameter on a CONSOLE initialization statement.

Parameter Default: M1

MODEL =nnnn

Specifies the 4-digit model number of the processor (for example, 0168, 3033, or 3081). This parameter is optional, but can be helpful when duplicate CPUIDs exist in the same JES3 complex. In a processor complex, however, such as a 3084, including the model number will not distinguish duplicate CPUIDs.

Coding this parameter incorrectly will cause JES3 processing to terminate.

PRTPAGE =([csapages] ,[auxpages])

Specifies the number of pages of storage that JES3 may use as USAM protected data buffers (PBUFs).

csapages

Specifies the number of pages of CSA that JES3 is to use.

auxpages

Specifies the number of pages in the JES3 auxiliary address space that JES3 is to use.

The total number of pages that JES3 may use is csapages + auxpages. The total number of pages specified (csapages + auxpages) must be between 16 and 256 inclusive.

If you do not want to allocate JES3 auxiliary storage, specify 0 as the auxpages value and exclude the AUX subparameter on the STXTNT keyword of the MAINPROC statement.

Parameter Defaults:

If the PRTPAGE parameter or both PRTPAGE subparameters are not specified:

csapages: 16 times the number of spool data sets or 256, whichever is smaller.
auxpages: 0

If only one subparameter is specified, the other subparameter defaults to zero.

RID = { **prefix** / **R** }

Specifies a message prefix (1 to 8 characters) for every message received *from* this JES3 main and logged on the MDEST console (determined by the MDEST parameter). This prefix will precede the R= prefix which automatically identifies every message that is received.

To suppress the automatic R prefix, code a pound sign (#) as the last character of the prefix. For example, RID=SYS2 provides SYS2R=message; but RID=SYS2# provides SYS2=message.

If you specify the RID parameter in addition to the ID parameter, the last such parameter in the initialization stream defines the default prefix.

Parameter Default: R

SELECT = { **selmode** / **JS3BATCH** }

Specifies the name of the job selection mode to be initially assigned to this main. The scheduling controls associated with this job selection mode should be appropriate for the main being defined by this MAINPROC statement. The selmode must match the NAME parameter on a SELECT initialization statement (except if the default JS3BATCH is used).

Parameter Default: JS3BATCH (the selection mode consisting of SELECT statement default values will be used).

SID = { **prefix** / **S** }

Specifies a message prefix (1 to 8 characters) for every message sent *to* this JES3 processor and logged on the MDEST console (determined by the MDEST parameter). This prefix will precede the S=prefix which automatically identifies every message that is sent.

To suppress the automatic S prefix, code a pound sign (#) as the last character of the prefix. For example, SID=SYS1 provides SYS1S=message; but SID=SYS1# provides SYS1=message.

If you specify the SID parameter in addition to the ID parameter, the last such parameter in the initialization stream defines the default prefix.

Parameter Default: S

SPART =partitionname

Specifies the spool partition that JES3 is to use for jobs that execute on the main defined by this statement. The partition name must match a partition name specified on an SPART statement. To specify the default spool partition, omit the SPART parameter.

If you specify an undefined partition name, JES3 uses the default partition.

A partition name specified on a //*MAIN JES3 control statement or on a CLASS statement can override the partition name specified by this parameter. For a discussion of the order of overrides and when they occur, see "Determining the Order of Spool Partition Overrides" and "How the User Can Request A Spool Partition" in Chapter 4, "Defining and Managing Spool Data Sets."

STXTNT =([primext / 50] [, [secdext / 10] , [noext / 10] , [%min / 90] [,AUX]])

Specifies:

- the amount of storage, common service area (CSA), and JES3 auxiliary address space that JES3 is to allocate for staging areas
- a limit that prevents individual address spaces from depleting the staging area pool.

primext

Specifies, in multiples of 1K, the size of the primary staging area pool. JES3 allocates this storage in CSA. The minimum value you can specify is 20 and the maximum value is 16384.

secdext

If you do not specify the AUX subparameter, the secdext subparameter specifies, in multiples of 1K, the size of the secondary staging area pool. JES3 allocates this storage in CSA only after the primary staging area pool is filled.

If you specify the AUX subparameter, the secdext subparameter multiplied by the noext subparameter (secdext X noext) specifies, in multiples of 1K, the amount of storage JES3 is to allocate for the secondary staging area pool. JES3 allocates this storage in the JES3 auxiliary address space during JES3 initialization. The secdext subparameter also specifies, in multiples of 1K, an additional amount of storage JES3 is to allocate after the primary and secondary staging areas are filled. JES3 allocates this additional storage in the JES3 auxiliary address space as often as needed after initialization.

The minimum value you can specify for the secdext subparameter is 0 and the maximum value is 16384.

noext

If you do not specify the AUX subparameter, the noext subparameter specifies the number of times JES3 may allocate the amount of storage specified by the secdext subparameter. In this case, if a job requests a staging area when both the primary and secondary staging areas are filled, JES3 abnormally terminates (abend code 6FB) the job.

If you specify the AUX subparameter, JES3 uses the noext subparameter in combination with the secdext subparameter. See the secdext subparameter for an explanation of how this is done.

% min

Specifies the percentage of the total staging area pool (primary + all secondary extents) to be allocated before a minimal storage condition results. When a minimal storage condition exists, all requestors (except JES3) of staging areas will have to wait until enough staging areas are freed to be 5% below the minimal limit.

If you specify the AUX parameter, the percentage is calculated from the primary extent plus the total secondary extent. For example:

```
(PRIMEXT + (SECDEXT x NOEXT))
```

To prevent a slow-down condition, increase the SECDEXT or NOEXT values. Increasing these values allows JES3 to allocate more AUX staging areas before a minimum condition is detected. For example, if you specify:

```
STXTNT=(50,10,20,90,AUX)
```

JES3 initially allocates 250K of staging areas. A slow-down condition occurs when 90% of these staging areas are in use, regardless of further secondary allocations in the auxiliary address space.

AUX
Specifies that JES3 is to allocate the secondary staging area pool in the JES3 auxiliary address space.

If you do not want to allocate JES3 auxiliary storage, exclude this subparameter and specify 0 as the auxpages value on the PRTPAGE= keyword of the MAINPROC statement.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: (50,10,10,90)

TRKGRPS $=\left(\left\{\begin{matrix}\textbf{prigrps}\\ \underline{\textbf{1}}\end{matrix}\right\}\quad\left\{\begin{matrix}\textbf{,secgrps}\\ \underline{\textbf{,2}}\end{matrix}\right\}\right)$

Specifies the number of track groups (as defined by the GRPSZ parameter on the BUFFER or SPART statement) JES3 is to allocate to jobs that execute on this main. For guidelines on how to determine the appropriate value for the TRKGRPS parameter for your installation, see "Determining Track Group Allocation Sizes" in Chapter 4, "Defining and Managing Spool Data Sets."

prigrps
Specifies the number of track groups to be initially allocated to jobs that execute on this processor. The specified value may be 1 through 9.

secgrps
Specifies the number of track groups to be allocated to jobs that execute on this main subsequent to their primary allocation. JES3 allocates the specified amount of spool space after the job uses up its initial allocation, and again (for an unlimited number of times) when the job uses up each secondary allocation and requests more spool space. The specified value may be 1 through 9.

Parameter Default: (1,2)

USRPAGE = { **nn** / **2** }

Specifies the number of 4K pages for each open SYSOUT data set. The USRPAGE parameter indicates pages of user address space for SYSOUT buffers. These pages are used as required and released when they are no longer required.

Specify a 1- or 2-digit number from 1 to 20.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 2

Associated Statement/Parameter: Each MAINPROC statement must have a DEVICE statement that specifies DTYPE=SYSMAIN associated with it.

The NAME parameter on the MAINPROC statement is referred to by the MDEPTH and MLIMIT parameters on the CLASS statement; by the UNIT and MAIN parameters on the CONSOLE statement; by the JNAME, JUNIT, and XUNIT parameters on the DEVICE statement; by the EXRESC parameter on the GROUP statement; by the SYSTEM parameter on the MSGROUTE statement; and by the VOL parameter on the SETACC statement.

The MDEST parameter on this statement is associated with the DEST parameter on a CONSOLE statement.

The SELECT parameter on this statement must correspond to the NAME parameter (specified value or default value) of a SELECT statement.

The SPART parameter on the MAINPROC statement must correspond to the NAME parameter on an SPART statement.

The TRKGRPS parameter on this statement specifies a number of track groups; the size of a track group is defined by the GRPSZ parameter on the BUFFER or SPART statement.

Overrides: The TRKGRPS parameter specification on the SYSOUT and CLASS initialization statements and on the //*MAIN JES3 control statement override the TRKGRPS parameter specification on the MAINPROC initialization statement.

Example 1: This example shows how to define a main, named SY1, on multiprocessor, such as a 3081, using a MAINPROC statement and a DEVICE statement.

```
MAINPROC,NAME=SY1,SYSTEM=JES3,CPUID=(023141,223141)
DEVICE,DTYPE=SYSMAIN,JNAME=SY1,JUNIT=(NONE,SY1)
```

Example 2: This example shows how to define a single main, named SY1, on a 3084 processor complex. The example uses a MAINPROC statement and a DEVICE statement.

```
MAINPROC,NAME=SY1,SYSTEM=JES3,                                  X
CPUID=(023100,223100,123100,323100)
DEVICE,DTYPE=SYSMAIN,JNAME=SY1,JUNIT=(NONE,SY1)
```

Example 3: This example shows how to define a 3084 processor complex to run physically partitioned as two multiprocessors, named SY1 and SY2. The example uses two MAINPROC statements (one for each multiprocessor) and two DEVICE statements.

```
MAINPROC,NAME=SY1,SYSTEM=JES3,CPUID=(023100,223100)
MAINPROC,NAME=SY2,SYSTEM=JES3,CPUID=(123100,323100)
DEVICE,DTYPE=SYSMAIN,JNAME=SY1,JUNIT=(NONE,SY1,,,5C0,SY2)
DEVICE,DTYPE=SYSMAIN,JNAME=SY2,JUNIT=(4C0,SY1,,,NONE,SY2)
```

Example 4: This example shows how to define two mains on a 3084 processor complex. If run in single-image, only one of the two mains can run at a time. If partitioned, both SYA and SYB can run simultaneously. The example uses the following MAINPROC and DEVICE statements.

```
MAINPROC,NAME=SYA,SYSTEM=JES3,CPUID=(033333,133333,233333,333333),...
MAINPROC,NAME=SYB,SYSTEM=JES3,CPUID=(033333,133333,233333,333333),...
DEVICE,DTYPE=SYSMAIN,JNAME=SYA,JUNIT=(.....,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYB,JUNIT=(.....,ON)
```

Example 5: This example shows how to define three mains on a 3084 processor complex to run as one main in single-image mode or two mains in partitioned mode. The single-image main is SYQ; the partitioned mains are SYA and SYB. The example uses the following MAINPROC and DEVICE statements:

```
MAINPROC,NAME=SYQ,SYSTEM=JES3,CPUID=(044444,144444,244444,344444),...
MAINPROC,NAME=SYA,SYSTEM=JES3,CPUID=(044444,244444),...
MAINPROC,NAME=SYB,SYSTEM=JES3,CPUID=(144444,344444),...
DEVICE,DTYPE=SYSMAIN,JNAME=SYQ,JUNIT=(.....,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYA,JUNIT=(.....,ON)
DEVICE,DTYPE=SYSMAIN,JNAME=SYB,JUNIT=(.....,ON)
```

Example 6: This example shows how to define a 3084 processor complex to run as one main with an MSS connection. An alias CPUID, 000001, is included. MSS Table Create output is modified to match this dummy CPUID.

The example uses the following MAINPROC and DEVICE statements.

```
MAINPROC,NAME=SYA,SYSTEM=JES3,                                  X
CPUID=(055555,155555,255555,355555,000001),...
DEVICE,DTYPE=SYSMAIN,JNAME=SYA,JUNIT=(.....,ON)
```

MSGROUTE (MVS Message Route Table)

You can use the MSGROUTE statement to control the routing of subsystem modifiable messages (such as most MVS-issued messages). If you do not include a MSGROUTE statement, any message that originates from that processor is sent only to the system log and to the originating processor's MCS consoles.

Refer to "Defining Message Routing" on page 5-21 for an overview of message routing. "Using MSGROUTE to Control Message Routing" on page 5-27 explains how to use the MSGROUTE statement and describes a sample statement. The following chart shows the parameters you should include or omit to route message to the desired destination:

To route messages:	Use these parameters:
Directly to JES3 or TYPE=MCS consoles	Specify ***console*** and any other parameter(s)
To only MCS consoles attached to the issuing processor	Omit ***destclass, console, J,*** or omit the routing code
To only JES3 and MCS consoles attached to the the global	***J*** and any other parameters
To JES3 consoles, as well as local and global MCS consoles	Omit ***J***
To only the log (MLOG and/or DLOG)	Specify ***J*** only

```
MSGROUTE,MAIN=main[,routecode=([destclass],[console],J ),]...
```

MAIN =main
Specifies a 1-to-8 character name of the main to which these definitions apply. The name must match the name you specify on the NAME= keyword on a MAINPROC statement.

routecode =([destclass],[console],J)
Specifies the MVS routing code and a console name and/or destination class to which you want messages sent.

routecode
Specifies an MVS routing code which between 1 and 128 inclusive.

destclass
Specifies a JES3 console destination class to which you want messages with the designated MVS routing code mapped. You assign destination classes to JES3 consoles using the DEST= keyword of the CONSOLE initialization statement.

console
Specifies the name of a JES3 or MCS-managed console with a logical association to which you want messages with the designated MVS routing code sent. You can specify the name of a JES3-managed console or a TYPE=MCS console.

J
Specifies that messages with the specified routing code are to be directed to only JES3 and MCS consoles attached to the global.

Associated Statement/Parameter: The destclass parameter must be the same as the DEST= keyword on a CONSOLE statement. The console parameter must be the same as the JNAME= keyword on a CONSOLE statement.

Override: You can use the JES3 *INQUIRY,M and *MODIFY,M commands to inquire about, and dynamically change the mappings of MVS routing codes and JES3 destination classes.

You can achieve different routing results by combining or omitting parameters on the MSGROUTE statement. Use Figure 12-7 to select the type of routing you want to establish for messages that originate from a local processor and the parameter(s) required to implement your choice:

Parameter(s) Specified	Available for Display at MCS Consoles Attached to the Local	Available for Display at JES3 Consoles	Available for Display at MCS Consoles Attached to the Global
None or routing code omitted	yes, using the message's original set of routing codes.	yes	no
Destination class only	yes, using the message's original set of routing codes and the equivalent routing code of the specified destination class.	yes, using the specified destination class.	yes, using the equivalent routing code of the specified destination class.
Console name only	yes, using the message's original set of routing codes.	yes, if the named console is a JES3-managed console.	yes, if the named console is associated to an MCS console on the global and is inactive to JES3.
J only	no	yes, if you've assigned destination class MLG to the console(s).	no
Destination class and console	yes, using the message's original set of routing codes and the equivalent routing code of the specified destination class and the console name if the named console is associated to an MCS console on the global and is inactive to JES3.	yes, using the specified destination class and the console name if the named console is a JES3-managed console.	yes, using the equivalent routing code of the specified destination class and console name if the named console is associated to an MCS console on the global and is inactive to JES3.
Destination class and J	no	yes, using the specified destination class.	yes, using the equivalent routing code of the specified destination class.
Console and J	no	yes, if the named console is a JES3-managed console.	yes, if the named console is associated to an MCS console on the global and is inactive to JES3.
Destination class, console, and J	no	yes, using the specified destination class and the console name if the named console is a JES3-managed console.	yes, using the equivalent routing code of the specified destination class and the console name if the named console is associated to an MCS console on the global and is inactive to JES3.

Figure 12-7. Parameter Combinations and Their Effects on Messages Issued from a Local Process

Use Figure 12-8 to select the type of routing you want to establish for messages that originate from a global processor and the parameter(s) required to implement your choice:

Parameter(s) Specified	Available for Display at MCS Consoles	Available for Display at JES3 Consoles
None or routing code omitted	yes, using the message's original set of routing codes.	no
Destination class only	yes, using the message's original set of routing codes and the equivalent routing code of the specified destination class.	yes, using the specified destination class.
Console name only	yes, using the message's original set of routing codes and the console if the named console is associated to an MCS console on the global and is inactive to JES3.	yes, if the named console is a JES3-managed console.
J only	no	yes, if you've assigned destination class MLG to the console(s).
Destination class and console	yes, using the message's original set of routing codes and the equivalent routing code of the specified destination class and the console name if the named console is associated to an MCS console on the global and is inactive to JES3.	yes, using the specified destination class and the name if the named console is a JES3-managed console.
Destination class and J	yes, using the equivalent routing code of the specified destination class.	yes, using the specified destination class.
Console and J	yes, if the named console is associated to an MCS console on the global and is inactive to JES3.	yes, if the named console is a JES3-managed console.
Destination class, console, and J	yes, if the named console is associated to an MCS console on the global and is inactive to JES3 and the equivalent routing code of the specified destination class.	yes, if the named console is a JES3-managed console and the specified destination class.

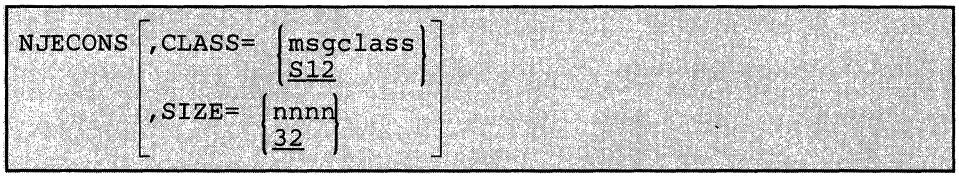
Figure 12-8. Parameter Combinations and Their Effects on Messages Issued from a Global Processor

Many factors influence the routing of message traffic. Refer to "Defining Message Routing" on page 5-21 for additional information.

If you have not defined any console to display a message, it is sent sent only to the system log provided you define at least one of the message's original routing code(s) on HARDCOPY keyword in the CONSOLxx member of SYS1.PARMLIB or the issuer originally specified that the message be sent to the log (and no other facility such as MPF overrides that specification). Otherwise, the message does not appear on any console or in the system log.

NJECONS (JES3 Networking Message Class Assignment)

Specifies the message class to which JES3 is to send messages about the JES3 job entry network. Also defines the number of entries in a table that JES3 networking uses to keep track of consoles that owe it a response.



CLASS = { **msgclass** / **S12** }

Specifies the message class to which JES3 is to send network messages.

Specify this message class in the DEST parameter on the CONSOLE statement for consoles that are to receive network messages.

SIZE = { **nnnn** / **32** }

Defines the number of entries in a table that JES3 networking uses to keep track of consoles that owe it a response.

nnnn

Specifies a value from 1-4095. The more commands your node receives or the faster it receives them, the larger this value should be. Initially, start with the default value and change it as experience dictates.

NJERMT (JES3 Network Node Definition)

Defines a node in the JES3 job entry network. You must code a NJERMT statement for the home node (your node) and one for each remote node that will communicate with the home node.

```
NJERMT,NAME=nodename,
       [,AUTO= { YES[,RDLY= {mm} ] }
               { NO         {5}  }
       ,BDTID=sysid
       ,BFSIZ= {nnnn}
               {400 }
       ,CTC= {YES}
             {NO }
       ,EXPWD=recpassword1
       ,EXSIG=recpassword2
       ,HOME= {YES}
              {NO }
       ,LINE=linename
       ,MAXLINE= {0}
                 {1}
                 {2}
                 {3}
       ,NJEPR= {nn}
               {3 }
       ,NJEPU= {nn}
               {3 }
       ,PATH=nodename
       ,PRTDEF=printdefault
       ,PUNDEF=punchdefault
       ,PWD=sendpassword1
       ,SIG=sendpassword2
       ,STREAM= {1}
                {2}
       ,TYPE={BSC}
             {SNA}
       ]
```

NAME = nodename

Specifies a 1-8 character node name. No two nodes should have the same node name.

If you are naming a remote node, see "Coding Considerations" following this statement description or "Defining a Remote Node" in Chapter 9, "JES3 Networking."

AUTO = { YES } [,RDLY = { mm }]
{ NO } **{ 5 }**

Specifies, for a directly-connected remote BSC node, whether JES3 is to automatically restart the line to the node if the remote node interrupts transmission. YES specifies that JES3 is to automatically restart the line.

If you specify NO or omit this parameter and the remote node interrupts transmission, the operator must restart the line.

Code this parameter on the NJERMT statement that defines a directly-connected remote BSC node.

RDLY = { **mm** / **5** }

Specifies, in minutes, the amount of time that JES3 networking is to wait before it automatically restarts an interrupted line. Valid values for mm are 0-99.

Code this parameter on an NJERMT statement that defines a directly-connected remote BSC node.

BDTID = sysid

Specifies the 1-8 character name of the MVS/BDT subsystem in your JES3 complex that is to process SNA/NJE transactions. This name must match the name specified on the MVS/BDT SYSID initialization statement.

Include this parameter only on the NJERMT statement that defines the home node. If this parameter is not included, JES3 sends SNA/NJE transactions to the default MVS/BDT subsystem that is named on the JES3 SYSID statement.

BFSIZ = { **nnnn** / **400** }

Specifies the buffer size to be used for communication with the directly-connected remote BSC node defined by this statement. The system programmer at the remote node must specify the same buffer size as you specify. Do not specify a buffer size of less than 400 bytes.

The maximum buffer size allowed is the size of the spool buffers (specified via the BUFSIZE parameter on the BUFFER statement) minus 44.

CTC = { **YES** / **NO** }

Specifies the type of connection between the home node and the directly-connected remote BSC node defined by this statement. For a node that is connected via a channel-to-channel (CTC) adapter, specify YES. For a node that is connected via a leased line or a dial-up line, omit this parameter or specify NO. This parameter is ignored when TYPE=SNA is specified.

EXPWD =recpassword1

Specifies a 1-8 character password that the home node expects to receive from the directly-connected remote BSC node defined by this statement. This password allows the remote node to start the line to the home node.

For the remote node to be able to send the password, the system programmer at the remote node must also specify the password. The system programmer does this by coding PWD=sendpassword1 on the NJERMT statement that defines your node.

If you omit this parameter, the remote node will be able to start the line without sending a password.

EXSIG =recpassword2

Specifies a 1-8 character password that the home node expects to receive from the directly-connected remote BSC node defined by this statement. This password identifies the remote node that started a dial-up line (with a dial-up line, several remote nodes can have the capability to start the same line). Once identified, the remote node can communicate with the home node.

The password, recpassword2, can be any character string. Code this password only for remote nodes that are connected to the home node by dial-up lines.

For the remote node to be able to send the password, the system programmer at the remote node must also specify the password. He does this by coding SIG=sendpassword2 on the NJERMT statement that he codes to define your node.

If you omit this parameter, any remote node that can start the line to the home node will be able to communicate with the home node.

HOME = { **YES** / **NO** }

Specifies whether this NJERMT statement defines the home node or a remote node. If it defines the home node, code YES. For all other nodes, omit this parameter or code NO.

LINE =linename

Specifies the 1-8 character name of the line that connects the home node to the directly-connected remote BSC node defined by this statement. This name must match the name specified via the JNAME= parameter on the DEVICE statement that defines the line.

When the operator starts the line, he can override the name you have specified with this parameter. If you omit the LINE= parameter, the operator must specify a line name when starting the line.

MAXLINE = { **0** / **1** / **2** / **3** }

Specifies the maximum number of lines that the operator can start to the directly-connected remote BSC node defined by this statement. For an indirectly-connected remote node or the home node, specify 0.

NJEPR = { **nn** / **3** }

Specifies the number (1-99) of logical network printers to be generated for the home node.

Code this parameter on the NJERMT statement that defines the home node.

NJEPU = { **nn** / **3** }

Specifies the number (1-99) of logical network punches to be generated for the home node.

Code this parameter on the NJERMT statement that defines the home node.

PATH =nodename

Specifies the name of the first node in the path to an indirectly-connected node. Code this parameter on the NJERMT statement that defines the indirectly-connected node.

If you omit this parameter, JES3 networking assumes the home node and the remote node are directly connected. This parameter is ignored when TYPE=SNA is specified.

PRTDEF =printdefault

Specifies the print class default for networking output received at the home node. If you omit this parameter, JES3 networking assumes print class A.

PUNDEF =punchdefault

Specifies the punch class default for networking output received at the home node. If you omit this parameter, JES3 networking assumes punch class B.

PWD =sendpassword1

Specifies a 1-8 character password that the home node must send to the directly-connected remote BSC node defined by this statement. This password allows the home node to start the line to the remote node.

If the remote node is expecting a password and you omit this parameter or code it incorrectly, the home node will be unable to start the line.

If the remote node is not expecting a password and you code this parameter, the remote node ignores the password and allows the home node to start the line.

SIG =sendpassword2

Specifies a 1-8 character password that the home node must send to the directly-connected remote BSC node defined by this statement. Code this password only if the home node and the remote node are connected by a dial-up line.

If the remote node is expecting a password and you omit this parameter or code it incorrectly, the home node will be unable to communicate with the remote node.

If the remote node is not expecting a password and you code this parameter, the remote node ignores the password and allows communication with the home node.

STREAM = {**1**|2}

Specifies the number of concurrent data streams that JES3 networking is to transmit on one line between the home node and the directly-connected remote BSC node defined by this statement.

TYPE = {**BSC**|SNA}

Specifies the networking protocol to be used for communicating with a directly-connected remote node. Include this parameter only when defining a directly-connected remote node.

BSC
Specifies a BSC networking protocol.

SNA
Specifies a SNA networking protocol.

Parameter Default: BSC

Associated Statements: All DEVICE statements with the parameter DTYPE=SYSMAIN specified must precede the NJERMT statement.

The JNAME= parameter on the DEVICE statement must match the LINE= parameter on the NJERMT statement.

The name specified on the MVS/BDT SYSID statement must match the name specified on the BDTID= parameter on the NJERMT statement.

Example 1: The following example defines a home node and a remote node in a BSC job entry network. The DEVICE statement defines the CTC connection between the remote node and the home node.

```
NJERMT,NAME=NODE1,HOME=YES,MAXLINE=0
NJERMT,NAME=NODE2,LINE=LINE1,CTC=YES,TYPE=BSC
DEVICE,DTYPE=NJELINE,JNAME=LINE1,JUNIT=(207,SY1,TP)
```

Example 2: The following example defines a home node and a remote node in a SNA job entry network.

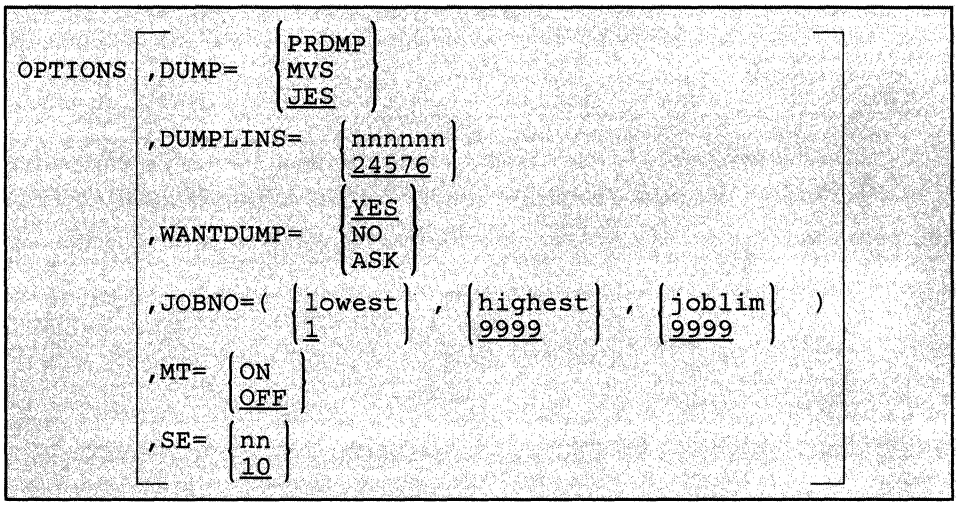
```
NJERMT,NAME=NODE3,HOME=YES,BDTID=SYSA1
NJERMT,NAME=NODE4,TYPE=SNA
```

OPTIONS (JES3 Options)

The OPTIONS statement can be used to specify:

- The type of MVS system dump to be taken if needed
- Whether or not a dump should be taken when a termination condition exists
- The job numbering limits for JES3 jobs
- Whether the writer output multitasking facility should be turned on (enabled) or turned off (disabled)
- The number of scheduler elements needed to support the largest job that will be run in the JES3 complex

Only one OPTIONS statement should be included in an initialization stream. If more than one is included, the parameters explicitly specified on the last OPTIONS statement dominate.



DUMP = { PRDMP / MVS / JES }

Indicates the type of MVS system dump to be taken in the event a JES3 abnormal termination or program check occurs.

PRDMP
Specifies that a dump of main storage is to be written to the data set SYS1.DUMPxx. To print this dump, use the MVS service aid program AMDPRDMP.

MVS
Specifies that the MVS system dump written to the SYSUDUMP or SYSABEND data set contains the MVS nucleus and SQA as well as the MVS JES3-related control blocks and JES3 region.

JES
Specifies that the MVS system dump written to the SYSUDUMP or SYSABEND data set contains only the MVS JES3-related control blocks and JES3 region.

The JES3 control blocks, if written, are formatted and always written to the JESABEND data set.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: JES

DUMPLINS = { **24576** / **nnnnnn** }

Specifies the maximum number of lines to be printed in a formatted dump. The maximum value allowed is 999999.

Parameter Default: 24576

WANTDUMP = { **YES** / **NO** / **ASK** }

Indicates the action to be taken when a JES3 failure condition occurs.

YES
Specifies that a dump should be taken when a failure occurs.

NO
Specifies that no dump should be taken when a failure occurs.

ASK
Specifies that, when a failure occurs, the operator is to be given the choice of specifying whether a dump is to be taken.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: YES

JOBNO =({ **lowest** / **1** } **,** { **highest** / **9999** } **,** { **joblim** / **9999** } **)**

Indicates a specific range of numbers for JES3 to assign to jobs that enter the JES3 complex. There must be at least five job numbers in this range. When the end of the range is reached, numbering resumes at the beginning of the range, skipping those numbers that are assigned to jobs still in the system. If all job numbers are assigned, the next JES3 DSP requiring a number is put into wait state and an operator message is issued to indicate that no job numbers are available.

Note: JES3 may assign job numbers to incoming jobs in excess of the number of jobs that input service can actually process. These jobs are pending input service, and are processed when buffers become available.

lowest

Specifies a number within the range 1 to 9994 to indicate the lowest job number. This number must be at least 5 job numbers preceding the value of 'highest' in the range.

highest

Specifies a number within the range 6 to 9999 to indicate the highest job number. This number must be at least 5 job numbers past the value of 'lowest' in the range.

joblim

Specifies the maximum number of jobs that may be in the JES3 complex at any given time (within the range 1 to 9999). JES3 uses as the actual job maximum the **smaller** of the following values:

- the *joblim* value
- the value obtained by the calculation: highest - lowest + 1
- the number of job control table (JCT) entries that will fit in the JCT data set (see "Determining the Size of the JCT Data Set" in Chapter 6, "Defining and Managing JES3 Resources" to determine this number)

If, during a warm start, you lower the upper end of the job number range, jobs that are in the system and were previously assigned job numbers above the new upper end of the range will be lost. For example, assume you have specified 1-50 as the range of job numbers and JES3 has assigned numbers 1-35. You now warm start JES3 and change the range of job numbers to 1-30. Jobs with numbers 31-35 will be lost.

There is a way to avoid losing jobs when you change the upper end of the range via a warm start. Before doing the warm start, execute the dump job (DJ) facility to save jobs that are in the job queue. After the warm start, again execute the dump job facility to restore these jobs to the job queue. How to invoke the DJ program is described in *MVS/Extended Architecture Operations: JES3 Commands.*

Parameter Default: (1,9999,9999)

MT = { **ON** / **OFF** }

Indicates whether JES3 is to turn the writer output multitasking facility on or off. To determine whether to turn the facility on or off, see the topic "Using the Writer Output Multitasking Facility" in Chapter 7, "Defining and Managing JES3 Mains and Storage."

ON

Requests that JES3 turn on (enable) the writer output multitasking facility. Specify ON only when the global processor is a multiprocessor.

OFF
Requests that JES3 turn off (disable) the writer output multitasking facility. Specify OFF when the global processor is a uniprocessor.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: OFF

SE = { **nn** / **10** }
Specifies the maximum number of scheduler elements (SEs) that may be constructed for any job. The value chosen should equal the number of scheduler elements required to support the largest job (in terms of SEs) to be run in the complex. A number between 10 and 90 may be specified. This specification is used to calculate the maximum size for a JCT and its SEs (JCT record size).

Parameter Default: 10

Statement Default:

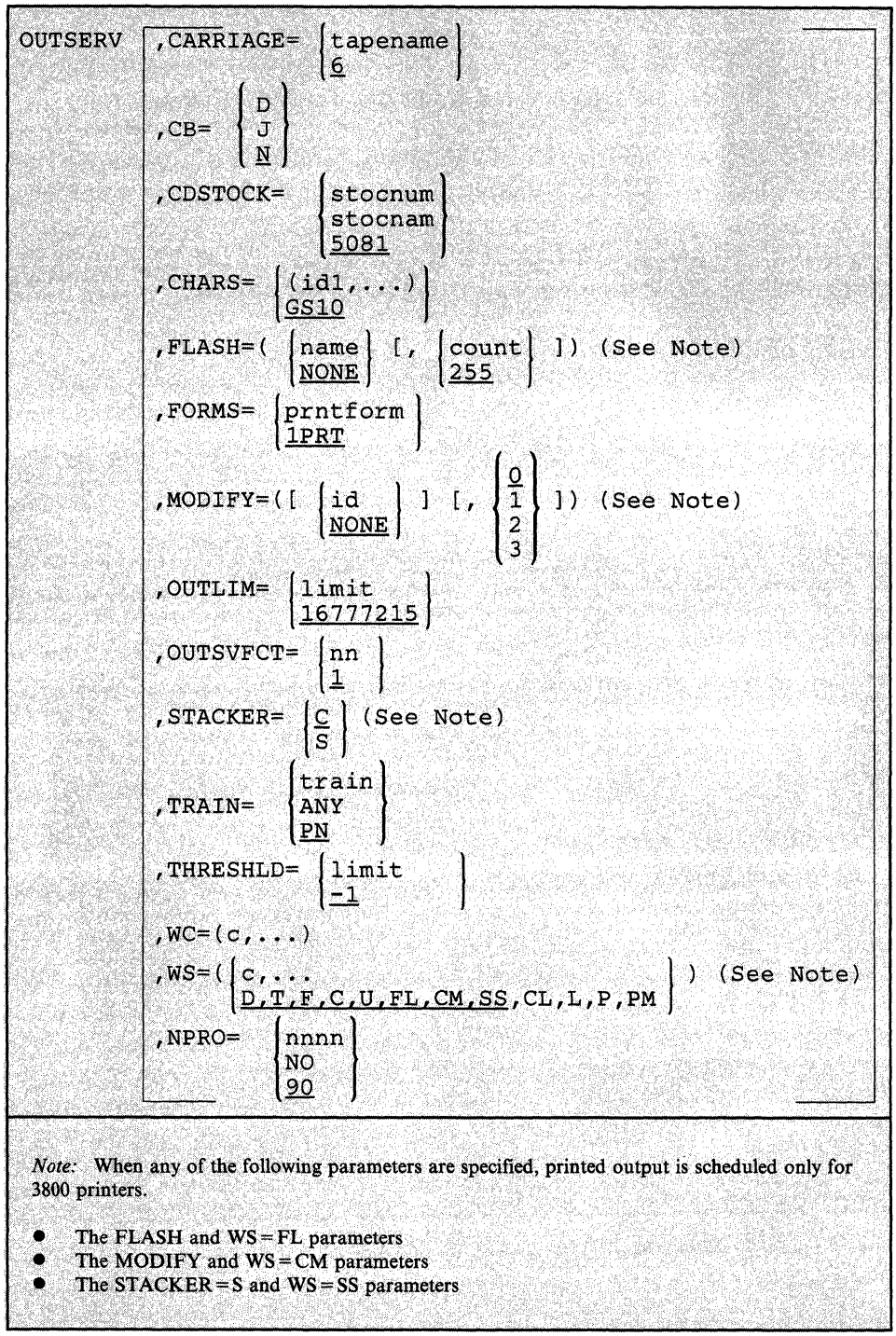
```
OPTIONS,DUMP=JES,DUMPLINS=24576,WANTDUMP=YES,      X
JOBNO=(1,9999,9999),MT=OFF,SE=10
```

Example: In the following example, the numbers 1000 to 4999 would be assigned to JES3 jobs, no more than 2000 jobs could be in the JES3 complex at any given time, the multitasking facility would be off, and the number of scheduler elements would default to 10. In the event of an abnormal termination or program check, a dump of MVS JES3-related control blocks and the JES3 region will be written to the SYSUDUMP or SYSABEND data set. The formatted JES3 control blocks will be written to the JESABEND data set.

```
OPTIONS,JOBNO=(1000,4999,2000)
```

OUTSERV (Output Service Defaults and Standards)

The OUTSERV statement specifies default values and standards for printers and punches.



Note: When any of the following parameters are specified, printed output is scheduled only for 3800 printers.

- The FLASH and WS=FL parameters
- The MODIFY and WS=CM parameters
- The STACKER=S and WS=SS parameters

CARRIAGE = { **tapename** / **6** }

Indicates the specific name (1 to 8 characters) or the number of characters in the name of the printer carriage tape that is to be the installation standard.

For 3211 printers the FCB module, which must be included in SYS1.IMAGELIB, is named FCB2, plus the first 4 characters of the carriage tape name. For 3800 printers the FCB module is named FCB3, plus the first 4 characters of the carriage tape name.

For more information about SYS1.IMAGELIB, see *MVS/Extended Architecture System Programming Library: Data Management.*

Parameter Default: 6

CB = { **D** / **J** / **N** }

Specified for the 3800 printer only. This parameter specifies when the device's buffer is to be cleared. The buffer is cleared by the clear printer channel command.

D

Indicates that the data set option is to be used. This causes output service to issue a clear-print channel command at the end of each data set. This causes a short pause at the end of each data set.

J

Indicates that the job option is to be used. This causes output service to issue a clear-printer channel command only at the end of each job. This causes a short pause at the end of each job.

N

Indicates that the device's buffer is not to be cleared unless required by a function. Maximum performance is obtained with CB=N. Examples of functions that require a clear printer command are:

- Loading new characters
- Requesting operator setup
- Waiting for work

When the device is in manual mode (M specified on *X, *R, or *S and M= is specified in message IAT8562), a clear printer command is issued before each data set and CB= has no effect.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: N

CDSTOCK = { **stocnum** | **stocnam** | **5081** }

Indicates the stock name (1 to 8 characters) or stock number of the cards to be considered the installation standard. This parameter is analogous to the FORMS parameter for printer output.

Parameter Default: 5081

CHARS = { **(idl,...)** | **GS10** }

Specifies the 1- to 4-character names of the image to be set up on the 3800 printer. Up to four names can be specified.

Parameter Default: GS10

FLASH =({ **name** | **NONE** } [, { **count** | **255** }])

Specifies the name (1 to 4 characters) of the forms flash cartridge to be used on the 3800 printer.

name

Specifies the name of the forms flash cartridge that is to be the standard; the subparameter default is NONE.

count

Specifies the standard flash count. If NONE is specified, then this subparameter should not be specified. If FLASH=name is specified and the count subparameter is not specified, then 255 is the default.

FORMS = { **prntform** | **1PRT** }

Specifies the name of the printer forms (1 to 8 alphanumeric or national characters) to be considered the installation standard. This parameter is analogous to the CDSTOCK parameter for punched output.

Parameter Default: 1PRT

MODIFY =([{ **id** | **NONE** }][, { **0** | **1** | **2** | **3** }])

Specifies the module identifier and reference character to be used as the standard copy modification option for the 3800 printer.

id

Specifies the name of the copy modification module to be used (one to four characters).

NONE

Specifies that the copy modification option is not being used.

0,1,2,3
Specifies the table reference character to be used with the copy modification option.

OUTLIM = { **limit** / **16777215** }
Specifies the default record limit for a SYSOUT data set. When this limit is exceeded, the installation SMF exit is entered. Upon return from the SMF routine, the job is either canceled or a new limit is used. The value of limit must be from 1 to 16777215.

If you specify an integer greater than the maximum allowable, JES3 uses the parameter default. If you specify a negative integer or a non-numeric character, JES3 issues message IAT3245 and initialization terminates.

Parameter Default: 16777215

OUTSVFCT = { **nn** / **1** }
Specifies the number (1-10) of available OUTSERV DSPs to process jobs from the output service queues. Defining more than one DSP allows output service to process multiple jobs simultaneously.

Parameter Default: 1

STACKER = { **C** / **S** }
Specifies which stacker option is to be standard for the 3800 printer.

C
Specifies that output is to be placed in the continuous forms stacker. If no 3800 described as BTS feature, STACKER=C is forced by initialization.

S
Specifies that output is to be placed in the sheet stacker, where offset stacking is automatically performed.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: C

TRAIN = { **train** / **ANY** / **PN** }
Indicates the installation standard printer train or band.

train
Specifies the name (1 to 4 characters) of a specific printer train or band to be used as the installation standard.

ANY
Specifies that any printer train or band that is mounted is acceptable, provided the printer can otherwise fulfill processing requirements.

PN
Specifies the PN Printer train.

Parameter Default: PN

THRESHLD = { **limit** / **-1** }

Specifies the default maximum size for a SYSOUT data set. The maximum value that may be specified is 99999999. THRESHLD=-1, which is the default, indicates that no threshold processing is in effect. This parameter is used by output service to build and queue OSEs for writers. The THRESHLD parameter assumes that the data set size is the number of records in the data set multiplied by the number of copies. Data sets that equal or exceed the value specified are queued as separate OSEs for output service writers.

A job that has OSEs created in this manner is eligible to be concurrently processed by several output service writers that have the same processing characteristics. Normally, similar OSEs for a job are assigned to the same writer. This parameter should be used when the user wants to permit concurrent printing or punching of multiple copies of data sets or a large volume of data sets for a job on several output service writers. This parameter is ignored when the copy distribution for a data set is 3000 or more.

Example: Job ABC generates three SYSOUT data sets with similar processing characteristics (such as same forms, carriage). Assume that data set A is 3000 records and the user needs 2 copies; data set B is 10000 records and the user needs 1 copy; data set C is 1000 records and the user wants 10 copies.

If the THRESHLD for the job is 20,000, then the first writer to select the job is assigned all three data sets.

If the THRESHLD for the job is 10000, then the first writer to select the job is only assigned data set A. Data sets B and C are eligible for selection by another writer.

WC =(c,...)
Specifies the writer classes. This parameter indicates SYSOUT classes in the order they are processed by the output service writers. Data sets for a class not specified will not be selected. The value of c specifies a valid SYSOUT class (A-Z, 0-9) that appears as a SYSOUT parameter on a DD statement. If a list of classes is specified, separate each class by a comma.

This parameter is effective only for writers that have a writer selection list that includes CL, the SYSOUT class. The list could have been specified explicitly or by default.

Parameter Default: The list is null; JES3 selects data sets, regardless of class, to be processed by output service writers.

WS =({ c,... / D,T,F,C,U,FL,CM,SS,CL,L,P,PM })

Specifies the writer selection criteria. The value of c indicates the items JES3 output service checks, in order of importance, when selecting a data set for output processing. (These WS parameter values remain in effect over a hot start.)

Specify the selections in order of importance and separate specifications with a comma.

Selection Character	Meaning
C	Carriage tape or FCB (printers only)
CL	SYSOUT class
CM	Copy modification (3800 printer only)
D	Data set destination
F	Forms requested
FL	Flash (3800 printer only)
L	Limit scheduling (line, page or record)
SS	Stacker (3800 printer only)
P	Data set priority
PM	Processing mode
T	Specific device type requested
U	Train image UCS (3525 printer or 3800 character arrangement)

All the selection characters are valid for printer output. C is not valid for punched output.

CAUTION

JES3 does not take any action (such as issue mount messages or load FCB and UCS) if you omit printer and punch setup characteristics on this keyword parameter. For example, a job that requires special forms may print on the wrong form if you omit forms as a selection criterion.

NPRO = { nnnn / NO / 90 }

Specifies the non-process run-out interval for a 3800 model 3 printer.

nnnn

Specifies the number of seconds the printer will wait for more data before forcing out the already-printed pages. The value specified must be between 0 and 9999, inclusive.

NO

Specifies that the run-out interval will not be used for this printer.

Parameter Default: 90

Associated Statement/Parameter: The CARR, CHARS, FLASH, FORMS, MODIFY, STACKER, AND TRAIN parameters on the SYSOUT statement may override values specified on the OUTSERV statement.

Statement Default:

```
OUTSERV,CARRIAGE=6,CB=N,CDSTOCK=5081,CHARS=GS10,FLASH=(NONE,255),          X
FORMS=1PRT,MODIFY=(NONE,0),OUTLIM=16777215,STACKER=C,TRAIN=PN,            X
THRESHLD=99999999,WS=(D,T,F,C,U,FL,CM,SS),NPRO=90
```

Example: This example specifies default values and standards for printers and punches as follows:

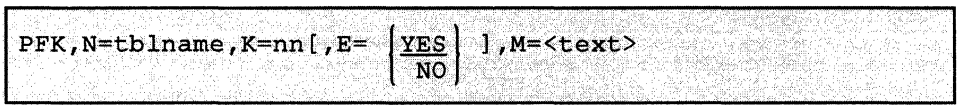
- the carriage tape name is 6
- the writer selection criteria are, in order of importance, data set priority (P), SYSOUT class (CL), processing mode (PM), limit scheduling (L), and data set destination (D)
- the run-out interval is 75 seconds

All other parameters use their default values.

```
OUTSERV,CARRIAGE=6,WS=(P,CL,PM,L,D),NPRO=75
```

PFK (Program Function Key)

Use the PFK statement to assign up to 12 operator commands to a program function key. For information about when and how to use this statement, see "Defining Program Function Keys" in Chapter 5, "Defining Consoles and Message Routing."



N = tblname
Specifies the name (1 to 8 characters) of a PFK table. PFK statements that you want to group together must specify the same value for the N parameter.

K = nn
Specifies the number (1 to 24) of the PFK that is to be associated with the command(s) specified by the M parameter.

E = { YES / NO }
Specifies whether JES3 is to execute the commands immediately when the PFK is depressed or is to display them.

YES
JES3 is to execute the commands immediately.

NO
JES3 is to display the commands and then position the cursor at the end of the command stream.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: YES

M = <text>
Specifies the text of one or more (up to 12) operator commands. To specify more than one operator command, the "new-line" character specified on the CONSTD statement is used to separate commands. The M parameter must be specified last. If a PFK statement must be continued, the M parameter must be totally specified on a single card.

Statement Default: Depressing a program function key that is not defined on a PFK statement deletes the current input line (CANCEL function).

Associated Statement/Parameter: The N parameter is used in the PFK and SP parameters of the CONSOLE statement to assign a PFK table to a program function key and/or selector pen of a specific console. You define program function keys for MCS consoles in the PFKTABxx member of SYS1.PARMLIB. See *MVS/Extended Architecture System Programming Library: Initialization and Tuning* for information about the SYS1.PARMLIB data set.

Examples: In the example which follows, an operator command is defined for program function key 2 in PFK table TBL2. When key number 2 is pressed, the disk reader processes the partitioned data set member, TESTJOB.

```
PFK,N=TBL2,K=2,M=<*X DR,M=TESTJOB>
```

In the following example, a series of operator commands are defined for program function key 4 in the PFK table TABL2. When key number 4 is pressed, each of the commands is executed. (In order to define the series of commands in the M parameter, the EDIT parameter of the CONSTD statement must specify semicolon (;) as the "new-line" character.)

```
PFK,N=TABL2,K=4,                                              X
M=<*F V 181,OFF; *F V 233,OFF;*F V SYS1,ON;*S JSS>
```

In the following example, the operator can complete the command before it is entered into the system. When program function key 5 is pressed, the command *S,RJP,L= is displayed with the cursor positioned after the equal sign.

```
PFK,N=PKFTAB2,K=5,E=NO,M=<*S RJP,L=>
```

Override: Use the command *F,K to define a different set of commands for a program function key. Use the command *F,O to assign a different PFK table to a console.

PROC (Frequently Used Procedures)

The PROC statement indicates frequently used cataloged procedures and the procedure library on which they are found. Using a PROC statement improves JES3 performance because the procedure library's directory does not have to be searched each time the procedure name appears on an EXEC statement.

```
PROC,BLDL=(procname[,procname]...)
   [,LIB= {ST} ]
          {nn}
```

BLDL =(procname[,procname]...)
Required to indicate the name of at least one frequently used procedure. A directory entry is maintained in storage for each procname that is found in either the library specified by the LIB parameter or, if LIB is not included, in the standard procedure library as defined by the ddname IATPLBST in the JES3 procedure.

LIB = {**ST**} {**nn**}
Optionally identifies the procedure library on which the procedures are to be found.

ST
Refers to the installation standard procedure library defined by the DYNALLOC initialization statement for the ddname IATPLBST, which is dynamically allocated or in the JES3 procedure. (As distributed, this library is SYS1.PROCLIB.)

nn
Refers to a 2-byte identifier which corresponds to the last 2 characters in the ddname IATPLBnn. This ddname identifies the IATPLBnn ddname which must be defined on a JES3 DYNALLOC statement or in the JES3 procedure.

Parameter Default: ST

Example: In the following example, directory entries for procedures ASMFCLG and COBLG are created in the BLDL list maintained in main storage for library IATPLB04. Any reference made by JCL to those two procedures results in a faster, more efficient search than if no PROC statement was specified.

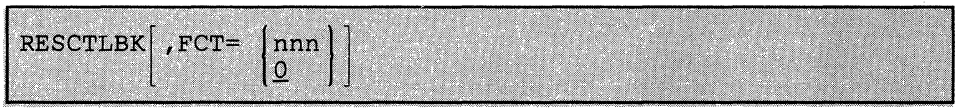
```
PROC,BLDL=(ASMFCLG,COBLG),LIB=04
```

Library definition on a JES3 DYNALLOC statement:

```
DYNALLOC,DDN=IATPLB04,VOLSER=DISK01,UNIT=3330,     X
DSN=MYPROCLIB
```

RESCTLBK (Resident Control Block)

The RESCTLBK statement is specified to preallocate storage for the highly used JES3 function control table (FCT) entries. This can reduce GETMAIN overhead and the associated paging overhead.



FCT = **{nnn / 0}**

Indicates the number of FCT entries to be preallocated. The value of nnn may be from 1 to 999. When determining the nnn value consider:

- One FCT entry per active JES3 DSP
- Each FCT entry is 224 bytes long

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0

Statement Default: RESCTLBK,FCT=0

Example: In the following example, enough FCT entries are preallocated for 30 DSPs. After 30 DSPs are active, JES3 allocates more FCT entries as they are required.

```
RESCTLBK,FCT=30
```

Use the RESDSN statement to name permanently resident data sets for which JES3 is to bypass setup processing. JES3 bypasses setup processing whenever the named data sets appear as cataloged references (no UNIT or VOLUME parameters are specified) on the DD statement of a job.

You cannot continue the RESDSN statement. If the dsname subparameters will not fit on one statement, specify the remaining subparameters on another statement.

Notes:

1. *JES3 provides no data set integrity checking for data sets that appear on a RESDSN statement.*

2. *JES3 does not check dynamically allocated data sets against the names specified on the RESDSN statement. To bypass integrity checks for a dynamically allocated data set, use the DYNALDSN statement.*

```
RESDSN,DSN=(dsname[,dsname]...)
```

DSN =(dsname[,dsname]...)
Specifies the names of permanently resident data sets for which JES3 is to bypass setup processing.

Example: The following example shows how to specify the data set names when they will not fit on one RESDSN statement.

```
RESDSN,DSN=(SYS1.PROCLIB,SYS1.LINKLIB,SYS1.MACLIB)
RESDSN,DSN=(SYS1.JES3LIB,SYS1.SVCLIB)
```

RJPLINE (BSC Remote Job Processing Line)

The RJPLINE statement defines the characteristics of a single BSC line (and its respective adapter) that will be used by the JES3 global processor for remote job processing. Additionally, this statement may assign a specific RJP work station, defined by the N parameter of an RJPTERM statement, to this line.

One RJPLINE statement is required for every adapter to be accessed by RJP. For further information about RJP, see the RJP description in Chapter 8, "JES3 Remote Job Processing."

```
RJPLINE,N=linename
       ,A=adaptadr
          [,F=DIAL
          ,F=NTRS
          ,G=grpname
          ,I={B}
             {A}
          ,I={Y}
             {N}
          ,O=AUTO
          ,P=password
          ,S={linespeed}
             {2400     }
          ,T=termname]
```

N =linename
Required to specify the name for this line up to 8 characters. It must be different from the N parameter of the RJPTERM statement.

A =adaptadr
Required to indicate the 3-character line adapter address associated with this line. For multiple processors, the parameter indicates the adapter addresses associated with each processor.

- For a single processor, specify:

  A=xxx (where xxx is the adapter address)

- For multiple processors, specify:

  A=(main1,xxx,main2,yyy,main3,zzz) (where main1, main2, and main3 are the main names and xxx, yyy, and zzz are the adapter addresses)

F =DIAL
F =NTRS
Used to indicate a dial feature for this line (DIAL) and/or the absence of a transparency feature for the line (NTRS). If both DIAL and NTRS are selected, both F parameters are specified, as shown previously.

Parameter Default: Dedicated or leased line environment. Transparency feature assumed on terminals on this line.

DIAL
Indicates that the line is a switched line. Do not specify F=DIAL if the line is to be used for the transmission of output restricted to authorized remote terminals. If not specified, a dedicated (or leased) line is assumed.

NTRS
Indicates that the line does not have the transparency feature; if not specified, the transparency feature is assumed.

G =grpname
Indicates a line group name of up to 8 characters, which associates several lines.

Parameter Default: 8 blanks (no group name).

I = { **A** / **B** }
Indicates the 1-character line interface.

A
Refers to the first or only interface of the BSC communications adapter.

B
Refers to the second interface of a BSC communications adapter having the dual communications interface feature.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: A

I = { **Y** / **N** }
Specifies the mode of operation.

Y
Specifies the interrupt mode of operation.

N
Specifies the noninterrupt mode of operation.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: Y

O = AUTO
Indicates the line option to have the RJP line start automatically; the *START operator command is not required.

Parameter Default: *START operator command is required.

P = password
Provides password protection for this line. password defines a line password up to 8 characters. This password must be used by any work station attempting to sign on to this line.

Parameter Default: 8 blanks (no password protection).

S = { linespeed / 2400 }
Specifies a line baud rate up to 6 characters.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 2400

T = termname
Indicates a terminal to be assigned to this line. If this parameter is included, a /*SIGNON card must not be used. T specifies the name (up to 5 characters) of a nonprogrammable terminal. This name must also appear in the N parameter of an RJPTERM statement and JNAME parameter of a CONSOLE statement. The T parameter must not be specified when F=DIAL is specified.

Associated Statement/Parameter: The T parameter on this statement must match the N parameter on an RJPTERM statement and the JNAME parameter on a CONSOLE statement.

Examples: In the following example, the LINE01 is defined as having an adapter address 001, without the dual communications interface feature. The line is a 50K BPS leased line with transparency. It will be automatically started when RJP is called. No password is assigned to LINE01.

```
RJPLINE,N=LINE01,A=001,S=50000,O=AUTO,T=FUL50
```

In the following example, LINE002B is defined as having an adapter address 002, with the second interface of the dual communications interface feature. It is a switched 2400-BPS line with transparency. The line must be started by the operator after RJP is called. Note that if another line specifies the same dual interface feature on line adapter 002, it cannot be active (started) concurrently on this line. Work stations must use the password, SECRET, to sign onto the line.

```
RJPLINE,N=LINE002B,A=002,I=B,F=DIAL,S=2400,P=SECRET
```

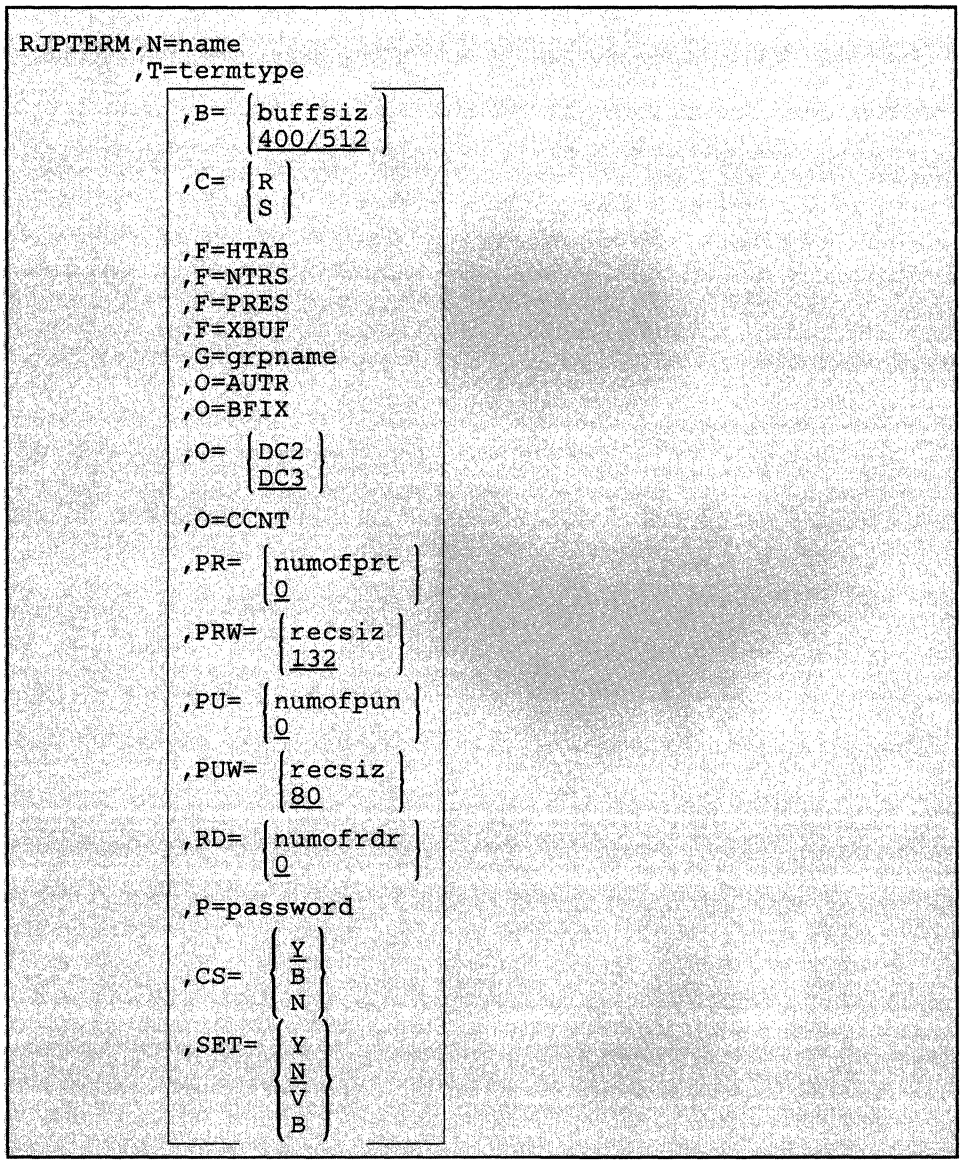
In the following example, LINE01 is defined as adapter address 001, interface A. It is a 4800-BPS line with transparency. The line will be automatically started when RJP is called, and the work station named T2780 will be automatically signed on to it. No password is assigned to LINE01.

```
RJPLINE,N=LINE01,S=4800,O=AUTO,T=T2780,A=001
```

RJPTERM (BSC Remote Job Processing Terminal)

The RJPTERM statement defines a single remote BSC work station to the JES3 system. This statement causes a default description to be provided for each work station device (printer, punch, or card reader) indicated by the PR, PU, or RD parameters along with the operating characteristics of the work station.

If the JES3 default characteristics for a remote printer or punch device are not acceptable, you should code a DEVICE statement to indicate desired characteristics. If a work station is to have the facilities of a JES3 operator console, then a CONSOLE statement must be coded. Figure 12-9 on page 12-138 shows the DEVICE statement defaults and the RJP DEVICE parameters. Figure 12-10 on page 12-139 shows the RJP parameters on the CONSOLE statement.



Parameter	Default (By Device Type)	
	Printer	Punch
FORMS	YES,STANDARD	YES,STANDARD
CARRIAGE	YES,STANDARD	N/A
TRAIN	NO,STANDARD	N/A
XLATE	YES	N/A
BURST	YES	YES
HEADER	YES	YES
LINELIM	0	0

Part A. DEVICE Statement Defaults

Parameter	Usage
DTYPE=(RMT1403) or (RMT3211)	Required (Note 1)
JNAME	Required (Note 2)
FORMS	Optional (Note 3)
CARRIAGE	Optional
TRAIN	Optional
XLATE	Optional
BURST	Optional
HEADER	Optional
LINELIM	Optional
JUNIT	Ignore (Note 4)
DGROUP	Ignore
XTYPE	Ignore
XUNIT	Ignore

Notes:

1. *The DTYPE must be specified, and the first characters must be RMT. If remote printer FCB loading is desired, specify 3211 as the last 4 characters; any characters other than 3211 in the last positions will be ignored.*
2. *The JNAME consists of an 8-character name, where characters 1 through 5 are the terminal name, characters 6 and 7 are PR or PU (for the device type), and the 8th character is a single digit (1 through 7). This digit must correspond to the device number specified during remote generation for the device being referenced.*
3. *"Optional" indicates that the parameter may or may not be coded.*
4. *"Ignore" indicates that the parameter has no meaning for remote devices, and will be ignored.*

Part B. DEVICE Statement Parameters

Figure 12-9. DEVICE Statement Defaults and Parameters Associated with the RJPTERM Statement

Parameter	Default	Usage
TYPE=RJP		Required (Note 1)
JNAME=terminal name		Required
DEST (Note 2)		Required
LEVEL	00	Optional (Note 3)
DEPTH	50	Optional
LL	120/PRW=	Optional
TIME		Rejected (Note 4)
PFK		Rejected
SP		Rejected
MAIN		Rejected
UNIT		Rejected

Notes:

1. *"Required" indicates that this parameter must be specified.*
2. *Destinations MLOG and DLOG should not be routed to a remote console.*
3. *"Optional" indicates that this parameter may be coded if the default is not acceptable.*
4. *"Rejected" indicates that this parameter must not be coded; the console definition will be rejected.*

Figure 12-10. CONSOLE Statement Parameters Associated with the RJPTERM Statement

N =name

Required to indicate the name for this work station. This name must be 5 characters and cannot be the same as an N parameter specified on an RJPLINE or RJPWS statement.

T =termtype

Indicates a 4-character identifier specifying the type of work station. The valid types for this parameter are:

ID	Work Station
1130	1130 processor terminal
2770	2770 nonprogrammable terminal
2780	2780 nonprogrammable terminal
2922	2922 programmable terminal
3741	3741 nonprogrammable terminal (See the topic "Restrictions and Limitations" at the end of this statement description.)
3747	3747 nonprogrammable terminal
3780	3780 nonprogrammable terminal
M202	System/360, Model 20, Submodel 2, processor terminal
M205	System/360, Model 20, Submodel 5, processor terminal
S360	All System/360 Model 22 and above processor terminals
S370	All System/370 programmable terminals
SYS3	System 3, Model 10, processor terminal

In specifying termtype, the following terminals are specified as 2770s: 3771, 3772, 3773, 3774, and 3775. The 3776 and 3777-1 are specified as 3780s. The 3777-2 is specified as M205.

The value of the T parameter determines the default value for the C and B parameters.

B = { **bufsiz** / **400/512** }

Indicates the terminal buffer size in bytes. The value specified can be up to 4 numeric characters. For programmable terminals, the bufsize must correspond to the work station package. For nonprogrammable terminals, the default buffer value is used.

Parameter Default: The default value for B is determined by the T parameter as shown below:

T Parameter	B Default
1130	400
2770	512*
2780	400
2922	400
3741	512
3747	512
3780	512
M202	400
M205	400
S360	400
S370	400
SYS3	400

*If F=XBUF is specified, then this default for B is 256.

For fixed buffer size devices, such as the 2780, the B parameter is forced to 400.

C = { **R** / **S** }

Indicates the type of console support for this work station. If the C parameter is specified, a CONSOLE statement should be specified also. If you omit a CONSOLE statement for this work station, a default JES3 message destination class of TP (teleprocessing) is assigned to the console. If a CONSOLE statement is found and the C parameter is not specified, then the default value for C is assumed.

R

Applies to programmable terminals only. This subparameter indicates the presence of a real console.

S

Applies to both programmable and nonprogrammable terminals. This subparameter specifies that regardless of any real console attached to this work station, the printer designated as PR1 is to be used as a simulated console output device.

If S is specified for a programmable terminal that has a real console, that console can still be used to enter console commands. However, all console output will be directed to the printer.

Parameter Default: The default value for C is determined by the T parameter as follows:

T Parameter	C Default
1130	R
2770	S
2780	S
2922	R
3741	S
3747	S
3780	S
M202	R
M205	R
S360	R
S370	R
SYS3	R

F =HTAB

Specifies that this terminal has the printer horizontal format control feature. This parameter is used to indicate a 2770, 2780, or 3780 with the horizontal format control feature.

Parameter Default: It is assumed that the terminal does not have this feature.

F =NTRS

Specifies that the terminal transparency feature is not present on this terminal.

Parameter Default: It is assumed that the terminal has the transparency feature.

F =PRES

Specified for a nonprogrammable terminal only, to indicate the blank compression/expansion feature for that terminal.

Parameter Default: It is assumed that the terminal does not have this feature.

F =XBUF

Specifies that a 2770 terminal has the extended buffer feature, that a 2780 terminal has the multirecord feature, or that a 3740 has the expanded communications feature. For a 2770, the B parameter must be specified large enough to incorporate this feature.

Parameter Default: It is assumed that the terminal does not have this feature.

G =grpname

Used for the groupname facility. This parameter associates this work station with a groupname. The groupname specified may be up to 8 characters.

Parameter Default: The terminal name specified in the N parameter.

O =AUTR
Indicates a work station option that provides the JES3 automatic call reader function in the local system. If the T parameter identifies a nonprogrammable terminal, this option is assumed; the O=AUTR parameter does not have to be specified.

O =BFIX
Indicates a work station option which specifies that the output buffers used by RJP must be dedicated, that is, obtained at OPEN time and freed at CLOSE.

O = { **DC2** / **DC3** }
Indicates the punch select character to be used for remote 2770 or 3780 terminals.

DC2 specifies device component position 2 on the terminal. When the 3777-1 is used as a 3780, the O parameter must indicate DC2.

DC3 specifies device component position 3 on the terminal.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: DC3

O =CCNT
Specifies, for 3740 terminals, the carriage control characters that should be sent to the remote printer. If the parameter is specified, the remote operator should load program buffer A prior to initiating printing.

PR = { **numofprt** / **0** }
Indicates the number of remote printers associated with this work station. This parameter specifies a single digit with a maximum value of 7. Nonprogrammable terminals can have a maximum of one printer.

Parameter Default: 0

PRW = { **recsiz** / **132** }
For programmable terminals, the value specified on the PRW parameter must not exceed the buffer size minus 16. The B parameter defines the buffer size.

For nonprogrammable terminals, the value specified on the PRW parameter must not exceed the maximum print record size defined by the design of the specific printer. The component manual for the printer defines the maximum print record size.

Parameter Default: 132

PU = { **numofpun** / **0** }

Indicates the number of remote punches associated with this work station. This parameter specifies a single digit with a maximum value of 7. Nonprogrammable terminals can have a maximum of one punch.

Parameter Default: 0

PUW = { **recsiz** / **80** }

Indicates the maximum punch card record size for remote punch devices. For programmable terminals, the value specified for recsiz must not exceed the buffer size minus 16. The B parameter defines the buffer size. If recsiz is larger than bufsiz, JES3 assumes the default, 80.

For nonprogrammable terminals, the value specified for recsiz must not exceed the maximum punch card record size defined by the design of the specific device. The component manual for the device defines the maximum record size.

Parameter Default: 80

RD = { **numofrdr** / **0** }

Indicates the maximum number of readers at this work station. This parameter specifies a single digit with a maximum value of 7. Nonprogrammable terminals can have a maximum of one reader.

Parameter Default: 0

P =password

Specifies the 1- to 8-character password of the terminal. This password must be used by any work station trying to sign on.

CS = { **Y** / **B** / **N** }

Specifies the type of compression, if any, that is desired. This parameter is for programmable terminals only.

Y

Indicates that equal-character compression is requested for this terminal.

B

Indicates that blank compression only is requested for this terminal.

N

Indicates that no compression is requested.

SET = { **Y** / **N** / **V** / **B** }

Specifies the type of setup that is desired on remote printers.

Y

Indicates that setup is required on remote printers each time a work station signs on.

N

Indicates that setup is not required on remote printers when a work station signs on.

V

Specifies that the work station printer automatically will go through setup procedures when varied online.

B

Specifies that the work station printer automatically will go through setup procedures when varied online, or the first time accessed after a LOGON.

Restrictions and Limitations

3741

- The 3741 accepts record formats of fixed (F), fixed blocked (FB), variable (V), or variable blocked (VB). JES3, however, transmits only F or FB record formats to the 3741.
- Each logical record that JES3 transmits to a 3741 is 128 bytes long. If the 3741 has the expanded communication feature, each physical record that JES3 transmits to the 3741 contains four 128-byte logical records. Otherwise, each physical record contains one 128-byte logical record.
- Because JES3 always transmits a 128-byte logical record to the 3741, do not use data compression. If you use data compression and JES3 compresses a record to fewer than 128 bytes, JES3 will then pad that same record to 128 bytes. Therefore, data compression serves no useful purpose on a 3741.

Associated Statement/Parameter: The CONSOLE statement defines console support at a remote work station. The DEVICE statement defines remote printers and punches. (A DEVICE statement must not be specified for remote devices other than printers and punches.)

Examples: These examples are provided as an aid in defining an RJP work station environment. Great care should be exercised in the definition of each work station, especially for console and device definitions.

In the following example, the work station named T3780 is defined as a 3780 nonprogrammable terminal with a 144-character printer and a card reader. The buffer size is set to 516 bytes, and the space compress/expand feature is invoked. Although the console mode is specified (C=S), no console support is provided for

this work station because no CONSOLE initialization statement was supplied. The group name defaults to the terminal name T3780.

```
RJPTERM,N=T3780,T=3780,F=PRES,F=NTRS,B=516,C=S,      X
PR=1,RD=1,PRW=144
```

In the following example, the work station named T2780 is defined as a 2780 nonprogrammable terminal with the multirecord and horizontal tab features installed. This work station is nontransparent, and therefore cannot send or receive object decks.

Work station devices include one printer, one punch, and one card reader. The printer width is 144 characters. Because T2780 is a nonprogrammable device, the console support mode is assumed to be simulated even though real (C=R) was specified. This means that the card reader and printer will be treated as the console input and output devices, respectively. The console authority level is 10; the default console width (equals printer width) is taken. The console queueing depth is set at 50 messages, the alternate console is CN1, and this console has the default route code of NONE. The group name defaults to the terminal name.

```
RJPTERM,N=T2780,T=2780,F=NTRS,F=HTAB,F=XBUF,         X
C=R,PR=1,PRW=144,PU=1,RD=1
CONSOLE,JNAME=T2780,TYPE=RJP,LEVEL=10,DEPTH=50,      X
ALTCON=CN1
```

In the following example, the work station named TERM2 is defined as a SYSTEM/360 having two printers, one punch, one card reader, and a console device. This work station buffer size is 600 bytes. The group name defaults to the terminal name. The console authority level is 10, the alternate console is CN1, and the console line width and queueing depth are defaulted to 120 and 50, respectively.

All devices on the work station TERM2, except PR1, are assigned JES3 default characteristics. PR1 is defined as a 3211.

```
RJPTERM,N=TERM2,T=S360,B=600,C=R,PR=2,PU=1,RD=1
CONSOLE,JNAME=TERM2,TYPE=RJP,LEVEL=10,ALTCON=CN1,    X
DEST=NONE
DEVICE,DTYPE=RMT3211,JNAME=TERM2PR1
```

The work station named FUL50 is defined as a SYSTEM/360. The work station buffer size is 800 bytes. The groupname assigned to this work station is DESIGN. It has five printers, two punches, and three card readers, as well as a real console device. The console line width is 124 characters. The console has no destination assignments, and is given an authority level of 15. If the console becomes unavailable, its messages will be routed to the console called CN1.

The first and second printers defined on FUL50 are 3211s. All devices attached to this work station, with the exception of PR2 and PU2, have default JES3 characteristics. PR2 may not have its forms, train, or FCB load changed. Its assumed setup is: forms (2PART7A), train (TN), and FCB load (SPDG8). JES3

header pages and burst pages will not be printed on PR2. PU2 on FUL50 may not have its forms changed, and is assumed to be setup with the forms called MICARD.

```
RJPTERM,N=FUL50,T=S360,B=800,G=DESIGN,C=R,PR=5,      X
PU=2,RD=3
CONSOLE,JNAME=FUL50,TYPE=RJP,DEST=NONE,LEVEL=15,    X
LL=124,ALTCON=CN1
DEVICE,DTYPE=RMT3211,JNAME=FUL50PR1
DEVICE,DTYPE=RMT3211,JNAME=FUL50PR2,BURST=NO,       X
FORMS=(NO,2PART7A),TRAIN=(NO,TN11),HEADER=NO,       X
CARRIAGE=(NO,SPDG8)
DEVICE,DTYPE=RMT2540P,JNAME=FUL50PU2,               X
FORMS=(NO,MICARD)
```

RJPWS (SNA Work Station Characteristics)

The RJPWS statement describes a single SNA work station's characteristics to the JES3 system. This statement causes a default description to be provided for each work station device (printer, punch, or card reader) indicated by the PR, PU, or RD parameter along with the operating characteristics of the work station.

If the JES3 default characteristics for a remote printer or punch device are not acceptable, you should code a DEVICE statement to indicate desired characteristics. Figure 12-9 on page 12-138 shows the DEVICE statement defaults and the RJP DEVICE parameters.

```
RJPWS,N=name
      [,RD=  {nn }]
             {1 }
      [,PR=  {nn }]
             {1 }
      [,PU=  {nn }]
             {0 }
      [,C=   {S }]
             {R }
      [,G=grpname]
      [,COMPACT= {NO      }]
                 {comptbl }
      [,P=password]
      [,PL=  {n }]
             {2 }
      [,AUTO= {(Y,luname[,modetabentry]) }]
              {N                         }
      [,LU=  {luname                     }]
             {(luname[,luname]...)       }
      [,TRACE=ON]
      [,SETUP=  {YES  }]
                {NO   }
                {VARY }
                {BOTH }
```

N =name

Specifies the 5-character name of the work station. This parameter cannot specify the same name specified in the N parameter on an RJPLINE or RJPTERM statement.

RD = { **nn** / **1** }

Specifies the maximum number of work station reader units. This number must be an integer between 0 and 15.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 1

PR = { **nn** / **1** }

Specifies the maximum number of work station printer units. This number must be an integer between 0 and 15.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 1

PU = { **nn** / **0** }

Specifies the maximum number of work station punch units. This number must be an integer between 0 and 15.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0

C = { **S** / **R** }

Specifies the type of console support for the work station.

S
indicates that console and print media share the same presentation surface. JES3 never interrupts data streams to send console messages.

R
indicates that the console and printer are separate real devices. JES3 interrupts data streams to send console messages.

Parameter Default: R

G =grpname

Used for the group name facility. This parameter associates the work station with a group name. The grpname specified can be up to 8 characters long. This name is also the default destination for any output data set from a job submitted from a work station in the group. BSC RJP and SNA RJP work stations can belong to the same group.

Parameter Default: The name specified in the N parameter on this statement.

COMPACT = { **NO** / **comptbl** }

Specifies the name of the default compaction table to be used for the work station if the name specified on the COMPACT initialization statement is not desired. If compaction is supported by the work station session and this parameter is specified, JES3 compacts all output data sets sent to the work station using this table unless another table is specified on the SYSOUT or //*FORMAT statement. COMPACT=NO indicates that there is no default compaction table.

P =password

Specifies the 1- to 8-character password to be used by the work station. This password must be included in the user data specified on the LOGON command passed to the JES3 LOGON exit by VTAM, or JES3 will not accept the LOGON.

Note: The password must not contain blanks, tabs, commas, or parentheses.

PL = { **n** / **2** }

Specifies the number of invalid LOGONs that can be attempted before the work station is rejected. Once the work station is rejected, the local operator must issue the *START or *RESTART commands to activate the work station. The number specified must be an integer from 0 to 9. If 0 is specified, there is no limit on invalid LOGONs.

If you specify an integer greater than the maximum allowable, JES3 uses the parameter default. If you specify a negative integer or a non-numeric character, JES3 issues message IAT3245 and initialization terminates.

Parameter Default: 2

AUTO = { **(Y,luname[,modetabentry])** / **N** }

Specifies whether automatic LOGON is supported for the work station.

(Y,luname [,modetabentry])

Indicates that automatic LOGON is supported for this work station. The luname is the name of the logical unit (LU). JES3 issues the SIMLOGON for the LU specified in luname when the JES3-VTAM interface is activated or when the local operator issues an *START or *RESTART command and an active session does not exist between JES3 and the work station. The modetabentry is the name of the mode table entry to be used for this work station. The default mode table is the first entry in the LOGMODE table for that LU.

Parameter Default: N

LU = { **luname** / **(luname,...)** }

Specifies the LU names associated with this work station as defined to VTAM. The names specified are those LUs which are permitted to LOGON as this work station. Up to 64 names can be specified.

Parameter Default: LOGONs are accepted from any LU.

TRACE =ON

Indicates that the SNA RJP trace facility is to be initiated during the work station LOGON process.

SETUP = { **YES** | **NO** | **VARY** | **BOTH** }

Specifies whether or not the work station printer will automatically go through setup procedures the first time it is accessed after work station LOGON.

YES
Specifies that the work station printer automatically will go through setup procedures. Do not specify SETUP=YES for unattended work stations because the system will wait for an operator response to the setup message.

NO
Specifies normal operation. The first time the printer is accessed after work station LOGON, JES3 assumes the printer is set up according to installation standards. The printer setup message will be sent to the work station operator only if a job requires special forms or features.

VARY
Specifies that the work station printer automatically will go through setup procedures when varied online.

BOTH
Specifies that the work station printer automatically will go through setup procedures when varied online, or the first time accessed after a LOGON.

Associated Statement/Parameter: The CONSOLE statement defines console support at a remote work station. The DEVICE statement may define remote printers and punches.

Example: The following example describes a SNA work station named WS1A1. The console and printer are separate real devices, the maximum number of work station punch units is 1, the work station is associated with a group named GROUP4A, and the default compaction table for the work station is named TN.

```
RJPWS,N=WS1A1,C=R,PU=1,G=GROUP4A,COMPACT=TN
```

SELECT (Job Selection Mode)

The SELECT statement defines scheduling controls to be associated with a particular job selection mode. The initial job selection mode is assigned to a JES3 main by the SELECT parameter on the MAINPROC statement. If a MAINPROC statement does not indicate a selection mode, the SELECT statement default values are assigned to that main. Each select mode defined can be dynamically changed using the *MODIFY,G,main,S operator command. In addition, the commands *MODIFY,G,main,G or *MODIFY,G,main,C can indirectly affect the select mode. A SELECT statement must be specified for each select mode indicated on a MAINPROC statement or in a *MODIFY,G,main,S command.

```
SELECT,NAME=modename
      ,CHOICE=  {BMIX }
                {FFIT }
                {FJOB }
                {FMIX }
      ,CLASS= ([/]jobclass[,jobclass]...)
      ,GROUP= {(groupnam,initnum[,groupnam,initnum]...)}
              {(/groupnam[,groupnam]...)               }
      ,INCL=  {nn }
              {14 }
      ,INCR=  {nn }
              {1  }
      ,JOBMIX=(n1[,n2...,n45])
      ,LSTOR= {nnnnn }
              {12000 }
      ,MAGEL= {nn }
              {14 }
      ,MAGER= {nnn}
              {0  }
      ,SAGEL= {nn }
              {14 }
      ,SAGER= {nnn}
              {0  }
      ,SBAR=  {PRTY}
              {nn  }
              {16  }
      ,SDEPTH= {nnn }
               {255 }
```

NAME =modename

Indicates the alphameric name (1 to 8 characters) of the job selection mode. NAME= must be the first parameter on a SELECT statement. This modename can be referenced with the SELECT keyword of the MAINPROC statement or the MODE subparameter on the JES3 command *F,G,main,S.

CHOICE = { **BMIX** | **FFIT** | **FJOB** | **FMIX** }

Specifies the job selection criteria to control the order of job selection on the main. The criteria are based on the size of the job and its I/O rate. JES3 uses the specified scheduling choice to select the most suitable jobs for execution. The choices available are BMIX, FFIT, FJOB, and FMIX. The following descriptions explain the job selection criteria associated with the different choices:

BMIX
Indicates that the first job in the queue that fits on the main and has a best mix IORATE is scheduled. JES3 determines IORATE from the job's //*MAIN statement or from the CLASS statement (IORATE parameter) that defines the class to which the job belongs. If none of the jobs meet these criteria, the first job that fits and has the alternate mix IORATE is scheduled. If none of the jobs in the queue have the alternate IORATE, the first job to fit will be scheduled.

FFIT
Specifies that the first job in the queue that fits on the main be scheduled. If none of the jobs fit, no job will be scheduled.

FJOB
Specifies that the first job in the queue be scheduled if it fits on the main. Otherwise, no job will be scheduled.

FMIX
Indicates that the first job in the queue that fits on the main and also meets the best or alternate IORATE requirement be scheduled. If none of the jobs in the queue meet these criteria, the first job that fits is scheduled.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: FFIT

CLASS =([/]jobclass[,jobclass]...)
Identifies either job classes that can be scheduled under this mode or, if a "/" precedes all class names, indicates those job classes that cannot be scheduled under this mode. Use the CLASS parameter to identify classes to be treated in a specific way or to include or exclude classes from a group for purposes of scheduling under this mode.

The CLASS parameter may be continued on another line if all class names cannot fit on one line. The continuation line must follow same format as the preceding line; that is, if the preceding line specified classes to be excluded (by a "/"), then the continuation line also must indicate exclusions. You cannot exclude certain classes in one specification of CLASS and include other classes in another specification.

The maximum number of job classes that can be specified is 255.

Parameter Default: All classes are eligible for scheduling.

GROUP = { **(groupnam,initnum[,groupnam,initnum]...)** | **(/groupnam[,groupnam]...)** }

Indicates either the names of job class groups (groupnam) that can be scheduled under this SELECT mode and the number of initiators (initnum) to be assigned to each of these groups; or, if a "/" precedes all the group names, this parameter identifies those job class groups that cannot be scheduled under this mode. The subparameter groupnam must match the NAME parameter on a GROUP statement. Jobs in the groups defined here as eligible for scheduling will not be scheduled until the execution resources are allocated. (Allocation is controlled by the allocation option with the EXRESC parameter on a GROUP statement.)

The GROUP parameter may be continued on another line if all class group names cannot fit on one line. A maximum of 255 groups for a single SELECT statement including continuation lines is allowed. The continuation must follow the same format as the preceding line; that is, if the preceding line specified groups to be excluded (by a "/"), then the continuation line must also indicate exclusions. You cannot exclude certain groups in one specification of GROUP and include other groups in another specification.

Note: The initnum value on the GROUP= keyword of the SELECT initialization statement overrides the value of the initcnt parameter on the EXRESC= keyword of the GROUP initialization statement.

Parameter Default: All groups are eligible for scheduling.

INCL = { **nn** | **14** }

Specifies a limit within the range of 0 to 15 past which a job's priority cannot be incremented by JES3 main device scheduling (MDS). Note that only jobs requiring volume mounting or referencing a volume that was mounted for another job are incremented by setup.

Parameter Default: 14

INCR = { **nn** | **1** }

Specifies a decimal number from 0 to 15 that is automatically added to the priority of the job which is set up. If a job has a priority of 5 when it is set up, and INCR=4 is specified, the job's priority is elevated to 9 (or to the value specified in the INCL parameter, whichever is less) after the devices have been allocated and set up. This parameter expedites the processing of jobs once devices have been assigned to them.

Parameter Default: 1

JOBMIX =(n1[,n2...,n45])

Specifies 1 to 45 values (ranging from 1 to 15) which indicate the optimal I/O rate job mix for 1 to 15 active initiators. These parameters update the 3 by 15 table (see Figure 12-11) of job mix values. Each row represents the job mix for a particular number of active initiators. The columns express the three job counts for low, high, and medium I/O rate. The JOBMIX parameters change the table as follows: the first three numbers n1, n2, and n3 are the optimal low, high, and medium I/O rate job counts when one initiator is active. The next three numbers n4, n5, and n6 are the job mixes when two initiators are active. The JOBMIX values for n7 through n45

continue across in rows and down a row at a time. If more than 15 initiators are active (as might be the case for a local main), the JOBMIX parameter is calculated to determine the number of active initiators. The scheduling of the next job is then performed based on the defined I/O rates in proportion to the number of active jobs for each I/O rate. If the number of JOBMIX operands exceeds one statement, the n-values can be continued in a JOBMIX keyword on a following statement. See the example of the SELECT statement.

Figure 12-11 indicates default JOBMIX values.

Active Initiators	Low	High	Medium
1	1	1	1
2	1	1	1
3	1	1	1
4	1	2	1
5	1	3	1
6	1	3	2
7	1	4	2
8	2	4	2
9	2	5	2
10	2	5	3
11	2	6	3
12	2	7	3
13	2	7	4
14	3	7	4
15	3	8	4

Figure 12-11. JOBMIX Default Values

LSTOR = { **nnnnn** / **12000** }

Defines the logical storage resource (in 1024-byte blocks) for a processor. The purpose of the LSTOR parameter is to allow the system programmer to control the number of jobs to be scheduled on a processor in such a way as to fully utilize real dynamic storage and at the same time minimize the probability of excessive paging (thrashing). The maximum value for the LSTOR parameter is 32767. Specifying 0 disables logical storage scheduling. Also, when insufficient logical storage is available, no further scheduling occurs. Jobs are scheduled on processors based on logical storage available.

Initially, it is recommended that LSTOR be defined to obtain maximum throughput with jobs using their normal region requests. Over-commitment of the MVS storage can then be planned by the systems programmer without the need for application programmer action. These recommendations should result in reasonable utilization of storage during the period when each job's actual real storage utilization (working set) is being determined.

For additional information about logical storage, see "Defining Logical Storage for Processors" in Chapter 2, "JES3 Job Management."

Coding Consideration: See the LSTRR parameter on the CLASS initialization statement. If LSTRR=0 is specified on the CLASS statement, logical storage *processing* is disabled for jobs of that class.

Parameter Default: 12000

MAGEL = { **nn** / **14** }

Specifies an aging priority limit (0-15) beyond which a job cannot be aged during GMS.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 14

MAGER = { **nnn** / **0** }

Specifies the number (0-255) of times a job must be eligible for aging before its job priority is actually incremented. If 0 is specified, no aging is done.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0 (a job will not be aged).

SAGEL = { **nn** / **14** }

Specifies an aging priority limit (0-15) beyond which a job cannot be aged during MDS setup for the job.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 14

SAGER = { **nnn** / **0** }

Specifies the number of times (0-255) a job must be eligible for aging during JES3 MDS processing before its job priority is actually incremented. If 0 is specified, no aging is done.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0

SBAR = { **PRTY** / **nn** / **16** }

Specifies a job priority barrier. Jobs equal to or above this barrier which cannot obtain all their required resources (volumes, data sets, and available devices) will reserve resources as they become available to prevent lower priority jobs from obtaining them.

PRTY
Indicates that the priority of the first job that cannot be setup is the priority barrier.

nn
Specifies a job priority level from 0 to 15.

16
Indicates there is no job priority barrier.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 16

SDEPTH = { **nnn** / **255** }

Specifies the maximum number (0-255) of jobs requiring operator mounts that may be set up at one time on any main for which this select mode is active. SDEPTH allows processors with different nonshared device configurations to be given a variable number of setup jobs, depending on the number of devices associated with the processor.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 255

Statement Default: JES3 assigns the following default select mode to any processor whose associated MAINPROC statement does not specify the SELECT= parameter.

```
SELECT,NAME=JS3BATCH,CHOICE=FFIT,LSTOR=12000,                X
SDEPTH=255,
MAGEL=14,MAGER=0,SAGEL=14,SAGER=0,SBAR=16,INCL=14,           X
INCR=1
```

Associated Statement/Parameter: The SELECT parameter specified on a MAINPROC statement as the initial select mode must match the NAME parameter on a SELECT statement. The CLASS parameter on the SELECT statement must match the NAME parameter on one or more CLASS statements. The GROUP parameter on the SELECT statement must match the NAME parameter on a GROUP statement.

Example: In the following example, the SELECT statement is specified for best performance in this environment:

Group	Job Class	I/O Rate	Jobs Per Day
AB	A	Low	100
	B	Low	
CD	C	Medium	200
	D	Medium	
EF	E	High	300
	F	High	
			600 Total

```
SELECT,NAME=SHIFT1,CHOICE=BMIX,CLASS=(A,B,C,D,E,F),          X
GROUP=(AB,2,CD,2,EF,2)
GROUP,NAME=AB,EXRESC=(SYS1,2)
GROUP,NAME=CD,EXRESC=(SYS1,4)
GROUP,NAME=EF,EXRESC=(SYS1,6)
```

Override: The *MODIFY,S operator command can be used to override some SELECT statement parameters.

SETACC (Specify Accessibility to Direct-Access Volumes)

The SETACC statement identifies those processors that normally have access to a permanently resident direct-access or MSS volume. The SETACC statement identifies the location of a volume on the uninitialized processors in a JES3 complex. SETACC prevents JES3 from setting up a job that needs the mounted volume until the processor is initialized. When all processors are initialized or the volume is found, the SETACC definition is no longer used and normal JES3 management of the volume and device occurs. The devices on which the volumes reside are defined on a DEVICE statement with an XTYPE parameter and PR subparameter.

```
SETACC,VOL=(volser,procname[,procname]...)
     [,VOL=(volser,procname[,procname]...)]
```

VOL =(volser,procname[,procname]...)
Associates one permanently resident direct-access or MSS volume with the specified processors. One VOL parameter is specified for each volume defined on the SETACC statement.

volser
Specifies the volume serial number of a direct access volume. JES3 is to make the volume "unavailable" until the named processor(s) complete initialization.

procname[,procname]
Specifies the name of the processors that must be initialized before JES3 will make the associated volume "available." Each processor name (procname) must match the name parameter on a MAINPROC statement.

Associated Statement/Parameter: The NAME parameter in the MAINPROC statement defines the processor name parameter specified in the SETACC statement. A DEVICE statement indicating the PR subparameter with the XTYPE parameter indicates a device where a permanently resident volume may reside.

Example: In the following example, two volumes are defined to be accessible only by the specified processors.

```
SETACC,VOL=(WORKDS,SYS1,SYS2),VOL=(SECURT,SYS1,SYS3)
```

Override: The SETACC function is nullified when the specified volume is located on a device or when all of the specified processors become initialized.

SETNAME (Set JES3 Device Names)

The SETNAME statement specifies all user-assigned names and device type names associated with MDS-managed devices, except those names used to define MSS virtual units (such as 3330V).

```
SETNAME,XTYPE=devicetype[{,NAMES=(name[,name]...)  }]
                         {,POOLNAMS=(pool[,pool]...)}
```

XTYPE =devicetype

Indicates a 1- to 8-character name which matches the first operand in the XTYPE parameter on a DEVICE statement. The XTYPE parameter associates this statement with a specific device definition indicated on a DEVICE statement. The XTYPE parameter must always precede the NAMES or POOLNAMS parameters.

Coding Consideration: Devices within a specific XTYPE should have compatible characteristics. If the XTYPE= parameter is the same on two different SETNAME statements, the NAMES= parameter on the two statements must be different.

NAMES =(name[,name]...)

Identifies all names (user-assigned and device type) used to refer to the device defined in the associated DEVICE statement. The NAMES parameter indicates what names are used to specify devices in the UNIT parameter of any DD statements. Each name must also have been indicated at system generation on the UNITNAME macro. Note that SYSGEN device type names must be included if allocation of cataloged data sets that reside on these device types is done by JES3 even though the devices are not directly referred to in the UNIT parameter of a DD statement. For example, if UNIT=2314 is specified, 2314 must be defined as a name.

By organizing names, the system programmer can identify unit allocation requirements (including specific device attributes, such as tapes with a special density) and can balance the load across available channel paths.

If the same names are specified in the NAMES parameter, but in a different order for the same XTYPE, an order of preference for device selection is created for requests that require volume mounting. (See the first example.) If an order of preference is created, it must be the same for both JES3 and MVS.

Note: If you have replaced one device type on your system with a different device type and the resource is still referred to using the old device name (for example, in catalog entries), do not remove the name of the old device type from the NAMES parameter. Leave it in and add the name of the new device type.

POOLNAMS =(pool[,pool]...)

Specifies the 1- to 8-character name(s) that may be used only for dedicating devices to job class groups or dependent job control jobs. The pool names specified may be used in the DEVPOOL parameter of the GROUP

statement and/or the //*NET statement. These names may not be used in the UNIT parameter of the DD statement.

Associated Statement/Parameter: The XTYPE parameter on this statement must match the first operand of the XTYPE parameter on one or more DEVICE statements. The DEVPOOL parameter on the GROUP statement may match the POOLNAMS parameter on the SETNAME statement. All names specified on the TYPE parameter of the HWSNAME statements must be specified on the NAMES parameter of the SETNAME statements.

Example 1: The following example shows how to achieve a preference order of device selection when devices are on three channel paths and volume mounting is required. As a result of these control cards, requests for DACH1 would attempt allocation on channel path 1, then on channel path 2, and finally on channel path 3. Similarly, requests for DACH2 would attempt allocation first on channel path 2, then on channel path 3, and then on channel path 1. By using DISK1, DISK2, or DISK3, strict channel path separation could be achieved.

```
DEVICE,XTYPE=(3330CH1,DA),XUNIT=(130,SY2,S1,ON)
DEVICE,XTYPE=(3330CH2,DA),XUNIT=(230,SY2,S1,ON)
DEVICE,XTYPE=(3330CH3,DA),XUNIT=(330,SY2,S1,ON)

SETNAME,XTYPE=3330CH1,NAMES=(DACH1,DISK1,3330)
SETNAME,XTYPE=3330CH2,NAMES=(DACH1,DACH2,DISK2,3330)
SETNAME,XTYPE=3330CH3,NAMES=(DACH1,DACH2,DACH3,DISK3,3330)
SETNAME,XTYPE=3330CH1,NAMES=(DACH2,DACH3)
SETNAME,XTYPE=3330CH2,NAMES=(DACH3)
```

Example 2: The following example illustrates how to dedicate two 3330 disks and three tapes on channel path 1. There is a two-channel main processor with four tape devices and four disk devices on each channel path. The accessibility desired by name is:

TAPE	Any tape
TAPEC1	Any tape on channel path 1
TAPEC2	Any tape on channel path 2
DISK	Any disk
DISKC1	Any disk on channel path 1
DISKC2	Any disk on channel path 2
3330	Any 3330 disk
3340	Any 3340 disk

The XTYPE names are chosen to indicate the model of a device and its channel path location. These must be specific enough to allow the separation of devices to satisfy the access level desired.

```
SETNAME,XTYPE=3340C1,NAMES=(DISK,DISKC1,3340)
SETNAME,XTYPE=3340C2,NAMES=(DISK,DISKC2,3340)
SETNAME,XTYPE=3330C1,NAMES=(DISK,DISKC1,3330),          X
POOLNAMS=NETDISK
SETNAME,XTYPE=3330C2,NAMES=(DISK,DISKC2,3330),          X
POOLNAMS=NETDISK
SETNAME,XTYPE=3400C1,NAMES=(TAPE,TAPEC1,3400-4),        X
POOLNAMS=NETTAPE
SETNAME,XTYPE=3400C2,NAMES=(TAPE,TAPEC2,3400-4)
```

The MVS generic names must be included if cataloged data sets residing on these types are to be allocated by JES3, even though direct reference to them is not made in the UNIT parameter of their DD statements.

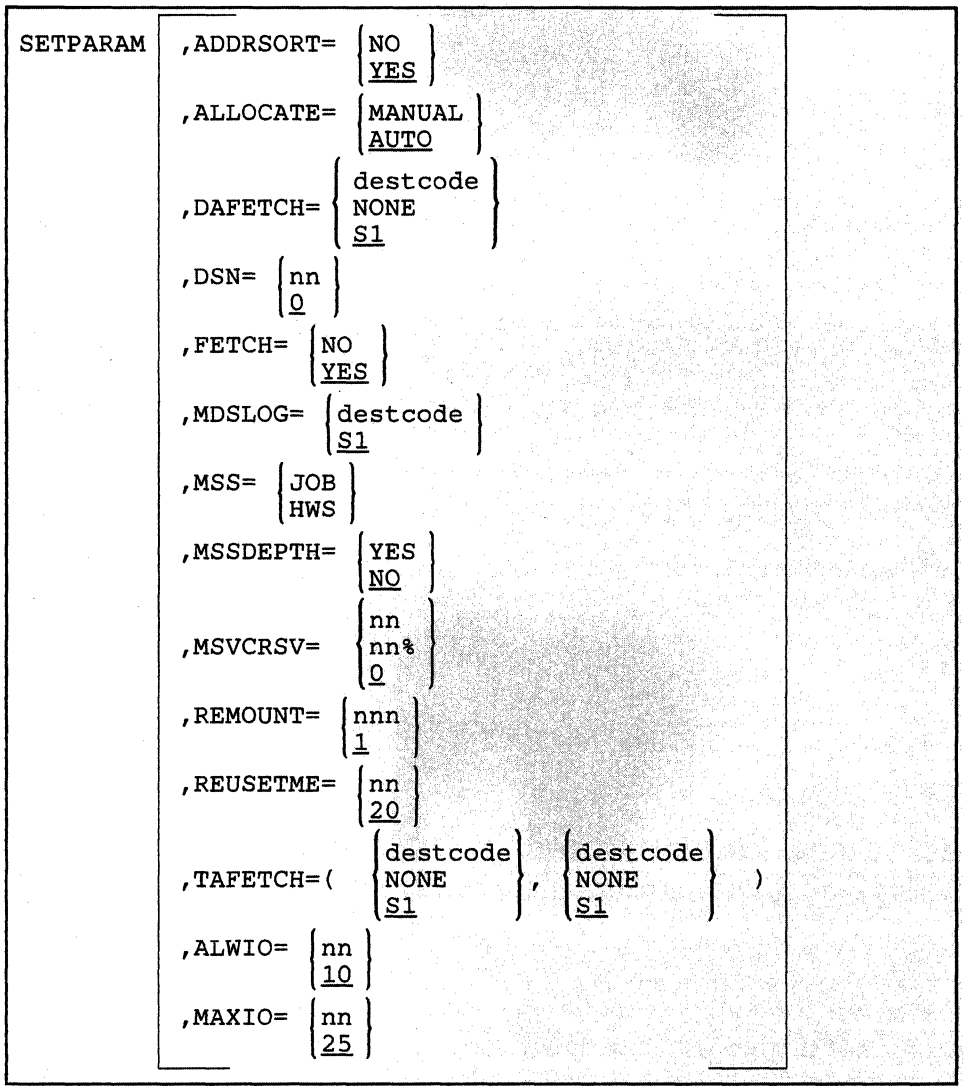
If a dynamic allocation is made for a time-sharing user and a UNIT parameter is not included in the DD statement, the UNIT parameter is obtained from the time-sharing user attribute data set (UADS). If the UADS does not contain a UNIT parameter, or if the user is not a time-sharing user, an MVS default of SYSALLDA (that is, all direct access devices) is assumed.

It is recommended that a default UNIT parameter value be defined in the UADS and that all dynamic allocations specify a UNIT parameter. Otherwise, if JES3 is to manage the dynamic allocations that do not have a UNIT parameter, the MVS default UNIT value (SYSALLDA) must be included in SETNAMES for all DA device types.

SETPARAM (Set MDS Parameters)

The SETPARAM statement specifies parameters that the JES3 main device scheduler (MDS) and the DYNAL DSP are to use in allocation, mounting, and deallocation of devices for jobs run on all processors. The SETNAME and DEVICE statements are used with the SETPARAM statements. SETNAME and DEVICE identify the devices to be managed by MDS. SETPARAM also indicates how MDS is to manage devices.

Note that if SETUP=NONE is specified on the STANDARDS statement to indicate no MDS, then SETPARAM is ignored and all devices are allocated, mounted, and deallocated by MVS.



ADDRSORT = { **NO** | **YES** }

Specifies the order in which JES3 MDS is to allocate devices.

NO
Indicates that devices within a device type are to be allocated in the same order as the DEVICE statements are placed in the initialization stream.

YES
Indicates that devices within a device type are to be allocated by the order of their device numbers, that is, 188, 189, 18A.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: YES

ALLOCATE = { **MANUAL** | **AUTO** }

Specifies whether or not automatic allocation of a job is to immediately follow MDS volume fetch. This parameter is ignored for jobs that reference only premounted volumes.

The FETCH parameter specified may override the ALLOCATE parameter.

MANUAL
Indicates that all jobs are to be suspended following volume fetch until the operator causes them to continue. Note that ALLOCATE=MANUAL is ignored if FETCH=NO is indicated; ALLOCATE=AUTO is assumed instead.

AUTO
Specifies that MDS will automatically attempt allocation of resources for all eligible jobs. Note that ALLOCATE=AUTO is assumed (ALLOCATE=MANUAL is ignored) if FETCH=NO is specified.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: AUTO

DAFETCH = { **destcode** | **NONE** | **S1** }

Specifies which console destination is to receive fetch messages for direct-access volumes.

destcode
Indicates a setup console destination assigned to at least one console by the DEST parameter on the CONSOLE statement. Direct-access volume fetch messages are issued to consoles with this destination code and also to the hard-copy log. For a description of the valid console destination codes, refer to the description of the DEST parameter on the CONSOLE statement.

NONE
indicates that volume fetch messages are not to be issued.

S1
indicates that volume fetch messages are to be issued to set up consoles with a destination code of S1; messages also are recorded on the hard-copy log.

Note that this parameter is ignored if FETCH=NO is also specified.

Parameter Default: S1

DSN = { **nn** / **0** }

Specifies the number of characters (0 to 44) of the data set name that is to be included in MDS volume fetch, mount, and breakdown messages. This parameter is used for message formatting.

If DSN=0 is specified, or assumed by default, then the data set name is omitted from these MDS messages.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0

FETCH = { **NO** / **YES** }

Indicates whether or not MDS is to issue volume fetch messages. Note that the FETCH parameter can override the ALLOCATE parameter.

NO
Specifies that MDS is not to issue volume fetch messages. If FETCH=NO is specified, ALLOCATE=MANUAL will be overridden (and ALLOCATE=AUTO assumed); MDS will automatically attempt to set up jobs.

YES
Indicates that MDS is to issue volume fetch messages

Parameter Default: YES

MDSLOG = { **destcode** / **S1** }

Specifies which console destination is to receive all nonaction messages (that is, job LOGON and error messages).

destcode
Indicates a setup console destination to which MDS directs all nonaction messages.

S1
Indicates that the nonaction messages are to be directed to the console(s) with the setup destination code S1.

Parameter Default: S1

MSS = { **JOB** | **HWS** }

Specifies whether or not MSS support is requested and the type of MSS setup requested. If this parameter is omitted, no MSS setup processing will occur.

JOB

Specifies that each 3330V unit requested by a job will result in a separate unit being assigned to the job by the MSS support.

HWS

Specifies that usage of the 3330V units will be minimized by reusing units in subsequent job steps. TYPE=3330V must be specified on the HWSNAME initialization statement or high watermark setup will not occur.

MSSDEPTH = { **YES** | **NO** }

Specifies whether jobs that use MSS virtual units are to be included in the setup depth counts.

YES

Specifies that any job that requests at least one MSS virtual unit is to be included in the setup depth counts for the job's job class, for the processor on which the job executes, and for the JES3 complex.

NO

Specifies that the use of an MSS virtual unit must not cause a job to be included in setup depth counts.

Parameter Default: NO

MSVCRSV = { **nn** | **nn%** | **0** }

Specifies how many MSS virtual units JES3 is to reserve in each staging drive group on each processor. You must reserve virtual units so JES3 can satisfy:

- Mass Storage Volume Group (MSVGP) allocation requests
- Dynamic allocation requests

JES3 will not use a reserved unit to satisfy batch allocation requests.

To reserve no virtual units, omit this parameter or specify MSVCRSV=0. When you reserve no virtual units, however, the probability increases that a job will fail because JES3 cannot satisfy an MSVGP allocation request or a dynamic allocation request.

You may specify nn as a count or as a percentage. In either case, nn must be a one-digit or two-digit number ranging from 0 to 99. To specify nn as a percentage, code nn%.

If you specify nn as a count, and the count is greater than the number of virtual units defined in a particular staging drive group, JES3 reserves all virtual units in that staging drive group.

To determine how many MSS virtual units are reserved, issue command *I,S,MR. To dynamically change the number of reserved MSS virtual units, issue command *F,S,MR=.

If you specify an integer greater than the maximum allowable, JES3 uses the parameter default. If you specify a negative integer or a non-numeric character, JES3 issues message IAT3245 and initialization terminates.

Parameter Default: 0

REMOUNT = { **nnn** / **1** }

Specifies the number of times that an operator can retry to correct volume mount errors for a job before the devices for the job are released and allocation is restarted. The value of nnn specifies the number of retries allowed, from 0 to 255. For example, if REMOUNT=1 is specified, the operator can make two attempts to mount the volume--the original mount request and one retry request.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 1

REUSETME = { **nnn** / **20** }

Specifies the time interval (nnn) in minutes between pauses of the periodic timer-scheduled reuse routine. The mean data reuse time is 1.5 times the (nnn) value specified or 30 minutes if the parameter is omitted.

The valid range for nnn is 0 to 999. Specifying 0 means that JES3 does not support the data reuse capability and volumes are deleted from the table as soon as they are used for the last time. However, 0 is not a recommended value since it causes inefficient use of MSS devices.

Any specification from 933 to 998 is converted to 932. If 999 is used, the timer is never set and the entries for the volumes remain in the SETVOL and SVX tables. Specifying 999 might be desirable in an installation where there is low MSS activity or where few virtual volumes are defined.

Volumes that have a data reuse time interval specified are placed in data reuse status after a volume is used for the last time and the MDS table entries that describe that volume are not deleted until the volume has remained unused for an entire time interval. The *MODIFY,S command can be used to immediately delete an MSS virtual volume entry from all MDS tables as soon as the volume is used for the last time. See *MVS/Extended Architecture Operations: JES3 Commands* for more information on this command.

For MSS devices, JES3 used the REUSETME parameter as a factor in selecting SDGs. The value specified in this parameter should be matched to the normal period for a volume to remain staged on a drive after the dismount is issued for that volume. If the REUSETME value is too high, JES3 may try to remount volumes on a particular SDG. If the REUSETME value is too low, JES3 may cause destaging and restaging.

Parameter Default: 20

TAFETCH =({**destcode** / **NONE** / **S1**} , {**destcode** / **NONE** / **S1**})

Specifies which console destination is to receive fetch messages for tape volumes. The first operand designates the console destination to receive specific (nonscratch) volume requests. The second operand designates the console destination to receive scratch volume requests.

destcode
Indicates a setup console destination assigned to at least one console by the DEST parameter on the CONSOLE statement. Tape volume (scratch or nonscratch) fetch messages are issued to consoles with this destination and also to the hard-copy log. For a description of the valid console destination codes, refer to the description of the DEST parameter on the CONSOLE statement.

NONE
Indicates that tape volume fetch messages are not to be issued.

S1
Indicates that tape volume fetch messages are to be issued to setup consoles with a destination code of S1; messages also are recorded on the hard-copy log.

Note that this parameter is ignored if FETCH=NO is also specified.

Parameter Default: (S1,S1)

ALWIO = {**nn** / **10**}

The ALWIO parameter specifies the current number of asynchronous I/O requests which can be processed at the same time. This value must be a number from 1 to the value specified in the MAXIO parameter.

Coding Considerations: The value specified in the ALWIO parameter must be less than or equal to the value specified in the MAXIO parameter.

This parameter can be displayed via the *INQUIRY,S,ALWIO=nn operator command, and modified via the *MODIFY,S,ALWIO=nn operator command.

Parameter Default: 10

MAXIO = { **nn** / **25** }

The MAXIO parameter specifies the maximum number of asynchronous I/O requests that can be processed simultaneously. Storage is obtained for the number of requests specified here. Note that an increase of one in this parameter results in a 76-byte increase in storage used. This parameter can only be changed when performing a warm start or cold start. The value specified in the MAXIO parameter may be a number from 1 to 99. The default value is 25.

Coding Consideration: The value specified in the MAXIO parameter must be greater than or equal to the value specified in the ALWIO parameter.

Parameter Default: 25

Statement Default: The statement default is as follows:

```
SETPARAM,ADDRSORT=YES,ALLOCATE=AUTO,DAFETCH=S1,          X
DSN=0,FETCH=YES,                                        X
MDSLOG=S1,MSSDEPTH=NO,MSVCRSV=0,REMOUNT=1,              X
TAFETCH=(S1,S1),REUSETME=20,ALWIO=10,MAXIO=25
```

Associated Statement/Parameter: The console destination codes specified in the MDSLOG, DAFETCH, and TAFETCH parameters must match the DEST parameter on a CONSOLE statement. The CONSOLE statement associates the destination code with a specific console. If MSS=HWS is specified, TYPE=3330V must be specified on the HWSNAME statement.

Example: In the following example, volume fetch messages would be issued to consoles with destination code:

S7 Nonscratch tape volume fetch messages.
S10 Scratch tape fetch messages.
S9 Direct-access volume fetch messages.

Also, MDS messages would identify the first 15 characters of the data set names. All nonaction messages would go to console destination S1. If necessary, one retry to mount any volume would be allowed. Allocation would occur automatically following volume fetch. Allocation order for devices would be by the order of their device numbers.

```
SETPARAM,FETCH=YES,TAFETCH=(S7,S10),DAFETCH=S9,DSN=15
```

Override: The SETPARAM statement is overridden if SETUP=NONE is specified on the STANDARDS statement; all devices are allocated, mounted, and deallocated by MVS instead of MDS.

SETRES (Mount Direct-Access Volumes)

The SETRES statement identifies frequently used direct-access or MSS volumes which are not permanently resident. The SETRES statement specifies volumes which may reside on devices at processor initialization time. When a specified volume is found to be present on an MDS-managed, removable, direct-access device during processor initialization, the volume is considered mounted by MDS, without a MOUNT command being necessary. Using SETRES, the time required to perform volume setup following a JES3 restart is reduced and management of highly used direct-access setup volumes is simplified. (No action is taken if a specified volume is not found during initialization.) JES3 treats the specified volumes as mounted until an MDS unload is issued.

A real direct-access device that is listed on a SETRES statement may also be assigned the reserved mount-attribute through a VATLST entry. (*MVS/Extended Architecture System Programming Library: System Modifications* describes VATLST entries.)

An MSS virtual volume that is listed on a SETRES statement must also be assigned the reserved mount-attribute through a VATLST entry.

```
SETRES,VOL=(volser[,volser]...)
```

VOL =(volser[,volser]...)
Indicates the volume serial numbers of volumes mounted on devices that can have these volumes removed (unmounted) by JES3 MDS. (Note that the XTYPE parameter on the DEVICE statement for the associated devices must include the RM subparameter.)

Example: In the following example, four volumes with these serial numbers are marked as mounted if found on devices at the time a processor is initialized.

```
SETRES,VOL=(111111,222222,2314AA,3330BB)
```

SPART (Spool Partition Definition)

The SPART statement defines one spool partition and specifies:

- The name of the partition
- Whether JES3 is to use the partition as the default partition
- Whether JES3 is to write initialization information to the named partition
- Whether the partition is to overflow into another partition
- The number of records in each track group

You may include up to 1024 SPART statements in the initialization stream. One or more of the spool partitions can be "dummy" partitions that do not contain any data sets.

The SPART statements must all appear before the ENDJSAM statement. If you add SPART statements during a warm start, include them *following* existing SPART statements in the initialization stream.

```
SPART,NAME=partitionname
      [,DEF= {YES}
             {NO }
      ,INIT= {YES}
             {NO }
      ,SPLIM=( {min} , {marg} )
              {10 }   {25  }
      ,OVRFL= {YES}
              {NO }
      ,GRPSZ= {partitionname}
              {nnn          }
              {135          }]
```

NAME =partitionname
Specifies the 1-8 character name of the spool partition. Do not name the partition "YES" or "NO."

DEF = { **YES** / **NO** }
Specifies whether this is the default spool partition.

YES
Specifies that this is the default spool partition.

NO
Specifies that this is not the default spool partition.

If you do not specify the DEF= parameter on any SPART statements, the partition defined on the first SPART statement in the initialization stream becomes the default partition.

If you have defined only one partition, that partition becomes the default, even if you have specified DEF=NO on the SPART statement.

If you specify DEF=YES on more than one SPART statement, the partition defined on the first statement that specifies DEF=YES becomes the default partition.

If you specify an invalid subparameter, JES3 uses the default.

Parameter Default: DEF=NO for all partitions except the first; DEF=YES for the first or only partition.

INIT = { **YES** / **NO** }

Specifies whether JES3 is to use this partition to write the initialization data needed for hot starts or local starts.

YES
Specifies that JES3 is to use this partition.

NO
Specifies that JES3 is not to use this partition.

If you omit INIT=YES from all SPART statements, JES3 writes the initialization data to the default spool partition.

If you specify INIT=YES on more than one SPART statement, JES3 writes the initialization data to the partition defined on the first SPART statement that specifies INIT=YES.

If you specify an invalid subparameter, JES3 uses the default.

Parameter Default: NO

SPLIM =({ **min** / **10** } , { **marg** / **25** })

Specifies the minimum and marginal percentages of spool space still available in active spool partitions. An active spool partition is one containing at least one spool data set. If the minimal or marginal percentages of spool space are reached, indicating that a spool partition is nearly full, JES3 issues action messages to the operator.

Define the character to be used by console service to warn the operator of minimum and marginal spool space conditions by using the FLAG parameter of the CONSTD statement.

min
Specifies the percentage of total spool space in an active spool partition which, when that percentage is all that is still available, defines a minimum spool space condition. For example, if *min* specifies 10, a minimum spool space condition exists when 10% or less of the spool space in an active spool partition is still available.

The percentage of spool space defining this condition may be between 0 and 99. It must, however, be smaller than or equal to the percentage defining a marginal spool space condition (see the next subparameter).

When a spool partition reaches a minimum spool space condition, JES3 issues a message stating that this condition has occurred. The message alerts the operator to inquire whether the spool partition automatically overflows into another partition. If the spool partition does overflow, no operator action is required. If the spool partition is not defined to overflow into another partition, the operator can use operator commands to take appropriate actions. For information on actions to take, see "Balancing the Workload Across Partitions" in Chapter 4, "Defining and Managing Spool Data Sets."

If a minimum spool space condition arises on the default spool partition, JES3 suspends all SYSOUT buffer processing. JES3 does not resume SYSOUT buffer processing until enough spool space is freed to reach a marginal spool space condition.

Always specify a minimum spool space percentage for the default spool partition so that enough spool space remains to perform a warm start. Otherwise, if JES3 requires a warm start and not enough spool space is available, you must perform a cold start.

marg

Specifies the percentage of total spool space in an active spool partition which, when that percentage is all that is still available, defines a marginal spool space condition. For example, if *marg* specifies 20, a marginal spool space condition exists when 20% or less of the spool space in an active spool partition is still available.

The percentage of spool space defining this condition may be between 0 and 99.

When a spool partition reaches a marginal spool space condition, JES3 issues a message stating that this condition has occurred. The message alerts the operator to inquire whether the spool partition automatically overflows into another partition. If the spool partition does overflow, no operator action is required. If the spool partition is not defined to overflow into another partition, the operator can use operator commands to take appropriate actions. For information on actions to take, see "Balancing the Workload Across Partitions" in Chapter 4, "Defining and Managing Spool Data Sets."

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: (10,25)

OVRFL = { **YES** | **NO** | **partitionname** }

Indicates where, if anywhere, data for this spool partition may be put when this partition becomes full. If the spool partition defined by this SPART statement is the default partition, this parameter is ignored. For guidelines on choosing overflow partitions, see "Defining Spool Partition Overflow" in Chapter 4, "Defining and Managing Spool Data Sets."

YES
Specifies that this spool partition may overflow only into the default spool partition.

NO
Specifies that this spool partition may not overflow into any other spool partition. Jobs requesting spool space in this partition when it is full will wait until space becomes available.

partitionname
Specifies the name of the spool partition into which this spool partition overflows.

GRPSZ = { **nnn** | **135** }

Specifies the number of spool records in each track group. (The BUFSIZE parameter determines the size of each spool record.) The number must not be greater than 999. JES3 rounds the specified value up to the number of records in the nearest whole physical track for the selected spool device type. For guidelines to help you select a value for this parameter, see "Determining the Size of a Track Group" in Chapter 4, "Defining and Managing Spool Data Sets."

Parameter Default: 135

If the initialization stream includes no SPART statements, JES3 defines one spool partition and names it JES3PART. JES3 then writes all spool data and the checkpoint information to this spool partition.

Associated Statement/Parameter: The SPART parameter specified on a CLASS, FORMAT, MAINPROC, SYSOUT or TRACK statement must correspond to the NAME parameter specified on an SPART statement.

Overrides: The SPLIM and GRPSZ parameter specifications on the SPART statement override the SPLIM and GRPSZ parameter specifications on the BUFFER statement.

Example 1: The following example defines two spool partitions, PARTA and PARTB. Partition PARTB is the default partition. JES3 also writes initialization information to partition PARTB.

```
SPART,NAME=PARTA
SPART,NAME=PARTB,DEF=YES,INIT=YES
```

Example 2: This example also defines two partitions, PARTC and PARTD. Because neither SPART statement specifies DEF=YES, JES3 uses the partition defined by the first SPART statement as the default partition (PARTC). Because neither SPART statement specifies INIT=YES, JES3 writes initialization information to the default partition, PARTC.

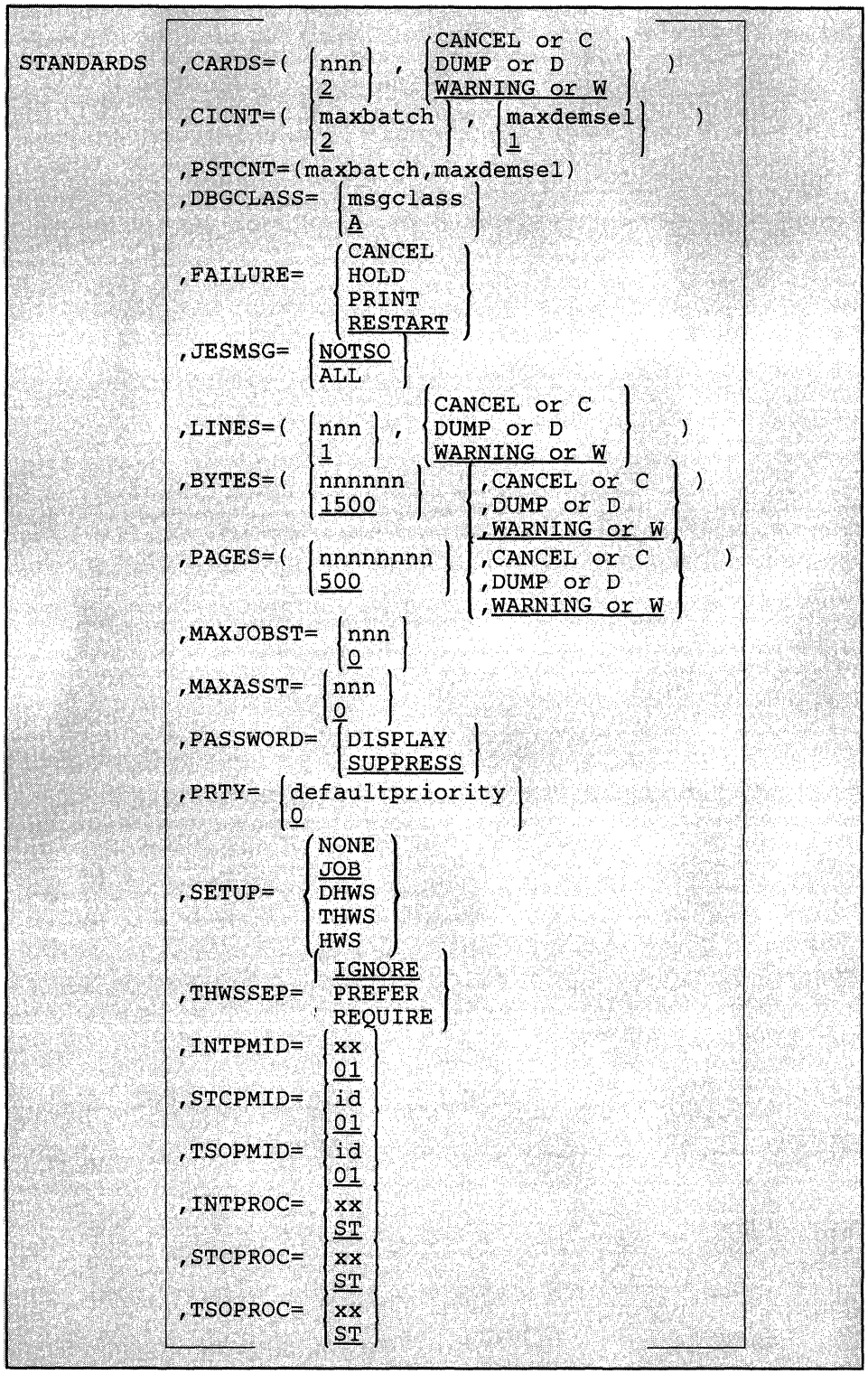
```
SPART,NAME=PARTC
SPART,NAME=PARTD
```

Example 3: This example defines 4 partitions. PART1 is the default partition. JES3 also writes initialization information to PART1. The second partition, PART2, overflows into the third partition, PART3, which overflows into the fourth partition. PART4 overflows into the default partition, PART1.

```
SPART,NAME=PART1,DEF=YES,INIT=YES
SPART,NAME=PART2,OVRFL=PART3
SPART,NAME=PART3,OVRFL=PART4
SPART,NAME=PART4,OVRFL=PART1
```

STANDARDS (Installation Defaults and Standards)

The STANDARDS statement specifies default values for information not provided on other JES3 initialization statements or on the //*FORMAT or //*MAIN JES3 control statements. It also provides standards to be applied to all jobs entering the system.



CARDS =({ nnn / 2 } , { CANCEL or C / DUMP or D / WARNING OR W })

Defines the maximum number of cards a job may punch and how to handle jobs exceeding this maximum.

nnn
Specifies the maximum number of cards, in hundreds, from 1 to 999. The default is 2; that is, 200 punched cards.

CANCEL or C
Specifies that a job be canceled without a dump if it exceeds nnn.

DUMP or D
Specifies that a job be canceled with a system abend dump if the job exceeds nnn.

WARNING or W
Specifies that JES3 issue a warning message to the operator console and that JES3 continue processing the job if the job exceeds the punch limit. JES3 reissues a warning message for every additional 200 cards the job punches. WARNING is the default.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: (2,W)

CICNT = { (maxbatch,maxdemsel) / (2,1) }

Specifies the maximum number of CI DSPs that can operate in the JES3 global address space at any time. The first subparameter (*maxbatch*) specifies the maximum number of CI DSPs that process batch jobs. The second subparameter (*maxdemsel*) specifies the maximum number of CI DSPs that process demand select jobs (that is, started tasks and TSO LOGONs).

The sum of the two subparameters cannot exceed 255. CI DSPs defined to process batch jobs cannot be used to process demand select jobs, and vice versa.

Specify as small a number as practical, since this parameter can be modified using an operator command. The count can be incremented or decremented using the *MODIFY,X operator command.

Notes:

1. *If you specify CICNT=(0,0), no CI DSPs will run in the JES3 global address space. However, specifying CICNT=0 defaults to CICNT=(0,1) and results in one demand select CI DSP in the JES3 global address space.*

2. *If you define C/I FSS address spaces for your installation, JES3 creates a special C/I subtask in the JES3 global address space for starting C/I FSS address spaces; it cannot be used for starting other tasks or TSO LOGONs. This subtask allows a C/I FSS address space to be started even if you specify CICNT=(0,0) in the initialization stream.*

Parameter Default: (2,1)

PSTCNT =(maxbatch,maxdemsel)

Specifies the maximum number of POSTSCAN DSPs that can operate in the JES3 global address space at any one time. The first subparameter *(maxbatch)* indicates the maximum number of POSTSCAN DSPs that can process batch jobs. The second subparameter *(maxdemsel)* indicates the maximum number of POSTSCAN DSPs that process demand select jobs (that is, started tasks and TSO LOGONs). A POSTSCAN DSP defined as processing batch jobs cannot be used to process demand select jobs, and vice versa.

The total of both numbers specified for this parameter must be between 1 and 32,767 (inclusive).

For guidelines on choosing a value for this parameter, see "Configuring C/I Processing" in Chapter 3, "Defining and Managing C/I Service."

Parameter Default: The combined maximum number of CI DSPs in the JES3 complex that can process batch jobs (maxbatch) and demand select jobs (maxdemsel). (That is, the total number of CI DSPs specified on the CICNT parameter of the STANDARDS statement, plus the number of CI DSPs specified on the DSPCNT parameter of every FSSDEF statement for a C/I FSS address space, including all values allowed to default.)

DBGCLASS = { **msgclass** / **A** }

Specifies the default messages class for the debug data set written by the debug facility.

Parameter Default: A

FAILURE = { **CANCEL** | **HOLD** | **PRINT** | **RESTART** }

Specifies the default job recovery option to be applied to jobs active at the time of a processor restart failure. This parameter is used if the FAILURE option is not indicated on the CLASS statement associated with the job and the job does not have a processor MVS checkpoint/restart option specified in its JCL.

CANCEL
Specifies that the job be canceled on the processor.

HOLD
Specifies that the job be held for restart on the processor.

PRINT
Specifies that the job be printed and held for restart on the processor.

RESTART
Specifies that the job be restarted on the processor at the beginning of the first step.

Parameter Default: RESTART

JESMSG = { **NOTSO** | **ALL** }

Specifies that messages sent to the JESMSG data set for TSO users are to be suppressed for all TSO users. Suppressing the messages sent to the JESMSG data set for TSO users significantly reduces the number of spool I/O requests needed to support dynamic allocation requests. NOTSO indicates that messages to the JESMSG file should not be written to the spool. The data set does exist and contains the job-started message.

ALL indicates that messages sent to JESMSG data set are written to the spool.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: NOTSO

LINES =({ **nnn** | **1** } **,** { **CANCEL or C** | **DUMP or D** | **WARNING or W** } **)**

Defines the maximum number of lines to be printed for a single job and how to handle jobs exceeding this maximum.

nnn
Specifies the maximum number of lines, in thousands, from 1 to 999. The default is 1; that is, 1000 lines.

CANCEL or C
Specifies that a job be canceled if it exceeds nnn.

DUMP or D
Specifies that a job be canceled with a system abend dump if the job exceeds nnn.

WARNING or W
Specifies that JES3 issue a warning message to the operator console and that JES3 continue processing the job if the job exceeds the line limit. JES3 reissues a warning message for every additional 3000 lines the job prints. WARNING is the default.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: (1,W)

MAXJOBST = { **nnn** / **0** }

Specifies the maximum number of JCL statements that a batch job can include. JES3 control statements are not counted. This limit applies no matter where in the complex the job's C/I processing takes place. For a discussion about how to select the job JCL limit, see "Preventing a Job from Dominating the SWA" in Chapter 3, "Defining and Managing C/I Service."

nnn
Specifies the total number of JCL statements that a single job may contain and still be processed by a converter/interpreter (CI) DSP. *nnn* may be any decimal integer from 0 to 99999999, inclusive. Specifying 0 indicates that you do not want to limit the total number of JCL statements in a single job.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0 (no limit)

MAXASST = { **nnnnnnnn** / **0** }

Specifies the maximum number of JCL statements for batch jobs that may be processed concurrently by all CI DSPs in the JES3 global address space. The value must be an integer between 0 and 99999999, inclusive. A value of 0 means no JCL statement limit applies; JES3 does not check how many JCL statements are being processed.

For guidelines on choosing a value for this parameter, see "Managing the Scheduler Work Area" in Chapter 3, "Defining and Managing C/I Service."

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0 (no limit)

PASSWORD = { **DISPLAY** / **SUPPRESS** }

Specifies that you want replies to security messages suppressed or displayed on JES3-managed consoles. You can dynamically change this specification using the *MODIFY,O,P= command.

DISPLAY
Specifies that you want replies to security messages entered from JES3-managed consoles displayed. Replies to security messages entered from MCS-managed consoles are always suppressed.

SUPPRESS
Specifies that you want to suppress replies to security messages entered from JES3-managed consoles.

Parameter Default: SUPPRESS

PRTY = { **defaultpriority** / **0** }

Specifies the default job priority (0 to 14) for a job whose priority is not specified in the JCL or on the CLASS statement.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: 0

SETUP = { **NONE** / **JOB** / **DHWS** / **THWS** / **HWS** }

Indicates the system standard for allocation of devices identified by the NAME parameter of SETNAME statement(s). The SETUP parameter specifies the type of setup processing, such as job setup, tape high watermark setup, or disk high watermark setup. This parameter also provides the default for the SETUP parameter on the //*MAIN JES3 statement. SETUP=NONE overrides the SETUP parameter on the //*MAIN statement.

NONE
Specifies that no pre-execution setup is to occur. All devices are allocated, mounted, and deallocated by MVS; the SETPARAM statement is ignored. The SETUNITS and SETNAME tables are created even though MDS processing does not occur.

JOB
Specifies that pre-execution setup is to occur by job for those devices (indicated in the UNIT parameter of DD statements for the job) which are identified in the SETNAME statement. All devices for the job are allocated to the job from first step to last. This type of setup improves job turnaround time at the expense of overall device usage efficiency.

DHWS
Specifies that high watermark setup is to occur for only MVS direct-access units identified in the SETNAME statement. The direct-access units must also be specified on the TYPE parameter of the HWSNAME statement. MDS will attempt to allocate the minimum number of direct-access devices for a job. Other devices are allocated based on the amount required for the entire job.

THWS
Specifies that high watermark setup is to occur for only tape units identified in the SETNAME statement. The tape units must also be specified on the TYPE parameter of the HWSNAME statement. MDS will attempt to allocate the minimum number of tape devices for a job. Direct-access devices are allocated based on the amount for the entire job.

HWS
Specifies that high watermark setup is to occur for all devices required for a job running on an MVS processor which are indicated in the SETNAME statement. The devices must also be specified on the TYPE parameter of the HWSNAME statement. MDS will attempt to allocate the minimum number of devices required for the job to run.

If you specify an invalid subparameter, JES3 uses the parameter default.

Parameter Default: JOB

THWSSEP = { **IGNORE** | **PREFER** | **REQUIRE** }

Specifies whether you want scratch and specific tape requests separated during high watermark processing. This keyword is valid only when you specify the THWS parameter of the SETUP= keyword. You can use this keyword to direct specific and scratch tape requests to different types of tape drives (for example, you may want JES3 to allocate only scratch tape requests to an IBM 3480 that is equipped with an automatic cartridge loader). An end user can override this keyword using a //*MAIN JECL statement. You can also modify these values using JES3 *MODIFY,S command. For information about using the JES3 *MODIFY,S command see *MVS/Extended Architecture Operations: JES3 Commands.*

IGNORE
Specifies that JES3 not split-up scratch and specific tape requests during high watermark processing. Both scratch and specific tape requests can be allocated on the same tape drive.

PREFER
Specifies that you want JES3 to attempt to allocate scratch and specific tape requests on separate tape drives without allocating additional devices. If JES3 cannot separate the requests, tape requests will be allocated on the same tape drive.

REQUIRE
Specifies that JES3 should not allocate scratch and specific tape requests on the same tape drive, even if JES3 must allocate additional tape drives to satisfy the request.

Parameter Default: IGNORE

INTPMID = { **id** / **01** }
Specifies the 2-byte identifier (ID) of the converter/interpreter options list for jobs entered using the internal reader. The ID must match the ID specified by the PARMID parameter on a CIPARM initialization statement. If no CIPARM statements are included in the initialization stream, the default value is used.

Parameter Default: 01

STCPMID = { **id** / **01** }
Specifies the 2-byte identifier of the parameter list to be used when interpreting started task jobs. The parameter list specified must be defined on a CIPARM statement.

Parameter Default: 01

TSOPMID = { **id** / **01** }
Specifies the 2-byte identifier of the parameter list to be used when interpreting TSO LOGON jobs. This parameter list also becomes the default for *any* job submitted under TSO, including internal reader jobs. The parameter list must be defined on a CIPARM statement.

Parameter Default: 01

INTPROC = { **xx** / **ST** }
Specifies the appropriate IATPLBxx procedure library to be searched by the MVS converter when resolving procedure references for jobs submitted using the internal reader. This procedure library must be defined by an IATPLBxx DD statement in the JES3 start procedure or by a DYNALLOC initialization statement. An individual job can override this value using the PROC parameter on the //*MAIN JES3 control statement.

Parameter Default: ST

STCPROC = { **xx** / **ST** }
Specifies the appropriate IATPLBxx procedure library to be used for started task jobs. This procedure library must be defined by a IATPLBxx DD statement in the JES3 procedure or a DYNALLOC initialization statement. The default is IATPLBST.

Parameter Default: ST

TSOPROC = { **xx** / **ST** }

Specifies the appropriate IATPLBxx procedure library to be used for TSO LOGON jobs. This procedure library must be defined on a IATPLBxx DD statement or a DYNALLOC statement. The default procedure library is IATPLBST.

Parameter Default: ST

BYTES = { **nnnnnn** / **1500** } { **,C or CANCEL** / **,D or DUMP** / **,W or WARNING** }

Specifies the maximum number of bytes of data that can be written to a spool data set for a job and the action to be taken for any job that exceeds the maximum.

nnnnnn
Specifies the maximum number of bytes of data that can be written to a spool data set for a job. The value specified must be between 1 and 999999, inclusive, and represents thousands of bytes (that is, 1000 times nnnnnn) of spool data. This value may be overridden by the BYTES parameter on the //*MAIN JES3 job control statement.

C or CANCEL
Specifies that a job whose output exceeds the byte limit is to be canceled without a dump.

D or DUMP
Specifies that a job whose output exceeds the byte limit is to be canceled with a system abend dump.

W or WARNING
Specifies that JES3 issue a warning message to the operator console and that JES3 continue processing the job if a job exceeds the byte limit. JES3 reissues a warning message for every additional 1,500,000 bytes the job writes to spool.

Parameter Default: (1500,W)

PAGES = { **nnnnnnnn** / **500** } { **,C or CANCEL** / **,D or DUMP** / **,W or WARNING** }

Specifies the maximum number of pages for a single job, and the action to be taken for any job whose output exceeds the maximum.

nnnnnn
Specifies the maximum number of pages for a single job. This value may be overridden by the PAGES parameter on the //*MAIN JES3 job control statement. The specified number must be between 1 and 16777215, inclusive.

C or CANCEL
Specifies that a job whose output exceeds the page limit is to be canceled without a dump.

D or DUMP
Specifies that a job whose output exceeds the page limit is to be canceled with a system abend dump.

W or WARNING
Specifies that JES3 issue a warning message and that JES3 continue processing the job if the job exceeds the page limit. JES3 reissues a warning message for every additional 500 pages that the output exceeds the limit.

Parameter Default: (500,W)

Statement Default: The statement default is as follows:

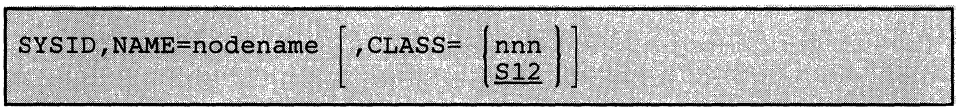
```
STANDARDS,CARDS=(2,W),CICNT=(2,1),DBGCLASS=A,FAILURE=RESTART,          X
JESMSG=NOTSO,LINES=(1,W),MAXJOBST=0,MAXASST=0,PRTY=0,SETUP=JOB,        X
INTPMID=01,STCMID=01,TSOPMID=01,INTPROC=ST,STCPROC=ST,TSOPROC=ST,      X
BYTES=(1500,W),PAGES=(500,W)
```

Example: This example specifies installation default values for information not provided on other JES3 initialization statements or JES3 control statements.

```
STANDARDS,CICNT=5,PRTY=2,FAILURE=PRINT,LINES=(100,W),CARDS=(200,W),     X
SETUP=NONE,PAGES=(1000,W),BYTES=(1500,W),INTPMID=I1,STCPMID=S1,         X
TSOPMID=T1,INTPROC=I1,STCPROC=S1,TSOPROC=T1
```

SYSID (Define the Default MVS/BDT Node)

The SYSID statement defines the default MVS/Bulk Data Transfer (BDT) node for this JES3 complex. If the JES3 complex includes one or more MVS/BDT facilities (program product 5665-302), you must include this statement in the JES3 initialization stream. JES3 submits MVS/BDT commands and transactions to the MVS/BDT node defined by this statement unless otherwise specified on the command or transaction. Include only one SYSID statement in the JES3 initialization stream.



NAME = nodename

Specifies the name of the default MVS/BDT node. This name must match the node name on a MVS/BDT SYSID initialization statement.

CLASS = { nnn / S12 }

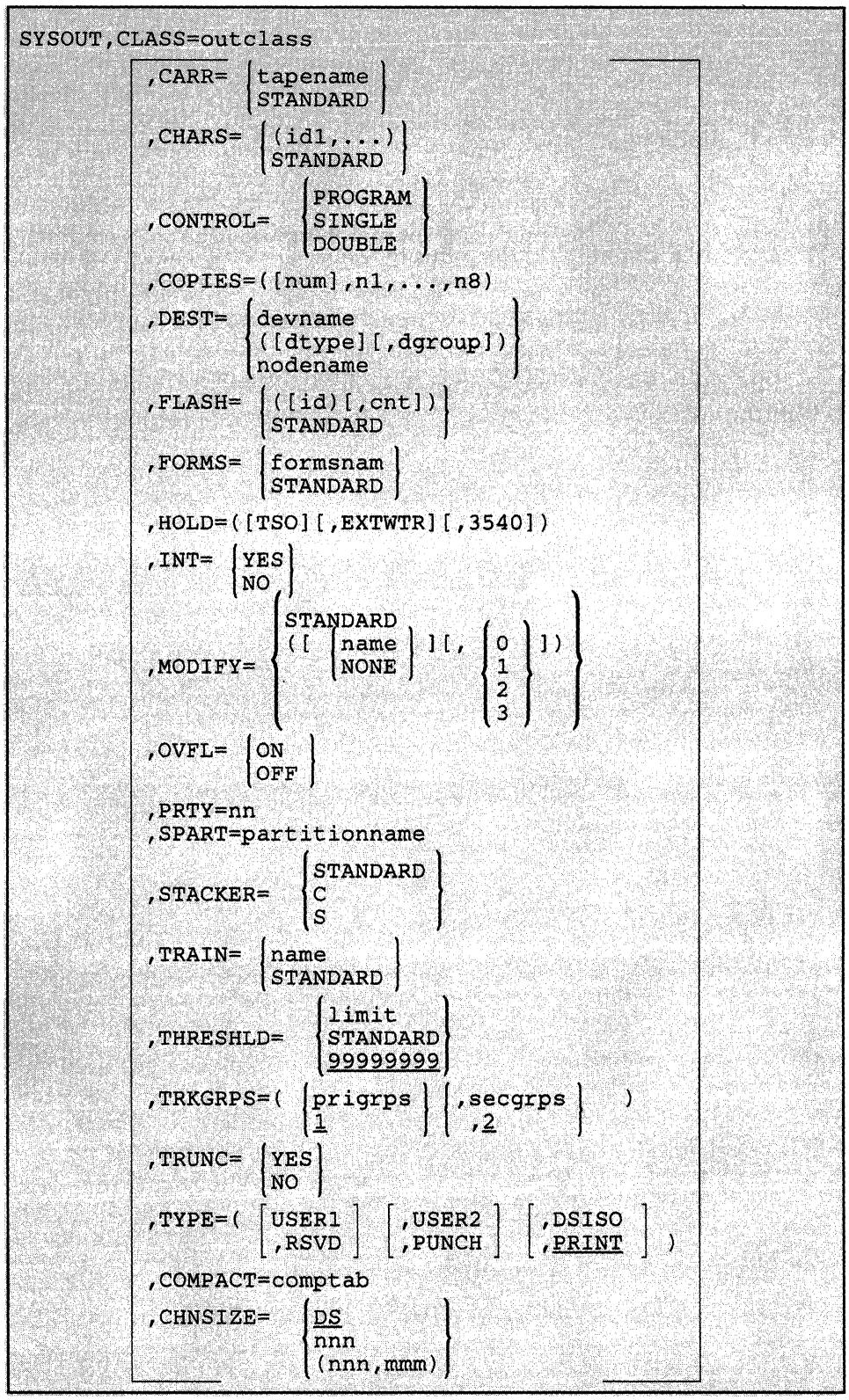
Specifies the message class to which MVS/BDT "system" messages that do not specify a message class are to be sent. MVS/BDT "system" messages are those issued by MVS/BDT dynamic application programs (DAPs) with no explicit transaction origin id. If you let this parameter default, all MVS/BDT "system" messages will appear on JES3 consoles that receive message class S12.

SYSOUT (SYSOUT Class Characteristics)

The SYSOUT statement is required for each JES3 output class that requires other than TYPE=PRINT processing (JES3 initially sets all SYSOUT classes to TYPE=PRINT.)

If STANDARD is specified for any parameters, the corresponding value from the OUTSERV statement is used. The CARR, FORMS, and TRAIN parameters on this statement override corresponding values on the OUTSERV and DEVICE statements.

The SYSOUT statement parameters are applicable only to JES3 processing. These characteristics are not available to external functions that process JES3 SYSOUT data (for example, external writer).


SYSOUT,CLASS=outclass
,CARR= tapename | STANDARD
,CHARS= (id1,...) | STANDARD
,CONTROL= PROGRAM | SINGLE | DOUBLE
,COPIES=([num],n1,...,n8)
,DEST= devname | ([dtype][,dgroup]) | nodename
,FLASH= ([id)[,cnt]) | STANDARD
,FORMS= formsnam | STANDARD
,HOLD=([TSO][,EXTWTR][,3540])
,INT= YES | NO
,MODIFY= STANDARD | ([name | NONE][,0 | 1 | 2 | 3])
,OVFL= ON | OFF
,PRTY=nn
,SPART=partitionname
,STACKER= STANDARD | C | S
,TRAIN= name | STANDARD
,THRESHLD= limit | STANDARD | 99999999
,TRKGRPS=(prigrps | 1) (,secgrps | ,2)
,TRUNC= YES | NO
,TYPE=(USER1 | ,RSVD) [,USER2 | ,PUNCH] [,DSISO | ,PRINT])
,COMPACT=comptab
,CHNSIZE= DS | nnn | (nnn,mmm)

CLASS =outclass
Specifies the SYSOUT class (A-Z,0-9) being defined.

CARR = { **tapename** | **STANDARD** }
Specifies the name (1 to 8 characters) of the carriage tape required to print this SYSOUT class. This parameter overrides the CARRIAGE parameter specified as an installation default on the OUTSERV statement and the CARRIAGE parameter on the DEVICE statement indicating the carriage tape initially mounted. If STANDARD is specified, the carriage tape name is taken from that specified on the OUTSERV initialization statement.

If the device is a 3211, 3203, or 3800 printer, a module must be included in SYS1.IMAGELIB, having the name FCB2xxxx or FCB3xxxx, where xxxx is the 1- to 4-character name of the FCB.

For more information about SYS1.IMAGELIB, see *MVS/Extended Architecture System Programming Library: Data Management.*

CHARS = { **(id1,...)** | **STANDARD** }
Specifies the name of the character image to be used in processing this SYSOUT class.

id1-id4
Specifies up to four images to be used for processing this SYSOUT class.

STANDARD
Specifies that the image to be used is as defined by the OUTSERV statement.

CONTROL = { **PROGRAM** | **SINGLE** | **DOUBLE** }
Specifies the type of carriage spacing control required to print this SYSOUT class.

PROGRAM
Specifies that the carriage control character is to be the first character of each logical record in the data set.

SINGLE
Specifies single spacing for the data set.

DOUBLE
Specifies double spacing for the data set.

COPIES =([num],n1,...,n8)

Specifies the number of copies of each data set to be produced.

num

Specifies the number of copies, from 0 to 255, to be produced for this SYSOUT class. If specified alone, it applies to all printers.

n1-n8

Specifies the number of transmissions of a data set and the number of copies, from 1 to 255, to be produced with each transmission. If n1 is specified, it is applicable only to impact printers. Example: COPIES=(1,5,2,6) produces one copy on an impact printer and three transmissions of the data set to a 3800 printer, where the first transmission will produce five copies, the second will produce two copies, and the third will produce six copies.

DEST = { **devname** / **([dtype][,dgroup])** / **nodename** }

Specifies the printer or punch to process this SYSOUT class. This parameter indicates the device specifically by name or nonspecifically by device type and/or device group.

devname

Specifies a 1- to 8-character name for the printer or punch. The name must match the JNAME parameter on a DEVICE statement defining this device.

dtype

Specifies a 1- to 8-character device type which matches the DTYPE parameter on a DEVICE statement for a printer or punch. If dtype only is specified, then any device of this type may process the output class. If dtype is specified with dgroup, then any device of that type within the group will process the output class.

dgroup

Specifies a 1- to 8-character device group name which matches the DGROUP parameter on a DEVICE statement for a printer or punch. If dgroup only is specified, then any device in that group may process the output class. If dgroup is specified with dtype, then any device in that group of the specified type will process the output class.

nodename

Specifies a 1- to 8-character name that is the name of the node to which JES3 networking output is directed.

FLASH = { **([id][,cnt])** | **STANDARD** }

Specifies the name of the forms flash cartridge for a 3800 printer to be used on this SYSOUT class.

id
Specifies the name of the forms flash cartridge.

cnt
Specifies the number of consecutive copies the forms flash is to print.

STANDARD
Specifies that the forms flash cartridge to be used was defined in the OUTSERV statement.

FORMS = { **formsnam** | **STANDARD** }

Specifies the name of the printer forms or card stock to be used when processing this output class. This parameter overrides the FORMS parameter specified on the OUTSERV or DEVICE statement. If STANDARD is specified, the forms or card stock specification is taken from that specified on the OUTSERV initialization statement.

HOLD =([TSO][,EXTWTR][,3540])

Specifies that the output data set is to be held for use by a system function.

TSO
Specifies that the data set is to be held for return to a TSO user.

EXTWTR
Specifies that the data set is to be returned to an external writer.

3540
Specifies that the data set is to be returned to the 3540 external writer.

INT = { **YES** | **NO** }

Specifies whether or not the interpret option is required for punched output.

YES
Indicates that a card punch with the interpret option (device type PUN3525I or 3525M) is to be used. Note that the DEST parameter must include a PUN3525I (or M) punch; otherwise, INT=NO is assumed.

NO
Indicates that the interpret option is not needed.

MODIFY = $\left\{ \begin{array}{l} \text{STANDARD} \\ \left(\left[\begin{array}{l} \text{name} \\ \text{NONE} \end{array} \right] \left[, \left\{ \begin{array}{l} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\} \right] \right) \end{array} \right\}$

Specifies the copy modification module (for 3800) and table reference character to be used for this SYSOUT class.

STANDARD
Specifies the copy modification module to be used as defined by the OUTSERV statement.

name
Specifies the name of the copy modification module to be used (1-4 characters).

NONE
Specifies that no copy modification module is to be used.

0-3
Specifies the table reference character to be used with the copy modification module.

OVFL = $\left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\}$

Specifies the overflow option.

ON
Indicates that the printer should eject whenever the end-of-forms indicator is sensed (skips to channel 12).

OFF
Indicate that the forms overflow control is not to be used.

PRTY =nn
Specifies the priority at which the data set is to enter the output queue. The value may be from 0 to 255.

SPART =partitionname
Specifies the spool partition that JES3 is to use when it writes data for this SYSOUT class. To specify the default spool partition, omit this parameter. If you specify a partition name, the name must match the NAME parameter on a SPART statement.

If you specify the SPART parameter, you must also specify TYPE=DSISO.

If you specify an undefined partition name, JES3 uses the default spool partition.

STACKER = { **STANDARD** | **C** | **S** }

Specifies the 3800 stacker in which this SYSOUT class output is to be placed.

STANDARD
Specifies the stacker as defined in the OUTSERV statement.

C
Specifies the continuous stacks. If no 3800 has BTS features, STACKER=C is forced.

S
Specifies the sheet stacker, where offset stacking takes place.

TRAIN = { **name** | **STANDARD** }

Specifies the print train or band required to print this SYSOUT class.

name
Specifies the name of the print train or band. This parameter specification overrides the TRAIN parameters specified on the OUTSERV or DEVICE statements.

STANDARD
Specifies that the value specified on the OUTSERV statement is to be used.

THRESHLD = { **limit** | **STANDARD** | **99999999** }

Specifies the default maximum size for a SYSOUT data set. This parameter is used by output service to build and queue OSEs for writers. The THRESHLD parameter assumes that the data set size is the number of records in the data set multiplied by the number of copies. Data sets that equal or exceed the value specified are queued as separate OSEs for output service writers. Specifying STANDARD causes the value specified on the OUTSERV statement to be used.

The THRESHLD parameter on the SYSOUT statement overrides the THRESHLD parameter on the OUTSERV statement.

TRKGRPS = ({ **prigrps** | **1** }) ({ **,secgrps** | **,2** })

Specifies the number of track groups (as defined by the GRPSZ parameter on the BUFFER or SPART statement) JES3 is to allocate to jobs within this SYSOUT class. For guidelines on how to determine the appropriate value for the TRKGRPS parameter for your installation, see "Determining Track Group Allocation Sizes" in Chapter 4, "Defining and Managing Spool Data Sets."

prigrps
Specifies the number of track groups to be initially allocated to jobs in this SYSOUT class. The specified value may be 1 through 9.

secgrps
Specifies the number of track groups to be allocated to jobs in this SYSOUT class subsequent to their primary allocation. JES3 allocates the specified amount of spool space after the job uses up its initial allocation, and again (for an unlimited number of times) when the job uses up each secondary allocation and requests more spool space. The specified value may be 1 through 9.

If you specify this parameter, you must also specify the TYPE=DSISO parameter.

Parameter Default: (1,2)

TRUNC = { YES / NO }

Specifies whether or not trailing blanks are to be truncated from SYSOUT data belonging to this SYSOUT class.

YES
Indicates that trailing blanks are to be truncated from SYSOUT data.

NO
Indicates that trailing blanks are not to be truncated from SYSOUT data.

Parameter Default: The default is the value specified on the TRUNC= parameter of the BUFFER statement. If the TRUNC= parameter is omitted from the BUFFER statement, the default for both parameters is TRUNC=YES.

TYPE =([USER1][,USER2] [,DSISO][,RSVD] [,PUNCH][,PRINT])
Specifies the action JES3 takes after a data set in this SYSOUT class is processed. Multiple subparameters must be specified for type DSISO and RSVD data sets if PRINT, PUNCH, or TSO type characteristics are also required. For example, TYPE=(PRINT,DSISO) causes the data set to be printed, but TYPE=DSISO never prints. Omission of the output type subparameter results in the data set becoming lost to output service.

USER1
Specifies that no action is to be taken by JES3 for this class. It is a user responsibility to process data sets in this SYSOUT class.

USER2
Specifies that no action is to be taken by JES3 for this class. It is a user responsibility to process data sets in this SYSOUT class.

DSISO
Specifies that each data set in this class is to have its own track allocation table (TAT). This increases spool utilization because each data set can be purged when the data set processing is complete

instead of when the job completes. (Note that the subparameter has no relationship to any other TYPE specification.) If you specify the SPART parameter on this statement, you must also specify the DSISO parameter.

RSVD
Specifies that this is a reserved SYSOUT class. Reserved classes may be used to hold SYSOUT output for TSO. A SYSOUT data set assigned to a reserved class is held for TSO if the MSGCLASS parameter specified for the generating job is also a reserved class.

PUNCH
Specifies that data sets in this SYSOUT class are to be punched.

PRINT
Specifies that data sets in this SYSOUT class are to be printed.

Parameter Default: PRINT

COMPACT =comptab
Specifies the 1- to 8-character name of a compaction table defined to JES3 by the COMPACT initialization statement. Any data set in this SYSOUT class which is sent to a SNA work station (which supports compaction) is compacted using this compaction table instead of the work station default compaction table.

Parameter Default: If this parameter is not specified, the default compaction table or default compaction table for the work station is used.

CHNSIZE = { **DS** | **nnn** | **(nnn,mmm)** }
Specifies the size of the RU chain to be transmitted to SNA workstations.

DS
Specifies that the entire data set is to be sent as a single chain.

nnn
Specifies the number of pages that the chain is to contain. Control characters in the data (skip to channel 1 for printers or eject for punches) delimit the page size.

The value for nnn can be any number from 1-255.

(nnn,mmm)
Specifies the number of pages (nnn) in the chain and the number of logical records (mmm) in each page. This form of the subparameter allows you to transmit, as a multiple chain, a data set that contains no control characters.

The value for nnn and for mmm can be any number from 1-255.

Notes:

1. *If you specify DS, JES3 takes output checkpoints as specified by the CKPNT parameter on this statement.*

2. *If you specify nnn or (nnn,mmm), JES3 takes an output checkpoint before each chain is transmitted.*

3. *CHNSIZE specified on a //*FORMAT statement overrides this specification. CHNSIZE specified on this statement overrides CHNSIZE specified on a DEVICE statement.*

4. *Sending the entire data set as a single chain gives the best performance. If, however, the receiving device requires an operator intervention or there is a transmission error, JES3 recovery will cause the entire data set to be resent.*

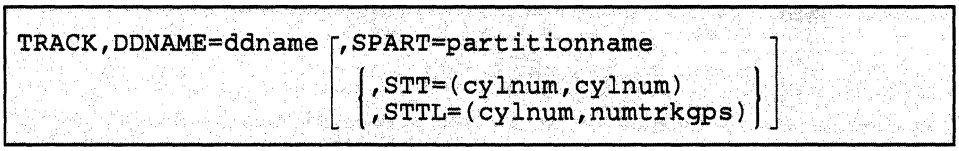
Overrides: The CARR, CHARS, FLASH, FORMS, MODIFY, STACKER, and TRAIN parameters on this SYSOUT statement override corresponding values on the OUTSERV and DEVICE statements.

The TRKGRPS parameter on this SYSOUT statement overrides corresponding values on the CLASS and MAINPROC initialization statements and on the //*MAIN JES3 control statement.

If specified, the TRUNC parameter on this SYSOUT statement overrides the corresponding value on the BUFFER initialization statement.

TRACK (Preformatted Spool Data Set)

The TRACK statement should replace a corresponding FORMAT statement in an initialization stream after the spool data set specified by the FORMAT statement has been formatted. The TRACK statement indicates that the corresponding data set has been formatted. The maximum number of TRACK statements is 1024.



DDNAME =ddname
Specifies the name (1 to 8 characters) of the DD statement that defines the spool data set. Once a spool data set is defined, it must be referred to by the same ddname.

SPART =partitionname
Specifies that the spool data set identified by this statement is a member of the named spool partition. The partition name must match a partition name specified on an SPART statement.

To specify that the data set is a member of the default partition, omit this parameter. You must assign at least one spool data set as a member of the default spool partition, or you must allow at least one spool data set to become a member by default. If you fail to do this, JES3 terminates its processing during initialization.

STT =(cylnum,cylnum)
Specifies the range of cylinders to be allocated to the single track table (STT). This range must be within the extent allocated to the data set. The value of *cylnum* specifies an absolute cylinder number. (Absolute cylinder numbers are device-dependent; the component description for the device describes the numbering scheme.)

The range indicated by *(cylnum,cylnum)* can be one cylinder (for example: 24,24) or several cylinders (for example: 24,28) and can be in ascending or descending order. JES3 allocates to the STT only those track groups that fall completely within the indicated range of cylinders. The value for *cylnum* cannot be 0.

For fixed head devices, allocate cylinders under the fixed heads for better performance. For other devices, allocate cylinders in the center of the data set.

If you omit either the STT= or STTL= keyword, or you specify an invalid range, JES3 allocates the centermost 2 track groups of each spool data set in the default partition as the initial STT allocation.

Note: If you change this parameter and want the change to go into effect for existing spool data sets, you must perform a cold start.

STTL =(cylnum,numtrkgps)

Specifies the location and number of track groups to allocate to the single track table (STT). These track groups must be within the extent allocated to the data set. The value for *cylnum* specifies an absolute cylinder number indicating the beginning cylinder number of the STT allocation in this extent. (Absolute cylinder numbers are device-dependent; the component description for the device describes the numbering scheme.) The value for *numtrkgps* specifies the number of track groups to allocate to this extent, beginning with the first track group that is located completely in cylinder *cylnum*. The maximum number of track groups that may be allocated to the STT is 9999.

For fixed head devices, allocate cylinders under the fixed heads for better performance. For other devices, allocate cylinders in the center of the data set.

If you omit either the STT= or STTL= keyword, or you specify an invalid range, JES3 allocates the centermost 2 track groups of each spool data set in the default partition as the initial STT allocation.

Note: If you change this parameter and want the change to go into effect for existing spool data sets, you must perform a cold start.

Associated Statement/Parameter: The TRACK statement should replace the FORMAT statement after each spool data set has been formatted. The SPART parameter specification on the TRACK statement must match the NAME parameter on a SPART statement.

Example: The following example shows how two spool data sets are specified on TRACK statements and on DD statements in the JES3 procedure.

Track statements.

```
TRACK,DDNAME=SPOOL1
TRACK,DDNAME=SPOOL2
```

Associated DD statements in the JES3 procedure:

```
//SPOOL1 DD DSN=JES3.QUES1,DISP=OLD,UNIT=3330,
VOL=SER=MVSRW1
//SPOOL2 DD DSN=JES3.QUES2,DISP=OLD,UNIT=3330,
VOL=SER=MVSRW2
```

Appendix A. RMT Option Statements

If RTP programs are to be generated, parameters that define those programs must be specified. Additionally, if changes are to be made to the RTP program source modules, these changes must be specified in control statements.

The RMT generation procedure is described in Chapter 6, "Defining and Managing JES3 Resources" under the topic "Generating Remote Terminal Processing Programs." The rest of this appendix describes the RMT option statements that you must code.

For an RMT generation, the input deck contains one or more RTP program descriptions. Each terminal program to be generated is described by card entries in the following order.

1. JES3 remote terminal processor program identification
2. RMT generation parameter cards
3. $.UPDATE control card (optional)
4. Update cards if $.UPDATE is specified
5. $.RMTEND end of RMT generation description

Each parameter is coded, beginning in column 1, in the format parameter=value, where:

- parameter represents a valid option specified in the appropriate RTP program options section (see "RMT Parameter Descriptions" later in this chapter).
- value represents a character string of up to 17 characters.

The parameter cannot have embedded blanks. Comments can be included in a parameter statement, but they must be separated from the value by one or more blanks.

RMT generation parameters can appear in any order after the RTP program identification card. If the same parameter occurs more than once in the input deck, the last occurrence determines the parameter value.

RMT Control Cards

The general format for RMT control cards is:

Columns	Field	Description
1-2	$.	Control card identification
3-71	operands	Variable length, separated by a comma and containing no embedded blanks (the last operand must be followed by a blank)
73-80	ignored	

The first card in the RMT generation input deck is a remote terminal processor program identification card. It serves two functions:

1. Selects the appropriate standard options group and source member from SYS1.HASPSRC

2. Sets the remote terminal identification number

The card format is

$.name,n

name
The name of the RTP program to be generated (see Figure A-1).

n
A terminal number, 1-4 digits, that specifies the remote sign-on number (the first digit cannot be 0). This number must be followed by a blank.

There are two additional control cards:

$.UPDATE
The $UPDATE control card sets the update mode and causes the cards following this card to be used to modify the RTP program source module for the current generation description.

$.RMTEND
The $.RMTEND control card is required to signal the end of the RMT generation description.

JES3 RTP Processor For	Terminal Program ID Card (First Card of Each Remote Description)
System/360 Model 20,2922	$.RMTM20,n
System 360(other than Model 20) or System/370	$.RMT360,n
1130 Loader	$,RTPLOAD,n
1130	$.RTP1130,n
System/3	$.RMTSYS3,n
	n = remote sign-on number

Figure A-1. RTP Program Identification Cards

RMT Update Control Cards

The update control cards may be used only during an update run--after a $.UPDATE CARD. The format of an update control card is:

(./)DELETE SEQ1 =serial1 ,SEQ2 =serial2

./are in columns 1 and 2.

One or more blanks must precede and follow the word DELETE. There are no blanks between serial1 and serial2.

serial 1 indicates the starting card serial number.

serial 2 indicates the ending card serial number.

The DELETE card is used to delete one or more source card images from the source code of the described RTP program (see previous section "RMT Control Cards") as the source code is being prepared for the assembler. The DELETE card may be mixed with insertion and replacement update cards containing new source statements for the assembler (see "RMT Update Cards," next). When a DELETE control card is specified, the source card images for the RPT program, starting with the serial number specified in SEQ1 through and including the serial number specified in SEQ2, are omitted from the assembler input source. ENDUP terminates the remote terminal program description. It may be replaced by $.RMTEND, which also serves this function.

RMT Update Cards

Update cards are assembly language source cards in the format described in *OS/VS-DOS/VS-VM/370 Assembler Language*. Each card can have a serial number in columns 73 through 80 or may have blanks in columns 73 through 80. Cards with blank serial numbers will be inserted in the source deck after the last serialized input card or, if following a DELETE control card, in place of the deleted source cards. All serialized input, including DELETE control cards, must have the serial numbers in ascending order.

RMT Parameter Descriptions

This section describes the parameters for each of five types of RMT generations: System/360 (models other than Model 20) and System/370 BSC remote terminal processor program, System/360 Model 20 BSC remote terminal processor program (including the 2922), 1130 remote terminal processor program, 1130 loader program, and System/3 remote terminal processor program.

Refer to the overview at the beginning of this chapter for the devices that are supported by each type of remote work station.

The following conventions are used in this section to describe the RMT parameters:

- The RMT parameters for each RTP program are discussed alphamerically; the first character is ignored if it is & or $.
- Capital letters must be coded as shown.
- Lowercase letters represent variables for which you must substitute specific information or specific values.
- If an item is underlined, it is the default value. This value will be used automatically if the parameter is not specified.

The System/360 Model 20 BSC RTP Program

This section describes the parameters used to specify the machine configuration and programming options required in the assembly of the System/360 Model 20 BSC remote terminal processor program.

&CCT Parameter

&CCT =nn

The &CCT parameter specifies a number (3-31) for all text compression (except trailing blank compression) the minimum number of characters to be compressed. A duplicate character string of fewer than the number specified is treated as a string of nonduplicate characters for compression purposes. If a small value is specified, efficiency of communication line usage is increased at the expense of the compute time that is required for compression. If the &CMPTYPE parameter is specified as 1, this parameter is ignored.

Default: 4

&CMPTYPE Parameter

&CMPTYPE =1/2/3

The &CMPTYPE parameter specifies the type of compression to be applied to all data transmitted from the Model 20 to JES3. Only the value 1, 2, or 3 is valid, where:

1 specifies trailing blank compression.
2 specifies compression of leading, embedded, and trailing blanks.
3 specifies compression of all duplicate character strings.

If this parameter is specified as 1, the &CCT parameter is ignored.

Default: 2

&CORESIZ Parameter

&CORESIZ =nn

The &CORESIZ parameter specifies the size in 1K bytes (8-32) of Model 20 main storage (1K bytes equals 1024 bytes). This parameter must never be greater than the actual storage size of the object machine.

Note: It is possible to specify combinations of parameters such that the resulting work station program is too large for the available storage (&CORESIZ=255). Such a program will fail to load into the object machine.

Default: 8

&ERRMSGN Parameter

&ERRMSGN =nn

The &ERRMSGN parameter specifies the number of 4-byte entries to be assembled in the Model 20 RTP program as an error message log table. This value must be greater than or equal to 8.

Default: 10

&LINESPD Parameter

&LINESPD =nnnn

The &LINESPD parameter specifies the speed, in bits per second, of the communication line to be used between the Model 20 and JES3.

Default: 2000

&MLBFSIZ Parameter

&MLBFSIZ =nnn

The &MLBFSIZ parameter specifies the size, in bytes, of each JES3 MULTI-LEAVING buffer. This value must match the B= parameter value specified on the RJPTERM JES3 initialization statement for this terminal.

Default: 400

&NUMBUFS Parameter

&NUMBUFS =n

The &NUMBUFS parameter specifies the number of teleprocessing buffers to be constructed by the Model 20 RTP program. The specification must be an integer no less than 2X +1, where:

X=1, if either a 2520 or a 2560 is to be used as both a reader and a punch

X=0, if a 2520 or a 2560 is not to be used as both a reader and a punch

The length of each buffer is the value specified in the &MLBFSIZ parameter plus 5 bytes (rounded upward to the next full word). If this parameter specifies more buffers than can be built in available storage, the RTP program will build as many buffers as it can. It is recommended that at least two buffers be provided for each output device and for the communication adapter.

Default: 8

&NUMTANK Parameter

&NUMTANK =n

The &NUMTANK parameter specifies the number of decompression buffers to be assembled in the Model 20 RTP program. This number must be an integer greater than or equal to 2. It is recommended that at least two buffers be provided for each printer and punch. The length of each decompression buffer is the value specified in the &PRTSIZE parameter plus 6.

For an 8K Model 20, specifying this parameter greater than 8 may cause the RTP program to assemble more than X'1F00' bytes (8K-256). If this occurs, the resultant program will fail to load.

Default: 8

&PDEV(1) Parameter

&PDEV(1) =nnnn

The &PDEV(1) parameter specifies the device type for the Model 20 printer. The specification must be either 1403 or 2203.

Default: 2203

&PRTCONS Parameter

&PRTCONS =0/1/2

The &PRTCONS parameter specifies the usage of the printer as an output console. This parameter is dependent upon the specifications (given during JES3 initialization) that pertain to the handling of messages for the remote station.

If JES3 is informed, by means of the RJPTERM statement, that the remote station has a console; &PRTCONS should be specified as one of the following:

0 specifies that error logging and display are suppressed and that operator messages that are created while the remote is online to JES3 are discarded.

1 specifies that the printer is to be used as an output console when sufficient operator messages from JES3 have been queued for output at the remote. If the printer is busy with job stream output, that output is interrupted for the printing of operator messages from JES3 and from the remote error log. When the console queue is empty, job stream output continues.

2 specifies that the printer is to be used as an output console but will not interrupt the printing of jobs. Operator messages received from JES3 while jobs are being printed are discarded.

If JES3 is informed, by means of the RJPTERM initialization statement that the remote station does not have a console, &PRTCONS should be specified as follows:

0 specifies that error logging and display are to be suppressed (JES3 will not return operator messages to the work station).

1 or 2 specifies that error log messages are to be displayed when the printer is free to print them and no job's printed output will be interrupted.

Default: 0

&PRTSIZE Parameter

&PRTSIZE =nnn

The &PRTSIZE parameter specifies the length, in bytes, of the text portion of each decompression buffer. Each buffer must be long enough to hold a maximum-length output record for either the printer, the punch, or the operator console. The specification must be an integer that is the largest of 80, if &UDEV(1) is not 0; 120, if &WDEV(1) is not 0; or the line width of the printer.

Default: 120

&RADR(1) Parameter

&RADR(1) =n

The &RADR parameter specifies the unit address of the Model 20 card reader. The specification must correspond to the specification for the &RDEV(1) parameter as follows:

&RDEV(1)	&RADR(1)
2501	1
2520	2
2560	2

This parameter should not be altered when generating a 2922 work station program.

Default: 1

&RDEV(1) Parameter

&RDEV(1) =nnnn

The &RDEV(1) parameter specifies the device type for the Model 20 card reader. The specification must be either 2501, 2520, or 2560. This parameter should not be altered when generating a 2922 work station program (refer to the &RADR(1) parameter for additional information).

Default: 2501

&SUBMOD Parameter

&SUBMOD =n

The &SUBMOD parameter specifies the submodel number of the Model 20 for the specified remote terminal. The specification must be a valid System/360 Model 20 submodel number. This parameter should not be altered when generating a 2922 work station program.

Default: 2

&UADR(1) Parameter

&UADR(1) =n

The &UADR(1) parameter specifies the unit address of the Model 20 card punch. The specification must correspond to the specification of &UDEV(1) as follows:

&UDEV(1)	&UADR(1)
1442	3
2520	2
2560	2
0	Not present

Default: 3

&UDEV(1) Parameter

&UDEV(1) =nnnn

The &UDEV(1) parameter specifies the device type for the Model 20 card punch. The specification must be either 1442, 2520, 2560, or 0. Specify 0 when the Model 20 does not include a card punch. Specify &UDEV(1)=0 for the 2922, unless the RPQ punch is included, in which case &UDEV(1) should not be altered (refer to the &UADR(1) parameter for additional information).

Default: 1442

&WDEV(1) Parameter

&WDEV(1) =n

The &WDEV(1) parameter specifies the device type for the Model 20 console. The specification must be either 2152, if a console is present; or 0, if a console is not present. If a console is present, console support should be indicated for this remote terminal during JES3 initialization.

Default: 0

&WTOSIZE Parameter

&WTOSIZE =nnn

The &WTOSIZE parameter is an integer less than or equal to 120 that specifies the maximum length, in bytes, of a JES3 operator command to be transmitted from the Model 20 to the central computer. If &WDEV(1) is specified as 0, this parameter is ignored.

Default: 0

&XPARENT Parameter

&XPARENT =NO/YES

The &XPARENT parameter specifies the inclusion or exclusion of support for the text-transparency feature. If the BSC adapters at both the Model 20 and the central computer have the text-transparency feature, the default should be used; otherwise, NO should be specified.

Default: YES

RMT Parameters for the 2922 Remote Work Station RTP Program

This section describes the parameters used to specify the machine configuration and program options required in the assembly of the 2922 remote terminal processor program.

To install a 2922 RTP program, the parameters and procedures for the System/360 Model 20 BSC should be used, subject to the following specific parameter settings:

```
&LINESPD=xxx   where xxx is the actual line speed used
&PDEV(1)=1403
&PRTSIZE=132
&UDEV(1)=0
&WDEV(1)=2152  if the optional typewriter console is installed
&XPARENT=NO    if the optional text transparency feature is not
               installed
```

The default values should be used for the following parameters:

```
&CORESIZ=
&RADR(1)=
&RDEV(1)=
&SUBMOD =
&UADR(1)=
```

The remaining Model 20 BSC parameters may be allowed to default or may be changed.

The System/360 (Except Model 20) and System/370 BSC RTP Program

This section describes the parameters used to specify the machine configuration and program options required in the assembly of the System/360 and System/370 BSC remote terminal processor program.

&ADAPT Parameter

&ADAPT =nnn

The &ADAPT parameter specifies the unit address of the BSC adapter used by the System/360 or System/370 remote terminal to communicate with JES3 at the central computer. The specification must be a valid unit address.

Default: 020

&CCT Parameter

&CCT =nn

The &CCT parameter is an integer from 3 to 31 that specifies, for all text compression (except trailing blank compression), the minimum number of characters to be compressed. A duplicate character string of fewer than the number specified is treated as a string of non-duplicate characters for compression purposes. If a small value is specified, efficiency of communication line usage is increased at the expense of the compute time that is required for compression. If the &CMPTYPE parameter is specified as 1, this parameter is ignored.

Default: 4

&CMPTYPE Parameter

&CMPTYPE =1/2/3

The &CMPTYPE parameter specifies the type of compression to be applied to all data transmitted from the System/360 or System/370 remote terminal to JES3, where:

1 specifies trailing blank compression.
2 specifies compression of leading, embedded, and trailing blanks.
3 specifies compression of all duplicate character strings.

If this parameter is specified as 1, the &CCT parameter is ignored.

Default: 2

&CORESIZ Parameter

&CORESIZ =nn

The &CORESIZ parameter is an integer from 8 to 32 that specifies the size of main storage for the System/360 or System/370 remote terminal in 1K bytes (1K byte equals 1024 bytes). If the System/360 or System/370 remote terminal is larger than 32K bytes, this parameter must be specified as 32.

Default: 8

&ERRMSGN Parameter

&ERRMSGN =nn

The &ERRMSGN parameter is a value greater than or equal to 8 that specifies the number of 4-byte entries to be assembled as an error message log table in the System/360 or System/370 remote terminal.

Default: 10

&LINESPD Parameter

&LINESPD =nnnn

The &LINESPD parameter is an integer that specifies the speed, in bits per second, of the communication line to be used between the System/360 or System/370 remote terminal and JES3.

Default: 2000

&MACHINE Parameter

&MACHINE =nn

The &MACHINE parameter specifies the model number of the System/360 or System/370 to be used as a JES3 remote terminal. The specification must be a valid number for a System/360 or System/370 that includes the standard instruction set and the decimal instruction set.

Default: 30

&MLBFSIZ Parameter

&MLBFSIZ =nnnn

The &MLBFSIZ parameter specifies the size, in bytes, of each JES3 MULTI-LEAVING buffer. This value must match the value of the B= parameter or the RJPTERM statement for this work station.

If this parameter specifies more buffers than can be built in available storage, the RTP program will build as many buffers as it can.

It is recommended that at least two buffers be provided for each output device and for the communication adapter.

Default: 400

&NUMBUFS Parameter

&NUMBUFS =nn

The &NUMBUFS parameter specifies the number of teleprocessing buffers to be constructed by the System/360 or System 370 remote terminal program. The specification must be an integer no less than 2X + 1, where:

X=n, the number of 2520 or 1442 units to be used as both readers and punches

X=0, if neither a 2520 nor a 1442 is to be used as both a reader and a punch

Default: 8

&NUMTANK Parameter

&NUMTANK =nn

The &NUMTANK parameter specifies the number of decompression buffers to be assembled in the System/360 or System/370 RTP program. The specification must be an integer greater than zero and not less than 2 (number of 2540 punches attached). The length of each decompression buffer is the value specified in the &PRTSIZE, plus 6.

It is recommended that at least two decompression buffers be provided for each printer and each punch (three buffers for a 2540 punch).

Default: 5

&PADR(n) Parameter

&PADR(n) =nnn

The &PADR(n) parameter specifies the unit address of each remote terminal printer defined by the &PDEV(n) parameter, where n is a sequential number (1-7) that you code to identify each device being specified. For each &PDEV(n) parameter that is not specified as 0, the corresponding parameter &PADR(n) must specify the device's 3-character hexadecimal unit address. All devices at the remote terminal work station must be on separate nonshared subchannels (that is, all I/O devices must be capable of running simultaneously).

If this parameter is not specified, the following values are used as defaults:

```
&PADR(1)=00E
&PADR(2)=00F
&PADR(3)=FFF
&PADR(4)=FFF
&PADR(5)=FFF
&PADR(6)=FFF
&PADR(7)=FFF
```

&PDEV(n) Parameter

&PDEV(n) =nnnn

The &PDEV(n) parameter specifies the existence and device type of each remote terminal printer. The specification must be either 1403, 1443, 3211, 3203, 5203, or 0. A specification of 0 indicates that the associated printer does not exist. The value of n is a sequential number (1-7) that you code to identify each device being specified.

If this parameter is not specified, the following values are used as defaults:

```
&PDEV(1)=1403
&PDEV(2)=0
&PDEV(3)=0
&PDEV(4)=0
&PDEV(5)=0
&PDEV(6)=0
&PDEV(7)=0
```

&PDEV(1) must not be specified as 0. If &PDEV(n+1) is specified as a device type, &PDEV(n) must be specified as a device type. If PDEV(n) is specified as device type, &UDEV(8-n) must be specified as 0. If more than one printer is specified, more than one printer should also be specified in the RJPTERM statement at JES3 initialization.

&PRTSIZE Parameter

&PRTSIZE =nnn

The &PRTSIZE parameter specifies the length, in bytes, of the text portion of each decompression buffer. Each buffer must be long enough to hold a maximum-length output record for either a printer, a punch, or the operator console. The specification must be an integer that is the larger of 120 or the line width of the widest printer.

Default: 132

&RADR(n) Parameter

&RADR(n) =nnn

The &RADR(n) parameter specifies the unit address of each remote terminal card reader defined by the &RDEV(n) parameter. The value of n is a sequential number (1-7) that you code to identify each device being specified. For each &RDEV(n) parameter that is not specified as 0, a corresponding parameter &RADR(n) must specify the device's 3-character hexadecimal unit address. All devices at the remote terminal work station must be on separate nonshared subchannels (that is, all I/O devices must be capable of running simultaneously.)

If this parameter is not specified, the following values are used as defaults:

```
&RADR(1)=00C
&RADR(2)=FFF
&RADR(3)=FFF
&RADR(4)=FFF
&RADR(5)=FFF
&RADR(6)=FFF
&RADR(7)=FFF
```

&RDEV(n) Parameter

&RDEV(n) =nnnn

The &RDEV(n) parameter specifies the existence and device type of each remote terminal card reader. Each specification must be either 1442, 2501, 2520, 2540, or 0. A specification of 0 indicates that the associated remote terminal card reader does not exist. The value of n is a sequential number (1-7) that you code to identify each device being specified.

If this parameter is not specified, the following values are used as defaults:

```
&RDEV(1)=2540
&RDEV(2)=0
&RDEV(3)=0
&RDEV(4)=0
&RDEV(5)=0
&RDEV(6)=0
&RDEV(7)=0
```

&RDEV(1) must not be specified as 0. If &RDEV(n+1) is specified as a device type, &RDEV(n) must be specified as a device type. If more than one reader is specified, more than one reader should also be specified in the RJPTERM statement at JES3 initialization.

&UADR(n) Parameter

&UADR(n) =nnn

The &UADR(n) parameter specifies the unit address of each remote terminal punch defined by the &UDEV(n) parameter. The value of n is a sequential number (1-7) that you code to identify each device being specified. For each &UDEV(n) parameter that is not specified as 0, the corresponding parameter &UADR(n) must specify the device's 3-character hexadecimal unit address. All devices at the remote terminal work station must be on separate nonshared subchannels (that is, all I/O devices must be capable of running simultaneously).

If this parameter is not specified, the following values are used as defaults:

```
&UADR(1)=00D
&UADR(2)=FFF
&UADR(3)=FFF
&UADR(4)=FFF
&UADR(5)=FFF
&UADR(6)=FFF
&UADR(7)=FFF
```

&UDEV(n) Parameter

&UDEV(n) =nnnn

The &UDEV(n) parameter specifies the existence and device type of each remote terminal punch. The specification must be either 1442, 2520, 2540, or 0. A specification of 0 indicates that the associated punch does not exist. The value of n is a sequential number (1-7) that you code to identify each device being specified.

If this parameter is not specified, the following are used as defaults:

```
&UDEV(1)=2540
&UDEV(2)=0
&UDEV(3)=0
&UDEV(4)=0
&UDEV(5)=0
&UDEV(6)=0
&UDEV(7)=0
```

If &UDEV(n+1) is specified as a device type, &UDEV(n) must be specified as a device type. If &UDEV(n) is specified as a device type, &PDEV(8-n) must be specified as 0. If more than one punch is specified, more than one punch should also be specified in the RJPTERM statement at JES3 initialization.

&WADR(1) Parameter

&WADR(1) =nnn

The &WADR(1) parameter specifies the unit address of the 1052 or 1052-compatible operator console on the System/360 or System/370 remote terminal. The specification must be a 3-character hexadecimal unit address.

Default: 01F

&WTOSIZE Parameter

&WTOSIZE =nnn

The &WTOSIZE parameter is an integer less than or equal to 120 that specifies the maximum length, in bytes, of a JES3 operator command to be transmitted from the System/360 or System/370 remote terminal to JES3.

Default: 120

&XPARENT Parameter

&XPARENT =NO/YES

The &XPARENT parameter specifies the inclusion or exclusion of support for the text transparency feature. If the BSC adapters at both the System/360 or System/370 remote terminal and the central computer have the text-transparency feature, the default value should be used; otherwise, NO should be specified.

Default: YES

The 1130 RTP Program

This section describes the parameters used to specify the machine configuration and program options required in the assembly of the 1130 remote terminal processor program.

&CLOCK Parameter

&CLOCK =0/1

The &CLOCK parameter specifies the type of communication adapter clocking available on the 1130.

0 specifies that data set clocking is being used
1 specifies internal 1130 clocking

The rate of insertion of the synchronous idle sequence in the transmitted data is determined by the &CLOCK, &LINESPD, and &TRANPRN parameters. The relationship of these parameters to the insertion rate is:

&CLOCK	&TRANPRN	Insertion Rate
0	0	Every &LINESPD/8 characters
0	1	Every &LINESPD/8 characters
1	0	Every 70 characters
1	1	Every &LINESPD/8 characters

The equation used for the insertion rate is (&LINESPD/8) T where T is 1.00 second, which is the nominal 2701 timer value.

Default: 0

&CMPTYPE Parameter

&CMPTYPE =0/1/2

The &CMPTYPE parameter specifies the type of compression to be applied to all data transmitted to JES3, where:

0 specifies no compression of duplicate characters and no truncation of trailing blanks.
1 specifies trailing blank truncation.
2 specifies full compression, trailing blank truncation, and encoding of duplicate characters.

The process of compressing input data offers optimum performance with respect to efficient line utilization. However, factors such as line speed, processor availability, buffer size, line turnaround time, and nature of the data to be compressed contribute to the overall operation of the RTP program. Since compression and truncation require considerable processor time, you may decide, on the basis of the other variables, to respecify the compression technique.

Default: 2

&DELAY Parameter

&DELAY =nn

The &DELAY parameter specifies the number of time intervals that the RTP program will delay in transmitting a "handshaking" sequence (DLE-ACK0) to the central computer. The machine program timer clock is used to measure the delay and is assumed to be set to a minimal value of 0.35 seconds. The purpose of the delay when "handshaking" is to minimize processing at the central computer when no data is being transmitted. Using the default value results in a delay of 1.05 seconds, assuming a timer interval of 0.35 seconds.

The value of this parameter must not be set to such a large increment that the delay will be greater than the time-out period of the central computer's 2701/2703.

Default: 3

&FULLIST Parameter

&FULLIST =0/1

The &FULLIST parameter specifies the type of assembly listing produced by the OS/VS assembler during RMT generation, where:

0 specifies that the assembly listing is to be produced according to the PRINT NOGEN stipulation of the assembler.

1 specifies that the listing is to be produced according to the PRINT GEN stipulation.

Because most of the code in the RTP1130 and RTPLOAD programs is created by macro instructions, the specification of 0 essentially produces a source listing (cross-referenced) without the 1130 assembled instructions.

Default: 1

&LINESPD Parameter

&LINESPD =nnnn

The &LINESPD parameter is an integer that specifies the speed, in bits per second, of the communication line to be used between the 1130 and the central computer. The value should correspond to the selected setting of the baud rate switch on the 1130 SCA control panel: 1200, 2000,.... The rate of insertion of the synchronous idle sequence (DLE-SYN or SYN-SYN) in the transmitted data is determined by the &CLOCK, &LINESPD, and &TRANPRN parameters (refer to the &CLOCK parameter for the relationship of these parameters).

Default: 2000

&MACHSIZ Parameter

&MACHSIZ =nnnn

The &MACHSIZ parameter specifies the amount of 1130 storage to be used by the RTP program. The value is specified in 1130 words. The value specifies the number of words, starting at location 0, that are available to the RTP programs (RTPBOOT, RTPLOAD, and RTP1130). The value specified may be less than the actual available storage but must not be greater. The same parameter must be defined for the assembly of the 1130 loader program and should have the same value.

Default: 8192

&MLBFSIZ Parameter

&MLBFSIZ =nnnn

The &MLBFSIZ parameter specifies the size, in bytes, of each JES3 MULTI-LEAVING buffer. This value must match the B= parameter value specified on the RJPTERM JES3 initialization statement for this work station.

Default: 400

&PN1442 Parameter

&PN1442 =0/1

The &PN1442 parameter specifies the inclusion or exclusion of support for the 1442 punch in the RTP program, where:

0 specifies that support is not to be included.

1 specifies that support for punched card output produced by jobs at the central computer is to be included.

Refer to the &RD1422 parameter for information about the reader function of the 1442.

Default: 1

&PRFOTLW Parameter

&PRFOTLW =120/132

The &PRFOTLW parameter specifies the line width of the 1403 printer. The specification of the line width for all printers on a remote terminal is a JES3 installation requirement.

Default: 120

&PR1132 Parameter

&PR1132 =0/1

The &PR1132 parameter specifies the inclusion or exclusion of the support for the 1132 printer in the RTP program, where:

0 specifies that support is not to be included.
1 specifies that support for printing job output using the 1132 is to be included.

Default: 0

&PR1403 Parameter

&PR1403 =0/1

The &PR1403 parameter specifies the inclusion or exclusion of the support for the 1403 printer in the RTP program, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

Default: 1

&RD1442 Parameter

&RD1442 =0/1

The &RD1442 parameter specifies the inclusion or exclusion of the support for a 1442 card reader in the RPT program, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

Default: 1

&RD2501 Parameter

&RD2501 =0/1

The &RD2501 parameter specifies the inclusion or exclusion of the support for the 2501 card reader in the RTP program, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

Default: 0

&RTPLORG Parameter

&RTPLORG =nnnnn

The &RTPLORG parameter defines the location in 1130 storage of the RTPLOAD program, which is used to load the 1130 RTP program. The RTPLOAD program must reside in an area of storage that is available between the beginning of the buffer pool and the end of main storage (as defined in the &MACHSIZ parameter), minus the length of the RTPLOAD program. The default value of this parameter allows 1024 words for the RTPLOAD program.

Assuming &MACHSIZ=8192, the default value is 14336. This value is twice the actual 1130 storage address because the value is used in an ORG operation and must be in terms of bytes, not 1130 words.

Default: 2 times the value of &MACHSIZ minus 1024

&TRANPRN Parameter

&TRANPRN =0/1

The &TRANPRN parameter specifies the simulation of the binary synchronous transparency feature, where:

0 specifies that no simulation will occur. In this case, data containing transparent characters cannot be properly processed by the RTP program.

1 specifies that simulation will occur in the same manner as the 2701 SDA-II adapter that is equipped with the transparency feature.

If 0 is specified, the conversion of card code data is monitored and all BSC control characters are converted to hexadecimal 0. This prevents mispunched data from causing an infinite error retry if the central computer does not have transparency. If 1 is specified, the RTP program will communicate only with a 2703 or with a 2701 adapter that has the text transparency feature.

Default: 1

The 1130 Loader Program

This section describes the parameters used to specify the machine size, loader origin, and assembly list option that are used in the assembly of RTPLOAD, the 1130 loader program that loads the 1130 remote terminal processor program.

RMT generation produces the object decks for the RTPLOAD and RTP 1130 programs. The bootstrap loader program (RTPBOOT) cannot be produced on System/360 or System/370 and must be keypunched as indicated in Appendix B, "Remote Terminal Bootstrap (RTPBOOT)."

&FULLIST Parameter

&FULLIST =0/1

The &FULLIST parameter specifies the type of assembly listing produced by the OS/VS assembler during the RMT generation, where:

0 specifies that the assembly listing is to be produced according to the PRINT NOGEN stipulation of the assembler.

1 specifies that the listing is to be produced according to the PRINT GEN stipulation.

Since most of the code in the RTP1130 and RTPLOAD programs is created by macro instructions, the specification of 0 essentially produces a source listing (cross-referenced) without the 1130 assembled instructions.

Default: 1

&MACHSIZ Parameter

&MACHSIZ =nnnnn

The &MACHSIZ parameter specifies the amount of 1130 main storage to be used by the RTP program. The value is specified in 1130 words. The value specifies the number of words, starting at location 0, that are available to the RTP programs (RTPBOOT, RTPLOAD, and RTP1130). The value specified may be less than the actual available storage but must not be greater. The same parameter must be defined for the assembly of the 1130 RTP program and should have the same value.

Default: 8192

&RTPLORG Parameter

&RTPLORG =nnnnn

The &RTPLORG parameter defines the location in 1130 main storage of the RTPLOAD program, which is used to load the 1130 RTP program. The RTPLOAD program must reside in an area of storage that is available between the beginning of the buffer pool and the end of main storage (as defined in the &MACHSIZ parameter), minus the length of the RTPLOAD program. The default value of this parameter allows 1024 words for the RTPLOAD program.

Assuming &MACHSIZ=8192, the default value is 14336. This value is twice the actual 1130 storage address because the value is used in an ORG operation and must be in terms of bytes, not 1130 words.

Default: 2 times the value of &MACHSIZ minus 1024

The System/3 RTP Program

This section describes the parameters used to specify the machine configuration and programming options in the assembly of the System/3 remote terminal processor program.

&COMP Parameter

&COMP =0/1/2

The &COMP parameter specifies the type of compression to be applied to all data transmitted to the central computer, where:

0 specifies that no compression of duplicate characters and no truncation of trailing blanks is performed.

1 specifies that trailing blanks are truncated.

2 specifies that compression takes place after truncation. Strings of from 2 to 31 blanks are compressed into a single byte; strings of from 3 to 31 duplicate characters are compressed into 2 bytes.

Default: 2

&DEBUG Parameter

&DEBUG =0/1

The &DEBUG parameter specifies the inclusion or exclusion of certain validity tests and a main storage dump program in the RTP program, where:

0 specifies that support is not to be included.

1 specifies that support is to be included.

Default: 0

&DIAL and &DIAL1 Parameters

&DIAL =nnnn
&DIAL1 =nnnn

The &DIAL and &DIAL1 parameters specify the telephone number to be used during JES3 initialization. The values specified will be included on the /*SIGNON statement that is assembled into the RTP program and will be preceded by the keyword DIAL (unless the default values are used). Each specification is a string of up to 8 decimal digits. If the telephone number has 8 or fewer digits, it should be specified in the &DIAL parameter. If the telephone number has more than 8 digits, the leftmost 8 digits are specified in the &DIAL parameter and the remaining digits are specified in the &DIAL1 parameter.

Default: Null string

&MACHSIZ Parameter

&MACHSIZ =nnnnn

The &MACHSIZ parameter specifies the size of System/3 main storage. The value specified must be the appropriate specification for the System/3 storage size, specified as follows:

Value	Main Storage Size
8192	8K
12288	12K
16384	16K
24576	24K
32768	32K

Default: 8192

&MLBFSIZ Parameter

&MLBFSIZ =nnnn

The &MLBFSIZ parameter specifies the size, in bytes, of each JES3 MULTI-LEAVING buffer. This value must match the B= parameter value specified on the RJPTERM JES3 initialization statement for this work station.

Default: 400

&PASSWD Parameter

&PASSWD =XXXXXXXX

The &PASSWD parameter specifies the password that is to be used during the sign-on process. The value specified will be included in the /*SIGNON statement that is assembled into the System/3 RTP program. The specification must be a character string of from 1 to 8 characters. If you want blanks, let the parameter default.

Default: Null string.

&PC(n) Parameter

&PC(n) =nn

The &PC(n) parameter specifies skip information on the 5203 or 1403 printer. The value of n is a number from 1 to 12 that specifies the channel. The value specified in this parameter determines the print line number to which paper will be skipped when the RTP program simulates the 1403 command "Skip to channel n." A specification of 0 causes no forms movement.

Default: If this parameter is not specified, the following values are used as defaults.

```
&PC(1)=1
&PC(2)=0
&PC(3)=0
&PC(4)=0
&PC(5)=0
&PC(6)=0
&PC(7)=0
&PC(8)=0
&PC(9)=0
&PC(10)=0
&PC(11)=0
&PC(12)=&S3FORML-5
```

&PRTCONS Parameter

&PRTCONS =0/1/2

The &PRTCONS parameter specifies utilization of the 5203 or 1403 printer as an operator's console, where:

0 specifies that the printer is not to be used as an operator's output console.

1 specifies that the RTP program will attempt to hold operator messages from JES3 until a job has completed printing. However, if two or more MULTI-LEAVING buffers contain JES3 operator messages, the printer ejects a page (skip to channel 1), prints the operator messages, ejects another page, and resumes printing the job.

2 specifies that the RTP program is to throw away all operator messages while the printer is printing a job. While the printer is dormant, it prints received messages.

Regardless of the setting of this parameter, messages temporarily saved on a direct-access volume for a remote terminal are printed to the terminal as a job. Thus, they will always appear on the printer, even if another console exists. If this parameter is specified greater than 0, MULTI-LEAVING console support should be specified in the RJPTERM statement for this work station at JES3 initialization. If &S3547 = 1, the value of &PRTCONS is ignored and assumed to be 0.

Default:. 2

&S3BSCA Parameter

&S3BSCA =1/2

The &S3BSCA parameter specifies the number of the System/3 BSC adapter (BSCA) to be used for RJE communication, where:

1 specifies the first BSCA.
2 specifies the second BSCA.

The assembled System/3 RTP program uses only the adapter specified in this parameter.

Default: 1

&S3CMDS Parameter

&S3CMDS =0/1

The &S3CMDS parameter specifies the inclusion or exclusion of a command facility and of commands to assist the System/3 operator, where:

0 specifies support is not to be included.
1 specifies that support is to be included.

The commands that are available with this facility are detailed in *Operator's Library: OS/VS2 Remote Terminals.*

Default: 0

&S3FORML Parameter

&S3FORML =nnn

The &S3FORML parameter is an integer greater than or equal to 6 that specifies the number of print lines on a page for the continuous forms used on the 5203 or 1403 printer.

Default: 66

&S3NPUNS Parameter

&S3NPUNS =1/2/3

The &S3NPUNS parameter specifies the maximum number of jobs that can be punched simultaneously at the System/3 remote terminal. A value of 3 allows simultaneous operation of both 5424 hoppers and the 1442 hopper as punches. If this parameter is specified greater than 1, additional punches should also be specified in the RJPTERM statement for this work station parameter at JES3 initialization.

Default: 1

&S3NRDRS Parameter

&S3NRDRS =1/2/3

The &S3NRDRS parameter specifies the maximum number of jobs that can read simultaneously from the System/3 remote terminal. A value of 3 allows simultaneous operation of both 5424 and 1442 hoppers as card readers. If this parameter is specified greater than 1, additional card readers should also be specified in the RJPTERM statement for this work station at JES3 initialization.

Default: 1

&S3OBJDK Parameter

&S3OBJDK =0/1

The &S3OBJDK parameter specifies the inclusion or exclusion of the facility to punch OS/VS2 object decks, where:

0 specifies support is not to be included.
1 specifies support is to be included.

If this facility is to be included, the text-transparency feature should be present. If 1 is specified, each card of an OS/VS2 object deck is expanded and punched into two 96-column cards. These cards are recognized when read by the System/3 RTP program. For each two 96-column cards read, one OS/VS2 object deck card image is transmitted.

Default: 0

&S3SIP Parameter

&S3SIP =0/1

The &S3SIP parameter specifies usage of those bytes of System/3 main storage between X'100' and X'1FF', where:

0 specifies that the RTP program is to use the bytes.

1 specifies that the bytes are to be used by the System/3 card system initialization program.

Default: 0

&S3TRACE Parameter

&S3TRACE =nn

The &S3TRACE parameter is an integer greater than 1 that specifies the number of 4-byte entries in the RTP program's internal error message table.

Default: 10

&S3XPAR Parameter

&S3XPAR =0/1

The &S3XPAR parameter specifies the presence or absence of the EBCDIC text-transparency feature, where:

0 specifies that the EBCDIC text-transparency feature is not present.

1 specifies that both the central computer's communications adapter and the System/3 BSCA have the EBCDIC text-transparency feature.

Default: 0

&S31442 Parameter

&S31442 =0/1

The &S31442 parameter specifies inclusion or exclusion of support for the 1442 card read punch, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

Default: 0

&S35424 Parameter

&S35424 =0/1

The &S35424 parameter specifies inclusion or exclusion of support for the 5424 Multi-function Card Unit, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

If this parameter is specified as 0, &S31442 must be specified as 1.

Default: 1

&S35471 Parameter

&S35471 =0/1

The &S35471 parameter specifies inclusion or exclusion of support for the 5471 printer-keyboard for use as an operator's input/output console, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

If console support is desired, MULTI-LEAVING console support must be specified in the RJPTERM statement for this work station at JES3 initialization. Regardless of the specification of this parameter, messages from JES3 can print on the printer. Refer to the &PRTCONS parameter in this section for additional information.

Default: 0

&S35475 Parameter

&S35475 =0/1

The &S35475 parameter specifies inclusion or exclusion of support for the 5475 data entry keyboard on the System/3 for use as an operator's console, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

If console support is desired, MULTI-LEAVING console support must be specified in the RJPTERM statement for this work station at JES3 initialization. If &S35471 = 1 is specified, this parameter is ignored.

Default: 0

&S396COL Parameter

&S396COL =0/1

The &S396COL parameter specifies inclusion or exclusion of support for the load-mode punch option, where:

0 specifies that support is not to be included.
1 specifies that support is to be included.

If this parameter is specified, the resultant System/3 RTP program will be capable of receiving the punched output of a System/3 RMT generation.

Default: 0

Output

The output from an RMT generation is a card deck for each RMT program generated. In addition, the GENRMT utility prints an information listing, the RMT parameter default values, and the parameter values you specified. Also, a listing of each assembly is produced.

All listings produced by the GENRMT utility and the assembler have the remote terminal sign-on identification number at the top of each page. With the exception of loader bootstrap cards, all object deck cards have the identification number punched in columns 74 through 76.

System/3 96-Column Card Output

The REMOTGEN utility produces the System/3 object-deck image on the SYSPUNCH data set. The cards created are 80-column cards which, if routed (by use of a /*ROUTE card to a System/3 remote terminal utilizing the System/3 starter system, are punched as 96-column System/3 load mode cards. They may also be punched locally or remotely as 80-column cards (with the punched output of other RMT generations) and later be separated and routed to a System/3 starter system, as the punched output of an 80/80 card-to-punch job. The utilities IEBPTPCH or IEBGENER may be used for this. (Refer to *Operator's Library: OS/VS2 Remote Terminals* for a description of the System/3 starter program and to *OS/VS Utilities* for a description of the utility programs.)

The 96-column System/3 load mode cards must be punched in order to use the output of an RMT generation on a System/3 if the System/3 configuration includes a 5424 Multi-function Card Unit and the RMT parameter &S35424 was specified as 1. The 80-column cards are loadable on a System/3 only if a 1422 card reader is attached and the RMT parameter &S35424 was specified as 0.

Instead of the System/3 starter system, any JES3 System/3 remote terminal processor program generated with &S396COL-1 specified may be used to punch RMT generation output that is routed to a System/3.

Appendix B. Remote Terminal Bootstrap (RTPBOOT)

The bootstrap loader distributed in object form as shown in the subsequent pages is specifically constructed to bootstrap the EBCDIC main loader (RTPLOAD) into the storage locations defined by &RTPLORG at RMTGEN time. RTPBOOT loads into lower 1130 storage via the load-mode format first card and following binary program cards and EBCDIC conversion table cards. RTPBOOT loads from a 1442 or 2501 Card Reader which is wired for the load-mode sequence initiated by the console LOAD button.

Remote Terminal Bootstrap Card Format

Card Col.	Card No. 1	Card No. 2
1	12-11-7	12
2	1-2-9	11-0-3-5
3	12-11-1-8	11
4	12-11-7-8-9	blank
5	11-0-1-6-9	5
6	0-2-6	11-0-1-5
7	4-7-8-9	12-11-0-1-2-4-5
8	blank	12-11
9	4-6	blanks
10	0-1-2	12-11-1-5
11	blank	5
12	11-2-5	1-2
13	4-5-9	12-11-0-1-4-5
14	12-0-1-2-5-6	12-11-1
15	1-2-8	blank
16	12-11-1-3-4-5-6-8-9	12-11-3
17	12-11-3-4-5-6-7	5
18	1-2-8-9	blank
19	12-11-1-3-4-5-6-8	12-11-0-1-2-3-4-5
20	12-3-4-5-7-9	11-0-1-3
21	12-11-1-3-4-5-7	11-2-4-5
22	12-11-4-7	blank
23	1-6	12-11-3-4-5
24	12-11-1-4-8	11-0-1
25	12-4-7-8	2-3-4-5
26	12-11-1-4-7-9	11-0-1-3
27	12-4-8	11-2-4
28	12-11-1-4-9	blank
29	12-11-3-4-6-9	12-11-3
30	1-6-9	11-0-1
31	12-11-1-3-4-6	3-5
32	1-2-6	11-0-5
33	12-11-1-4-6-7-9	3-4-5
34	12-11-1-5-6-7-8	11-0-1-3
35	12-11-1-5-6-8-9	11-2-4
36	12-11-1-3-4-8	blank
37	12-11-1-3-4-7-9	blank
38	2-3-5-6-7-8	11-0-3-4
39	2-3-5-6-7-8-9	11-0-2-4-5
40	11-0-1-3-4-5-6-7-8-9	4
41	9	1-3
42	2-3-4-8	11-0-1-3
43	12-11-3-5-6-7-8-9	2-3
44	12-8-9	blank
45	12-11-1-3-5-6-7-9	12-0-1

Card Col.	Card No. 1	Card No. 2
46	2-3-5-6-7	11-0-1-4
47	11-2-3-4-5-6	12-0-4-5
48	9	11-0-2-4
49	11-0-1-3-4-5-6-7-8-9	12-1-2-3-4
50	9	11-0-2-4
51	12-11-6-7-9	11-2-3-4-5
52	12-3-4-5-6-8-9	12-11
53	12-11-1-6-8-9	blank
54	12-11-6	12-11-1-4
55	12-3-4-5-6	12-11-0-1-2-3-4-5
56	12-11-1-8-9	11-0-1-5
57	12-3-4-5-7-8	12-0-1-2-5
58	12-11-1-7	11-0-1
59	3-7	1-2-4-5
60	blank	11-0-1-3
61	1-2-6	11-2-4
62	1-2	blank
63	blank	12-0-1-3-4
64	1	11-0-1
65	blank	blank
66	12-11-7-8-9	11-0-3-5
67	12-1-3-4-6-7	12-11-1-2-3-5
68	12-11-1-7-9	blank
69	11-2-4	11-3-4
70	11-0-1-3-4-6-7	11
71	2-3-7-9	blank
72	2-3	12-11-1-5
73	11-2-3-4-5-6	12
74	4-7-8-9	11-0-3-4
75	11-0-1-7	12-11-1-2-3-5
76	8	blank
77	blank	12
78	blank	11-0-3-4-5
79	0	0
80	1	2

Card Col.	Card No. 3	Card No. 4
1	12-11-1-2-3-4-5	12-11-1
2	blank	12
3	5	12-11-2-3-4-5
4	11-0-1-5	12-11-1
5	4	5
6	11-0-1-5	11-0-1-5
7	0-1-2-3-4	12-11-0-2-3-4-5
8	11	11-0-1
9	blank	blank
10	12-11-1-4	11-0-3-5
11	5	0-3
12	11-0-1-4	5
13	12-11-0-1-2-3-4-5	blank
14	11-0-1-4-5	12-1-5
15	12-11-0-1-2-4	5
16	11-0-1	12-1-5
17	12-3-4-5	12-11-2
18	12-11	12-11-1
19	12-0-3	1-5
20	11-0-1	11
21	blank	12-11-3-4
22	12-11-3	12-11-0-1
23	12-11-1-2-3	1-2
24	blank	11-2-3
25	blank	11-0-2-3-4-5
26	11-0-3-4-5	blank
27	2-3-4-5	12-11-0-1-2-3-4-5
28	4	11-0-1
29	0-2-4	5
30	11	11-0-1-4-5
31	5	12-11-0-1-2-3-4-5
32	11-0-1	11-0-1-4
33	3-4	12-11-0-2
34	11-0-1	11-0-1
35	blank	12-11-0-1-2-3-4-5
36	11-0-3-4-5	11-0-1
37	12-11-0-1-5	5
38	5	blank
39	0-3-5	blank
40	11	12-11-0-1-4-5
41	12-11-0-1-4-5	0
42	11-0-1	11-2-3
43	11-2-3-4-5	12-0-1-3-5
44	11-0-1-3	blank
45	12-0-2-5	5
46	blank	11-0-1-3
47	1	12-0-2-3-4-5
48	1-2	blank
49	12-11-1-2-3-4-5	blank
50	12-11-1	12-11-0-1-4-5

Card Col.	Card No. 3	Card No. 4
51	12-0-1-3-4	12
52	11-0-5	11-2-3
53	blank	12-0-2-3-4-5
54	12-11-3-5	blank
55	0-3-4	11-1
56	5	blank
57	blank	blank
58	11-0-3-4-5	12-11-5
59	0-2-3	2
60	5	1
61	blank	5
62	11-0-3-4	12-11-0-2-5
63	blank	12
64	5	11-2-3
65	1-2	12-0-1-2
66	11	blank
67	1-4-5	blank
68	11-0-1	11-0-3-5
69	blank	12-11-1-2-3-5
70	12-11-3	blank
71	4-5	12-11-0-1-3-4-5
72	blank	11
73	12-11-0-1-2	5
74	12-1	12-11-1
75	blank	blank
76	12-11-1-3-5	11-2-3
77	0-3-4	blank
78	5	blank
79	0	0
80	3	4

Card Col.	Card No. 5	Card No. 6
1	0	11-0-1-8
2	1	11-0-1
3	2	11-0-2
4	3	11-0-3
5	4	11-0-4
6	5	11-0-5
7	6	11-0-6
8	7	11-0-7
9	8	11-0-8
10	9	11-0-9
11	12-11-0-2-8-9	11-0-2-8
12	12-11-0-3-8-9	11-0-3-8
13	12-11-0-4-8-9	11-0-4-8
14	12-11-0-5-8-9	11-0-5-8
15	12-11-0-6-8-9	11-0-6-8
16	12-11-0-7-8-9	11-0-7-8
17	.	12-11-0-1-8
18	.	12-11-0-1
19	.	12-11-0-2
20	.	12-11-0-3
21	.	12-11-0-4
22	.	12-11-0-5
23	.	12-11-0-6
24	.	12-11-0-7
25	.	12-11-0-8
26	.	12-11-0-9
27	.	12-11-0-2-8
28	.	12-11-0-3-8
29	.	12-11-0-4-8
30	.	12-11-0-5-8
31	.	12-11-0-6-8
32	.	12-11-0-7-8
33	.	12-0
34	.	12-1
35	.	12-2
36	.	12-3
37	.	12-4
38	.	12-5
39	.	12-6
40	.	12-7
41	.	12-8
42	.	12-9
43	.	12-0-2-8-9
44	.	12-0-3-8-9
45	.	12-0-4-8-9
46	.	12-0-5-8-9
47	.	12-0-6-8-9
48	.	12-0-7-8-9
49	.	11-0
50	.	11-1

Card Col.	Card No. 5	Card No. 6
51	.	11-2
52	.	11-3
53	.	11-4
54	.	11-5
55	.	11-6
56	.	11-7
57	.	11-8
58	.	11-9
59	.	12-11-2-8-9
60	.	12-11-3-8-9
61	.	12-11-4-8-9
62	.	12-11-5-8-9
63	.	12-11-6-8-9
64	.	12-11-7-8-9
65	.	0-2-8
66	.	11-0-1-9
67	.	0-2
68	.	0-3
69	.	0-4
70	.	0-5
71	.	0-6
72	.	0-7
73	.	0-8
74	.	0-9
75	.	11-0-2-8-9
76	.	11-0-3-8-9
77	.	11-0-4-8-9
78	.	11-0-5-8-9
79	.	11-0-6-8-9
80	blank	11-0-7-8-9

Card Col.	Card No. 7	Card No. 8
1	12	12-0-1-8-9
2	12-11-1-9	12-1-9
3	12-11-2-9	12-2-9
4	12-11-3-9	12-3-9
5	12-11-4-9	12-4-9
6	12-11-5-9	12-5-9
7	12-11-6-9	12-6-9
8	12-11-7-9	12-7-9
9	12-11-8-9	12-8-9
10	11-1-8	12-1-8-9
11	11-2-8	12-2-8-9
12	11-3-8	12-3-8-9
13	11-4-8	12-4-8-9
14	11-5-8	12-5-8-9
15	11-6-8	12-6-8-9
16	11-7-8	12-7-8-9
17	11	12-11-1-8-9
18	0-1	11-1-9
19	11-0-2-9	11-2-9
20	11-0-3-9	11-3-9
21	11-0-4-9	11-4-9
22	11-0-5-9	11-5-9
23	11-0-6-9	11-6-9
24	11-0-7-9	11-7-9
25	11-0-8-9	11-8-9
26	0-1-8	11-1-8-9
27	12-11	11-2-8-9
28	0-3-8	11-3-8-9
29	0-4-8	11-4-8-9
30	0-5-8	11-5-8-9
31	0-6-8	11-6-8-9
32	0-7-8	11-7-8-9
33	12-11-0	11-0-1-8-9
34	12-11-0-1-9	0-1-9
35	12-11-0-2-9	0-2-9
36	12-11-0-3-9	0-3-9
37	12-11-0-4-9	0-4-9
38	12-11-0-5-9	0-5-9
39	12-11-0-6-9	0-6-9
40	12-11-0-7-9	0-7-9
41	12-11-0-8-9	0-8-9
42	1-8	0-1-8-9
43	2-8	0-2-8-9
44	3-8	0-3-8-9
45	4-8	0-4-8-9
46	5-8	0-5-8-9
47	6-8	0-6-8-9
48	7-8	0-7-8-9
49	12-0-1-8	12-11-0-1-8-9
50	12-0-1	1-9

Card Col.	Card No. 7	Card No. 8
51	12-0-2	2-9
52	12-0-3	3-9
53	12-0-4	4-9
54	12-0-5	5-9
55	12-0-6	6-9
56	12-0-7	7-9
57	12-0-8	8-9
58	12-0-9	1-8-9
59	12-0-2-8	2-8-9
60	12-0-3-8	3-8-9
61	12-0-4-8	4-8-9
62	12-0-5-8	5-8-9
63	12-0-6-8	6-8-9
64	12-0-7-8	7-8-9
65	12-11-1-8	blank
66	12-11-1	12-0-1-9
67	12-11-2	12-0-2-9
68	12-11-3	12-0-3-9
69	12-11-4	12-0-4-9
70	12-11-5	12-0-5-9
71	12-11-6	12-0-6-9
72	12-11-7	12-0-7-9
73	12-11-8	12-0-8-9
74	12-11-9	12-1-8
75	12-11-2-8	12-2-8
76	12-11-3-8	12-3-8
77	12-11-4-8	12-4-8
78	12-11-5-8	12-5-8
79	12-11-6-8	12-6-8
80	12-11-7-8	12-7-8

Index

C

E

MVS/Extended Architecture
System Programming
Library: JES3
Initialization and Tuning

READER'S
COMMENT
FORM

SC23-0059-5

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning

SC23-0059-5 S370-36

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning

SC23-0059-5

S370-36

SC23-0059-05