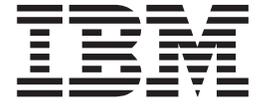
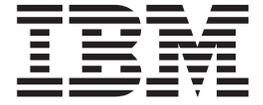


z/OS



JES3 Initialization and Tuning Guide

z/OS



JES3 Initialization and Tuning Guide

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 349.

Third Edition, September 2002

This is a major revision of SA22-7549-01.

This edition applies to Version 1 Release 4 of z/OS (5694-A01), z/OS.e Version 1 Release 4 (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order documents through your IBM® representative or the IBM branch office serving your locality. Documents are not stocked at the address below.

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

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

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

FAX (Other Countries):

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

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

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

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

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

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

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

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

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

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

Contents

Figures	vii
--------------------------	------------

About This Document	ix
Who Should Use This Document	ix
How to Use This Document	ix
Where to Find More Information	ix
Accessing z/OS™ licensed documents on the Internet	x
Using LookAt to look up message explanations	xi
Additional Information	xi

Summary of Changes	xiii
-------------------------------------	-------------

Part 1. Introduction **1**

Chapter 1. Introduction **3**

Developing Your Installation Plan	3
Preparing for JES3	4
Installing JES3	4
Running the Hardware Configuration Definition Program	4
Initializing MVS	4

Part 2. Initializing JES3 **5**

Chapter 2. Initializing JES3 **7**

Modifying the JES3 Cataloged Start Procedure	7
Modifying or Creating a JES3 Initialization Stream	11
Organizing the Initialization Stream	12
Using a Segmented Initialization Stream	13
Testing Your Initialization Stream	15
How to Run Step 1	16
How to Run Step 2	17
Storage Requirements for the Initialization Stream Checker	18
Abnormal Termination of the Initialization Stream Checker	18
Starting JES3	19
Hot Start	19
Hot Start with Analysis	20
Hot Start with Refresh	20
Hot Start with Refresh and Analysis	22
Warm Start	22
Warm Start with Analysis	23
Warm Start to Replace a Spool Data Set	23
Warm Start with Analysis to Replace a Spool Data Set	23
Cold Start	24
Local Start	25
Tuning JES3	26
Using JMF to Measure JES3 Performance	26
Maintaining JES3	27
JES3 Maintenance Philosophy	27

Dynamically Changing the JES3 Configuration	31
---	----

Part 3. Providing Security for JES3 **37**

Chapter 3. Providing Security for JES3 **39**

Planning for Security	39
Using RACF to Provide Security	40
How JES and RACF Work Together	43
Defining JES in the RACF Started Procedures Table	43
Forcing Batch Users to Identify Themselves to RACF	43
Defining and Grouping Operators	44
Understanding How to Provide Access to Resources	44
Understanding Security Labels	45
Security Labels for JES Resources	46
JES3 Installation Exits that Affect RACF-Provided Security	46
Giving JES3 Access to Resources	47
JES User ID Early Verification	47
User ID Propagation When Jobs Are Submitted	47
Control of User ID Propagation	48
Where NJE Jobs are Verified	48
How SYSOUT Requests are Verified	49
Controlling Access to Data Sets JES Uses	50
Controlling Input to Your System	50
How RACF Validates Users	50
Controlling the Use of Job Names	51
Authorizing the Use of Input Sources	55
Authorizing Network Jobs and SYSOUT (NJE)	56
Authorizing Inbound Work	56
Authorizing Outbound Work	70
Controlling Access to Data That Resides on Spool	72
Protecting SYSIN and SYSOUT	72
Protecting JESNEWS	75
How RACF Affects Jobs Dumped from and Restored to Spool (JES3 Only)	76
Authorizing Console Access	76
MCS Consoles	76
Remote Workstations (RJP/RJE Consoles)	76
Controlling Where Output Can Be Processed	79
Authorizing the Use of Your Installation's Printers	80
Using RACF to Authorize the Use of Operator Commands	81
Profile (Entity) Names for JES3 Commands	82
Who Authorizes Commands When RACF Is Active	84
Using JES3 to Provide Security	85
JES3 Initialization Statements that Affect Security	86
JES3 Installation Exits that Affect JES3-Provided Security	88
Using JES3 to Control Access to RJP Workstations	88
Using JES3 to Authorize the Use of Operator Commands	89

Part 4. JES3 Job Management . . . 93

Chapter 4. JES3 Job Management . . . 95

Input Service	95
Reader Phase	95
Control Statement Processing Phase	95
Converter/Interpreter Service	99
Converter/Interpreter Phase	99
Prescan Phase	100
Postscan Phase	100
JES3 Resource Allocation	101
MVS Allocation Compared with JES3 MDS Allocation	102
Types of Setup	106
Initializing MDS	108
Operator Control of MDS	111
Job Selection and Scheduling	111
Comparison of JES3 and WLM Initiator Management	111
Classifying Jobs	116
Pre-Execution Delays	117
Defining Service Classes and Performance Goals	120
Job Selection Algorithm for JES3-Managed Initiators	121
Job Selection Algorithm for WLM-Managed Initiators	123
Deadline Scheduling	125
Dependent Job Control	125
Controlling Job Selection	128
Controlling Job Scheduling	133
Output Service	138
Queueing Output	139
Scheduling Output	151
Writing Output	152
External Writers/SAPI Applications	154
NJERDR	154
Internal Reader	154
Accessing Job Output Through TSO	155
Output Service Installation Exits	155
Purge	156

Part 5. Defining and Managing C/I Service 157

Chapter 5. Defining and Managing C/I Service 159

Setting Up C/I Service	160
Advantages to Using C/I FSS Address Spaces	160
How Many C/I FSS Address Spaces and Where to Put Them	160
Defining a C/I FSS Address Space	162
Defining the Maximum Number of CI and POSTSCAN DSPs	163
Controlling Jobs Through C/I Service	166
Controlling Job Flow with Installation Exits	166
Assigning Jobs to the Appropriate Processor and Address Space for C/I Service	167
Using System Symbols in Source JCL for Demand Select Jobs	169

Defining a Converter/Interpreter Options List	169
Managing the Scheduler Work Area	170
Creating SWA Space	170
Preventing a Job from Dominating the SWA	171
Preventing Abnormal Termination of JES3 or a C/I FSS Address Space	171
Monitoring and Modifying C/I Service	174
Determining the Status of C/I Service	174
Modifying C/I Service	175
Managing Procedure Libraries	176
Updating Procedure Libraries	177
Displaying the Status of Procedure Library Data Sets	178

Part 6. Defining and Managing Spool Data Sets 179

Chapter 6. Defining and Managing Spool Data Sets 181

Defining Spool Data Sets	181
Determining How Many Spool Data Sets You Should Allocate	181
Allocating Spool Data Sets	182
Formatting Spool Data Sets	182
Using Spool Partitions	184
Advantages to Spool Partitioning	184
Isolating Different Spool Data Types	185
Defining Spool Partitions	185
Defining Spool Partition Overflow	186
Specifying Spool Data Sets as Members of Spool Partitions	187
Specifying a Spool Partition for Spool Data	188
Determining the Order of Spool Partition Overrides	188
How the User Can Request a Spool Partition	189
Defining Spool Space Allocation Units	192
Defining a Track Group	192
Determining Track Group Allocation Sizes	194
Managing Spool Space	196
Adding or Deleting a Spool Data Set	196
Replacing a Spool Data Set	196
Moving a Spool Data Set to Another DASD Volume	197
Balancing the Work Load Across Spool Partitions	197
Deleting Output Data Sets	199
Freeing Spool Space Using the Dump Job Facility	199
How to Use Performance Measurement Tools to Tune JES3 Spool	199

Part 7. Defining Consoles and Message Routing 201

Chapter 7. Defining Consoles and Message Routing 203

Defining Consoles	203
Defining MCS Consoles	203

Defining RJP Consoles	204
JES3 Console Management	204
Operator Communication	204
Input Processing	204
MCS Console Management	205
Entering Commands	205
Defining the Hardcopy Log	206
Recording Message Traffic	208
Rules	208
Defining Message Routing	209
Where and How Messages Originate	210
Where Messages Can Go	211
Understanding the General Path of a Message	211
JES3 Destination Classes and MVS Routing	
Codes	214
Two Types of Messages	215
Routing JES3 Messages to Consoles	216
Using MSGROUTE to Control Message Routing	216
Message Routing Exceptions	217
Diagnosing Misrouted Message Traffic	220
Suppressing the Display of Messages	220
Automating Message Processing	223

Part 8. Defining and Managing JES3 Resources 225

Chapter 8. Defining and Managing JES3 Resources 227

JES3 Data Sets	227
Allocating JES3 Data Sets	227
Allocating the JES3 Checkpoint Data Set(s)	228
Defining the Maximum Number of Jobs for	
Your Complex	229
JCT Utility (IATUTJCT)	237
Identifying Resident Data Sets	242
Using System Catalogs	243
I/O Devices	243
Defining I/O Devices to JES3	243
Defining Process Modes	245
Running a Printer Under an Output Writer	
Functional Subsystem	246
Using the Infoprint Server for z/OS Printer	
Inventory	251
Defining a Device Configuration	251
Maintaining a Device Configuration Using HCD	253
Allocating an I/O Device to JES3	254
Grouping I/O Devices	254
Specifying Device Fencing	255
Dynamically Reconfiguring I/O Devices	255
Volumes	256
How Resource Definition Affects JES3 Resource	
Management	257

Part 9. Defining and Managing JES3 Mains and Storage 265

Chapter 9. Defining and Managing JES3 Mains and Storage 267

Defining Mains	267
General Considerations	267
Defining a JES3 XCF Group	268
Running JES3 in a Sysplex Environment	270
Defining Storage	271
Defining Buffers	271
Using the Writer Output Multitasking Facility	275

Part 10. Remote Job Processing 277

Chapter 10. JES3 Remote Job Processing 279

Binary Synchronous Communication Remote Job	
Processing	279
Data Security Considerations	279
Data Compression	279
Operator Communications	280
Debugging Facilities	280
Initialization Statements that Affect BSC RJP	281
Systems Network Architecture Remote Job	
Processing	281
SNA RJP Implementation of SNA Concepts	281
Function Management Presentation Services	
(FMPS)	284
JES3 to VTAM Interface	287
Initialization Statements that Affect SNA RJP	289
Maximum Record Lengths for SNA RJP Devices	289
Basic Exchange Support	289
Exchange Support	290
Exchange and Basic Exchange Initialization	
Considerations	290

Part 11. JES3 Networking 291

Chapter 11. JES3 Networking 293

Networking Protocols	293
BSC Versus SNA	293
Converting Networking Protocols	294
Types of Nodes	295
Defining the Home Node	296
Defining a Remote Node	297
Specifying a Communications Path for	
Indirectly-Connected Nodes	298
Defining an Alias	299
Defining a Spool Partition	299
BSC Considerations	300
Defining the Buffer Size	300
Specifying Passwords	300
Defining BSC Communication Lines	301
Logical Senders: How JES3 Names Them	304
SNA Considerations	306
Changing Node Definitions	306
How Restarts Affect Networking Jobs	306
Rerouting Jobs and SYSOUT	307
Networking Job Numbers	307
Defining a Network Message Destination	307
Monitoring the Job Entry Network with	
Installation Exits	307

Monitoring Files Sent via TSO/E TRANSMIT or CMS SENDFILE	310
Deleting Files from the Spool	311

**Part 12. JES3 IBM Tape Library
Dataserver 313**

**Chapter 12. JES3 IBM Tape Library
Dataservers 315**

JES3 Support Overview	315
Allocation for 3495 Tape Data Sets	315
JES3 Initialization Statements	315
3495 Configuration Example	318
Operational Considerations.	320
Unchanged Functions	321
Modified Functions	321
JES3 Limitations	322
Mount Failures	324
IBM 3590 Installed in IBM Tape Library	
Dataservers	324
Integration into Existing Tape Library	325
JES3	326

Part 13. Appendixes 329

Appendix A. The External Writer 331

Overview of the IBM-Supplied External Writer	331
Characteristics of the IBM-Supplied External Writer	331
How to Set Up and Start the External Writer	332
The External Writer Cataloged Procedure	332
How the IBM-Supplied Output Writer Routine Works	336
Functions of the Output Writer Routine	336
How to Add Your Own Output Writing Routine	338
Replacing the IBM-Supplied Routine	339
Using Your Routine for Only Certain Jobs	339

Appendix B. Accessibility 347

Using assistive technologies	347
Keyboard navigation of the user interface	347

Notices 349

Programming Interface Information	350
Trademarks	351

Glossary 353

Index 363

Figures

1. Sample JES3 Cataloged Start Procedure	8	23. Simplified Path of a Message Issued from a	
2. Structure of the JES3 Initialization Stream	13	Global Processor	213
3. Example of JES3 Initialization Stream With		24. Subgeneric Groups.	254
INCLUDE Statements	15	25. JES3-Managed, MVS-Managed, and Jointly	
4. Editing to Add a FSS Managed Printer	33	Managed Device Determination	258
5. Editing to Add a SNA/RJP Work Station and		26. Defining a JES3 XCF Group - Putting the	
Its Devices	33	Pieces Together	269
6. The ADDFSS Member	34	27. Overview of SNA Environment for JES3	283
7. *MODIFY,CONFIG Log For Member ADDFSS		28. JES3-VTAM Interface	284
- Duplicate FSS Names.	34	29. How JES3 Creates a Logical Sender Name	305
8. *MODIFY,CONFIG With IATUX15 Exit		30. Job Related Installation Exits	309
Processing	35	31. SYSOUT Related Installation Exits	309
9. Sample Section of a CLIST that Defines RACF		32. Command Related Installation Exits	310
Profiles	42	33. 3495 Libraries Defined for the Configuration	
10. Which NODES Profiles Are Used?	60	Example	318
11. Example: Simple NJE User Translation	66	34. MVS UNITNAMEs Defined for the	
12. Example: Simple NJE User Translation Using		Configuration Example	319
&SUSER	67	35. SETNAME Statements for the Configuration	
13. Example: Trusted, Semitrusted, and Untrusted		Example	319
Nodes	68	36. HWSNAME Statements for the Configuration	
14. Overview of Job Flow to Input Service	96	Example	320
15. Overview of Pre-Execution Job Delays	119	37. DEVICE Statements for the Configuration	
16. Response Time/Velocity Goals	121	Example	320
17. JES3 Job Selection Environment	130	38. Sample JES3 Definitions for IBM 3590 with an	
18. Sample Job Using IEBDG to Format a Spool		IBM 3494 or 3495 Automated Tape Library	
Data Set	183	Dataserver	327
19. Hardcopy Log Migration Scenario	207	39. General Flow of IBM's Output Writing	
20. Sample Log Entries (with 2-digit-year dates)	208	Routine	338
21. Sample Log Entries (with 4-digit-year dates)	209		
22. Simplified Path of a Message Issued from a			
Local Processor	212		

About This Document

This document supports z/OS (5694–A01) and z/OS.e (5655–G52).

This document is for any JES3 complex that runs Multiple Virtual Storage/Enterprise Systems Architecture/390. This document provides guidance information about initializing, tuning, and managing JES3.

Who Should Use This Document

JES3 system programmers or anyone who is responsible for initializing, tuning, or managing JES3 should use this document. This document also contains information that is useful when planning for JES3.

How to Use This Document

To use this document you need not read it from cover to cover. Depending on your knowledge of JES3 and your information needs, you can read selected topics. The first time you come to this document, however, you should read the table of contents, the figure list, and the “Introduction” to help you understand the type of information presented and the organization of the information.

Where to Find More Information

The following table lists document that contain information related to the information provided in this document.

When this document references information in other documents, the shortened version of the document title is used. The following table shows the shortened titles, complete titles, and order numbers of the document that are not listed in *z/OS Information Roadmap*. Refer to that document for all z/OS documents.

Short Title	Title	Order Number
z/OS Parallel Sysplex Application Migration	z/OS Parallel Sysplex Application Migration	SA22-7662
HLASM Language Reference	High Level Assembler for MVS & VM & VSE Language Reference	SC26-4940
HLASM Programmer's Guide	High Level Assembler for MVS & VM & VSE Programmer's Guide	SC26-4941
z/OS DFSMS: Managing Catalogs	z/OS DFSMS: Managing Catalogs	SC26-7409
z/OS DFSMS Checkpoint/Restart	z/OS DFSMS Checkpoint/Restart	SC26-7401
<i>NetView Installation Guide</i>	<i>NetView Installation and Administration Guide</i>	SC30-3360
IBM IP PrintWay Guide	IBM IP PrintWay Guide	S544-5379
z/OS Communications Server: SNA Network Implementation Guide	z/OS Communications Server: SNA Network Implementation Guide	SC31-8777

Short Title	Title	Order Number
z/OS RMF Reference Summary	z/OS RMF Reference Summary	SX33-9033
z/OS SDSF Operation and Customization	z/OS SDSF Operation and Customization	SA22-7670
z/OS Security Server RACF General User's Guide	z/OS Security Server RACF General User's Guide	SA22-7685
z/OS Security Server RACF Security Administrator's Guide	z/OS Security Server RACF Security Administrator's Guide	SA22-7683
z/OS SDSF Operation and Customization	z/OS SDSF Operation and Customization	SA22-7670
<i>3480 Planning and Migration Guide</i>	<i>IBM 3480 Magnetic Tape Subsystem Planning and Migration Guide</i>	GC35-0098
<i>IBM 3390 DASD</i>	<i>3390 Direct Access Storage Introduction</i>	GC26-4574
<i>IBM 3390 DASD</i>	<i>Using 3390 DASD in an MVS Environment</i>	GC26-4573
<i>IBM 3390 DASD</i>	<i>Reference Summary for the 3390</i>	SX26-1676
<i>IBM 3380 DASD</i>	<i>3380 Direct Access Storage Introduction</i>	GC26-4491
<i>IBM 3380 DASD</i>	<i>Using 3380 DASD in an MVS Environment</i>	GC26-4492

Accessing z/OS™ licensed documents on the Internet

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

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

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code.¹

To obtain your IBM Resource Link user ID and password, log on to:

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

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

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

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

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

Using LookAt to look up message explanations

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

You can access LookAt from the Internet at:

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

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

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

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

Additional Information

Additional information about z/OS elements can be found in the following documents.

Title	Order Number	Description
z/OS Introduction and Release Guide	GA22-7502	Describes the contents and benefits of z/OS as well as the planned packaging and delivery of this new product.
z/OS and z/OS.e Planning for Installation	GA22-7504	Contains information that lets users: <ul style="list-style-type: none">• Understand the content of z/OS• Plan to get z/OS up and running• Install the code• Take the appropriate migration actions• Test the z/OS system
z/OS Information Roadmap	SA22-7500	Describes the information associated with z/OS including z/OS documents for the participating elements.
z/OS Summary of Message Changes	SA22-7505	Describes the changes to messages for individual elements of z/OS. Note: This document is provided in softcopy only on the message bookshelf of the z/OS collection kit (SK3T-4269).
<i>IBM 3495 Tape Library Dataserver Operator's Guide</i>	GA32-0235	Describes the procedures that are used in operating the IBM 3495 Tape Library Dataserver.

Summary of Changes

Summary of changes for SA22-7549-02 z/OS Version 1 Release 4

This document contains information previously presented in *z/OS JES3 Initialization and Tuning Guide*, SA22-7549-01, which supports z/OS Version 1 Release 2.

New information:

- An appendix with z/OS product accessibility information has been added.
- Information is added to indicate that this document supports z/OS.e.
- Use of the SPART= keyword on the NJERMT initialization statement can be used to define a spool partition for NJE streams received from the defined node. See “Defining a Spool Partition” on page 299.
- Added a description of the JCL SYSIN DD limit parameter to Input Service section of JES3 Job Management. See “Preventing a Job from Dominating JSAM Buffers” on page 98.

Changed information:

- Changed Terminology, Intended Programming Interfaces. See Appendix A, “The External Writer” on page 331.
- Added reference in Defining a C/I FSS Address Space. See “Defining a C/I FSS Address Space” on page 162.
- Added XNAMEREQ and DEFCLASS parameters to NJERMT. See Table 49 on page 296.
- Added description of DEFCLASS and XNAMEREQ to Defining the Home Node, “Defining the Home Node” on page 296, and Defining an Alias, “Defining an Alias” on page 299.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes for SA22-7549-01 z/OS Version 1 Release 2

This document contains information previously presented in *z/OS JES3 Initialization and Tuning Guide*, SA22-7549-00, which supports z/OS Version 1 Release 1.

New and changed information:

- Use of the ALTJCL= keyword on the STANDARDS initialization statements that governs how JES3 interprets JES3 job control statements.

- The default names for JES3 libraries installed for z/OS, formerly *SYS1.VvRnMm.dddef*, in OS/390, are changed to *SYS1.dddef*.
- Updates to reflect the changes to JES3 processing that allow or suppress the writing of JESlog data sets, JESMSGSLG and JESYSMSG, during the execution of a job.
- Updates in Chapter 7, “Defining Consoles and Message Routing” on page 203 to highlight JESMSGSLG processing changes in the JES3 address space or the JES3 global.
- Updates in Chapter 8, “Defining and Managing JES3 Resources” on page 227:
 - To allow a maximum of 999,999 job numbers in a JES3 system.
 - Describing the use of **All Function Presentation (AFP)** printers and the use of the Print Services Facility (PSF) with JES3.
- Updates in Chapter 12, “JES3 IBM Tape Library Dataservers” on page 315 to include the 3590 tape device.
- Use of Advanced Function Presentation (AFP) Terminology
 AFP was first introduced in 1984 to support the IBM 3800 Model 3 high-speed printer. AFP now supports new printing technology and new functions. Within JES3 initialization, the IBM 3800 Model 3 is not a unique AFP printer. JES3 provides output data to other AFP printers as well. References to unique model of IBM AFP printers have been replaced with just the term, AFP printer, where appropriate.
- Various additions and corrections to the glossary to improve its technical accuracy.

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

Part 1. Introduction

Chapter 1. Introduction	3
Developing Your Installation Plan	3
Preparing for JES3	4
Installing JES3	4
Running the Hardware Configuration Definition Program	4
Initializing MVS	4

Chapter 1. Introduction

You must make many decisions and perform many tasks to install, initialize, and start JES3. This document is intended to guide you through the decision making process and help you 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 z/OS Program Directory that you receive with JES3 explains how to install JES3.

Before you read this document, you should read z/OS JES3 Introduction to gain a basic understanding of JES3. For information on software and hardware requirements, and compatibility and coexistence requirements, see z/OS Information Roadmap. You should also read z/OS JES3 Migration, which explains the JES3 migration process and summarizes the new, changed, and deleted functions for each release of 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 run the hardware configuration definition (HCD) program?
- 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 processor complexes in ways that will allow reconfiguration without having to restart JES3. For more information about reconfiguring a processor complex, see "Defining Mains" in Chapter 9, "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 *z/OS MVS Recovery and Reconfiguration Guide*.

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.

Preparing for JES3

Before you can initialize JES3, you must first install the MVS base control program (BCP) and JES3, and then initialize MVS. The following steps precede JES3 initialization:

1. Install the MVS BCP
2. Install JES3
3. Run the MVS hardware configuration definition (HCD) program.
4. Update the MVS SYS1.PARMLIB data set
5. Initialize MVS

For more detailed information, see *z/OS and z/OS.e Planning for Installation*.

Installing JES3

Use the System Modification Program/Extended (SMP/E) to install JES3. SMP/E provides two methods for installing JES3, the GENERATE command and the RECEIVE, APPLY, and ACCEPT (RECEIVE/APPLY/ACCEPT) command set. For more information about using either of these methods, see *z/OS Program Directory* and *SMP/E User's Guide*.

Running the Hardware Configuration Definition Program

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

HCD enables you to maintain multiple I/O configurations. For more information about HCD, see *z/OS HCD User's Guide*.

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 predefined 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 *z/OS MVS Initialization and Tuning Guide* for information about how to use SYS1.PARMLIB.

Part 2. Initializing JES3

Chapter 2. Initializing JES3	7
Modifying the JES3 Cataloged Start Procedure	7
Modifying or Creating a JES3 Initialization Stream	11
Organizing the Initialization Stream	12
Using a Segmented Initialization Stream.	13
Testing Your Initialization Stream	15
How to Run Step 1.	16
Using MVS Hardware Configuration Definition (HCD)	16
How to Run Step 2.	17
Storage Requirements for the Initialization Stream Checker	18
Abnormal Termination of the Initialization Stream Checker	18
Starting JES3	19
Hot Start	19
Hot Start with Analysis	20
Hot Start with Refresh.	20
Hot Start with Refresh and Analysis	22
Warm Start	22
Warm Start with Analysis	23
Warm Start to Replace a Spool Data Set	23
Warm Start with Analysis to Replace a Spool Data Set	23
Cold Start	24
Local Start.	25
Tuning JES3	26
Using JMF to Measure JES3 Performance	26
Maintaining JES3	27
JES3 Maintenance Philosophy	27
General Overview of JES3 Service	27
JES3 Restart Information	28
APAR Cover Letter Information	28
Dynamically Changing the JES3 Configuration	31

Chapter 2. Initializing JES3

Each time JES3 starts, initialization occurs. During initialization, MVS runs the JES3 cataloged procedure. The **JES3 cataloged procedure** is JCL that 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 start procedure
- Modifying the JES3 initialization stream
- Using a segmented initialization stream
- Creating and using an alternate initialization stream
- Coding a program for installation exit IATUX15. An **installation exit** is a part of JES3 specifically designed for installations that want to augment or change JES3 processing. (For a description of this and other exits, see *z/OS JES3 Customization*.)
- Dynamically reconfiguring JES3

Modifying the JES3 Cataloged Start Procedure

The JES3 cataloged start procedure contains the job control language (JCL) statements needed to allocate the data sets required by JES3. IBM provides a basic cataloged start procedure shipped with JES3. During the construction of the Base Control Program, MVS stores this procedure in a member of the SYS1.PROCLIB data set. MVS names the member JES3 unless the installation specifies a different name during system installation.

You can change this procedure by using a text editor or by using the IEBUPDTE utility program. Before you can make changes, however, JES3 must complete initialization. Changes do not take effect until you restart JES3. Figure 1 shows a sample of the JES3 procedure. This sample contains all of the required JCL statements. Table 1 explains the purpose of each DD statement in the procedure.

```

//IEFPROC EXEC PGM=IATINTK,DPRTY=(15,15)
//STEPLIB DD DISP=SHR,DSN=SYS1.SIATLIB
//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.SIATSAMP(JES3IN00)

```

If the disk reader facility (DR) is desired, specify:

```
//JES3DRDS DD DISP=SHR,DSN=dsn
```

Figure 1. 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.

To help you better understand the cataloged start procedure, Table 1 describes the purpose and content of each JCL statement shown in the previous example.

Table 1. Description of the Statements in the JES3 Cataloged Start 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.
//CHKPNT 1 //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, 3380, 3390 or 9345 spool volume. See Chapter 8, "Defining and Managing JES3 Resources" on page 227 for further allocation information.

Table 1. Description of the Statements in the JES3 Cataloged Start Procedure (continued)

Statement	Notes	Description
//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.
//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 de-allocated after initialization completes. You can tailor the block size (BLKSIZE) and logical record length (LRECL) values to improve performance. The values you can specify are device-dependent.
//JES3SNAP	O,D	Defines the data set used if JES3 produces a dump during a hot start, hot start with analysis, hot start with refresh, hot start with refresh and analysis, warm start, or warm start with analysis. This data set contains important diagnostic information. The information will not be available if you define JES3SNAP as a dummy data set.
//JESABEND	O,D	Defines the data set used for a JES3 formatted dump. If omitted, a formatted dump of JES3 control information will not be produced.
//SYSABEND or //SYSUDUMP	O,D	Defines the data set for JES3 system dumps.
//IATPLBST . . .	R,D	Defines the installation's standard procedure library.
//IATPLBnn		Note: If a data set is dynamically allocated as both a JES3 DISKRDR data set and a JES3 PROCLIB data set, the UPDATE= parameter on the JES3 /*MAIN statement (JES3 procedure library update facility) can not be used to move the data set.

Table 1. Description of the Statements in the JES3 Cataloged Start Procedure (continued)

Statement	Notes	Description
//JES3IN	R	Defines the data set containing the JES3 initialization stream. This data set must be a blocked or unblocked partitioned data set. The default initialization stream is read from SYS1.SIATSAMP(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.
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 mains.		

The following considerations and restrictions apply to JCL statements in the JES3 cataloged start procedure.

- Do not code a REGION parameter the JCL EXEC statement of the start procedure. Doing so would affect performance.
- Do not code FREE=CLOSE on any DD statement.
- Do not code a ddname of JS3Dnnnn (nnnn is a 4-digit number) on a DD statement. These ddnames define data sets dynamically allocated by the DYNALLOC initialization statement when there is more than one DYNALLOC statement with the same DDN parameter value.
- Do not code a ddname of J3INCLnn (nn is 2-digit number) on a DD statement. These ddnames define data sets dynamically allocated as a result of the INCLUDE 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.
- The data set specified on the JES3IN DD statement must be cataloged if:
 - Your initialization stream contains INCLUDE statements.
 - You want to use the *MODIFY,CONFIG command.
- Do not use the same data set name for JES3OUT, JES3SNAP, or JESABEND DD statements if a pre-allocated data set is being used. The results may be unpredictable.
- Be aware that JES3 does not hold any data set ENQUEUE (major name=SYSDSN, minor name=dsname) while it is running regardless of the type of allocation (JCL or dynamic). JCL allocation ENQUEUEs are based on the DSI subparameter of the PPT parameter of the SCHEDxx member of SYS1.PARMLIB. See *z/OS MVS Initialization and Tuning Reference* for more information on this subparameter. Dynamic allocations by JES3 use an equivalent parameter to prevent a data set ENQUEUE.

Not holding an ENQUEUE allows other jobs or address spaces to access data sets in a serialized manner. For example, if a job needs to update a member of a PROCLIB data set, it can use DISP=OLD on the DD statement and be guaranteed that no other job can access the same data set at the same time.

However, the absence of an ENQUEUE can mislead functions that rely on ENQUEUE for data protection. Those functions can manipulate JES3 data sets successfully, but end up with unwanted results; such as a JES3 coldstart.

Reliance on an ENQUEUE is insufficient since an ENQUEUE is only in effect while JES3 is running. It is your responsibility to ensure protection of your JES3 data sets to prevent their unauthorized modification or destruction when JES3 is running and when it is not. See “Controlling Access to Data Sets JES3 Uses” for more information on how to protect spool and checkpoint data sets.

To ensure performance and integrity, do not:

- Allocate additional extents or release extents
- Compress a PROCLIB data set
- Move a volume containing a PROCLIB.

Note: When compressing and moving PROCLIB, be sure that JES3 is up and running and you use the PROCLIB update facility to unallocate the PROCLIB data sets while the compress or move is being performed.

Modifying or Creating a JES3 Initialization Stream

You can modify the basic initialization stream provided with JES3 or you can create an alternate stream that is tailored to the needs of your installation. An **initialization stream** is a collection of JES3 initialization statements used to define the JES3 system configuration. You also use it to define how you want JES3 to manage resources and jobs. These statements tell JES3 how to manage the following:

- Jobs, job classes, and job class groups
- Mains (global and local)
- I/O devices
- Main and external storage
- The system log
- Communication lines and/or protocols
- Operator communication

During the construction of the MVS Base Control Program, MVS creates a basic initialization stream and stores it in SYS1.SIATSAMP 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 your favorite text editor or a utility program such as IEBUPDTE. Macros that can be used for editing an initialization deck can be found in SYS1.SIATSAMP. The macro prologs contain the information on how to install these macros. However, instead of using these macros you may find it more convenient to code your DEVICE statements using the NUMDEV parameter and with *ALL for a system name on the JUNIT and XUNIT parameter.

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:

- ENDINISH
- ENDJSAM
- FORMAT or TRACK
- MAINPROC

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

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. *z/OS JES3 Initialization and Tuning Reference* contains tables that list interdependent parameters.

Organizing the Initialization Stream

Statements must be placed into the initialization stream in a specific order. Figure 2 shows how to order the statements.

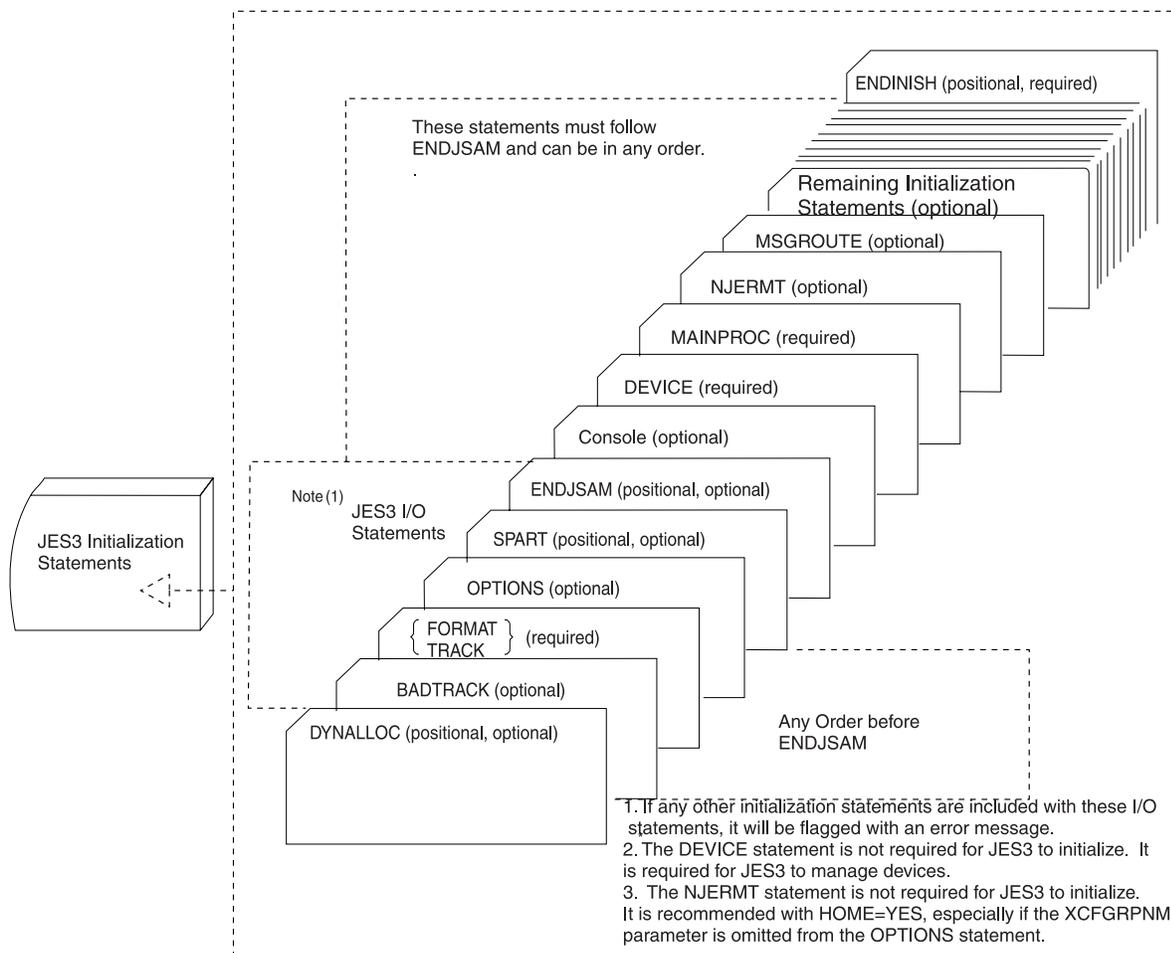


Figure 2. Structure of the JES3 Initialization Stream

Using a Segmented Initialization Stream

JES3 allows your initialization stream to be segmented. You put different sections of the initialization stream into different members of a partitioned data set and use the INCLUDE initialization statement to include the members at the appropriate places in the primary initialization stream. Each INCLUDE initialization statement refers to a member of the partitioned data set. The member contains the initialization statements you want added to the initialization stream.

For example, you can put all RJPWS initialization statements into member "RJPWS" and include this member at the appropriate place in the initialization stream.

The "member" you refer to is the name of the member in the data set specified on the JES3IN DD statement. The JES3IN DD statement must refer to a cataloged data set and be part of the cataloged JES3 start procedure. You can have as many INCLUDE statements as you wish in your initialization stream. You can have up to four member levels: the primary initialization member and up to 3 levels of INCLUDE statements.

For example, if the members of a PDS are JES3INXX, A, B, and C you can specify your INCLUDE initialization statements as follows:

```
Member JES3INXX      INCLUDE,member=A
Member A             INCLUDE,member=B
Member B             INCLUDE,member=C
Member C
```

The INCLUDE statement can occur anywhere after the DYNALLOC statements.

Note: If JES3IN DD data set is concatenated, the INCLUDE statement only works for members in the first data set in the concatenation.

The following example illustrates how you could organize your JES3 initialization stream with INCLUDE statements.

```

DYNALLOC,...
-
-
-
DYNALLOC,...
*
INCLUDE, MEMBER=JSAM           Include JSAM related statements up to ENDJSAM
*
INCLUDE, MEMBER=MAINS          Include main processor information
                                (MAINPROC, DEVICE DTYPE=SYSMAIN)
*
INCLUDE, MEMBER=BSCRJP         Include BSC related information
                                (RJPLINE, RJPTERM, CONSOLE, DEVICE)
*
INCLUDE, MEMBER=CI             Include C/I related information
                                (CIPARM, HWSNAME, RESDSN)
*
INCLUDE, MEMBER=CONSOLES       Include consoles related information
                                (CONSTD, MSGROUTE)
*
INCLUDE, MEMBER=DEADLINE       Include deadline algorithms (DEADLINE)
*
INCLUDE, MEMBER=FSS            Include FSS related information (FSSDEF)
*
INCLUDE, MEMBER=GMS            Include GMS job classes, groups, select mode
                                (CLASS, GROUP, SELECT)
*
INCLUDE, MEMBER=MDS            Include MDS related information
                                (SETPARAM, SETNAME, DYNALDSN)
*
INCLUDE, MEMBER=MISC           Include miscellaneous information
                                (STANDARDS, SYSID, RESCTLBK)
*
INCLUDE, MEMBER=NJE            Include NJE definitions
                                (NJERMT, DEVICE DTYPE=NJELINE)
*
INCLUDE, MEMBER=OUTSERV        Include output service definitions
                                (OUTSERV, SYSOUT)
*
INCLUDE, MEMBER=SNARJP         Include SNA related information
                                (COMDEFN, COMPACT, RJPWS, CONSOLE, DEVICE)
*
INCLUDE, MEMBER=UR             Include unit record devices
*
INCLUDE, MEMBER=TAPE           Include tape devices
*
INCLUDE, MEMBER=DASD           Include direct access storage devices
*
ENDINISH

```

Figure 3. Example of JES3 Initialization Stream With INCLUDE Statements

Testing Your Initialization Stream

You use the JES3 initialization stream checker utility to test your JES3 initialization statements before you perform a hot start with refresh, warm, or cold start of JES3. The initialization stream checker detects most syntax errors and some logical errors in the initialization stream. You can run this utility as a batch job or as a TSO job using a command list (CLIST).

You must perform two steps to use the initialization stream checker:

1. Step 1 gathers data for the initialization stream checker.
2. Step 2 runs the initialization stream checker.

Step 1 obtains hardware 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 runs the initialization stream checker. The initialization stream checker examines all initialization statements for correct syntax, except the DYNALLOC 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 “Grouping I/O Devices” on page 254 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 Run Step 1

If you want the initialization stream checker to check for logical errors in your initialization stream, first obtain the MVS configuration data by running the hardware configuration definition (HCD) program. Otherwise, you can omit this step.

Using MVS Hardware Configuration Definition (HCD)

The MVS hardware configuration definition (HCD) provides an interactive, panel-driven dialog that you can use to define hardware configurations. HCD also provides an option that creates JES3 initialization stream data for use in step 2 of the initialization stream checker utility. Perform the following steps to obtain the JES3 configuration data using HCD:

1. Select the “Activate Configuration Data” option on the HCD primary panel. HCD next displays the JES3 panel.
2. Select the “Create JES3 Initialization Stream Data” option from the JES3 panel.
3. Specify the following items on the JES3 panel:
 - Data set name (This name will be used later in the STG1CODE DD when running IATUTIS.)
 - Eight character MVS configuration identifier
 - Eligible device table (EDT) identifier

You can also create the initialization stream data using a batch job like the following:

```
//jobname JOB 'ACCTINFO', 'NAME', MSGLEVEL=(1,1),
// MSGCLASS=R
//BUILD EXEC PGM=CBDMGHCP, PARM='CONFIG, JES,ccccccc,ee'
//HCDIODFS DD DSN=SYS0.IODFxx, DISP=SHR
//HCDDECK DD DSN=dsname(member), DISP=OLD
```

where:

- xx is the suffix of the I/O definition file to be used as the basis for this data.
- ccccccc is the eight character MVS configuration identifier.
- ee is the eligible device table (EDT) identifier.
- dsname is the data set name to contain the output data (and to be used later in the STG1CODE DD when running IATUTIS).
- member is the member of the output data set to contain the data for a particular processor.

You must create this data for all processors defined in the initialization stream. Each member has the same name as one of these processors.

MVS places the configuration data in the data set you specify. IBM recommends that you place the data into a partitioned data set (PDS) since the initialization stream checker requires, as input, a PDS whose member name(s) match the names of the mains that you specify in the JES3 initialization stream.

See *z/OS HCD User's Guide* for information about using the HCD dialog.

How to Run Step 2

When you have obtained the configuration data for each main in your complex, submit the JCL that runs the initialization stream checker. Your JCL should contain the following data sets:

Table 2. Data Sets Required to Run the Initialization Stream Checker

DDNAME	PURPOSE
STEPLIB	Specifies the data set that contains the JES3 load modules that are used during initialization. If you are running the initialization stream checker against the level of JES3 installed on your system, this would be the same as the data set specified on the STEPLIB statement in the JES3 cataloged procedure. If you are running the initialization stream checker against the level of JES3 installed on your system, and the JES3 load library is installed in your LNKSTxx concatenation, you do not need the STEPLIB DD statement.
JES3IN	For the data set that contains your JES3 initialization stream. You can make this a PDS and specify one member.
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.
STG1CODE	For the PDS created as a result of the HCD program, if you want the initialization stream checker to test for logical errors; otherwise specify a dummy data set.
JESABEND	An optional dummy data set that you can define to avoid a warning message.

Step 2: Sample JCL for Running the Initialization Stream Checker

```

//INITCHK    JOB 'ACCTINFO','NAME',MSGLEVEL=(1,1),
//           MSGCLASS=R,...
//IATUTIS    EXEC PGM=IATUTIS,PARM='P=1F1R'
//STEPLIB   DD DSN=SYS1.SIATLIB,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 Running the Initialization Stream Checker

```

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.SIATLIB(IATUTIS)' 'P=1F1R'

```

Notes:

1. You can use the initialization stream checker to validate initialization streams that contain up to a maximum of 2048 generic and esoteric device names.
2. 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 runs the initialization stream checker.

The optional P= parameter is used to pass a character string to IATUX15 (exit 15) if you need to add, delete, or modify JES3 initialization statements and subsequent start processing. This is the same parameter you can supply in response to message IAT3012 SELECT INISH ORIGIN (N OR M=) OR OPTIONAL EXIT PARM (,P=) OR CANCEL when warm or cold starting JES3.

Storage Requirements for the Initialization Stream Checker

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 to contain the load module. The load module is the size of the JES3 nucleus (IATNUC) plus approximately 8 kilobytes 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 PDS that contains the output from the hardware configuration program.

When the initialization stream checker runs in your user address space, it occupies approximately 540 kilobytes of virtual storage.

Abnormal Termination of the Initialization Stream Checker

If your initialization stream contains critical syntax errors, the initialization stream checker does not check for logical errors and ends with a JES3 dump code of U001. See *z/OS JES3 Diagnosis* for a description of JES3 dump codes. Correct the critical errors as described in JES3OUT, and re-submit the initialization stream checker.

If you code a JES3 INTDEBUG statement in your JES3 initialization stream and want a dump, include a SYSUDUMP DD statement in the job's JCL.

Starting JES3

Here are the ways to start JES3:

- Hot start
- Hot start with analysis
- Hot start with refresh
- Hot start with refresh and 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 (used to start local mains only)

The type of start you select 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. See *z/OS JES3 Commands* for information on how to perform each type of start.

Hot Start

Use a hot start to start JES3 on the global:

- After an orderly shutdown
- After a JES3 failure on the global from which JES3 cannot automatically recover
- After an MVS failure that terminates all functions in the global
- 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 during the previous cold or warm start. In addition, certain changes made by the operator after the previous restart remain in effect. For a list of these changes, see *z/OS JES3 Commands*.

If you perform an IPL of MVS before the hot start, JES3 restarts jobs that were running on the global 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 FSSs that were running on the global and reschedules all jobs that were active in the FSS at the time of the IPL. If you do not perform an IPL of MVS before the hot start, all jobs and FSSs that were running before the hot start continue to run after the hot start. During a hot start, JES3:

- Examines the job queue to verify that it can restart each job
- Prompts the operator if it locates a job that contains invalid control information
- Removes jobs that contain invalid control information
- Records the job control information associated with the removed jobs (they are recorded in the data set defined by the //JES3SNAP DD statement)
- Continues to process valid jobs that were in the job queue before the hot start. Internal job numbering resumes with the next available job number.

Jobs and FSSs running on locals before the hot start continue to execute after the hot start. Any changes to the definition of an FSS brought about by using the *MODIFY operator command before the hot start remain in effect across a hot start with or without an IPL of MVS. 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.

Hot Start with Analysis

Use hot start with analysis after a hot start fails and/or you suspect problems with the JES3 job queue. A hot start with analysis performs the same functions as a hot start. In addition, JES3 also:

- Performs further job validation
- Gives the operator an opportunity to delete invalid jobs.

You are not required to IPL or to restart the local mains, although you may optionally do so.

Hot Start with Refresh

Use hot start with refresh when you want to change the initialization stream without having to IPL the entire complex.

Performing a hot start with refresh avoids disrupting your system since you need only restart the JES3 global address space. You do not need to IPL all processors in the complex like you do with a warm start.

JES3 is automatically restarted on all local processors that are at least at OS/390 Version 2 Release 9 JES3. (For this topic, any processor that is not OS/390 Version 2 Release 9 JES3 is termed a down level processor.). An IPL on processors at OS/390 Version 2 Release 9 JES3 is not required, although you may optionally do so. If you have added, deleted, or changed any SETNAME statements or DEVICE XUNITS, you must restart JES3 on all down level processors. You may need to IPL down level processors depending on how many of these changes you have made since they were last IPLed. If you have added, deleted, or changed any DYNALDSN statements you must IPL all down level processors. If you have not made any changes to SETNAME or DYNALDSN statements or DEVICE XUNIT parameters, you need not IPL or restart JES3 on down level processors although you may optionally do so.

During a hotstart with refresh, JES3 processes only those statements that can be changed. Those statements and parameters that cannot be changed are checked for syntax errors, and the information from the previous warm or cold start is used. If a hot start with refresh fails, a subsequent hot start will reinstate the configuration from the last successful hot start with refresh, warm start, or cold start.

The initialization statements listed below in Table 3, are the ones you can change, add, or delete during a hot start with refresh.

Table 3. Modifiable JES3 Initialization Statements for Hot Start with Refresh

ACCOUNT	BUFFER	CIPARM	CLASS
COMMDEFN	COMPACT	CONSOLE	DEADLINE
DESTDEF	DEVICE	DYNALLOC	DYNALDSN
FSSDEF	GROUP	HWSNAME	INCLUDE
INTDEBUG	MSGROUTE	NJECONS	NJERMT
OPTIONS	OUTSERV	RESCTLBK	RESDSN
RJPLINE	RJPTERM	RJPWS	SELECT
SETACC	SETNAME	SETPARAM	SETRES
STANDARDS	SYSID	SYSOUT	

You need to be careful when changing the initialization stream during a hot start with refresh, because not all of the initialization statements are processed. If you add or change one of the initialization statements that is not processed during a hot start with refresh, errors can occur when it is referenced by another initialization statement that is processed during a hot start with refresh. The examples that follow show errors that can occur during a hot start with refresh, because of dependencies between statements that can and cannot be changed.

- Example (1)

You add a new SPART initialization statement to your initialization stream and change the SPART parameter of a SYSOUT statement to reference the new spool partition. During initialization, JES3 issues message IAT3222 to indicate that an incorrect spool partition was specified; this happens during a hot start with refresh, because SPART statements are ignored and the SPART statements from the previous warm or cold start are used. The new SPART statement you added is ignored. The error message is issued when the SYSOUT statement references the new SPART statement.

- Example (2)

In your old initialization stream, you have a MAINPROC statement that refers to a SELECT statement named "A".

```
MAINPROC,NAME=SY1,SELECT=A,...
SELECT,NAME=A,....
```

In your new initialization stream, you delete SELECT statement "A", add a new SELECT statement "B", and change the MAINPROC statement to refer to the new SELECT statement.

```
MAINPROC,NAME=SY1,SELECT=B
*SELECT,NAME=A,...           Delete select mode
SELECT,NAME=B,....
```

During initialization, JES3 tries to activate select mode A for main processor SY1 and fails. This failure occurs during a hot start with refresh because the new MAINPROC statement is ignored and the old MAINPROC statement is used. The SELECT parameter (NAME=A) from previous warm or cold start is used. This causes an error when JES3 tries to activate select mode "A" for SY1 since select mode "A" is no longer defined in the initialization stream.

Notes:

1. You must restart C/I FSS address spaces if you changed initialization statements or parameters that C/I FSS address spaces use (for example, CIPARM, HWSNAME, REDSN, and SETNAME).
2. If a hot start with refresh fails, the old configuration remains in effect. A subsequent JES3 hot start reinstates the configuration from the last successful hot start with refresh, warm start, or cold start.
3. If you perform an IPL of MVS before the hot start with refresh, JES3 restarts jobs that were running on the global and that are eligible for restart. If a job is not eligible for restart, JES3 processes it based on the job's failure options. (See the FAILURE parameter on the CLASS initialization statement for details.) JES3 restarts all FSSs that were running on the global and reschedules all jobs that were active in the FSS at the time of the IPL. If you do not perform an IPL of MVS before the hot start, all jobs and FSSs that were running before the hot start with refresh continue to run after the hot start with refresh.
4. Jobs and FSSs running on locals before the hot start with refresh continue to run after the hot start with refresh. The following table shows whether changes

to the definition of an FSS (made by using the *MODIFY operator command) remain in effect across a hot start with refresh.

Table 4. Hot Start With Refresh - Effect on FSSs

Status of the System Where the FSS Runs	Is the System IPLed ?	Effect on FSS
Not Active	N/A	The FSSDEF statement overrides the changes.
Global	No	Changes remain in effect.
Local	No	Changes remain in effect.
Global	Yes	The FSSDEF statement overrides the changes.
Local	Yes	Changes remain in effect.

5. Overflow partition relationships, including any changes caused by using the *MODIFY operator command before the hot start with refresh, remain in effect across a hot start with refresh.
6. Refer to the Functional Summary tables in *z/OS JES3 Initialization and Tuning Reference* for more specific information on modifying initialization statements during a hot start with refresh.
7. A hot start with refresh will undo many of the changes made by operator commands that are related to initialization statements. For example, most parameters on the *MODIFY,G command will no longer be in effect after a hot start with refresh; the information from the initialization statements will be used. Refer to *z/OS JES3 Commands* for those commands and parameters that do not remain in effect during a hot start with refresh.

Hot Start with Refresh and Analysis

Use hot start with refresh and analysis after a hot start with refresh fails and/or you suspect problems with the JES3 job queue. A hot start with refresh and analysis performs the same functions as a hot start with refresh. In addition, JES3 also:

- Performs further job validation
- Gives the operator an opportunity to delete invalid jobs.

For restart of local mains, see Note 4 on page 21.

Warm Start

Use a warm start to restart JES3 on the global:

- After either type of hot start or hot start with refresh fails.
- After a failure of the global because of a software/hardware failure.
- When you want to change the initialization stream and the changes cannot be performed with a hot start with refresh.
- Following the construction of the MVS Base Control Program.

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, use caution when changing the following statements:

- BUFFER
- CLASS
- DEVICE
- FORMAT
- MAINPROC
- OPTIONS
- SETNAME
- TRACK

Refer to the Statement Summary charts that precede each initialization statement in *z/OS JES3 Initialization and Tuning Reference* for restart modification considerations for each of the statements listed above.

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 information that contain invalid data from the job queue
- Records job control information that is associated with the removed jobs (they are recorded in the data set defined by the statement //JES3SNAP)

After performing a warm start with analysis, you must perform an IPL of MVS and then restart (local start) all local mains.

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 function 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 of MVS and then restart (local start) all local mains. See “Replacing a Spool Data Set” in Chapter 6, “Defining and Managing Spool Data Sets” for information on how to replace a spool data set using a warm start.

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 function as a warm start. In addition, you can replace a spool data set and JES3:

- Removes from the job queue all jobs that have invalid control information

- Records job control information associated with the removed jobs (they are recorded in the data set defined by the statement `//JES3SNAP`)

After performing a warm start with analysis to replace a spool data set, you must IPL MVS, then restart (local start) all local mains.

Cold Start

Use a cold start to start JES3 on the global:

- When all types of warm starts are unsuccessful.
- When you want to change the size of the job control table data set (JCT), move the JCT to a device of a different device type, or relocate the JCT to a device of the same device type.

You can avoid a cold start in this case by using the JCT Utility, IATUTJCT, to copy the contents of your old JCT data set to a new JCT data set. Refer to “JCT Utility (IATUTJCT)” on page 237 for more information on the JCT Utility. You cannot use any other utility to do this copy because the JCT data set contains information that is dependent on the volume and the location on that volume where the data set resides.

- When you have changed the BUFSIZE parameter on the BUFFER initialization statement.
- When you have changed the SE parameter on the OPTIONS statement.
- When you have 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.
- When you have changed the sequence of 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 FSSs terminate at IPL time. Any changes to the definition of an FSS brought about by using the *MODIFY operator command before the cold start are lost. To restart a C/I FSS after a cold start, you must have specified START=YES on its FSSDEF statement or you must issue the *MODIFY,F,FSSNAME=fssname,ST=YES operator command.

If JES3 is running properly before the cold start, you can save jobs by using the dump job (DJ) facility and then restore them after the cold start has been performed. This facility is described in *z/OS JES3 Commands*.

Table 5 on page 25 summarizes the types of starts you can use to start the global. This figure uses the following mnemonics to identify each start type:

- Hot start (H)
- Hot start with analysis (HA)
- Hot start with refresh (HR)
- Hot start with refresh and analysis (HAR)
- 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)

Table 5. Characteristics of Global Processor Starts

Characteristics	H	HA	HR	HAR	W	WA	WR	WAR	C
Retains Job Queue	Y	Y	Y	Y	Y	Y	Y	Y	N
Read Init. Stream	N	N	Y	Y	Y	Y	Y	Y	Y
Spool Data Lost	N	N	N	N	N	N	N	N	Y
Replace Spool Data Set	N	N	N	N	N	N	Y	Y	Y
Reinstate Spool Data Set	Y	Y	Y	Y	Y	Y	Y	Y	Y
Remove Spool Data Set	Y	Y	Y	Y	Y	Y	Y	Y	Y
Add Spool Data Set	N	N	N	N	Y	Y	Y	Y	Y
IPL Global	O	O	O	O	R	R	R	R	R
IPL Locals	O	O	O	O	R	R	R	R	R
Job Disposition (see notes)	1,2	1,2,5	1,2	1,2,5	4	4,5	4	4,5	3

1. If you do not perform an IPL on the global processor, jobs that were running at the time of the hot start continue to execute.
2. If you perform an IPL on the global processor, JES3 restarts jobs that are eligible for restart and were running 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 option.
5. JES3 removes jobs that could cause a restart failure from the job queue and records control blocks that are associated with the job.

KEY Y=yes, N=no, O=optional, R=required

Local Start

Use a local start to start JES3 on a local main:

- After a cold start or any type of warm start on the global.
- After a hot start with refresh if you changed initialization statements or parameters that affect the local, and the local is below the HJS6609 level.
- After JES3 fails on the local.
- After a processor complex has been partitioned.

After a cold start or any type of warm start, initialization of the global must be complete before you can start or restart a local main. If you do not automatically start JES3 as part of the MVS IPL, you can use the ROUTE *OTHER command to start JES3 on all the local processors simultaneously, once those processors have joined the sysplex.

JES3 does not read the initialization stream to initialize a local. Instead, the global provides the initialization values that it read from the initialization stream during the last cold start, warm start, or hot start with refresh.

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 of MVS on the local, jobs and FSSs that were active at the time of the local start 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 is connected to the global, JES3 on the global restarts FSS address spaces that were active on the local 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 FSSs running on other mains.

Tuning JES3

You can affect how JES3 performs. Factors that you should consider when tuning JES3 are:

- Hardware used
- Availability of resources
- JES3 work load
- Job processing options selected

You should develop a set of performance objectives that state what you expect of JES3. After you develop performance objectives and install JES3, you can tune it. For guidelines about how to tune JES3, see the section of this document that describes the JES3 function or the specific resource you want to tune.

Tuning is not a task that you perform only once. It is an iterative process; you measure the performance and make adjustments, measure the performance, make additional adjustments, and so forth. You can use the following facilities to collect data and measure the performance of JES3 and MVS:

- The JES3 Monitoring Facility (JMF)
- The MVS Resource Monitoring Facility (RMF)
- The MVS System Monitoring Facility (SMF)
- JES3 Commands

This document refers to the reports produced by JMF and RMF, and records produced by SMF where applicable. Refer to *RMF Monitor I and II Reference Summary* for descriptions and examples of RMF reports and *z/OS MVS System Management Facilities (SMF)* for descriptions and examples of SMF records.

Using JMF to Measure JES3 Performance

The JES3 Monitoring Facility (JMF) is a JES3 diagnostic and performance utility that you can use to help monitor, diagnose, and tune your JES3 complex. You can use JES3 operator commands to produce JMF reports that contain data about:

- JES3 spool data management
- JES3 CPU/storage
- JES3 functions
- JES3 device scheduling
- JES3 job throughput

Since JMF measurements vary from installation to installation, you must first establish your own baseline measurements by running JMF over a period of time under normal operating conditions. See *z/OS JES3 Diagnosis* for sample JMF reports and the commands that you need to generate those reports.

Maintaining JES3

JES3 Maintenance Philosophy

Maintaining JES3 is a periodic activity. As you receive updates or enhancements for your JES3 product, you need to apply these updates to your JES3 component. Although IBM delivers JES3 and MVS service in the same manner, how you install the service can differ.

To minimize the impact of applying maintenance and still maintain continuous operations, you need to understand the JES3 maintenance philosophy. You use this maintenance philosophy to apply APARs/PTFs that fix problems that can arise between your scheduled maintenance upgrades, which you apply through the PUT maintenance process.

When installing APARs, there are new terms, which apply. These terms are explained in this section. The intent is to limit the number of complex-wide restarts required by installing APARs to as few systems as possible. Due to design limitations of the JES3 product, it may not be possible in all cases to avoid the need for a complex-wide restart; however, in these cases there is a way to "bypass" applying specific fixes through the use of the ++HOLD statement for a particular PTF. The reasons for this "bypass" are clearly indicated in the APAR closure test.

In addition, JES3 internally documents why the complex-wide restart can not be avoided. It is expected that JES3 maintenance is "rolled" across all systems in the JES3 complex in the order specified and in a timely manner (hours not days). Without the fix on all processors, you run the risk of hitting the problem again.

Also, the restart information does not take into account a dynamic system interchange (DSI). Therefore, if you use DSI, be aware that when "global" is specified, this implies the "current global" and all potential globals.

General Overview of JES3 Service

To maintain the JES3 component, it is necessary to understand how JES3 structures its maintenance in an System Modification Program (SMP) environment. It is assumed that the reader is familiar with the terminology and operation of SMP. Refer to *SMP/E User's Guide* for more information on using SMP.

JES3 is a "source maintained" product, which means that corrective fixes (APARs) are distributed in source format while preventive fixes (PTFs) are distributed in source (non-cumulative) and object code format. It is a SMP requirement that every new fix (SYSMOD) must either PRE-REQ (prerequisite) or SUP (supersede) all previous maintenance to the same module. In this way, SMP can ensure that no previous service is dropped or regressed.

As an example, assume the first PTF, PTF1, addresses a problem in module IATOSWD, which is the permanent fix for APAR1. APAR2 is then created to address another problem in the same module, IATOSWD. The PTF, PTF2, created for APAR2 must then PRE-REQ or SUP PTF1 because both PTF1 and PTF2 apply to the same module, IATOSWD.

JES3 APARs do not follow this same requirement. A JES3 APAR only PRE-REQs or SUPs the code (APAR) that is required to allow the fix to work. JES3 APARs never PRE-REQ or SUP PTFs. Applying APARs in this manner allows the user to install APAR maintenance to the system regardless of the existing PTF level currently installed.

During the application of JES3 APARs, SMP SYSMOD regression reports can indicate that other SYSMODs are being regressed. This regression occurs because the APAR did not PRE-REQ or SUP the previously applied maintenance PTF. Each regression error message indicates the specific APAR that is doing the regress and the specific PTF that was previously applied. Regression error messages, GIM38201W and GIM31901L, issued during the application of JES3 APARs can be ignored when they list PTFs. You must not ignore these messages if they list APARs, and you must not ignore any other error messages.

During the application of JES3 PTFs, regression error messages can be issued. If the regressed SYSMOD is another PTF, then the PTF currently being applied may be in error, or the already installed PTFs have been erroneously applied. In either case, you must determine why the regression message was issued. If the SYSMOD being regressed is an APAR, the APAR needs to be restored, the new PTF applied, then the APAR reapplied. SMP gives you another choice instead of restoring the APAR; it is to ACCEPT the APAR; however, JES3 recommends that APARs never be accepted. Do not ignore any SMP error messages when processing PTFs.

JES3 Restart Information

After the maintenance has been applied to your JES3 system, you need to determine how JES3 is to be restarted. Each JES3 APAR and PTF includes installation and activation information that define the type of JES3 and MVS restarts required to properly install the changes made by the APAR or PTF. The information provided takes into account module residency, operating environment, and how the change interacts with control blocks and other data structures created prior to the installation of the change. This restart information must be followed to ensure proper implementation of the changes.

The restart information includes:

- Which processors the APAR must be installed on.
- The order of installation.
- The order of activation.
- The type of JES3 restart required.
- Whether or not CLPA is required.

APAR Cover Letter Information

An example of a typical cover letter information found in the APAR follows.

Restart Information - Install On

This information describes which processors the APAR must be installed on.

This processor can only be one of the following:

All processors with C/I FSS

All processors in the complex with C/I FSS address spaces.

All processors with WTR FSS

All processors in the complex with WTR FSS address spaces.

All processors with an FSS

All processors in the complex with FSS address spaces.

Locals

All local processors in the complex.

Global

The global processor and all potential global processors in the complex.

All processors

All processor in the complex regardless of whether C/I or WTR FSS is active on the processor.

RESTART Information - Order of Installation

This information describes the order that the APAR must be installed. This order can be one of the following:

ANY

The order of installation does not matter.

Global first, then locals

Install on the global processor first, then install on all local processors in the complex.

Locals first, then Global

install on all local processors in the complex first, then install on the global processor.

RESTART Information - Type of JES3 Restart Required

This information describes the type of JES3 restart that is required to activate the change after installing the APAR. This restart can be one of the following:

NONE

No JES3 restart is needed.

LOCAL

Perform a local start of all local processors in the complex.

C/I FSS

Terminate and restart all C/I FSS address spaces in the complex.

WTR FSS

Terminate and restart all WTR FSS address spaces in the complex.

ALL FSS

Terminate and restart all FSS address spaces in the complex.

HOT

Perform a hot start of the JES3 address space on the global processor.

HOT/COMMAND

For OS/390 JES3 Version 1 Release 3 and below, perform a hot start of the JES3 address space.

For OS/390 JES3 Version 2 Release 4 and above, a JES3 global restart is not required. Instead, the new level of the code can be activated by issuing a MODIFY LLA,REFRESH or MODIFY LLA,UPDATE command, and then issuing a *MODIFY,X,M=MODNAME,REFRESH command. The *MODIFY,X command deletes the old version of the module, and causes a new copy of the module to be loaded the next time it is used.

If the module refresh is successful, the following message is displayed:

```
*F,X,M=IATIIDR REFRESH  
IAT8489 (MODX) - MODULE IATIIDR HAS BEEN REFRESHED
```

If the module is not in storage when the refresh request is issued, the following message is displayed:

```
*F,X,M=IATIIDR REFRESH  
IAT8482 (MODX) - NO MODULES WERE REFRESHED
```

If the module is currently in use when the *MODIFY,X command is issued, the old version is not deleted and a new version is not loaded until there

are no longer any JES3 functions using the module. If the module refresh cannot be satisfied, the following message is displayed:

```
*F,X,M=IATIIDR REFRESH
IAT8489 (MODX) - MODULE IATIIDR REFRESH IS PENDING
```

An

```
*INQUIRY,X,M=MODNAME
```

command can be used to display whether a module refresh is pending:

```
*I,X,M=IATIIDR
IAT8476 IATIIDR = RES=00005 USE=00001 LOADS=00001 EP=050F5000..DELETE=Y
IAT8476 REL=HJS6066 DATE=02/27/98 TIME=17:48 APAR=NONE PTF=OS260
IAT8476 SIZE=000021BB AMODE=31 RMODE=ANY REFRESH PENDING
```

If a module cannot be refreshed, the DSP that is using it must be cancelled and restarted. If this is not possible, a hot start is required.

HOT/REFRESH

Perform a hot start with refresh of the JES3 address space on the global processor.

WARM

Re-IPL all processors in the complex. Perform a warm start of the JES3 address space on the global processor. Perform a local start of all local processors in the complex.

No IPL Required

In cases where you do not need to IPL and instead just restart the JES3 address space, issue a *MODIFY,X command to refresh the module within JES3 (or use the SETPROG command for Dynamic LPA). You still need to issue a MODIFY LLA,REFRESH or MODIFY LLA,UPDATE command first.

RESTART Information - Type of IPL Required

This information describes whether or not an IPL is required. If an IPL is required, this information indicates which systems should be IPLed after installing the APAR. This information can be one of the following:

NONE

No IPL is required.

ROLLING

IPL one processor at a time until all processors in the complex have been IPLed.

ROLLING/A

You can pick up the change on each processor where the fix is to be activated by either using the "dynamic LPA facility" on the processor or by performing an IPL on the processor. See on page 31 for more information.

GLOBAL

IPL the global processor only.

GLOBAL/A

You can pick up the change by using the "dynamic LPA facility" on the global processor or by performing an IPL on the global processor. See on page 31 for more information.

LOCAL

IPL the local processors only.

LOCAL/A

You can pick up the change by using the "dynamic LPA facility" on each local processor or by performing an IPL of each local processor. See on page 31 for more information.

GLOBAL TO LOCAL

IPL the global processor first followed by the local processors.

LOCAL TO GLOBAL

IPL the local processors first followed by the global processor.

COMPLEX

IPL all processors at the same time.

Dynamic LPA Facility - Definition of A

You can activate the change using the "dynamic LPA facility" followed by a hot or local start without an IPL of the required processors. See *z/OS JES3 Migration* for more information.

New versions of JES3 LPA modules can be dynamically added to LPA by issuing the following SETPROG command:

```
SETPROG LPA,ADD,MODNAME=IATSICA,DSN=LNKLST
CSV551I 10.16.14 LPA ADD 868
SUCCESSFUL: 1 UNSUCCESSFUL: 0 NOT PROCESSED: 0
MODULE      RESULT
IATSICA     SUCCESSFUL
```

This command can be issued while JES3 is active. The change does not take effect until JES3 is restarted. This allows you to back out the change if necessary.

If the fix must be installed on more than one system, a SETPROG command must be issued for each system.

Once the required modules have been added to LPA, perform a hot start or local start of JES3 (without an IPL of the system) to activate the change. During initialization, JES3 picks up the new versions of the modules and issues the following:

```
IAT3085 LPA MODULES CHANGED: IATSICA ...
```

On the JES3 global processor, the above message appears in JES3OUT only. On the JES3 local processors, the above message appears on the console.

Restart Information - CLPA required

This information describes whether or not a CLPA (create link pack area) is required when the system is IPLed after installing the APAR. The information can be as follows:

YES

Perform a CLPA.

NO

No CLPA is required.

Dynamically Changing the JES3 Configuration

You can make changes to the JES3 configuration dynamically after the system has been initialized and is running by using the *MODIFY,CONFIG operator command.

The *MODIFY,CONFIG command allows you to add the following definitions to JES3 without having to restart the JES3 global and local address spaces:

- SNA RJP work stations and their associated consoles and devices
- Non-channel attached printers

You use *MODIFY,CONFIG to specify the name of a member in the data set allocated to the JES3IN DD statement in the JES3 cataloged start procedure. This member contains the initialization statements associated with definitions that you want added to the JES3 configuration. The following initialization statements can be coded in the member specified on the *MODIFY,CONFIG command:

- RJPWS to define SNA/RJP work station characteristics
- CONSOLE to define SNA/RJP console
- DEVICE to define SNA/RJP devices and non-channel attached FSS managed printers
- FSSDEF to define writer FSSs
- INTDEBUG to establish the Initialization Debugging Facility
- INCLUDE to include another initialization stream member

Note: If the JES3IN DD data set is concatenated, only the members in the first data set of the concatenation are used in processing the INCLUDE statement.

*MODIFY,CONFIG command processing occurs in three stages:

Statement Syntax Checking

This process checks the statements for validity and issues error messages if they contain errors. This stage only checks for errors on the statement; it does not check for cross-statement dependencies. At the end of this stage, you can look at the log that was generated, and cancel *MODIFY,CONFIG processing if there are errors.

Cross Checking/Table Build

This process checks for cross statement dependencies (e.g. was a CONSOLE defined for an RJPWS), and builds the tables to represent the definitions being added; it issues error messages if the definitions contain errors.

Commit

This process updates the JES3 configuration with the new definitions; these definitions are now active in the JES3 configuration. If you perform a hot start, the changes you made via the *MODIFY,CONFIG command still remain in effect.

The following examples illustrate how you would edit a member to:

1. Add an FSS managed printer (Figure 4 on page 33).
2. Add a SNA/RJP work station and its devices (Figure 5 on page 33).

```

EDIT      MYOWN.COMMON.DECKS(ADDFSS)
Command ==>
***** ***** Top of Data *****
*****
000001 *
000002 FSSDEF,
000003   TYPE=WTR,FSSNAME=DYNFSS01,PNAME=PRTSIM02,TERM=NO
000004 *
000005 DEVICE,
000006   DTYPE=PRTAFP1,JNAME=DYNFSS01,DYNAMIC=NO,CARRIAGE=(YES,8),
000007   JUNIT=(,SY1,S1,ON,,SY2,S2,ON,,SY3,S3,ON,,SY4,S4,ON,,SY5,S5,ON),
000008   HEADER=NO,BURST=NO,NPRO=1,PAGELIM=0+,LINELIM=0+,CHARS=(YES,GT15),
000009   MODE=FSS,PM=(LINE,PAGE)
000010 *
***** ***** Bottom of Data *****

```

Figure 4. Editing to Add a FSS Managed Printer

```

EDIT      MYOWN.COMMON.DECKS(ADDRJP)
Command ==>
***** ***** Top of Data *****
*****
000001 *
000002 RJPWS,
000003   N=ABC12,C=S,PR=3,PU=3
000004 *
000005 CONSOLE,
000006   TYPE=RJP,JNAME=ABC12,LEVEL=15,DEST=ALL,LL=80
000007 *
000008 DEVICE,
000009   DTYPE=RMTPRINT,JNAME=ABC12PR2,BURST=NO,HEADER=NO
000010 *
***** ***** Bottom of Data *****

```

Figure 5. Editing to Add a SNA/RJP Work Station and Its Devices

After the members are created, the operator uses the *MODIFY,CONFIG command to add the FSS printer or add the SNA/RJP work station and its devices as follows:

*F,CONFIG,ADD=ADDFSS

*F,CONFIG,ADD=ADDRJP

Notes:

1. During *MODIFY,CONFIG processing, JES3 calls your installation’s IATUX15 exit routine to examine the statements. If IATUX15 requires parameters to be passed to it, they can be specified on the *MODIFY,CONFIG command using the P= parameter.
2. You can request that a log be generated on the *MODIFY,CONFIG command. If requested, the log contains the initialization statements and any error messages that were generated.
3. Refer to *z/OS JES3 Commands* for more detail on the use of the *MODIFY,CONFIG command.

Below is an illustration of initialization statements in member ADDFSS.

```

*
FSSDEF,          Can't add a C/I FSS
  TYPE=CI,FSSNAME=DYNCFSS,PNAME=JES3CI,START=YES,TERM=YES

FSSDEF,          Duplicate FSS name as one defined above
  TYPE=WTR,FSSNAME=DYNFSS01,PNAME=PRTSIM02,TERM=NO
*

```

Figure 6. The ADDFSS Member

When you issue the *MODIFY,CONFIG to add the C/I FSS, you get a log and error messages. Some errors are detected in the syntax checking phase of processing and others errors are found in the table build phase.

```

*** LOG OUTPUT ***

IAT8351 *MODIFY,CONFIG - DATE = 1997.140, TIME = 14.59.10

IAT8351 DSN = MYOWN.COMMON.DECKS , MEMBER = ADDFSS , PARMS =

  FSSDEF,
    TYPE=WTR,FSSNAME=DYNFSS01,PNAME=PRTSIM02,TERM=NO
  *
  DEVICE,
    DTYPE=PRTAFP1,JNAME=DYNFSS01,DYNAMIC=NO,CARRIAGE=(YES,8),
    JUNIT=(,SY1,S1,ON,,SY2,S2,ON,,SY3,S3,ON,,SY4,S4,ON,,SY5,S5,ON),
    HEADER=NO,BURST=NO,NPRO=1,PAGELIM=0+,LINELIM=0+,CHARS=(YES,G,T15),
    MODE=FSS,PM=(LINE,PAGE)
  *
  FSSDEF,          Can't add a C/I FSS
    TYPE=CI,FSSNAME=DYNCFSS,PNAME=JES3CI,START=YES,TERM=YES
IAT3055 TYPE=CI FSSDEF STATEMENT CANNOT BE ADDED DYNAMICALLY

  *
  FSSDEF,          Duplicate FSS name as one defined above
    TYPE=WTR,FSSNAME=DYNFSS01,PNAME=PRTSIM02,TERM=NO
  *
IAT8348 WARNING LEVEL MESSAGE(S) ISSUED DURING INITIALIZATION PROCESSING
  *
IAT3165 MULTIPLE FSS DEFINITIONS FOR FSS DYNFSS01

```

Figure 7. *MODIFY,CONFIG Log For Member ADDFSS - Duplicate FSS Names

After the *MODIFY,CONFIG command is processed, the appropriate tables are built to represent the information that was added. The changes you made are preserved across a JES3 hot start; you do not have to issue another *MODIFY,CONFIG command. However, the changes are not preserved across a cold start, warm start, or hot start with refresh so remember to add the initialization statements to your initialization stream.

You can use the INCLUDE statement to use the same members for both *MODIFY,CONFIG and JES3 initialization. For example, suppose that the INCLUDE statement is used to group related initialization statements into separate members, and the RJPWS and its related statements are in a member named "RJPWS". Also, suppose your IATUX15 exit supports conditional logic to skip around statements depending on the parameter string passed to it. To add a new RJPWS definition, you would then do the following:

1. Add the new RJPWS definition to the existing "RJPWS" member as illustrated below.

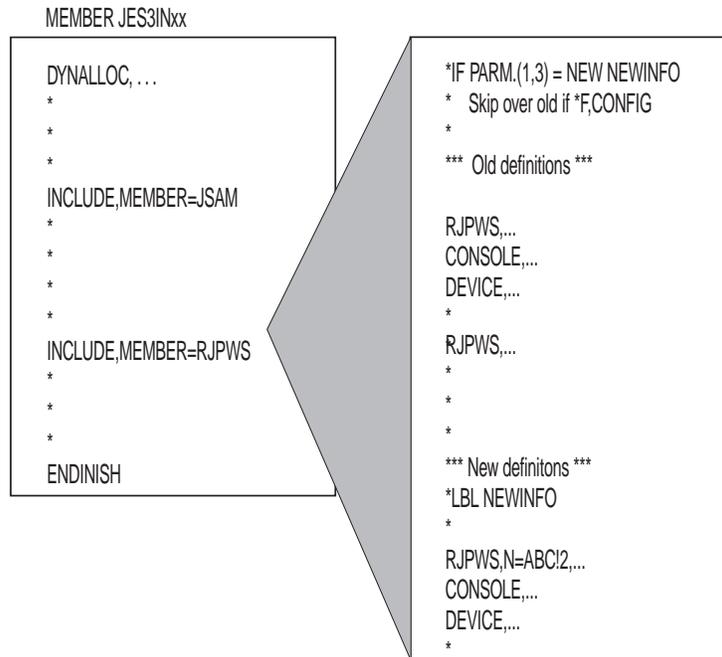


Figure 8. *MODIFY,CONFIG With IATUX15 Exit Processing

2. Add conditional logic in the RJPWS member to process only the new statements when a *MODIFY,CONFIG command is issued. You use the *IF and *LBL statements as in the example above. (This example assumes that IATUX15 has special code to recognize these statements.)
3. Issue the following *MODIFY,CONFIG command:

```
*MODIFY, CONFIG, ADD=RJPWS, P=NEW
```

By specifying "P=NEW" on *MODIFY,CONFIG, the old RJPWS definitions are skipped and only the new RJPWS definitions are processed. Since the "RJPWS" member is included by the member JES3INxx, these new RJPWS definitions are automatically picked up the next time a hot start with refresh, warm start, or cold start is performed. You do not have to update your initialization stream after issuing the *MODIFY,CONFIG command.

Part 3. Providing Security for JES3

Chapter 3. Providing Security for JES3	39
Planning for Security	39
Using RACF to Provide Security	40
How JES and RACF Work Together	43
Defining JES in the RACF Started Procedures Table	43
Forcing Batch Users to Identify Themselves to RACF	43
Defining and Grouping Operators	44
Understanding How to Provide Access to Resources	44
Universal Access Authority	44
Access Lists	45
Understanding Security Labels	45
Security Labels for JES Resources	46
JES3 Installation Exits that Affect RACF-Provided Security	46
Giving JES3 Access to Resources	47
JES User ID Early Verification	47
User ID Propagation When Jobs Are Submitted	47
Control of User ID Propagation.	48
Where NJE Jobs are Verified.	48
How SYSOUT Requests are Verified	49
Controlling Access to Data Sets JES Uses	50
Controlling Input to Your System	50
How RACF Validates Users	50
Propagation of Security Information	51
Propagating Security Information across a Network	51
Controlling the Use of Job Names	51
Controlling Who Can Submit Jobs by Job Name	52
Controlling Who Can Cancel Jobs by Job Name	53
Allowing a TSO/E User to CANCEL All Jobs Originating from Local Nodes	54
Surrogate Job Submission.	55
Authorizing the Use of Input Sources.	55
Authorizing Network Jobs and SYSOUT (NJE)	56
Authorizing Inbound Work	56
Understanding NODES Profiles.	57
Understanding Mixed Security Environments	60
Authorizing Jobs	61
Authorizing SYSOUT	63
Validating SYSOUT Based on the Submitter	65
Translating Security Information	66
Understanding Default User IDs	68
How JES Sends Security Information	69
Defining Profiles in the NODES Class	69
Defining Nodes as Local Input Sources	70
Authorizing Outbound Work	70
Controlling Password Encryption for NJE Jobs	71
Using Security Labels to Control Writers	72
Controlling Access to Data That Resides on Spool	72
Protecting SYSIN and SYSOUT	72
Defining Profiles for SYSIN and SYSOUT Data Sets	73
Allowing Users to Create Their Own JESSPOOL Profiles	75
Protecting JESNEWS	75
Protecting JESNEWS for JES3	75
How RACF Affects Jobs Dumped from and Restored to Spool (JES3 Only)	76
Dumping Jobs	76
Restoring Jobs	76
Authorizing Console Access	76
MCS Consoles	76
Remote Workstations (RJP/RJE Consoles)	76
Controlling Where Output Can Be Processed	79
Authorizing the Use of Your Installation's Printers	80
Using RACF to Authorize the Use of Operator Commands	81
Profile (Entity) Names for JES3 Commands.	82
Who Authorizes Commands When RACF Is Active	84
Using JES3 to Provide Security	85
JES3 Initialization Statements that Affect Security	86
JES3 Installation Exits that Affect JES3-Provided Security	88
Using JES3 to Control Access to RJP Workstations	88
Controlling Passwords for SNA RJP Workstations	89
Controlling Passwords for BSC RJP Workstations	89
Using JES3 to Authorize the Use of Operator Commands	89
Using JES3 to Authorize Commands from RJP Consoles	89
Using JES3 to Authorize Commands from NJE TSO/E	91
Using JES3 to Authorize Commands from MCS Consoles	91
Using JES3 to Authorize Commands Entered in Job Streams	92

Chapter 3. Providing Security for JES3

You can use initialization statements, JES installation exits, RACF 1.9, or any other functionally equivalent product, to provide security for JES.

This section describes how to use RACF to provide security for JES. The term “JES” here refers to both JES2 and JES3 except when there is a difference between the two. For additional information about RACF support for JES, see “Appendix D, Providing Security for JES” in *z/OS Security Server RACF Security Administrator’s Guide*.

Planning for Security

While every installation may have a unique set of reasons to secure data system resources, the basic reason is to prevent unauthorized personnel or programs from accessing, modifying, copying, or destroying valuable data or from damaging the system itself.

Table 6 lists some of the common benefits that you can obtain for your installation by activating security facilities for system resources:

Table 6. Benefits for Protecting JES3 Resources

Resource:	Benefits:
System data sets	Prevents unauthorized users or jobs from accessing, modifying, copying, or destroying valuable system data.
SYSIN and SYSOUT data sets	Prevents unauthorized users or jobs from reading, copying, printing, or destroying job data.
Job entry Jobnames Input sources RJP NJE	Prevents unauthorized users from entering jobs into the system with specific jobnames or entering jobs from specific ports of entry.
Console access	Prevents unauthorized operators from gaining access to MCS or RJP consoles.
Job output (writers)	Prevents unauthorized users from sending output to specific devices or remote locations.
Commands	Prevents unauthorized operators from entering commands from operator consoles.

In addition to controlling access to resources, JES3, in conjunction with (system management facilities) SMF, provides the ability to audit specific actions on resources.

You and your JES system programmer should develop a security plan for JES. Together, you should determine which resources you want to protect and decide who should have access to those resources. Your security plan should address questions such as:

- What resources must I protect?
- Should I restrict jobs and users from certain information depending on criteria such as security labels?

- Should I limit the job names users can submit or cancel?
- Is it important to protect SYSIN and SYSOUT?
- Which remote workstations should access my system?
- Can other nodes submit jobs to my system?
- To which nodes should I allow my system to send data?
- Should I limit the commands an operator can use?
- Do I want to restrict the consoles an operator can use to enter certain commands?
- What commands will I allow jobs, workstations, and nodes to submit to my system?
- Do I want only selected output devices to process particular output?
- Should the security label of the output appear on the header pages?

If JES3 is installed on your system, the JES system programmer will be using the following documents:

- z/OS Introduction and Release Guide*
- z/OS and z/OS.e Planning for Installation*
- z/OS JES3 Migration*
- z/OS JES3 Initialization and Tuning Reference*
- z/OS Security Server RACF Security Administrator's Guide*
- z/OS Security Server RACF General User's Guide*

Using RACF to Provide Security

JES3 accesses RACF services through the MVS System Authorization Facility (SAF). Through SAF, JES3 passes security information about jobs and resources to RACF, which evaluates the security information and returns the results of that evaluation to JES3. JES3 then enforces the results of the security check, such as permitting or denying access to a data set or allowing a job to process. If RACF is inactive or not installed, SAF returns default security information to JES3. JES3 may or may not perform alternative security processing depending on how you tailor your system through JES3 initialization statements or installation exits. See *z/OS MVS Programming: Authorized Assembler Services Guide* for additional information about the SAF interface.

To activate RACF protection, have your security administrator enable the RACF class for each type of resource that you want to protect. A RACF **class** is a collection of RACF-defined entities, such as data sets or devices, that have similar characteristics. In addition to activating RACF classes, the security administrator will usually also need to define RACF profiles that grant users access to these resources. Table 7 on page 40 lists the resources that you can protect, their corresponding RACF classes, and the RACF profile name format (which is sometimes referred to as an entity name) for each resource. Figure 9 on page 42 shows a cross-section of a sample CLIST that contains RACF profiles.

For additional information about defining RACF classes and profiles, see *z/OS Security Server RACF System Programmer's Guide*.

Table 7. RACF Classes Used to Protect JES3 Resources

Resource:	RACF Class:	RACF Profile (Entity) Names:(2)
Access to system data sets that JES3 uses	DATASET (always active)	data set name
Access to SYSIN and SYSOUT data sets	JESSPOOL	node.userid.jobname.jobid. dsnumber.dsname

Table 7. RACF Classes Used to Protect JES3 Resources (continued)

Access to SYSLOG	JESSPOOL	node.+MASTER+.SYSLOG.*.*?
Access to operator consoles	CONSOLE	console-id
Use of specific input sources (such as RJP, NJE, and readers)	JESINPUT (1)	devicename
Use of output destinations	WRITER	jesname.LOCAL.devicename jesname.RJP.devicename jesname.NJE.nodename
Job submission from network nodes	NODES	node.keyword.name
Use of jobnames	JESJOBS+	JESJOBS+
SUBMIT.node.jobname.userid CANCEL.nodename. userid.jobname		
Use of printers	DEVICE	sysname.dev-class. modelno.devnum (or /devnum)
Use of commands (other than RJP/NJE)	OPERCMDS	jesname.command[.qualifier]
Use of commands (from RJP workstations)	USER (always active) OPERCMDS FACILITY	workstationname jesname.command[.qualifier] (OPERCMDS) RJE.workstationname (FACILITY)
Use of commands (from NJE nodes)	USER (always active) OPERCMDS FACILITY	nodename jesname.command[.qualifier] (OPERCMDS) NJE.node (FACILITY)
<p>Notes:</p> <ol style="list-style-type: none"> 1. One profile must exist in this class when the class is active or all jobs will fail. 2. The name you specify for the variable <i>jesname</i> when creating RACF profiles is the name of your subsystem as specified in the IEFSSNxx member of the MVS SYS1.PARMLIB data set. The name you specify for the <i>sysname</i> variable is specified in the IEASYSxx member of SYS1.PARMLIB. 		

```

/*****/26500000
/* */26510000
/* CREATE OPERATOR COMMAND PROFILES */26520000
/* */26530000
/*****/26540000
/*-----*/ 26550000
/* JES3 DSP FUNCTION COMMANDS */ 26560000
/*-----*/ 26570000
26580000

RDEFINE OPERCMDS
  JES3.CALL.DR.* UACC(NONE) 26590000
PERMIT JES3.CALL.DR.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26600000
  ACCESS(UPDATE) 26610000
26620000

RDEFINE OPERCMDS JES3.CALL.* UACC(NONE) 26630000
PERMIT JES3.CALL.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26640000
  ACCESS(UPDATE) 26650000
26660000

RDEFINE OPERCMDS JES3.CANCEL.DEV.* UACC(NONE) 26670000
PERMIT JES3.CANCEL.DEV.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26680000
  ACCESS(UPDATE) 26690000
26700000

RDEFINE OPERCMDS JES3.CANCEL.* UACC(NONE) 26710000
PERMIT JES3.CANCEL.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26720000
  ACCESS(UPDATE) 26730000
26740000

RDEFINE OPERCMDS JES3.START.DR.* UACC(NONE) 26750000
PERMIT JES3.START.DR.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26760000
  ACCESS(UPDATE) 26770000
26780000

RDEFINE OPERCMDS JES3.START.DEV.* UACC(NONE) 26790000
PERMIT JES3.START.DEV.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26800000
  ACCESS(UPDATE) 26810000
26820000

RDEFINE OPERCMDS JES3.START.* UACC(NONE) 26830000
PERMIT JES3.START.* CLASS(OPERCMDS) ID(OPER7 OPER21 OPER24) + 26840000
  ACCESS(UPDATE) 26850000
  /*****/ 29940000
  /* CREATE JES3 SPOOL DATASET PROFILES */ 29960000
  /*****/ 29990000
30000000

RDEFINE JESSPOOL NODE1.JES3.*.JOB00000.D0000000.JNEWSLCL UACC(READ) + 30010000
  AUDIT(ALL) 30020000
RDEFINE JESSPOOL NODE3.JES3.*.JOB00000.D0000000.JNEWSRJP UACC(READ) + 30030000
  AUDIT(ALL) 30040000
RDEFINE JESSPOOL NODE3.JES3.*.JOB00000.D0000000.JNEWSTSO UACC(READ) + 30050000
  AUDIT(ALL) 30060000
30070000

PERMIT NODE1.JES3.*.JOB00000.D0000000.JNEWSLCL CL(JESSPOOL) + 30080000
  ACCESS(UPDATE) ID(OPER0 OPER24 TSOG0 SUPERU) 30090000
PERMIT NODE2.JES3.*.JOB00000.D0000000.JNEWSRJP CL(JESSPOOL) + 30100000
  ACCESS(UPDATE) ID(OPER0 OPER24 TSOG0 SUPERU) 30110000
PERMIT NODE3.JES3.*.JOB00000.D0000000.JNEWSTSO CL(JESSPOOL) + 30120000
  ACCESS(UPDATE) ID(OPER0 OPER24 TSOG0 SUPERU) 30130000

```

Figure 9. Sample Section of a CLIST that Defines RACF Profiles

How JES and RACF Work Together

JES requests RACF services by issuing the RACROUTE macro. SAF processes the RACROUTE macro invocation. If RACF is installed, SAF passes the security information specified by JES on the RACROUTE macro invocation to RACF. RACF evaluates the security information and returns the results of that evaluation to JES. JES then enforces the results of the security check, such as permitting or denying access to a data set or allowing a job to run.

JES Code That Can Bypass RACF Protection

Your installation can use JES initialization statements and JES installation exits to provide protection and, in some cases, to bypass RACF protection. For more information, see *z/OS JES3 Initialization and Tuning Reference* and *z/OS JES3 Initialization and Tuning Guide*.

Defining JES in the RACF Started Procedures Table

For performance and ease of migration, when RACF 1.9 is first installed, JES must be defined as a started procedure *with the trusted attribute* in the RACF started procedures table (module ICHRIN03).

To do this, take the following steps:

1. Ask your JES system programmer for the name of your JES started procedure.
2. Verify that the ICHRIN03 module contains the name of your JES started procedure. If none is defined, create an entry for JES. Make sure that JES has the *trusted* attribute.
3. Create a user profile for the JES started task:

```
ADDUSER  jes-userid
         DATA('JES started task')
         PASSWORD(password)
         DFLTGRP(appropriate-group)
```

For more information about adding a started procedure, see “Using Started Procedures on MVS” in *z/OS Security Server RACF Security Administrator’s Guide* and *z/OS Security Server RACF System Programmer’s Guide*.

Forcing Batch Users to Identify Themselves to RACF

To prevent unauthorized users from running batch jobs, you can require all batch jobs to have RACF identification. To do this, enter the following:

```
SETROPTS JES(BATCHALLRACF)
```

When you specify BATCHALLRACF, any batch job that does not have a RACF-defined user specified on the USER parameter of the JOB statement, or propagated security information associated with it, will fail.

Specifying NOBATCHALLRACF allows such jobs to run.

Defining and Grouping Operators

Your security plan may require that system programmers and operators at your installation be defined to RACF. To improve accountability, you should create user profiles for any persons who:

- Enter JES commands
- Enter commands from an MCS-managed console
- Update system data sets that JES uses.

You can organize your installation's support personnel, particularly operators, into groups that are responsible for a particular area. For example, you may want to group your installation's support personnel by shift, functional area, or both.

To identify and group your installation's support personnel you should:

- List the userIDs of all the system programmers and operators at your installation
- Group any of the userIDs together if they:
 - Perform similar tasks
 - Work on the same shift
 - Are responsible for the same area

If you decide to combine users into groups, you will need to:

- Record all the userIDs in the group
- Describe why the users were grouped together
- Assign a unique name to the group.

You will want to keep a record of userIDs and group IDs handy for use in securing other system resources such as spool data, console access, and commands as well as for updating groups in the future.

Understanding How to Provide Access to Resources

RACF allows you to permit access to resources with:

- Universal access authority (UACC)
- Access lists

Universal Access Authority

The universal access authority assigned to a resource defines the access that all users have to that resource unless the users are explicitly defined in an access list. The meaning of universal access values varies depending on the type of resource that is being defined. For example, the following UACC values are valid for resources defined through the RACF DATASET class:

NONE

Specifies that the user or group of users is not permitted to access the resource.

READ Specifies the user can only read from the resource.

UPDATE

Specifies the user can read from or write to the resource.

EXECUTE

Specifies a user can load and run a program from a private program library but cannot read or copy the program. This access authority applies only if you have installed Version 3 Release 1 or a subsequent release of the MVS Data Facility Product (5665-XA3).

ALTER

Specifies a user has the maximum access RACF allows to the resource and permits a user to update discrete profiles.

For example, if you assign a UACC value of READ to a data set named WORKLOAD, then all users would be able to read the data set. However, users could not update WORKLOAD unless they were explicitly allowed to do so through an access list.

Access Lists

You can define exceptions to the universal access authority assigned to a resource by defining the authority of a user in an access list associated with a resource. For example, you can assign a UACC value of READ to WORKLOAD and assign an access value of UPDATE to userid TOMN. In this case, TOMN would be able to write to WORKLOAD. The authority granted to a user or group of users through an access list effectively overrides the universal access authority associated with a resource. See *z/OS Security Server RACF Security Administrator's Guide* for additional information about defining access authority.

Understanding Security Labels

When you use RACF, you can provide an additional layer of security by assigning security labels to users and resources. When active, security labels identify the security clearance of users (such as TSO/E users) and the classification or sensitivity of resources (such as data sets).

These security labels are used to evaluate the security label assigned to a user against the security label assigned to a resource. The highest label in an installation is termed **SYSHIGH**, the lowest, **SYSLOW**. Both SYSHIGH and SYSLOW can be used as valid security labels when defining RACF profiles. In addition to SYSHIGH and SYSLOW, your installation can define other security labels in between.

Attention: Security labels defined through RACF must be consistent throughout a JES3 complex to prevent information from being declassified.

To create security labels, the security administrator defines a security level and zero or more categories to a security label. The security administrator then assigns a security label to each user and resource that requires one.

Table 8 on page 46 illustrates how the resulting security system works for the XYZ Corporation. It shows which employees are allowed to access what kind of data and resources. In the table, TOP SECRET, CONFIDENTIAL, and NON-SENSITIVE are **security levels**; Project A, Project B, and Project C are **security categories** and the names within parentheses are **security labels**.

As the security level becomes higher, fewer people are allowed access to specific data. For instance, Tony is the only person assigned a security label **LABEL7**, which allows him to access top secret resources associated with Project A. Another person, Joan, can access data and resources for both Project A and Project B. However, while she can access both CONFIDENTIAL and NON-SENSITIVE data on Project B, in Project A she is restricted to NON-SENSITIVE resources.

Table 8. Assigning Security Labels for the XYZ Corporation

	Project A	Project B	Project C
TOP SECRET	(LABEL7) Tony	(LABEL8) Cathy	(LABEL9) Jim
CONFIDENTIAL	(LABEL4) Tony, Rita, Philip	(LABEL5) Cathy, Joan	(LABEL6) Jim, Mike, Curtis
NON-SENSITIVE	(LABEL1) Tony, Rita, Philip, Joan	(LABEL2) Cathy, Joan, Ron, Maureen	(LABEL3) Jim, Mike, Curtis, Jerry

You can use RACF to set up a security system like the one illustrated in Table 8. See *z/OS Security Server RACF Security Administrator's Guide* for additional information about how to define and use RACF security labels.

Security Labels for JES Resources

When your installation uses security labels, each protected JES resource can have a security label associated with it. For spooled data sets, JES maintains the security label with the data set itself (not in a RACF profile). For other resources, such as consoles and DASD data sets, RACF maintains the security label in the resource profile protecting the resource.

Attention: Security labels should be consistent throughout a JES complex to prevent information from being declassified. To check what your system configuration should be, see “The Trusted Computing Base”.

JES3 Installation Exits that Affect RACF-Provided Security

You can modify JES3 to accept or reject RACF's security evaluations through JES3 installation exits. Table 9 on page 46 lists the installation exit(s) available for customizing security.

Table 9. Installation Exits that Affect JES3 Security When RACF Is Active

JES3 Installation Exit:	Purpose:
IATUX18	This exit allows you to modify a JES3 command and validate the console's authority level for entering the command.
IATUX30	This exit allows an installation that does not use the standard TSO/E jobname (userid plus one character) to authorize the use of STATUS, CANCEL, and OUTPUT commands. For the TSO/E CANCEL command, you can use IATUX30 to authorize or reject the request. If IATUX30 rejects the request, no further security processing occurs; the request is canceled. If accepted by IATUX30, then the request must also pass authority checks in IATUX58 and IATUX59 (if present) before the command is allowed to process.
IATUX35	This exit allows you to authorize commands received from network nodes. You can use this exit to accept or reject commands, or have JES3 use RACF command authorization.
IATUX58	This exit, taken prior to a security call to SAF, allows you to: <ul style="list-style-type: none"> • Accept the security request and bypass SAF • Reject the security request and bypass SAF • Use JES3 default processing and bypass SAF • Call SAF and make the subsequent call to installation exit IATUX59 • Call SAF but not IATUX59 • Change most of the parameters passed to SAF.

Table 9. Installation Exits that Affect JES3 Security When RACF Is Active (continued)

IATUX59	This exit, taken after a security call to SAF, allows you to: <ul style="list-style-type: none"> • Accept a security request, regardless of the SAF decision • Reject a security request, regardless of the SAF decision • Use JES3 default processing, regardless of the SAF decision • Honor the decision by SAF • Pass back messages to be written in the JESMSGLG data set.
IATUX60	This exit is invoked when RACF determines that the receiver of the data set currently has insufficient authority to receive data when using a TSO/E RECEIVE command, nor does the receiver have the capability of running with sufficient authority to receive the data set (that is, logoff and then logon again with a higher security label). You can use this exit to specify whether the data set should be retained or deleted (default).
IATUX67	This exit, taken when receiving a SYSOUT data set from another node and RACF has rejected validation for the job, allows you to override the security decision. You can direct the SYSOUT data set to be purged (default), held for TSO/E, or allow JES3 to process the job normally (for example, when a data set needs to print). When you allow a job to process normally, you are overriding the RACF decision.

See *z/OS JES3 Customization* for information about how to use these exits.

Giving JES3 Access to Resources

JES3 uses many resources to control its own processing and requires sufficient authority to access them. **Your RACF support personnel must define JES3, JES3 functional subsystems (CI and writer), and the names of started procedures in the RACF started procedures table with the trusted attribute so JES3 has access to every resource it might use.** The name you supply to your administrator is in member IEFSSNxx of SYS1.PARMLIB. The following sections discuss the resources you can control, the value of controlling the resource, a methodology for controlling the resource, and how to control the resource using RACF.

JES User ID Early Verification

Early verification is always done even if the SETROPTS command has been issued with JES(NOEARLYVERIFY) specified.

User ID Propagation When Jobs Are Submitted

For each previously-validated RACF user who is submitting a batch job to JES through a JES internal reader, SAF propagates the following security information to the batch job:

- If USER is not specified on the JOB statement, the current RACF userID is used.
- If PASSWORD is not specified on the JOB statement, the current user password is not required if the submitter propagates.
- If SECLABEL is not specified on the JOB statement, the submitter's current security label is used.

Note: If GROUP is not specified on the JOB statement, the default connect group is used from the user profile of the user used for the job.

This has the following advantages:

- It reduces the possible exposure of security information (especially passwords) stored in clear text in JCL.

- It reduces administrative overhead of maintaining RACF userIDs, passwords, and security labels in the JOB statements for all batch jobs.

As a result, a TSO/E user, for example, is not required to specify this security information for each job submitted.

Note: You can prevent userID propagation for specific users. See “Control of User ID Propagation”.

Control of User ID Propagation

In some environments, such as CICS, jobs submitted without the USER operand specified on the JOB statement run under a userID other than the user submitting the job. For example, if a user running under CICS submits a batch job without specifying a userID on the JOB statement, the job runs under the CICS userID and has the access authorities of the CICS userID.

You can prevent the CICS userID from being propagated to these batch jobs by defining a profile whose name is the CICS userID.

1. Define a profile in the PROPCNTL class where the profile name is the userID that is not to be propagated (in this case, userID CICS1 is not to be propagated):

```
RDEFINE PROPCNTL CICS1
```

2. Activate the PROPCNTL class:

```
SETROPTS CLASSACT(PROPCNTL)
```

3. Enter the SETROPTS command with the RACLIST operand to activate SETROPTS RACLIST processing for the PROPCNTL class:

```
SETROPTS RACLIST(PROPCNTL)
```

If you do not activate SETROPTS RACLIST processing for the PROPCNTL class, RACF will ignore the profiles you create in this class.

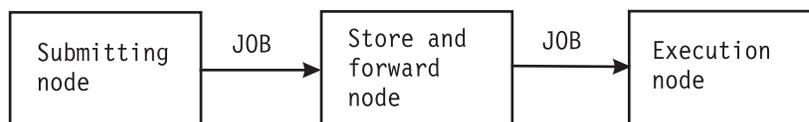
The above sequence of commands eliminates userID propagation of the userID CICS1.

Notes:

1. For a profile in the PROPCNTL class, UACC and entries in an access list only have a meaning for maintaining the profile itself and for controlling propagation. RACF checks only for the presence or absence of a profile in this class. If a profile exists for a particular userID, userID propagation will not occur for that userID.
2. RACF performs no logging and issues no messages for profiles in the PROPCNTL class.

Where NJE Jobs are Verified

The following is a simple network showing the path of a job.



User verification for NJE jobs normally is done at the execution node. However, RACF authorization checking may occur additionally at the submitting node, if:

- Using the JES2 /*ROUTE XEQ statement or /*XEQ statement are verified at the execution node only.
- Using the JES2 /*XMIT statement or the JES3 /*ROUTE XEQ or //XMIT statement have their first JOB statement verified at the submitting node and their second JOB statement verified at the execution node.

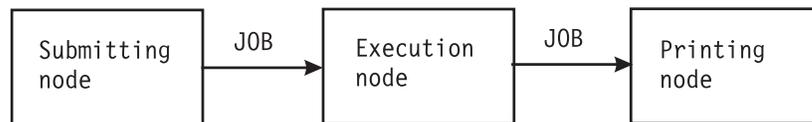
Submitter information is propagated from trusted nodes. The submitter information is:

- The token for a verified first job card
- The original submitter's token if the job was submitted from an internal reader
- The "unknown user" token if the job was submitted from a physical reader
- NJE header information (no token available) if the job was submitted from a downlevel node.

Whether a job is accepted is based on some combination of the submitter's ID, group, or seclabel. Whether security information is propagated and translated is based on the submitter's ID (as taken from above). Job acceptance and translation is done using profiles in the NODES class. RACF finds the best fit among the profiles in the NODES class and uses the information specified in the UACC and ADDMEM information.

How SYSOUT Requests are Verified

The following is a simple network showing the path of a job:



For inbound SYSOUT, user verification occurs at the printing node instead of the submitting node (as it can for inbound jobs). On the printing node, RACF authorization checking occurs in the NODES class, as it does for inbound jobs. RACF finds the best fit among the profiles in the NODES class and uses the information specified in the UACC and ADDMEM information.

Whether the SYSOUT is accepted is based on some combination of the owner's ID, group, or seclabel. Whether the security information is accepted and translated is based on the owner's ID taken from:

- The job token from the NJE header as verified at the executing node
- If no token is available (SYSOUT is from a downlevel node), the owner is considered to be the NJE undefined user as defined by:
`SETROPTS(JES(NJEUSERID(userid)))`

In addition, if &SUSER (submitting user) is specified on the ADDMEM operand, the submitter can be used as the owner if one of the following is also true:

- The submitting node is defined as a local node in the &RACLNDE profile in the RACFVARS class.
- The NODES profile that matches is the profile named *submitting-node.USERS.submitter* and UACC(CONTROL) is specified.

Controlling Access to Data Sets JES Uses

The JES spool and checkpoint data sets are critical for proper operation of your JES system. It is critical that JES be the only user that can update the information in these data sets. However, it is also important for a limited group of users to be able to recreate the spool and checkpoint data sets (should the data set become unusable because of hardware problems). Also, restrict access to the data set that contains the modules that JES uses. Make sure profiles exist for any data sets you might use for JES checkpoint reconfiguration.

You can define DATASET profiles to protect the system data sets that JES uses to control its own processing. Protecting your installation's system data sets prevents unauthorized users or jobs from accessing, modifying, or destroying valuable system data.

The JES system programmer should supply you with the following information for each data set to be protected:

- The name of the data set
- The universal access authority to be associated with the data set
- The security label to be associated with the data set (if labels are being used)
- Whether audit records should be generated:
 - Each time the data is accessed
 - When an unauthorized attempt is made to access the data set
 - When an authorized attempt is made to access the data set.

Make sure to define JES as a started procedure with the trusted attribute. See "Defining JES in the RACF Started Procedures Table" on page 43.

Controlling Input to Your System

You can use RACF to ensure that jobs entering your JES system are authorized for processing. JES carries security information about the submitter of a job internally and invokes the services of RACF at key points during JES processing, such as when a job enters the system through a TSO/E SUBMIT command or through a device reader.

You can protect several job entry resources including the use of job names and sources of job entry such as internal readers, device readers, RJP, RJE, and network nodes. This section describes how RACF validates users, how to control the use of job names, and how to control the use of input sources. "Authorizing Network Jobs and SYSOUT (NJE)" on page 56 provides a separate discussion about how to control security for inbound and outbound network jobs and SYSOUT.

How RACF Validates Users

When RACF is active, RACF ensures that the job's password, userID, group ID, and security label are valid before allowing the job to be processed. If security labels are being used, JES obtains the label from the job card. If the job card does not specify a label, RACF obtains the security label from the RACF profile associated with that job's userID. If no security label exists in the RACF profile, the job is automatically assigned a security label of SYSLOW.

The extent to which RACF performs user validation for jobs entering the system through NJE nodes depends on the universal access authority assigned to that

node. “Authorizing Network Jobs and SYSOUT (NJE)” on page 56 lists those values and their effects on user validation.

You can allow the setting up of surrogate users. A surrogate user is a user who submits jobs on behalf of another user.

Surrogate job submission allows a user to submit jobs on behalf of another user without having to specify the original user’s password. Jobs submitted by a surrogate user run with the identity of the original user. Although the surrogate user does not have to provide the password of the original user, RACF ensures that the job’s security label overrides the surrogate’s security label and that the original user is authorized to use the security label associated with the job. For additional information about defining surrogate job submission, see “Surrogate Job Submission” on page 55.

Propagation of Security Information

If RACF is active and a job enters the system with some or all security information missing, SAF propagates the submitter’s security information to the job. For example, if JOB1 submits JOB2, and JOB2 does not have SECLABEL= specified on the JOB JCL statement, SAF passes the value of SECLABEL= from JOB1’s JOB statement to JOB2.

Any user in a currently active session (for example, TSO/E or an executing batch job) can have SAF propagate the security information associated with the session by omitting the information on the JOB statement for the job. If SAF can not determine the group or Security label information from the current session, SAF uses the default information in the RACF profile. However, SAF will not propagate security information to a job that enters the system from an RJE work station or a physical card reader.

Propagating Security Information across a Network

To allow users to submit jobs without specifying security information such as a userID, JES propagates the submitter’s security information when the information is omitted. For example, you can have a user omit a password from the JOB statement if you do not want to send passwords through the network. Propagation also occurs when you use the /*XMIT card to transmit a job to another node in the network. In this case, SAF passes the information that appears in the first JOB statement to the JOB statement that follows the /*XMIT card.

If SAF is unable to verify the security information on the first JOB statement:

- If a TSO/E user submitted the job, SAF passes the TSO/E user’s security information to the second JOB statement.
- If the job entered the system through a local card reader, SAF uses a default userID for propagation purposes. (See “Understanding Default User IDs” on page 68 for information about the default userID.)

“How JES Sends Security Information” on page 69 describes what security information RACF propagates for NJE jobs.

Controlling the Use of Job Names

You can use profiles in the JESJOBS class to control which job names users can submit or cancel.

Because most installations have many jobs and users, it might not be practical to define all of the profiles needed to authorize every job name. A more reasonable approach would be to restrict the use of only certain job names.

You can specify which job names can be entered from a specific device. For more information, see “Conditional Access Lists for General Resource Profiles” in *z/OS Security Server RACF Security Administrator’s Guide*.

Notes:

1. TSO/E installation exit IKJEFF53 must be modified to become a dummy exit when the JESJOBS class is active. For specific information, see *z/OS TSO/E Customization*.
2. At least one generic profile with a universal access of READ is required for the TSO/E SUBMIT command when the JESJOBS class is active. A generic profile is not required for the TSO/E CANCEL command. Without a generic profile, however, only the owner of a job or those granted access through RACF profiles can use the CANCEL command on data sets associated with that job.

Controlling Who Can Submit Jobs by Job Name

To control who can submit jobs by job name, take the following steps:

1. Ask your TSO/E system programmer to ensure that TSO/E installation exit IKJEFF53 checks if the JESSPOOL or JESJOBS class is active, and if either is active, returns to the caller with no action. For specific information, see *z/OS TSO/E Customization*.

Note: SYS1.SAMPLIB contains a sample of such an exit (supplied with TSO/E 2.1.1 or later).

2. Define at least one profile with a universal access of READ to allow users to submit jobs when the JESJOBS class is activated:

```
RDEFINE JESJOBS ** UACC(READ)
```

Note: This example assumes that a SETROPTS GENERIC(JESJOBS) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

3. Define profiles with UACC(NONE) for the job names you want to protect.

```
RDEFINE JESJOBS SUBMIT.nodename.jobname.userid UACC(NONE)
```

where:

nodename

is your local node name.

Note: It is recommended that you define a profile in the RACFVARS class named &RACLNDE, and use &RACLNDE for all profiles in the JESJOBS class.

jobname

is the name of the job specified on the JOB statement.

userid is the userID the job will run under (either the USER operand on the JOB statement or the propagated userID).

For example,

```
RDEFINE JESJOBS SUBMIT.*.PAYROLL*.* UACC(NONE)
```

This command would prevent any user from submitting jobs whose job names begin with PAYROLL.

4. To allow users to submit jobs protected by the profile, give them READ access to the profile:

```
PERMIT SUBMIT.*.PAYROLL*.* CLASS(JESJOBS) ID(PAYGROUP) ACCESS(READ)
```

Note: By denying a user sufficient access to a SUBMIT profile, you can prevent that user from submitting jobs protected by the profile *even if that user knows the password or is an authorized surrogate user.*

For example, the following profile would prevent users from submitting jobs with USER=IBMUSER specified on the JOB card:

```
RDEFINE JESJOBS SUBMIT.*.*.IBMUSER UACC(NONE)
```

You can also define conditions that prevent the users from submitting jobs except when using a specific device. For example, you can specify the WHEN(JESINPUT) operand on the PERMIT command as follows:

```
PERMIT SUBMIT.*.PAYROLL*.* CLASS(JESJOBS) ID(PAYGROUP)
ACCESS(READ) WHEN(JESINPUT(device-name))
```

where *device-name* is the JES input device that the user must use.

5. When you are ready to use the JESJOBS class to control who can submit jobs, activate the JESJOBS class:

```
SETROPTS CLASSACT(JESJOBS)
```

Note: If you activate this class and create no profiles for it, users will be prevented from submitting batch jobs.

Controlling Who Can Cancel Jobs by Job Name

A user is always authorized to cancel a job that has been submitted. Using RACF, you can control who can use the TSO/E CANCEL command to cancel jobs, depending on the job names. To do this, take the following steps:

1. Ask the TSO/E system programmer to change TSO/E installation exit IKJEFF53 to become a dummy exit. For specific information, see *z/OS TSO/E Customization*.
2. Define profiles for the job names you want to protect:

```
RDEFINE JESJOBS CANCEL.nodename.userid.jobname UACC(NONE)
```

Note: The qualifiers for CANCEL profiles have the same meaning as for SUBMIT profiles. The *jobname* and *userid* qualifiers, however, are reversed in CANCEL and SUBMIT profiles because of the expected use of the profiles:

- It is likely that many users would submit jobs having common job names, with certain exceptions. For example, the following profiles:

```
RDEFINE JESJOBS SUBMIT.*.PAYROLL*.* UACC(READ)
```

```
RDEFINE JESJOBS SUBMIT.*.PAYROLL*.BEN UACC(NONE)
```

would allow many users to submit jobs whose names begin with PAYROLL, except when those jobs run with BEN's authority.

- It is likely that one user would give another user the authority to cancel all of the first user's jobs, with certain exceptions. For example, the following profiles:

```
RDEFINE JESJOBS CANCEL.*.BEN.* UACC(NONE)
```

```
PERMIT CANCEL.*.BEN.* CLASS(JESJOBS) ID(JOE) ACCESS(ALTER)
```

```
RDEFINE JESJOBS CANCEL.*.BEN.PAYROLL* UACC(NONE)
```

would allow JOE the authority to cancel BEN's jobs, except for his PAYROLL jobs.

- These examples assume that a SETROPTS GENERIC(JESJOBS) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.
3. Give users the appropriate access authority to the profile:

```
PERMIT CANCEL.*.*.PAYROLL* CLASS(JESJOBS) ID(PAYGROUP) ACCESS(ALTER)
```

Users must have ALTER access authority to the profile to enter the CANCEL command for the job.
 4. If the JESJOBS class is not already active, activate the JESJOBS class by entering:

```
SETROPTS CLASSACT(JESJOBS)
```

Allowing a TSO/E User to CANCEL All Jobs Originating from Local Nodes

To allow a TSO/E user to cancel all jobs that originate on nodes you treat as local nodes, do the following:

1. Define a profile named &RACLNDE in the RACFVARS class, specifying on the ADDMEM operand which nodes are to be treated as local:

```
RDEFINE RACFVARS &RACLNDE UACC(NONE)
        ADDMEM(POKMVS1 POKMVS2)
```

Note: UACC(NONE) is recommended to protect the &RACLNDE profile itself.

2. Define a profile in the JESJOBS class as follows:

```
RDEFINE JESJOBS CANCEL.&RACLNDE *.* UACC(NONE)
```

Note: This example assumes that a SETROPTS GENERIC(classname) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

3. Give the appropriate access to the TSO/E user, using the ALTER option of the ACCESS parameter.

```
PERMIT CANCEL.&RACLNDE *.* CLASS(JESJOBS)
        ID(USER1) ACCESS(ALTER)
```

Note: If there are any other JESJOBS profiles that begin with CANCEL, you may also need to permit the user to access those profiles.

4. If you have not already done so, activate the JESJOBS and RACFVARS classes:

```
SETROPTS CLASSACT(JESJOBS RACFVARS)
```

5. Refresh SETROPTS RACLIST processing for the RACFVARS class for the change to take effect:

```
SETROPTS RACLIST(RACFVARS) REFRESH
```

If, later, you decide that node POKMVS2 should no longer be treated as a local node, do the following:

```
RALTER RACFVARS &RACLNDE DELMEM(POKMVS2)
```

```
SETROPTS RACLIST(RACFVARS) REFRESH
```

```
SETROPTS GENERIC(JESJOBS) REFRESH
```

If, later, you decide that USER2 should also be allowed to cancel local jobs, do the following:

```
PERMIT CANCEL.&RACLNDE *.* CLASS(JESJOBS)
        ID(USER2) ACCESS(ALTER)
```

```
SETROPTS GENERIC(JESJOBS) REFRESH
```

Surrogate Job Submission

With RACF 1.9 or later and JES 3.1.3 or later installed, you can allow the use of surrogate users on MVS. A surrogate user is a RACF-defined user who has been authorized to submit jobs on behalf of another user (the original user) without having to specify the original user's password. Jobs submitted by the surrogate user process with the identity of the original user. This can be useful when a person is assuming a production workload for someone going on vacation or a leave of absence.

For information about setting up surrogate users, see "surrogate user" on page 51.

Authorizing the Use of Input Sources

You can use RACF to limit which sources of input are valid for job submission, including RJP workstations, device readers, nodes, and internal readers. For example, you might want to prevent certain users from entering jobs from a particular RJP workstation.

To authorize the submission of work from specific input sources, take the following steps:

1. Ask your JES system programmer for the following information:
 - The name of the device. This is described in the section on authorizing the use of input sources in the JES3 Initialization and Tuning document for your system.
 - The userID or group ID of the users you want to authorize or restrict.
 - The universal access authority to associate with each device. Valid access authorities for input devices are:

NONE

Specifies that the input device can be used only by those users explicitly permitted through the access list.

READ

Specifies the minimum authority required to use the input source.

2. Define a profile for each input source, as follows:

```
RDEFINE JESINPUT source-name UACC(NONE)
```

3. IBM *strongly recommends* that you create a profile with a UACC of READ for all JES input sources that are otherwise not defined:

```
RDEFINE JESINPUT ** UACC(READ)
```

If you do not, users will be able to access only JES input sources to which they (or their groups) are explicitly authorized.

Note: This example assumes that a SETROPTS GENERIC(JESINPUT) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

4. For each protected input source, grant access to the users or groups who need to use it:

```
PERMIT source-name CLASS(JESINPUT)  
      ID(user or group) ACCESS(READ)
```

5. When you are ready to start using the protection provided by the profiles you have created, activate the JESINPUT class:

```
SETROPTS CLASSACT(JESINPUT)
```

Note: If you activate this class and create no profiles for it, users will be prevented from submitting batch jobs.

Authorizing Network Jobs and SYSOUT (NJE)

Use RACF to ensure that all work entering or leaving your node complies with your installation's security policy. You can control:

- Jobs and data received from other nodes in a network, as long as the inbound job or data includes a standard NJE header. This includes jobs or data from an RSCS node.
- The extent of security validation performed at your node.
- Jobs and data destined for other nodes in a network.

JES does not validate work passing through your node on its way to another node in the network, but it does protect the work from unauthorized access while the work is temporarily stored on spool at your node.

To provide security for Network Job Entry (NJE), activate the NODES class (for inbound work) or the WRITER class (for outbound work), and define the profiles needed to enforce your installation's NJE security policy. As with other RACF-protected resources, you must gather information from your JES system programmer to define profiles.

Before you request this information, you must first work with your JES system programmer to decide whether you want to protect inbound work, outbound work, or both. For inbound work, you must decide whether you want to protect jobs, SYSOUT, or both. You must also determine which users and/or groups of users you will allow to submit work, and the security labels that are valid for processing on your node.

Authorizing Inbound Work

The following list outlines the topics discussed in this section:

- "Understanding NODES Profiles" on page 57 explains how the values you select for the NODES profile determines what type of work and which users or security labels you want RACF to validate.
- "Understanding Mixed Security Environments" on page 60 explains how different levels of JES and RACF affect security processing.
- "Authorizing Jobs" on page 61 explains how you can use RACF to validate inbound jobs from other nodes in a network.
- "Authorizing SYSOUT" on page 63 explains how you can use RACF to validate inbound job output (SYSOUT) from other nodes in a network.
- "Validating SYSOUT Based on the Submitter" on page 65 explains how RACF can be used to validate inbound work using the submitter's security information instead of the owner's security information.
- "Translating Security Information" on page 66 explains how RACF can be used to replace inbound userIDs, group IDs, or security labels with locally-defined values.
- "Understanding Default User IDs" on page 68 describes how JES handles security information for work from incompatible nodes and explains how JES handles security information belonging to store-and-forward work. This section also describes how you can use RACF to manipulate default security information.
- "How JES Sends Security Information" on page 69 explains where JES obtains missing security information for NJE work.

- “Defining Profiles in the NODES Class” on page 69 shows you how to set up the protection defined in the profiles and how to activate not only the NODES class but also the SETROPTS RACLIST processing for the class.
- “Defining Nodes as Local Input Sources” on page 70 explains how you can use RACF to treat SYSOUT from another node as if the SYSOUT originated at the home node.

Understanding NODES Profiles

You can use profiles in the NODES class to control how RACF validates inbound work on an NJE network. As with other RACF profiles, NODES profiles consist of a profile name, a profile class, a universal access authority, and an ADDMEM value. The profile name is a three-part identifier that indicates the origin of the work and the type of security information you want to validate. The universal access authority determines the actions that RACF performs on the inbound work. This information is described in Table 10 on page 62 and Table 11 on page 64.

Note: Access lists do not apply to NODES class profiles. The ADDMEM value is used to translate to locally-defined values.

A NODES profile name has the following format:

nodeid.keyword.name

where:

nodeid

is the name of the node from which you expect inbound work. For jobs, this is the submitting node. For SYSOUT, this is the execution node.

Notes:

1. If &SUSER is specified as an ADDMEM value in a profile that controls SYSOUT, a second check is done where the nodeid is the submitting node.
2. IBM recommends that you define a profile in the RACFVARS class named &RACLNDE, and use &RACLNDE for all nodes that are considered local to your system. For more information, see “Setting up NODES Profiles” on page 59, step 4. 4 on page 59.

keyword

is the type of work to be controlled by the profile. (Notice that the *last character*, J or S, indicates the type of work to be validated. J indicates jobs; S indicates SYSOUT.)

RUSER

This profile controls commands originating from NJE nodes. The *nodeid* is used as the name on the third qualifier.

USERJ

This profile controls jobs by the userID specified on the third qualifier. The job is controlled by the submitter. This type of profile is also used to determine the amount of trust the job has. Refer to “Understanding Mixed Security Environments” on page 60 for details.

USERS

This profile controls SYSOUT by the userID specified on the third qualifier. The SYSOUT is controlled by the owner. This type of

profile is also used to determine the amount of trust the SYSOUT has. Refer to “Understanding Mixed Security Environments” on page 60 for details.

GROUPJ

This profile controls jobs by the group ID specified on the third qualifier.

GROUPS

This profile controls SYSOUT by the group ID specified on the third qualifier.

SECLJ This profile controls jobs by the security label specified on the third qualifier.

SECLS

This profile controls SYSOUT by the security label specified on the third qualifier.

For example, a value of USERJ specifies that you want RACF to use the profile to validate inbound jobs; a value of USERS specifies that you want RACF to use the profile to validate inbound SYSOUT.

name is the actual userID, group ID, or security label you want validated. If you are using NODES profiles to allow the use of these input values, you must either define these values in your RACF database or use the ADDMEM operand to translate them into acceptable values for your system. For jobs, the submitter information is substituted. For SYSOUT, the owner information is used. (See “Understanding Mixed Security Environments” on page 60.)

For example, the following profile:

```
BERMUDA.USERJ.WAYNE
```

controls whether jobs coming from userID WAYNE at node BERMUDA can process here. You can optionally associate a local userID with userID WAYNE by specifying the userID on the ADDMEM operand.

You can specify generic characters in the profile name to control a wider range of work. For example, if you place an asterisk in place of the nodeid value, RACF performs the requested type of validation for work from all nodes in the network (unless a more specific profile exists). Examples of generic profiles in the NODES class are shown later. For more information, see “Choosing between Discrete and Generic Profiles in General Resource Classes” in *z/OS Security Server RACF Security Administrator's Guide*.

If you have installed RACF 1.9 and do not activate the NODES class, JES validates jobs and SYSOUT in the following manner:

- JES runs only those jobs that are destined for your node and that have a valid userID and password on the job card if BATCHALLRACF is active. If BATCHALLRACF is not active, the job may run without a RACF userID.
- A security label of SYSHIGH is assigned to all SYSOUT destined for your node (if security labels are being used) and may be printed only on those devices permitted to SYSHIGH data. JES assigns the default userID to this SYSOUT. See “Understanding Default User IDs” on page 68 for information about default userIDs.
- All work destined for another node remains unchanged.

If you choose to activate the NODES class, you must gather information from your JES system programmer so that you can set up profiles to control the work entering your system. The following sections identify the appropriate values for each type of work.

Setting up NODES Profiles: Use the following approach to setting up NODES profiles:

1. Define a profile for each node for which you want to control inbound work. (If you have several nodes that you are treating identically, consider creating RACFVARS profiles and using the RACF variables in NODES profile names. This can reduce the number of NODES profiles that you must maintain.)
2. Define a “top” generic profile to control all work not controlled by more specific NODES profiles.
3. For each node, define profiles with USER x , SECL x or GROUP x keywords.
 - Prevent work with the specified userID, security label, or group ID from entering your node (determined by the UACC of the profile).
 - Translate the specified userID, security label, or group ID to a local value (specify the ADDMEM operand).
4. Define the local node or nodes in the &RACLNDE profile in the RACFVARS class. The syntax is:

```
RDEFINE RACFVARS &RACLNDE ADDMEM (nodea nodeb...)
```

This allows security information to be accepted for verification without the use of NODES profiles. That is, the information is used as passed because it is considered local. For SYSOUT, this allows the owner information to be used without a NODES lookup, or automatically allows the submitter to become the SYSOUT owner when &SUSER is used (see “How SYSOUT Requests are Verified” on page 49). For jobs, this allows the special JES2 pre-execution reroute case to use the information as passed without translation, and allows the spool unload and reload of jobs to propagate the information automatically without requiring NODES profiles. See “Defining Nodes as Local Input Sources” on page 70.

5. If an inbound job has been submitted as a surrogate job on its originating system (see “surrogate user” on page 51) the PASSWORD parameter will not be specified on its JOB statement. Therefore, you must specify UACC(CONTROL) or higher in the NODES profile controlling such jobs, or UACC(UPDATE) or higher if the job is from an uplevel node to prevent requiring password verification (see “Understanding Mixed Security Environments” on page 60).

RACSLUNK Seclabel: RACSLUNK is a special SECLABEL, assigned to SYSOUT, that is received from another node if the SECLABEL class is active on the receiving node and the NJE header contains an unknown or blank SECLABEL. If the SECLABEL class is inactive on the receiving node and if RACF is at release 1.9, the received SYSOUT retains its original value; if RACF is pre-RACF 1.9, the SECLABEL is set to blanks.

Since the RACSLUNK SECLABEL is not predefined, no one is able to access the SYSOUT. This situation can be handled in one of two ways:

- Define the RACSLUNK to RACF as a valid SECLABEL and authorize users to access it.
- Use the NODES class translation facility to translate it to a known SECLABEL.

See Figure 10 on page 60 as an exercise to learn which NODES profiles are used.

Assume the following profiles:

(1)	POKMVS.SECLJ.A	ADDMEM(ALPHA)	UACC(READ)
(2)	POKMVS.SECLS.A	ADDMEM(ALPHA)	UACC(READ)
(3)	POKMVS.SECL%.A		UACC(NONE) /*never used*/
(4)	POKMVS.USERJ.JOHN	ADDMEM(JOHNNY)	UACC(UPDATE)
(5)	POKMVS.USERS.JOHN	ADDMEM(JOHNNY)	UACC(UPDATE)
(6)	POKMVS.USER%.JOHN		UACC(NONE) /*never used*/
(7)	POKMVS.USER%.TOM		UACC(NONE)
(8)	POKMVS.USER%.*	ADDMEM(NONAME)	UACC(UPDATE)
(9)	POKMVS.*.*	ADDMEM(X)	UACC(READ)
(10a)	*		UACC(NONE)
(10b)	*.USERJ.*		UACC(NONE)

1. If a job is submitted from user JOHN at node POKMVS, with SECLABEL A, the profiles (1), (4), and (9) are used.
 - Profile (4) translates the userID to JOHNNY.
 - Profile (9) translates the group ID to X (there is no profile with the GROUP keyword).
 - Profile (1) translates the SECLABEL to ALPHA.
2. Profile (3) would never be used because (1) and (2) are discrete profiles that cover all work from node POKMVS that has security label A.
 - Profile (6) would never be used because (4) and (5) are discrete profiles that cover all work from user JOHN at node POKMVS.
3. If jobs or SYSOUT come in from user TOM at POKMVS, profile (7) fails the job or purges the output.
4. If a job comes in from anyone other than JOHN or TOM at POKMVS, with SECLABEL A, profiles (1), (8), and (9) are used.
 - Profile (8) translates the userID to JOHNNY.
 - Profile (9) translates the group ID to X (there is no profile with the GROUP keyword).
 - Profile (1) translates the SECLABEL to ALPHA.

Note: Profile (8) translates many userIDs to one. You might do this to create a guest userID that will be used by any otherwise unknown user coming in from POKMVS. With such a userID, you can allow people from POKMVS to access certain resources, without having to give each of them a userID on your system.
5. Because there is no POKMVS profile with a keyword of GROUPx, profile (9) is the generic that is used to translate group IDs. Therefore all jobs and SYSOUT that come from POKMVS get group X. (If profile (9) did not have ADDMEM specified, there would be no translation of group names.)
 - Also, all security labels from POKMVS, except security label A, are translated to X.
6. Profile (10a) fails all NJE jobs and SYSOUT for any other user, group, or SECLABEL that is not covered by a more specific NODES profile. If you want to have just default control for any NJE jobs, and not control SYSOUT, use profile (10b) instead.

Figure 10. Which NODES Profiles Are Used?

Understanding Mixed Security Environments

Your network can be a mixed environment; that is, it can contain nodes in which different levels of JES and RACF, or non-JES systems, are installed. Networking in a mixed environment causes JES and RACF to validate work differently in some cases. For example, certain security information, such as security labels, can not be sent with work from some systems. The following list categorizes the various environments into three groups:

- Uplevel security systems

- Systems in which JES2 or JES3 3.1.3 or later is installed and RACF is installed and active
- Downlevel security systems
 - Systems in which a release prior to JES2 or JES3 3.1.3, or a non-MVS system, is installed
- Default security systems
 - Systems in which JES2 or JES3 3.1.3 or later is installed and one of the following conditions exists:
 - RACF is not installed.
 - RACF is not active.

The terms (uplevel, downlevel, and default) describe the security systems of the source nodes. This tells JES and RACF how much of the security information has been verified at the source. They are used to determine how much the receiving node trusts the source nodes. For jobs, the amount of trust determines under what circumstances NJE propagates the submitter. For SYSOUT, it determines under what circumstances we accept the owner information. NJE uses the NODES profile UACC to determine level of trust. Use these definitions when the remainder of this section refers to uplevel, downlevel, and default security systems. See Table 10 on page 62 and Table 11 on page 64.

Authorizing Jobs

You can control which network jobs are authorized for processing at your installation based on the submitter's userID, group ID, or security label associated with the inbound job.

To authorize or restrict jobs entering your system from another node, define a NODES profile that specifies the criteria upon which jobs are accepted. Ask your JES system programmer for the following:

- The node names from which you expect jobs.
- The userIDs or group IDs from which you expect jobs.
- The security labels that you will accept.
- The universal access authority, which determines how JES3 will process the job. Table 10 on page 62 lists the universal access authorities you can assign and defines the validation that RACF performs.

Table 10. NODES Class Keywords and the UACC Meaning for Inbound Jobs

Type of Check (keyword)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
User ID (USERJ)	FAILS the job.	Verifies all security information available including password validation.	<p>If the job is from an uplevel node, that is, if a nondefault valid security token is passed, propagates the submitter's or translated security information as the owning security information without password validation. See "Understanding Mixed Security Environments" on page 60.</p> <p>If the job is from a default or downlevel node, processing is the same as for a UACC of READ.</p>	Same as UPDATE, but default security or downlevel information is allowed. CONTROL allows a pre-SP3.1.3 system to send jobs to your node without passwords. RACF does not validate passwords. See "Understanding Mixed Security Environments" on page 60.
Group ID (GROUPJ)	FAILS the job.	Translates GROUPID to that specified in ADDMEM. If ADDMEM is not specified, uses the group ID received.		
Security Label (SECLJ)	FAILS the job.	Translates the submitter's SECLABEL to that specified in ADDMEM. If ADDMEM is not specified, uses the security label received.		

Notes:

1. RACF can perform only userID, group ID and password validation for jobs entering from a pre-SP3.1.3 system because security labels do not exist for pre-SP3.1.3 systems.
2. RACF can perform userID, group ID and password validation without performing any translation, if no profile exists for a job when the NODES class is active or if the NODES class is inactive.
3. RACF verifies all security information available, and you must specify a valid password and userID on the job card if no profile exists for a job when the NODES class is active.

You can further reduce the risk of security exposures by allowing jobs to be submitted from other nodes without requiring a password if the sending node(s) properly validates and transmits a user's identity. You can either allow the submitter's identity (that is, the userID and security label) to be propagated to the job or you can specify that the submitter is a surrogate submitter who can submit jobs on behalf of other users without needing a password. For either case, you indicate in NODES class profiles which nodes are trusted to provide valid submitter identity information. You can restrict the information that will be trusted to specified user IDs, group IDs or security labels, if desired.

This submitter identity information in combination with user data on the job card is used to determine the user identity to be used for the job. If no userID or password is specified on the job card, the submitter's identity is propagated to the job. If a userID but no password is specified, the userID will be allowed if the submitter is authorized as a surrogate for that userID.

Note: In either of these cases, if SECLABEL is specified on the job card, it is used. If not, the SECLABEL of the submitter is propagated to the job.

If both userID and password are specified on the job card, the submitter's identity information is not used and normal password validation is performed. If the NODES class profile does not identify the submitter's identity information as valid, then you must specify a userID and password on the job card.

Authorizing SYSOUT

You can control the processing of SYSOUT at your installation based on the userID, group ID, or security label associated with the inbound SYSOUT.

To authorize or restrict SYSOUT entering your system from another node, define NODES class profiles that identify the criteria upon which SYSOUT is accepted. Ask your JES system programmer for the following:

- The node name(s) from which you expect SYSOUT.
- The userID(s), group ID(s), and/or security labels from which you expect SYSOUT.
- The universal access authority, which determines how JES processes the SYSOUT. RACF can assign ownership based on either the userID and node that created the SYSOUT or the userID and node that submitted the job that created the SYSOUT.

Note: If the NODES profile allows the userID to be associated with the SYSOUT, but the user security information is incorrect, a message is issued and processing continues with the NJE unknown user as set by SETROPTS JES(NJEUSERID(userid)).

Table 11 on page 64 lists the universal access authorities you can assign and defines the validation that RACF performs.

Table 11. NODES Class Keywords, UACC and SYSOUT Ownership When Node Is Not Defined to &RACLNDE

Type of Check (keyword)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
User ID (USERS)	Check of User ID and Node That Created SYSOUT			
	Purges the output.	<p>If the translation value from ADDMEM is &SUSER, check submitting userID and node.</p> <p>Otherwise, assigns ownership of the output to the default NJE userID (default is ???????).</p>		
	<p>If default or no security information is available, that is, from a downlevel or default node, processing is the same as a UACC of READ.</p> <p>If security information is from an uplevel node, that is, a nondefault valid security token is passed, assigns the translation value from ADDMEM to the output. When ADDMEM is not specified, ownership is assigned to the userID that created the output. See "Understanding Mixed Security Environments" on page 60.</p> <p>If the translation value from ADDMEM is &SUSER, check submitting userID and node.</p>	<p>Processing is similar to UACC(UPDATE) except RACF translates any available information from any type of security system. This allows RACF to assign local userIDs to output from pre-SP3.1.3 nodes. See "Understanding Mixed Security Environments" on page 60.</p> <p>If the translation value from ADDMEM is &SUSER, check submitting userID and node.</p>		
Check of Submitting User ID and Node (Only When &SUSER Is Specified for ADDMEM)				

Table 11. NODES Class Keywords, UACC and SYSOUT Ownership When Node Is Not Defined to &RACLNDE (continued)

Type of Check (keyword)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
	Assigns ownership of the output to the default NJE userID (default is ???????).	Assigns ownership of the output to the default NJE userID (default is ???????).	Assigns ownership of the output to the default NJE userID (default is ???????).	Assigns the translation value from ADDMEM to the output, if available. If the translation value from ADDMEM is &SUSER, assigns the submitting userID to the output. Otherwise, assigns ownership of the output to the default NJE userID (default is ???????).
<p>Note: When you specify &SUSER for ADDMEM and the submitting node <i>is</i> defined to &RACLNDE, ownership is assigned to the submitter. Refer to “How SYSOUT Requests are Verified” on page 49.</p>				
Security Label (SECLS)	Purges the output	Translates SECLABEL to that specified in ADDMEM. If ADDMEM is not specified, uses the security label received.		
Group ID (GROUPS)	Purges the output	Translates GROUPID to that specified in ADDMEM. If ADDMEM is not specified, uses the group ID received.		
<p>Notes:</p> <ol style="list-style-type: none"> 1. If the nodeid is specified in the RACFVARS profile named &RACLNDE, the node is treated as a locally attached node and RACF verifies the supplied security information. 2. Also see “Authorizing Jobs” on page 61 for more details on how NJE jobs are processed. 				

Validating SYSOUT Based on the Submitter

JES normally validates SYSOUT based on the owner’s security information. The owner’s security information accompanies each piece of SYSOUT as it travels through the network.

You can define profiles that cause RACF to assign ownership of the SYSOUT to the submitter. For example, you can allow a user to submit a job to another node, have the job run under another userID, and allow the submitting user to view the output upon its return.

To translate inbound SYSOUT ownership to the submitter, specify &SUSER as the value on the ADDMEM keyword of the NODES profile.

Translating Security Information

You can avoid having to maintain identical userIDs, group IDs and security labels in RACF databases throughout a network by translating inbound userIDs, group IDs, and security labels into values your node has predefined.

Use the ADDMEM operand on the RDEFINE or RALTER command to specify the translation values for inbound security information. For example, if you want all inbound work with a security label of VERYCONF to be translated to a security label of NOLOOKAT at your system, specify the following:

```
RDEFINE NODES *.SECL*.VERYCONF ADDMEM(NOLOOKAT) UACC(READ)
```

Note: You should specify only one value on the ADDMEM operand; any subsequent values are ignored.

If you do not define profiles that translate inbound userIDs, group IDs, and security labels, then those inbound values must be defined in your RACF database or the work will not pass RACF validation.

Note: If the SECLABEL class is not active on your system, inbound security labels are ignored.

Example: Simple NJE User Translation: The example shown in Figure 11 shows how userIDs are translated.

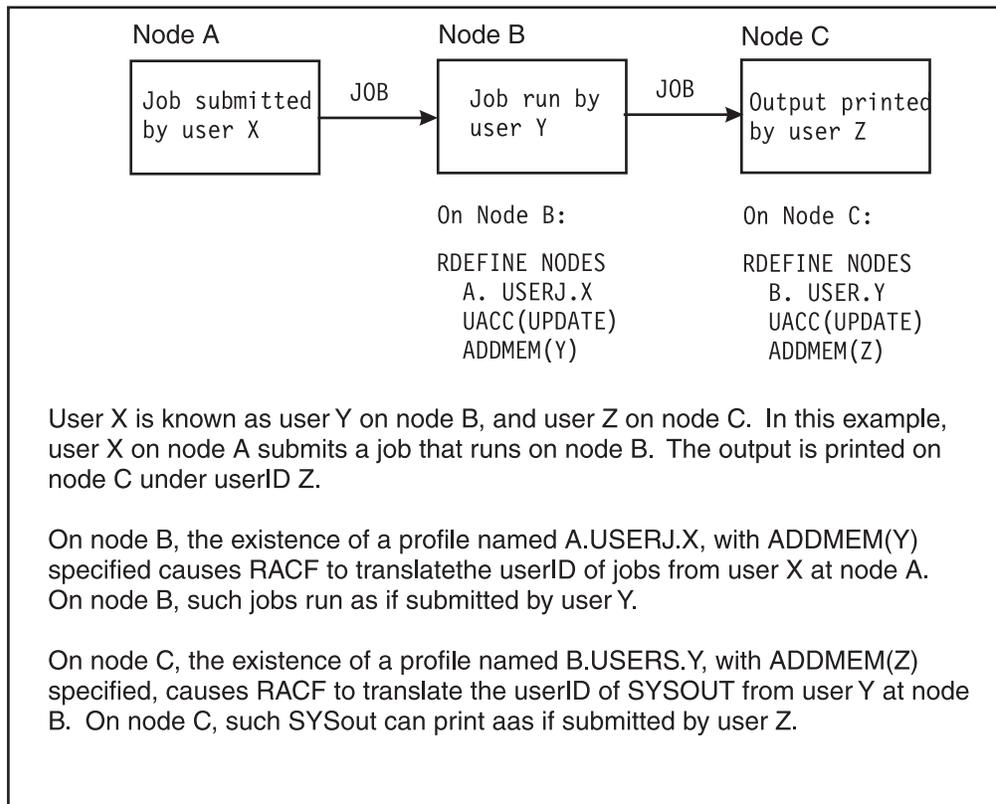


Figure 11. Example: Simple NJE User Translation

Example: Simple NJE User Translation Using &SUSER: The example shown in Figure 12 on page 67 shows how userIDs are translated when &SUSER is specified

on the ADDMEM operand. This can be useful when jobs are run on a remote system, but the output is printed on the submitter's system.

Note: If you wish, you could specify &SUSER on a third system (as in node C in Figure 11 on page 66).

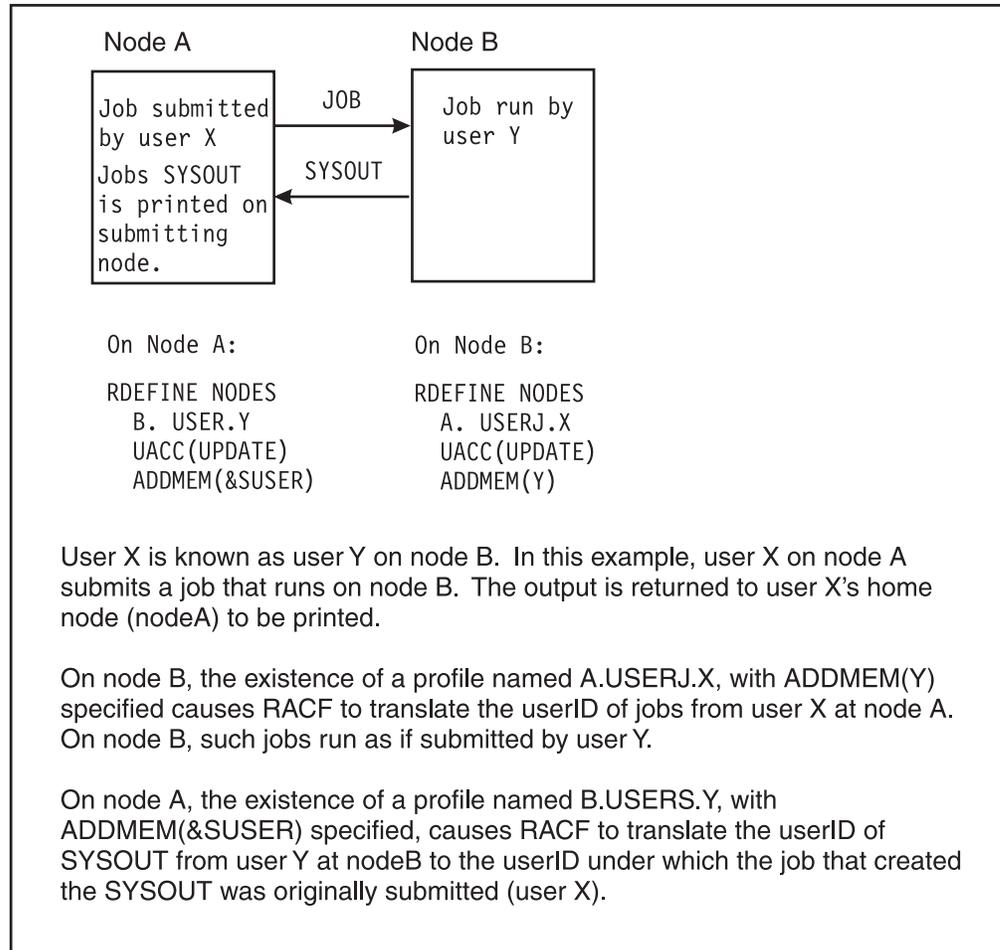


Figure 12. Example: Simple NJE User Translation Using &SUSER

Example: Trusted, Semitrusted, and Untrusted Nodes: The example shown in Figure 13 on page 68 shows a sample NJE network in which some nodes are trusted (see "Understanding Mixed Security Environments" on page 60), some nodes are semi-trusted (verification is done on inbound work), and some nodes are not trusted (no inbound work is allowed to run).

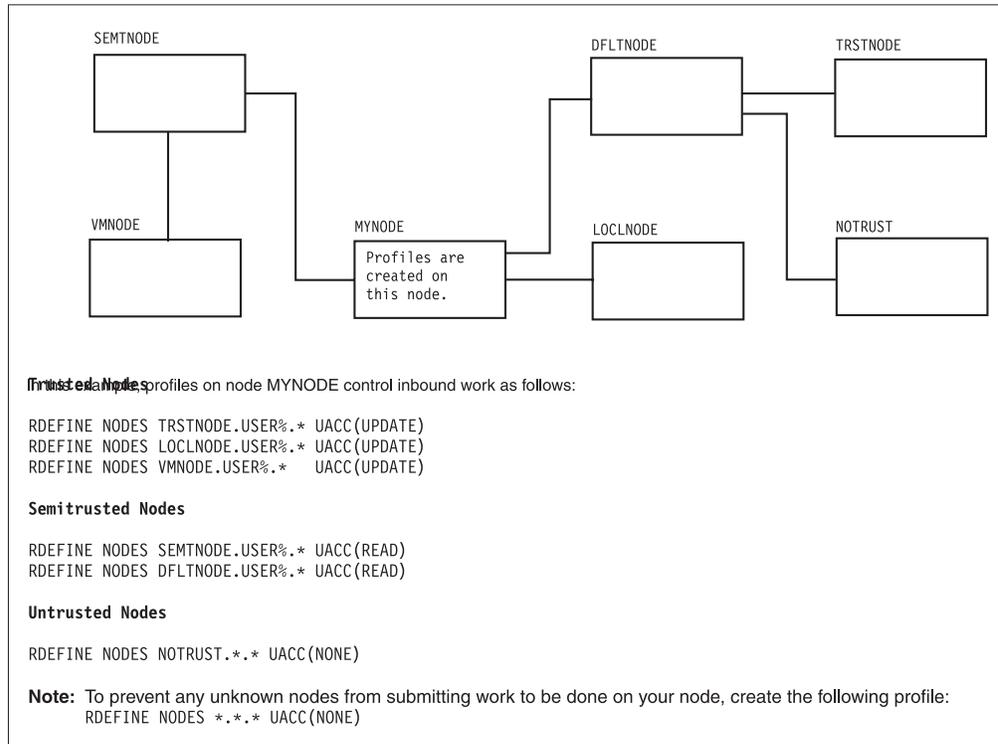


Figure 13. Example: Trusted, Semitrusted, and Untrusted Nodes

Understanding Default User IDs

RACF assigns a default userID to all work that enters your node when:

- SYSOUT enters from one of the following:
 - A downlevel node
 - A default node

For more information, see “Understanding Mixed Security Environments” on page 60.

- SYSOUT or a job enters your node, but your node is an intermediate (store-and-forward) node on the path to the work’s final destination. The default userID protects work while it resides on spool awaiting transmission.

RACF uses eight question marks (???????) as the userID for all inbound work meeting the above criteria. RACF also assigns the default userID to all store-and-forward work that resides temporarily at your node. The default userID protects work while it resides on spool.

You cannot directly permit the default userID (either IBM-supplied or installation-defined) to any resource. However, you can translate the default userID to a valid userID if you want to process any of this type of work at your system.

You can change the ???????? userID by using the NJEUSERID operand on the SETROPTS command:

```
SETROPTS JES(NJEUSERID(?NETWORK))
```

The userID you specify on the NJEUSERID operand cannot be a userID defined in the RACF database. Also, if you specify a userID on the NJEUSERID operand, you

cannot later define a user profile for that userID. This prevents network jobs from having access to RACF-protected resources on your system.

The following example shows how to do this for jobs:

```
RDEFINE NODES nodename.USERJ.????????? UACC(READ or higher)
ADDMEM(NJEJOBS)
```

The following example shows how to do this for SYSOUT:

```
RDEFINE NODES nodename.USERS.????????? UACC(UPDATE or higher)
ADDMEM(NJESOUT)
```

The following example shows how to do this for both SYSOUT and jobs:

```
RDEFINE NODES nodename.USER%.????????? UACC(READ or higher)
ADDMEM(NJEWORk)
```

Note: This example assumes that a SETROPTS GENERIC(NODES) command was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

You would also need to create user profiles for the translated user IDs (NJEJOBS, NJESOUT, or NJEWORK), and permit the userIDs to appropriate resource profiles (or connect them to appropriate groups).

Local jobs that enter the system without a userID are assigned a userID of ++++++++ (8 plus signs). You can specify which userID to assign to such jobs by entering the following command:

```
SETROPTS JES(UNDEFINEDUSER(userid))
```

Note: The userID you specify on the UNDEFINEDUSER operand cannot be a userID defined in the RACF database. Also, if you specify a userID on the UNDEFINEDUSER operand, you cannot later define a user profile for that userID. This prevents undefined users from having access to RACF-protected resources on your system. These userIDs can, however, be used in JESSPOOL profile names. JES uses these names to associate an owner with the spool data and to keep logical undefined users from accessing the data of network undefined users.

How JES Sends Security Information

Security information is sent from node to node in an NJE network. When a node receives a job through a network, RACF determines who submitted the job. After determining the submitting userID, RACF can translate the submitting userID to a valid userID on this system if a profile on the receiving node specifies that the userID must be translated. RACF uses the submitting userID or its translation to supply any missing security information. Security information RACF propagates is:

- Userid
- Password
- Security label.

Defining Profiles in the NODES Class

To create profiles in the NODES class, take the following steps:

1. Ask your JES system programmer for the information needed to create the profiles. This includes the following:
 - Information for specifying profile names
 - For each profile to be created, the UACC to be specified

- For each profile to be created, the values to be translated (to be specified on the ADDMEM operand).

Note: You should work with your JES system programmer to determine which user or group should be specified in the OWNER field of the profiles. This user or group will be responsible for maintaining the profiles.

2. Use the RDEFINE command to create the profiles. For examples, see Figure 10 on page 60, Figure 11 on page 66, Figure 12 on page 67, and Figure 13 on page 68.
3. When you are ready to start using the protection defined in the profiles, activate the NODES class and activate SETROPTS RACLIST processing for the class. You can do these two actions in one command:

```
SETROPTS CLASSACT(NODES) RACLIST(NODES)
```

Note: Any time you make a change to a NODES profile, you must also refresh SETROPTS RACLIST processing for the NODES class for the change to take effect.

Defining Nodes as Local Input Sources

To use RACF to treat nodes the same as locally attached devices, use the &RACLNDE profile in the RACFVARS class to identify the nodes that you want RACF to consider as local. See “Setting up NODES Profiles” on page 59. A node name defined to &RACLNDE either shares your RACF database or is the value you are using to rename your node when you are changing node names.

To setup nodes profiles take the following steps:

1. Ask your JES system programmer for the names of the nodes to be treated as local.
2. Define profile &RACLNDE in class RACFVARS:

```
RDEFINE RACFVARS &RACLNDE UACC(NONE)
```

3. Using the ADDMEM operand on the RALTER command, identify which nodes are to be treated as local nodes:

```
RALTER RACFVARS &RACLNDE ADDMEM(node1 node2 node3 ...)
```

Note: If you define a node as a local node, you must ensure that its RACF database is identical to the one on your node.

4. When you are ready to start using the protection defined in the profiles, activate the RACFVARS class and activate SETROPTS RACLIST processing for the class. You can do these two actions in one command:

```
SETROPTS CLASSACT(RACFVARS) RACLIST(RACFVARS)
```

Notes:

- a. Any time you make a change to a RACFVARS profile, you must also refresh SETROPTS RACLIST processing for the RACFVARS class for the change to take effect.
- b. This will also activate other functions that are administered through the RACFVARS class.

Authorizing Outbound Work

To use the WRITER class to control whether work is authorized for transmission to a specific NJE node, create profiles in the WRITER class that have profile names with the following format:

```
jesname.NJE.nodename
```

For more information, see “Controlling Where Output Can Be Processed” on page 79.

Controlling Password Encryption for NJE Jobs

You can use the PWCNTL keyword on the JES3 NJERMT initialization statement to control whether passwords defined on JCL JOB statements for NJE jobs are encrypted before the job is sent across the network, or you can define JES3 to validate passwords at the originating node.

If you do not request local password verification, user verification is performed at the remote node. If you request local password verification, user verification is performed at the local node, and no passwords are sent to the remote node.

NJE jobs contain two JOB statements. The first JOB statement is used to route the work to the remote node. The second JOB statement is the statement used to process the work. Only the password on the first JOB statement is affected when you request password encryption or verification. If you request password encryption the password on the second JOB statement is not affected.

Notes:

1. Ensure that the userid on the first and second JOB statements match if no passwords are coded on the second JOB statement and password encryption is specified. For example:

```
//JOB1 JOB USER=SUMBODY,PASSWORD=AMSCRAY
//SEND XMIT DEST=OUTHERE
//JOB1 JOB USER=SUMBODY
```

In the example above, the userid on the first JOB statement is used as the key to encrypt the password which is sent to the execution node. This password is used along with the userid on the second JOB statement. Because the second JOB statement userids password has been encrypted in the RACF data base using the userid as key, the userids must be identical in order to ensure that passwords match.

2. Ensure that passwords are coded on the second job statement when requesting local verification unless reviewing node profiles allow jobs without passwords. If a password is omitted, NODES class profiles must be defined to allow user propagation or surrogate job submission.
3. When you request local verification, the userid on the first job statement is used for job validation and is used to identify the submitter. If you do not request local verification, the userid who submits the job (through the internal reader) is used to identify the submitter. However, if the job is submitted through an external reader, the default userid is used as the submitter. In all of these instances, the resulting userid is propagated or used in surrogate checking as appropriate, based on the RACF profiles and options in effect at the execution node.

The following examples show where validation is performed when you include the PWCNTL parameter on the NJERMT statement.

Example 1: In the following example, PWCNTL=SENDENC has been specified on the NJERMT statement. Validation for JOB1 is performed at the execution node (NODE B). No job validation is performed at the originating node (NODE A).

NODE A (No job validation)	NODE B (Job validated here)
//JOB1 USER=X,PASSWORD=AMSCRAY	USER=X (second JOB card)
//XMIT DEST=NODEB	PASSWORD=AMSCRAY (decrypted from
//JOB1 USER=X	1st job card)

Example 2: In the following example, PWCNTL=LOCALCHK has been specified on the NJERMT statement. The userid (TOM) and the password from the first JOB statement (AMSCRAY) is validated at the originating node (NODE A). The job is also validated on the execution node (NODE B).

NODEA (Job validated here)	NODEB (validation also done here)
//JOB1 USER=TOM,PASSWORD=AMSCRAY	USER=MIKE (from 2nd JOB card)
//XMIT DEST=NODEB	PASSWORD=LETME (sent without being encrypted)
//JOB1 USER=MIKE,PASSWORD=LETME	

JES3 uses a security product, such as RACF, to validate job passwords for NJE. If you specify PWCNTL=LOCALCHK and RACF is not installed or it is inactive on your local node, password validation is performed only at the execution node.

Using Security Labels to Control Writers

If both the WRITER and the SECLABEL class are active when RACF checks a writer's authority to process outbound work, RACF uses "reverse MAC" (mandatory access checking). That is, the outbound work must have a security label, and the security label of the writer must be equal to or greater than the outbound work's security label.

You can use this to limit the sensitivity of the data that writers can process. For example, if you have some writers that process low-sensitivity information, you can assign those writers a low-sensitivity security label, such as SYSLOW. This prevents sensitive work from leaving your node via those writers.

Controlling Access to Data That Resides on Spool

You can use a security product, such as RACF, 1.9 to provide security for your spool data including:

- SYSIN and SYSOUT data sets
- JESNEWS
- SYSLOG
- For JES2, trace data sets
- Data sets dumped from and restored to spool
- Access to spool data.

The following sections identify what spool resources you can protect, why you might want to protect each resource, and what information you must gather from your JES system programmer so that you can implement RACF protection.

Protecting SYSIN and SYSOUT

You can use RACF to provide access to SYSOUT data sets that reside on spool, including spool files that JES appends to job output, such as JESNEWS. Using RACF permits users other than the owner of a data set to read, copy, print, or delete sensitive job data.

To enable RACF protection of SYSIN and SYSOUT data sets, activate the JESSPOOL class:

```
SETROPTS CLASSACT(JESSPOOL)
```

Profiles are not required in the JESSPOOL class for protection to be in effect.

Notes:

1. When the JESSPOOL class is active, RACF ensures that only authorized users obtain access to job data sets on spool. RACF user profiles provide

authorization to job data sets. If there is no profile for a data set, only the user that created the data set can access, modify, or delete it.

2. While a job is running, RACF optionally audits actions against SYSIN and SYSOUT data sets. For SYSIN data sets, JES invokes RACF each time a SYSIN data set is allocated, opened, or deleted. For SYSOUT data sets, JES invokes RACF each time a SYSOUT data set is created, opened, deleted, or selected for output.
3. For output selection, a data set can be selected by a TSO/E user via the TSO/E OUTPUT command. A profile must exist to enable users other than the creator to access data sets using the TSO/E OUTPUT command.
4. External writers, which are usually started tasks that process output to special devices (such as microfiche), require at least ALTER access to the spool data sets they process. If your installation has external writers, and you activate the JESSPOOL class, you must either ensure that the external writers have ALTER access to appropriate JESSPOOL profiles, or define the external writers in the RACF started procedures table with the trusted attribute. Otherwise, the external writers will not be able to process output. Because external writers are installation-written programs, IBM strongly recommends that you avoid giving these programs the trusted attribute.
5. SYSOUT application programming interface (SAPI) applications, which are usually started tasks that process SYSOUT datasets, require at least UPDATE access to the spool data sets they process. If your installation has SAPI applications, and you activate the JESSPOOL class, you must either ensure the SAPI application has UPDATE access to the appropriate JESSPOOL profile or you must define the SAPI application in the RACF started procedures table with the trusted attribute. Otherwise, the SAPI application will be unable to process output. Because SAPI applications are non-IBM written products or programs, IBM strongly recommends that you avoid giving these programs the trusted attribute.

Defining Profiles for SYSIN and SYSOUT Data Sets

Activating the JESSPOOL class provides protection for SYSIN and SYSOUT data sets. You might however, want to allow specific users to see or work with the SYSIN and SYSOUT data sets created by other users, take the following steps:

1. If you need to create JESSPOOL profiles longer than 39 characters, ask your RACF system programmer to restructure the RACF database, using the IRRDSC00 utility. For information about using the IRRDSC00 utility, see *z/OS Security Server RACF System Programmer's Guide*.
2. Create JESSPOOL profiles for the spool data sets:

```
RDEFINE JESSPOOL profile-name UACC(NONE)
```

where *profile-name* is a six-part name with the following format:

```
localnodeid.userid.jobname.jobid.dsnumber.name
```

where:

localnodeid

is the name of the node on which the SYSIN or SYSOUT data set currently resides. The local nodeID appears in the JES job log of every job.

Note: IBM recommends that you define a profile in the RACFVARS class named &RACLNDE and use &RACLNDE for all profiles in the JESSPOOL class.

- userID** is the userID associated with the job. This is the userID RACF uses for validation purposes when the job runs.
- jobname** is the name that appears in the name field of the JOB statement.
- jobid** is the job ID assigned to the job by JES. The job ID appears in notification messages and the JES job log of every job.
- dsnumber** is the unique data set number JES assigned to the spool data set. A "D" is the first character of this qualifier.
- name** is the name of the data set specified in the DSN= parameter of the DD statement. This name cannot be JESYSMSG, JESJCLIN, JESJCL, or JESMSGGLG and follows the naming conventions for a temporary data set. See *z/OS MVS JCL Reference* for the temporary data set naming conventions. If the JCL did not specify DSN= on the DD statement that creates the spool data set, JES uses a single question mark (?).
- Note:** You can specify generic characters for any of the qualifiers in the profile name. For example, you can substitute an asterisk (*) for one of the qualifiers, such as *jobid*, if it is not known.

A sample JESSPOOL profile name could be as follows. If user MYUSER submits a job named MYJOB to run on NODEA, and JES assigns a job ID of JOB08237, and the value of DSN= for a SYSOUT data set is OUTPUT, the profile name for a SYSOUT data set created by this job could be:

```
NODEA.MYUSER.MYJOB.JOB08237.D0000112.OUTPUT
```

If job MYJOB is run several times, and the same protection is desired for the OUTPUT data set each time, the profile name could be:

```
NODEA.MYUSER.MYJOB.*.*.OUTPUT
```

3. Give users the appropriate access authority as follows:

```
PERMIT profile-name CLASS(JESSPOOL)
      ID(userid or group-name) ACCESS(access-authority)
```

where *access-authority* is one of the following:

NONE

Allows no access.

READ Allows user to view the spool data set but not change its attributes. For example, this does not allow the following keywords on the OUTPUT command: NOKEEP, NOHOLD, DELETE, NEWCLASS, and DEST.

UPDATE

Allows to update a spool data set.

CONTROL

Equivalent to UPDATE.

ALTER

Allows any operand specified on the TSO/E OUTPUT command, including deleting and printing. Also, when specified for a discrete profile, allows the user to change the profile itself.

Note: If SDSF 1.3 or later is installed on your system, JESSPOOL profiles control which action characters and overtypable fields a user can enter

on SDSF panels. For complete information about creating JESSPOOL profiles for use with SDSF 1.3, see *SDSF Guide and Reference*.

Allowing Users to Create Their Own JESSPOOL Profiles

You can allow users to create their own JESSPOOL profiles by giving them CLAUTH authority to the JESSPOOL class. If your installation decides to place the SETROPTS GENERICOWNER option in effect, then you can restrict each user to creating JESSPOOL profiles only for his or her own spool data.

To do this, take the following steps:

1. Enter the SETROPTS GENERICOWNER command:

```
SETROPTS GENERICOWNER
```

2. For each user who should be able to create JESSPOOL profiles for his or her own spool data, create a JESSPOOL profile with the user's userID specified. Make the user the owner of the profile. For example, for users SMITH and BEN:

```
RDEFINE JESSPOOL nodename.SMITH.** OWNER(SMITH) UACC(NONE)
```

```
RDEFINE JESSPOOL nodename.BEN.** OWNER(BEN) UACC(NONE)
```

Note: These examples assume that a SETROPTS GENERIC(JESSPOOL) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

3. Give the users CLAUTH in the JESSPOOL class:

```
ALTUSER SMITH CLAUTH(JESSPOOL)
```

```
ALTUSER BEN CLAUTH(JESSPOOL)
```

Protecting JESNEWS

JESNEWS is a spool file that contains data to be printed following each job's output. Protecting JESNEWS prevents unauthorized users from adding, modifying, or deleting these files, or (if security labels are used) writing data with a higher security label into these files.

Protecting JESNEWS for JES3

To protect JESNEWS for JES3, take the following steps:

1. Ask the JES3 system programmer for the following information:
 - The fully qualified name(s) for the JESNEWS file(s) to be protected.
 - The universal access authority to be associated with each JESNEWS file. For JESNEWS, this value should always be READ to allow all JES users to receive JESNEWS.
 - The userIDs or group IDs of operators and users that are to be authorized to update JESNEWS. Assign each of these users an access authority of UPDATE.
 - The security label to be associated with each JESNEWS file (if security labels are being used). For JESNEWS, this value should always be the lowest security label (SYSLOW) to allow JESNEWS to be printed for all users.
2. Create profiles as indicated by the JES3 system programmer, for example:

```
RDEFINE JESSPOOL node.jesname.JOB00000.D0000000.JNEWSLCL
UACC(READ)
```

```
RDEFINE JESSPOOL node.jesname.JOB00000.D0000000.JNEWSRJP
UACC(READ)
```

```
RDEFINE JESSPOOL node.jesname.JOB00000.D0000000.JNEWSTSO
UACC(READ)
```

Note: To improve system performance, you should consider including entries for JESNEWS in the global access checking table, for example:

```
node.jesname.JOB00000.D0000000.JNEWSLCL/READ
```

```
node.jesname.JOB00000.D0000000.JNEWSRJP/READ
```

```
node.jesname.JOB00000.D0000000.JNEWSTSO/READ
```

3. For users who will update JESNEWS, give UPDATE authority:

```
PERMIT profile-name CLASS(JESSPOOL)  
ID(userid or group-name) ACCESS(UPDATE)
```

How RACF Affects Jobs Dumped from and Restored to Spool (JES3 Only)

RACF performs security validation for all jobs restored to your system using the JES3 Dump Job facility.

Dumping Jobs

JES3 dumps all security information associated with each job when you use the dump job facility. JES3 does not however, perform security validation while dumping jobs.

Restoring Jobs

JES3 calls RACF to revalidate the job. RACF validates the job using the security information saved when the job was dumped and writes an SMF audit record for each restored job.

Attention: Jobs and/or data transported to a complex that uses different security labels can be inadvertently declassified.

Authorizing Console Access

MCS Consoles

Your MVS system programmer can require operators to LOGON to and LOGOFF from MCS-managed consoles by specifying options in the CONSOLxx member of the MVS SYS1.PARMLIB data set. When the CONSOLE class is active and a console being used is protected by a profile in the CONSOLE class, RACF ensures that the person attempting to LOGON has the proper authority to do so.

To control access to MCS consoles, take the steps described in “Protecting Consoles on MVS” in *z/OS Security Server RACF Security Administrator’s Guide*.

Remote Workstations (RJP/RJE Consoles)

Your JES system programmer can require that remote workstation operators enter a password during workstation logon. This can be done through RACF or by using JES initialization statement parameters.

Note: In JES2, remote workstations are called RJE consoles; in JES3, they are called RJP consoles. If the workstation is connected using BSC, the operator must issue a /*SIGNON statement. If the workstation is connected using SNA, the operator must issue a LOGON statement.

If you want RACF to check LOGON or /*SIGNON passwords, you must activate the FACILITY class and define a profile for each workstation in both the FACILITY and USER classes. You should also ask your JES system programmer for the

workstation name. If JES2 is installed, the workstation name has the form RMTnnnn, where nnnn is the remote workstation number. If JES3 is installed, the workstation name is derived from the RJPWS initialization statement for an SNA workstation or the RJPTERM initialization statement for BSC. This workstation name serves as the userID for the workstation console. Users of the RJP console have to log on using this terminal ID and supply the same password.

You might also need similar support for NJE nodes for command authorization from the network. NJE nodes do not sign on as RJE workstations do, but rather perform the FACILITY/USERID verification as each command is issued. See "Using RACF to Authorize the Use of Operator Commands" on page 81.

Command validation in JES is composed of two parts:

1. Validating that the originator of the command can enter the command
2. Validating that the originator is authorized to the object of the command.

RACF control is only applied to the issuance of the command. JES continues to validate what object a particular workstation or node can affect.

Notes:

1. JES password protection or command authorization is used instead of RACF protection if any of the following conditions exist:
 - RACF is not installed.
 - RACF is active, but the FACILITY class is not active.
 - No NJE node or remote workstation profile exists in the FACILITY class.
2. If RACF is installed but not active, control returns to JES, and JES does its own password checking or command authorization.
3. Workstation operators can change their user passwords only at logon time.
4. RACF password protection replaces JES password protection for remote workstations. That is, either RACF or JES, but not both, will verify logons and passwords. Similarly, RACF command authorization across the network replaces JES NJE command authorization. That is, RACF or JES, but not both, verifies these commands.
5. The password for an RJE workstation must be changed the first time the workstation issues a LOGON or /*SIGNON.
6. Because the remote workstation or nodeID is also used as a port of entry, it needs to be defined to the JESINPUT class (if active). If it is not defined and the class is later activated, RJE signons or NJE command authorizations fail because of an incorrect port of entry. You can find other useful guidance in *Security Implementation Guide*.

To use RACF to check LOGON or /*SIGNON passwords, take the following steps:

1. For each remote workstation or node to be protected, ask your JES system programmer for the following:
 - The ID of the remote workstation. The ID will serve as the userID of the remote workstation. All users using a particular remote workstation must log on using this ID and supply the same password. (The password never expires.) The ID is one of the following:
 - If JES2 is installed, the remote ID of the RJE console to be protected, which takes the form RMTnnnn.
 - If JES3 is installed, the ID of the console you want to protect.
 - For NJE nodes, the name of the node to be used as the userID of that node.

- For each remote workstation or NJE node, create a user profile:

```
ADDUSER userid
        DATA('RJE console at xxx, phone yyy')
        PASSWORD(initial-password)
        DFLTGRP(group-id)
```

where:

userid is the RJE remote ID or NJE node name.

PASSWORD

is the initial password (to be changed immediately to another password that will never expire).

DATA is installation-defined.

DFLTGRP

is a group which you allow to use certain RACF-protected resources, such as commands.

Specify that the passwords for these profiles are never to expire:

```
PASSWORD USER(userid) NOINTERVAL
```

- For each workstation for which you want RACF to check the user's password, create a FACILITY profile in the FACILITY class, as follows:

```
RDEFINE FACILITY RJE.workstation
```

where *workstation* has been supplied by the JES system programmer.

Note: The existence of a profile in the FACILITY class for a remote workstation forces the user to enter a password to be checked by RACF, rather than by JES. The specification of UACC for these profiles has no effect.

- For each NJE node for which you want RACF to check the user's command authorization, create a FACILITY profile in the FACILITY class, as follows:

```
RDEFINE FACILITY NJE.nodename
```

where *nodename* has been supplied by the JES system programmer. The specification of UACC for these profiles has no effect.

- Run a batch job with old and new passwords specified to set a new password (which will never expire) per step 2.
- When you are ready to start using the protection provided by the profiles you have created, activate the FACILITY class:

```
SETROPTS CLASSACT(FACILITY)
```

- If the class is active, define the workstation or nodeID to the JESINPUT class, as follows:

```
RDEFINE JESINPUT workstation UACC(appropriate-access)
```

```
RDEFINE JESINPUT node-id UACC(appropriate-access)
```

If the workstation or nodeID is not defined and the class is later activated, sign on or command authorization fails because of an incorrect port of entry. You can also find other useful information in *Security Implementation Guide*.

Controlling Where Output Can Be Processed

You can use the WRITER class to control where output can be printed. For example, you can authorize or restrict the use of writers for local printers and punches, remote workstations (RJE and RJP devices) and network nodes. You can also limit which classification of data can be sent to a particular device or node. See “Authorizing Outbound Work” on page 70 for information about how to use the WRITER class to control outbound jobs and SYSOUT for NJE.

When the WRITER class is active, RACF ensures that the user is authorized to use a writer. For network devices, RACF also verifies the security of outbound data sets to ensure that the originator is authorized to send the data set to another node in a network.

To control where output can be sent, do the following:

1. Ask your JES system programmer for the following information:
 - The name of your JES system
 - If you are protecting local printers and punches, or RJE devices, their device names
 - If you are protecting network devices, the name of the node that will ultimately receive the output

Note: The node name as specified in the JES initialization stream.

- The security label, if you want to limit which classifications of output can be sent to a particular output destination
 - The list of users to be authorized or restricted from using a specific output destination.
2. Create a profile in the WRITER class to protect each writer:

```
RDEFINE WRITER profile-name UACC(appropriate-access)
```

where *profile-name* has one of the following formats:

- For local printers and punches:
jesname.LOCAL.devicename
- For JES2 RJE devices:
jesname.RJE.devicename
- For JES3 RJP devices:
jesname.RJP.devicename
- For data whose destination is a node:
jesname.NJE.nodename

where *nodename* is the name of the node to ultimately receive the output.

Also, UACC can be one of the following:

NONE

Allows no access

READ Allows all users to send output to the protected device or node.

3. Give the appropriate access to users and groups:

```
PERMIT profile-name CLASS(WRITER) ID(user or group)  
ACCESS(appropriate-access)
```

where *appropriate-access* is one of the following:

NONE

Allows no access

READ Allows the user or group to send output to the protected device or node.

4. When you are ready to start controlling access to writers based on the profiles you have defined, activate the WRITER class:

```
SETROPTS CLASSACT(WRITER)
```

Note: If SDSF 1.3 or later is installed on your system, WRITER profiles control which operations related to printers (such as displaying information about a printer or purging output) users can enter on SDSF panels. For complete information about creating WRITER profiles for use with SDSF 1.3, see *SDSF Guide and Reference*.

Authorizing the Use of Your Installation's Printers

You can use RACF to control who can use your installation's printers. Printers at your installation are defined to JES3 by DEVICE statements in the JES3 initialization stream. Printers are also defined in the MVS Configuration Program.

To authorize or restrict the use of your installation's printers, take the following steps:

1. Ask your JES system programmer for the following information:

- A 4-part profile name that represents the printer. The format of the 4-part profile name is:

```
sysname.dev-class.modelno.devnum
```

where:

sysname

identifies the name of the system.

dev-class

specifies the type of device. For printers, you must always specify unit record (UR).

modelno

specifies the model number of the printer.

devnum specifies the 3 or 4-digit hexadecimal device number associated with the printer. Device numbers can be specified by any of the following formats:

```
ddd  
dddd
```

- The universal access authority associated with the printer. A UACC of READ indicates the printer can be allocated to all users in your installation. A UACC of NONE indicates the printer can be allocated only to the users you specify.
- A list of users and groups that have access other than the UACC. READ access allows the device to be allocated to the job submitted by the specified user.
- The security label associated with the printer (if security labels are being used).

2. Create a profile in the DEVICES class to protect each writer:

```
RDEFINE DEVICES profile-name UACC(NONE)
```

- When you are ready to start using the protection provided by the profiles you have created, activate the DEVICES classes:

```
SETROPTS CLASSACT(DEVICES)
```

Using RACF to Authorize the Use of Operator Commands

You can control which groups of users (system programmers and operators) can enter commands. You can use RACF to authorize or restrict users from entering some or all commands, or specific variations of commands, or the consoles from which commands can be entered.

To control the use of operator commands, ask your security administrator to activate the appropriate RACF classes that enable RACF command authorization. The class or classes that must be activated depends on the source from which commands are entered. Table 12 lists the RACF classes that must be activated for each input source:

Table 12. RACF Functions Required to Protect JES3 Resources

Source of Commands	RACF Security Class to Activate
Commands from RJP workstations or NJE nodes	USER (always active) OPERCMDS, and FACILITY
Commands other than RJP or NJE	USER (always active) and OPERCMDS
//*PROCESS statements	OPERCMD5

You must also provide the security administrator with the following information needed to define RACF profiles for commands you want to authorize:

- The command you want to authorize and the profile name associated with that command. Table 13 on page 82 lists the profile names assigned to each JES3 command.
- The userid of the operator or group of operators to whom you want to grant authority.
- For each operator or group of operators:
 - The access authority to be assigned to the operator or group of operators. This is the minimum authority level required to enter each command without being explicitly defined to RACF. Table 13 on page 82 lists the minimum RACF authority level needed to enter each JES3 command.
 - Any conditional restrictions. You can request that operators be restricted to using specific consoles (using the console-id) when entering certain commands. This conditional processing is provided by conditional RACF profile statements. See *z/OS Security Server RACF Security Administrator's Guide* for information about defining conditional processing.

Notes:

- To control commands entering from RJE workstations and NJE nodes, the security administrator must activate the RACF USER class (which is always active by default) and add the workstation or nodeID to the list of operators authorized to enter the command. Also, any NJE node or RJP workstation you are controlling must be defined to the FACILITY class, and the administrator must activate the FACILITY class.
- Your security policy might require auditing of all commands entered, even if they are not valid on your system. You can audit commands that are unknown to JES3 by defining a profile called jesname.UNKNOWN with a universal access value of NONE and an AUDIT value of ALL.

Profile (Entity) Names for JES3 Commands

To authorize or restrict the use of commands, all JES3 commands are associated with an entity name which JES3 passes to RACF each time a command is entered. The security administrator uses the entity names to define command profiles. Depending on how the security administrator defines a command profile, the entity name can be used to authorize all commands belonging to a specific group (for example, all variations of the *MODIFY command) or authorize only a specific command variation (for example, *MODIFY,C). Table 13 lists the entity name associated with each JES3 command and the minimum authority level required to enter that command.

Table 13. RACF Profile Names for JES3 Commands

Command Type	Command	Minimum Authority Needed to Enter Command	Entity Name Format
Dynamic Support Programs	*CALL,DR,M=	UPDATE	jesname.CALL.DR.membername
	*CALL	UPDATE	jesname.CALL.dspname
	*CANCEL,DEV	UPDATE	jesname.CANCEL.DEV.dev
	*CANCEL	UPDATE	jesname.CANCEL.name
	*START,DR,M=	UPDATE	jesname.START.DR.membername
	*START,DEV	UPDATE	jesname.START.DEV.dev
	*START	UPDATE	jesname.START.name
	*RESTART,DEV	UPDATE	jesname.RESTART.DEV.dev
	*RESTART	UPDATE	jesname.RESTART.name
	*FAIL,DEV	CONTROL	jesname.FAIL.DEV.dev
	*FAIL	CONTROL	jesname.FAIL.name
General Commands	*VARY	UPDATE	jesname.VARY.DEV
	*VARY RECOVER	CONTROL	jesname.VARYRECOVER.DEV
	*SWITCH	UPDATE	jesname.SWITCH
	*FREE	CONTROL	jesname.FREE
	*TRACE	UPDATE	jesname.TRACE
	*MESSAGE	READ	jesname.MESSAGE
	*SEND	UPDATE	jesname.ROUTE.CMD.system
	*DUMP	CONTROL	jesname.STOP.DUMP
	*RETURN	CONTROL	jesname.STOP.RETURN
Unknown Commands	unknown	NONE	jesname.UNKNOWN
Inquiry Commands	*INQUIRY,A	READ	jesname.DISPLAY.A
	*INQUIRY,B	READ	jesname.DISPLAY.B
	*INQUIRY,C	READ	jesname.DISPLAY.C

Table 13. RACF Profile Names for JES3 Commands (continued)

Command Type	Command	Minimum Authority Needed to Enter Command	Entity Name Format
	*INQUIRY,D	READ	jesname.DISPLAY.D
	*INQUIRY,F	READ	jesname.DISPLAY.F
	*INQUIRY,G	READ	jesname.DISPLAY.G
	*INQUIRY,J	READ	jesname.DISPLAY.JOB
	*INQUIRY,J,E	READ	jesname.DISPLAY.JOBE
	*INQUIRY,L	READ	jesname.DISPLAY.L
	*INQUIRY,M	READ	jesname.DISPLAY.M
	*INQUIRY,MT	READ	jesname.DISPLAY.MT
	*INQUIRY,N	READ	jesname.DISPLAY.N
	*INQUIRY,NJE	READ	jesname.DISPLAY.NJE
	*INQUIRY,O	READ	jesname.DISPLAY.O
	*INQUIRY,P	READ	jesname.DISPLAY.P
	*INQUIRY,PROCLIB	READ	jesname.DISPLAY.PROCLIB
	*INQUIRY,Q	READ	jesname.DISPLAY.Q
	*INQUIRY,Q,N	READ	jesname.DISPLAY.QN
	*INQUIRY,S	READ	jesname.DISPLAY.S
	*INQUIRY,T	READ	jesname.DISPLAY.T
	*INQUIRY,U	READ	jesname.DISPLAY.U
	*INQUIRY,WTO	READ	jesname.DISPLAY.WTO
	*INQUIRY,X	READ	jesname.DISPLAY.X
Modify Commands	*MODIFY,C	UPDATE	jesname.MODIFY.C
	*MODIFY,CONFIG	UPDATE	jesname.MODIFY.CONFIG
	*MODIFY,E	UPDATE	jesname.MODIFY.E
	*MODIFY,F	UPDATE	jesname.MODIFY.F
	*MODIFY,G	UPDATE	jesname.MODIFY.G
	*MODIFY,J	UPDATE	jesname.MODIFY.JOB
	*MODIFY,J,P	UPDATE	jesname.MODIFY.JOBP
	*MODIFY,L	UPDATE	jesname.MODIFY.L
	*MODIFY,M	UPDATE	jesname.MODIFY.M
	*MODIFY,MT	UPDATE	jesname.MODIFY.MT
	*MODIFY,N	UPDATE	jesname.MODIFY.N
	*MODIFY,NJE	UPDATE	jesname.MODIFY.NJE
	*MODIFY,O	UPDATE	jesname.MODIFY.O
	*MODIFY,Q	UPDATE	jesname.MODIFY.Q

Table 13. RACF Profile Names for JES3 Commands (continued)

Command Type	Command	Minimum Authority Needed to Enter Command	Entity Name Format
	*MODIFY,S	UPDATE	jesname.MODIFY.S
	*MODIFY,T	UPDATE	jesname.MODIFY.T
	*MODIFY,U	UPDATE	jesname.MODIFY.U
	*MODIFY,V	UPDATE	jesname.MODIFY.V
	*MODIFY,V,RECOVER	CONTROL	jesname.MODIFYRECOVER.V
	*MODIFY,W	UPDATE	jesname.MODIFY.W
	*MODIFY,WTO	UPDATE	jesname.MODIFY.WTO
	*MODIFY,X	UPDATE	jesname.MODIFY.X

A sample list of JES3 commands, their entity names, and minimum authority levels is provided in SIATSAMP member IATSM010.

Certain JES3 utilities can also be invoked by placing job entry control language (JECL) `//*PROCESS` statements in job streams. Table 14 lists the profile names and minimum authority needed to authorize the use of each utility.

Table 14. RACF Profile Names for `//*PROCESS` Commands

Statement	Parameter	Minimum Authority Needed	Profile Name
<code>//*PROCESS</code>	CBPRNT	UPDATE	jesname.PROCESS.CBPRNT
	DISPDJC	UPDATE	jesname.PROCESS.DISPDJC
	DISPLAY	UPDATE	jesname.PROCESS.DISPLAY
	DJCPROC	UPDATE	jesname.PROCESS.DJCPROC
	DR,M=	UPDATE	jesname.PROCESS.DR.membername
	ISDRVR	UPDATE	jesname.PROCESS.ISDRVR
	JESNEWS	UPDATE	jesname.PROCESS.JESNEWS
	name (installation DSP)	UPDATE	jesname.PROCESS.name

Some of the profile names provided might not be specific enough to suit your installation's security requirements. For example, your installation might not want the operations personnel to enter a command with a particular parameter. You will need to create a new profile name for the command using installation exit IATUX58. See *z/OS JES3 Customization* for information about using IATUX58.

Who Authorizes Commands When RACF Is Active

If you are using MCS-managed consoles and enable RACF command authority checking, RACF performs all command authorization. Table 15 on page 85 shows command authority checking based on the source of the command.

Table 15. Command Authority Checking When RACF OPERCMDS Class Is Active

Source of Command:	Command Type:	Default Authorization Performed By:	JES3 Installation Exits that Can Affect RACF Authorization
MCS-managed consoles	JES3 commands	RACF	IATUX18 IATUX58 IATUX59
MCS-managed consoles	MVS commands	RACF	None
RJP consoles (see Note 1)	JES3 commands	RACF	IATUX18 IATUX58 IATUX59
NJE consoles (see Note 2)	JES3 commands	RACF	IATUX35 IATUX58 IATUX59
Internally generated commands	JES3 commands	RACF	IATUX58 IATUX59
Device readers (see Note 3)	JES3 commands	RACF	IATUX18 IATUX58 IATUX59
Automation consoles (SVC34)	JES3 commands	RACF	IATUX18 IATUX58 IATUX59
Automation consoles (SVC34)	MVS commands	RACF	None

Notes:

1. An RJP workstation must be defined in profiles for the RACF FACILITY and USER classes if you want RACF to perform command authority checking. If an RJP console is not defined to RACF, JES3 will perform command authorization instead.
2. A node must be defined to RACF if you want RACF to perform command authority checking. If a node is not defined to RACF, JES3 will perform command authorization instead.
3. *Readers Called from MCS Consoles:* Commands entered through device readers called from MCS consoles are assigned the authority of the MCS operator (if console LOGON is required) or the console identifier. The success or failure of these commands depends on the authority of the calling operator.

Using JES3 to Provide Security

JES3 has a number of initialization statement parameters and installation exits that you can use to protect JES3. Some of the resources that you can protect using JES3-provided facilities include:

- NJE communication lines
- RJP communication lines
- Remote workstations
- VTAM sessions
- Tape labels
- Commands
- Output

JES3 Initialization Statements that Affect Security

The following initialization statements identify protection capabilities using JES3 features. You can use RACF to provide additional security protection. Table 16 lists the resources you can protect and the initialization statements and parameters you can use to enable security protection:

Table 16. JES3 Initialization Statements that Provide Security Protection

To protect this JES3 resource	Use this initialization statement	Keyword	Purpose and Considerations
Tape labels	CIPARM	PARM=	The PARM= keyword defines a 21-character parameter string that the MVS converter/interpreter uses when processing jobs. The fourteenth character specifies whether to ignore the bypass label processing (BLP) JCL option.
VTAM sessions	COMMDEFN	P=	The communication subsystem interface definition records initialization statement (COMMDEFN) allows you to define optional user communication subsystem interface (VTAM) parameters. It contains the password keyword (P=) that JES3 uses when issuing an OPEN ACB macro. COMMDEFN uses this password to verify the application programs that have authority to run. This parameter must be the same as the password specified in the PRCTCT parameter of the VTAM APPL application definition statement. See <i>VTAM Installation and Resource Definition</i> for information about the VTAM APPL statement and the PRCTCT parameter.
Commands	CONSOLE	LEVEL=	The CONSOLE statement defines RJP consoles. You can use the LEVEL= keyword to specify an authority level for each console. The authority level defines the commands that operators can enter at consoles. This keyword is ignored when the RACF OPERCMDS class is active and the RJP workstation is defined to RACF.
Commands	CIPARM	AUTH	COMMAND
			The AUTH and COMMAND parameters on the CIPARM initialization statement control whether a command can be issued via the input stream (through a batch job or disk reader for example). Also, the COMMAND JCL statement can be used to enter commands into the system. AUTH and COMMAND do not override RACF security checking.
BSC/NJE nodes	NJERMT	EXPWD=	PWD=

Table 16. JES3 Initialization Statements that Provide Security Protection (continued)

To protect this JES3 resource	Use this initialization statement	Keyword	Purpose and Considerations
EXSIG=	SIG=	PWCNTL=	The NJERMT statement defines a node in a job entry network. You can use the expected password (EXPWD=) keyword to specify a password that the home node expects to receive from a directly connected node at line /*SIGNON time. The password allows a remote node to start a communication line to the home node. You can define the password (PWD=) keyword that the home node must send to a directly connected node at line /*SIGNON time. This password allows the home node to start a communication line to the remote node. In addition, the NJERMT statement contains the EXSIG= and SIG= passwords for dial-up lines. The PWCNTL= keyword allows you to specify whether job passwords should be encrypted, sent without encryption, or checked only at the local node.
Output	OUTSERV	FLASH=	The OUTSERV statement defines default values and standards for printers and punches. You can use the FLASH= keyword to print security classifications on 3800 output. Do not confuse the FLASH= security classifications with RACF security labels that can also be printed on job output. If you are using both FLASH= and RACF to print security classifications, ensure that their meanings agree and that you do not overlay the security label printed by PSF.
RJP lines	RJPLINE	P=	The RJPLINE statement defines the characteristics of a single BSC line (and its respective adapter) that the JES3 global uses for remote job processing. You can use the P= keyword to specify a password that protects the RJP line. This password must be used by any workstation attempting to sign on to this line.
RJP (BSC) workstations (/*SIGNON)	RJPTERM	P=	The RJPTERM statement defines a single remote BSC workstation to JES3. You can specify a /*SIGNON password using the P= keyword. This password must be specified on the /*SIGNON statement used to establish a BSC RJP session. This keyword is ignored when RACF is used to control RJP LOGON passwords.

Table 16. JES3 Initialization Statements that Provide Security Protection (continued)

To protect this JES3 resource	Use this initialization statement	Keyword	Purpose and Considerations
RJP (SNA) workstations (LOGON)	RJPWS	P=	The RJPWS statement defines a single remote SNA workstation to JES3. You can specify a LOGON password using the P= keyword to be used by the workstation operator. This password must be included in the user data specified on the LOGON command passed to the JES3 LOGON exit by VTAM; otherwise JES3 will reject the LOGON. This keyword is ignored when RACF is used to control RJP LOGON passwords.
Print security classifications on output	SYSOUT	FLASH=	The SYSOUT statement defines JES3 SYSOUT class characteristics. You can use the FLASH= keyword to print security classifications on 3800 output. Do not confuse the FLASH= security classifications with RACF security labels which can also be printed on job output. If you are using both FLASH= and RACF to print security classifications, ensure that their meanings agree and that you do not overlay the security label printed by PSF.

JES3 Installation Exits that Affect JES3-Provided Security

Several JES3 installation exits affect or provide security for JES3 resources such as JES3 commands, job output, and TSO/E commands that access spool. Table 17 lists the installation exits available for controlling JES3-provided security.

Table 17. Installation Exits that Affect JES3-Provided Security

JES3 Installation Exit	Purpose:
IATUX18	Allows you to modify a JES3 command and validate the console's authority of entering the command.
IATUX19	Provides the user with the contents of the temporary output service element (OSE) constructed for a data set. This exit can be used to set the FLASH= default value to print security classifications on 3800 output.
IATUX30	Examines the authority level for TSO/E terminal commands. This exit can be used to allow or disallow TSO/E STATUS, CANCEL, or OUTPUT commands if authority checking is not being performed by TSO/E installation exit IKJEFF53.
IATUX35	Allows you to authorize commands received from network nodes.

Using JES3 to Control Access to RJP Workstations

Remote job processing (RJP) console operators must /*SIGNON (for BSC/RJP workstations) or LOGON (for SNA/RJP workstations) before using them. You can require that RJP operators enter a password during workstation LOGON using JES3 initialization statement parameters. You can also use RACF to provide password protection for RJP workstations. See "Authorizing Console Access" on page 76 for information about using RACF to control the use of RJP passwords.

Controlling Passwords for SNA RJP Workstations

You can use the P= keyword on the JES3 RJPWS initialization statement to define workstation passwords and require operators to give the workstation password when they LOGON at SNA RJP workstations. You can also define the number of times that an operator can attempt to LOGON before the workstation must be restarted by the JES3 operator. RACF password protection and JES3 password protection for SNA RJP workstations are mutually exclusive. That is, JES3 ignores the RJPWS statement if you enable RACF password protection.

Controlling Passwords for BSC RJP Workstations

You can use the P= keyword on the JES3 RJPTERM initialization statement to define workstation passwords and require operators to give the workstation password when they /*SIGNON at BSC RJP workstations. RACF password protection and JES3 password protection for BSC RJP workstations are mutually exclusive. That is, JES3 ignores the RJPTERM statement if you enable RACF password protection.

Using JES3 to Authorize the Use of Operator Commands

MVS, JES3, and RACF allow you to control which commands operators can enter at consoles. Controlling the use of operator commands prevents unauthorized operators from entering commands that can violate your installation's operating guidelines or cause potential damage to your system or data. For example, your installation might want to limit the use of certain commands only to experienced operators or system programmers, such as those commands that initiate a dynamic system interchange or dump the contents of storage.

MVS and JES3 perform independent authority checking.

When RACF is active, RACF performs authority checking for all commands. You can also modify RACF command authorization using JES3 installation exits. "Using RACF to Authorize the Use of Operator Commands" on page 81 describes how you can use RACF to provide command authority checking.

Using JES3 to Authorize Commands from RJP Consoles

JES3 groups its commands into command authority levels that range from 0 to 15. You can assign a JES3 authority level to an RJP console by specifying the LEVEL= keyword on the JES3 CONSOLE initialization statement for each console.

For every JES3 command entered, JES3 checks the authority of the console from which you entered the command. If you have authorized the console to enter commands at that authority level or lower, JES3 processes the command; otherwise, JES3 rejects the command and displays an error message.

After initialization you can alter a console's authority level by entering a *MODIFY,O command. Use the *INQUIRY,O command to display a console's current authority level. You can use the JES3 installation exit IATUX18 to redefine authority levels in specific cases. See *z/OS JES3 Customization* for information about IATUX18.

Table 18 on page 90 shows the commands and authority levels for RJP consoles (BSC and SNA).

Table 18. Authority Levels for Remote Consoles

JES3 Authority Level	Allowable JES3 Commands	Restrictions
0-4	No remote console support exists.	
5-9	*START *RESTART *CANCEL *INQUIRY *MESSAGE	<p>For BSC terminals, the *START, *RESTART, and *CANCEL must specify a remote terminal name which is part of the remote workstation.</p> <p>For SNA terminals, the *START, *RESTART, and *CANCEL must specify a logical device name which belongs to the SNA workstation.</p> <p>The only *INQUIRY commands allowed are *I,B, *I,D, *I,J, *I,O, *I,P, *I,Q, and *I,U. The T=wsname parameter is inserted into the command by JES3 if it is not specified.</p> <p>The *I,D,D= command will only be accepted for a single device associated with the remote workstation. You can specify a long form (such as D=WS001PR1) or a short form (D=PR1).</p>
10-14	All of the commands listed above plus: *CALL*MODIFY *FREE (non-directed)	<p>The input (IN=) and output (OUT=) keywords specified on a *CALL command must be devices associated with the remote workstation. 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.</p> <p>The restrictions discussed previously for the *START, *RESTART, *CANCEL and *VARY commands also apply for this authority level. The *VARY,RECOVER and *MODIFY,V,RECOVER commands can be entered only with a JES3 authority level of 15. The only DSPs that can be invoked by the *CALL command are CR, and WTR. These DSPs can only be invoked from a BSC RJP workstation because the only utility supported by SNA RJP is the JESNEWS DSP. Card-reading DSPs cannot be called from remote nonprogrammable work stations, but they can be called to a remote reader by the local operators.</p> <p>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.</p> <p>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 workstation.</p> <p>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.</p>
15	All commands except: *DUMP *RETURN *SWITCH *TRACE	JES3 does not provide any default parameters.

Using JES3 to Authorize Commands from NJE

Certain JES3 commands can be entered into your system from other nodes in a NJE network. JES3 allows only *INQUIRY and *MODIFY commands to be entered through NJE.

The *INQUIRY commands allowed through NJE are:

- *INQUIRY,Q,[N]
- *INQUIRY,B
- *INQUIRY,J,[E]

The *MODIFY commands allowed through NJE are:

- *MODIFY,J,C
- *MODIFY,J,CP
- *MODIFY,J,CO
- *MODIFY,J,H
- *MODIFY,J,R

For additional information about using Network Job Entry to send and receive work throughout a network, see Chapter 11, “JES3 Networking” on page 293.

Using JES3 to Authorize Commands from TSO/E

If TSO/E installation exit IKJEFF53 is not performing authority checking, you can use JES3 installation exit IATUX30 to authorize the use of the TSO/E CANCEL,STATUS, or OUTPUT commands. JES3 invokes IATUX30 each time a user enters one of these TSO/E commands.

Although IATUX30 gives you the ability to determine whether a user is authorized to enter a TSO/E CANCEL command, RACF determines on a profile-by-profile basis whether the CANCEL command is valid for the affected job (That is, when the RACF JESJOBS class is active.).

Using JES3 to Authorize Commands from MCS Consoles

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 JES3 commands are allowable from MCS consoles.

Table 19. JES3 Commands Allowed from MCS Consoles

MCS Authority Level (Command Group)	Corresponding JES3 Authority Level	Allowable JES3 Commands From MCS Consoles
Informational	0	*INQUIRY *MESSAGE
System Control	5	All of the commands listed above plus: *CALL *CANCEL *RESTART *SEND *START

Table 19. JES3 Commands Allowed from MCS Consoles (continued)

MCS Authority Level (Command Group)	Corresponding JES3 Authority Level	Allowable JES3 Commands From MCS Consoles
Console Control I/O Control	10	All of the commands listed above plus: *MODIFY *SWITCH *TRACE *VARY
Master	15	All of the commands listed above plus: *FREE,con *DUMP *FAIL *MODIFY,V,RECOVER *RETURN *VARY,RECOVER

Using JES3 to Authorize Commands Entered in Job Streams

JES3 allows MVS and JES commands to be entered in a job stream through the use of the COMMAND JCL statement. The COMMAND= and AUTH= parameters on the CIPARM initialization statement allow you to control what jobs can use COMMAND statements and which commands they can issue. Refer to *z/OS JES3 Initialization and Tuning Reference* for more information on these parameters.

Part 4. JES3 Job Management

Chapter 4. JES3 Job Management	95	Initializing the DJC Job Network	126
Input Service	95	Scheduling the DJC Job Network	126
Reader Phase	95	Modifying the DJC Job Network	127
Control Statement Processing Phase	95	Terminating the DJC Job Network	127
Selecting a Main for Job Execution.	97	DJC Completion Option	127
Modifying JCL and JES3 Control Statements	98	Nonstandard DJC Job Processing	128
Preventing a Job from Dominating JSAM		Controlling Job Selection	128
Buffers	98	Determining Main Eligibility	128
Converter/Interpreter Service	99	Determining Job Eligibility	128
Converter/Interpreter Phase.	99	Controlling the Job Mix on Each Processor	128
Prescan Phase	100	Defining the Job Selection Environment	128
Postscan Phase	100	Default Job Selection	131
Cataloged Data Set Resolution.	100	Starting and Stopping Initiators	131
C/I Service Preparation for the Main Device		Defining Logical Storage for Processors.	132
Scheduler.	101	Controlling Job Scheduling	133
JES3 Resource Allocation	101	Assigning a Job Selection Mode	133
MVS Allocation Compared with JES3 MDS		Defining Job Selection Parameters	133
Allocation	102	Defining JES3 Job Classes	135
MVS Allocation.	102	Grouping JES3 Job Classes	136
Main Device Scheduler (MDS) Allocation	102	Automatic Restart Management	137
Volume Fetch	103	Output Service	138
System Select	104	Queueing Output	139
Allocation	104	Override Sequences for Output Data Set	
Volume Verification	105	Information	140
System Verify	105	Scheduling Output	151
Breakdown	106	Determining Which Output Parameters	
Types of Setup	106	Apply	151
Job Setup.	107	Writing Output.	152
High Watermark Setup	107	External Writers/SAPI Applications	154
Explicit Setup	108	NJERDR	154
Initializing MDS	108	Internal Reader.	154
CLASS Initialization Statement	108	Accessing Job Output Through TSO.	155
DEVICE Initialization Statement	108	Output Service Installation Exits	155
GROUP Initialization Statement	109	Purge	156
SELECT Initialization Statement	110		
SETACC Initialization Statement	110		
SETNAME Initialization Statement	110		
SETPARAM Initialization Statement	111		
SETRES Initialization Statement	111		
STANDARDS Initialization Statement	111		
Operator Control of MDS	111		
Job Selection and Scheduling	111		
Comparison of JES3 and WLM Initiator			
Management.	111		
Classifying Jobs	116		
Pre-Execution Delays	117		
Defining Service Classes and Performance Goals	120		
Recommendations For Defining Service			
Classes	121		
Job Selection Algorithm for JES3-Managed			
Initiators	121		
Job Selection Algorithm for WLM-Managed			
Initiators	123		
Deadline Scheduling	125		
Dependent Job Control	125		
Early Dependent Job Control (DJC) JCL Scan	126		

Chapter 4. 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 section describes how each phase contributes to job management and discusses how you can use JES3 initialization statements and JES3 installation exits to tailor job management. See *z/OS JES3 Customization* for a description of how to code installation 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 them 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 workstation. The reader phase treats jobs from a remote workstation as though the job came from a card reader.

Figure 14 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 installation exit `IATUX17` to modify the sequence of scheduler elements. For additional information about installation exits, see *z/OS JES3 Customization*.

Figure 14 shows job flow during the control statement processing phase. The control statement processing phase also determines which procedure library

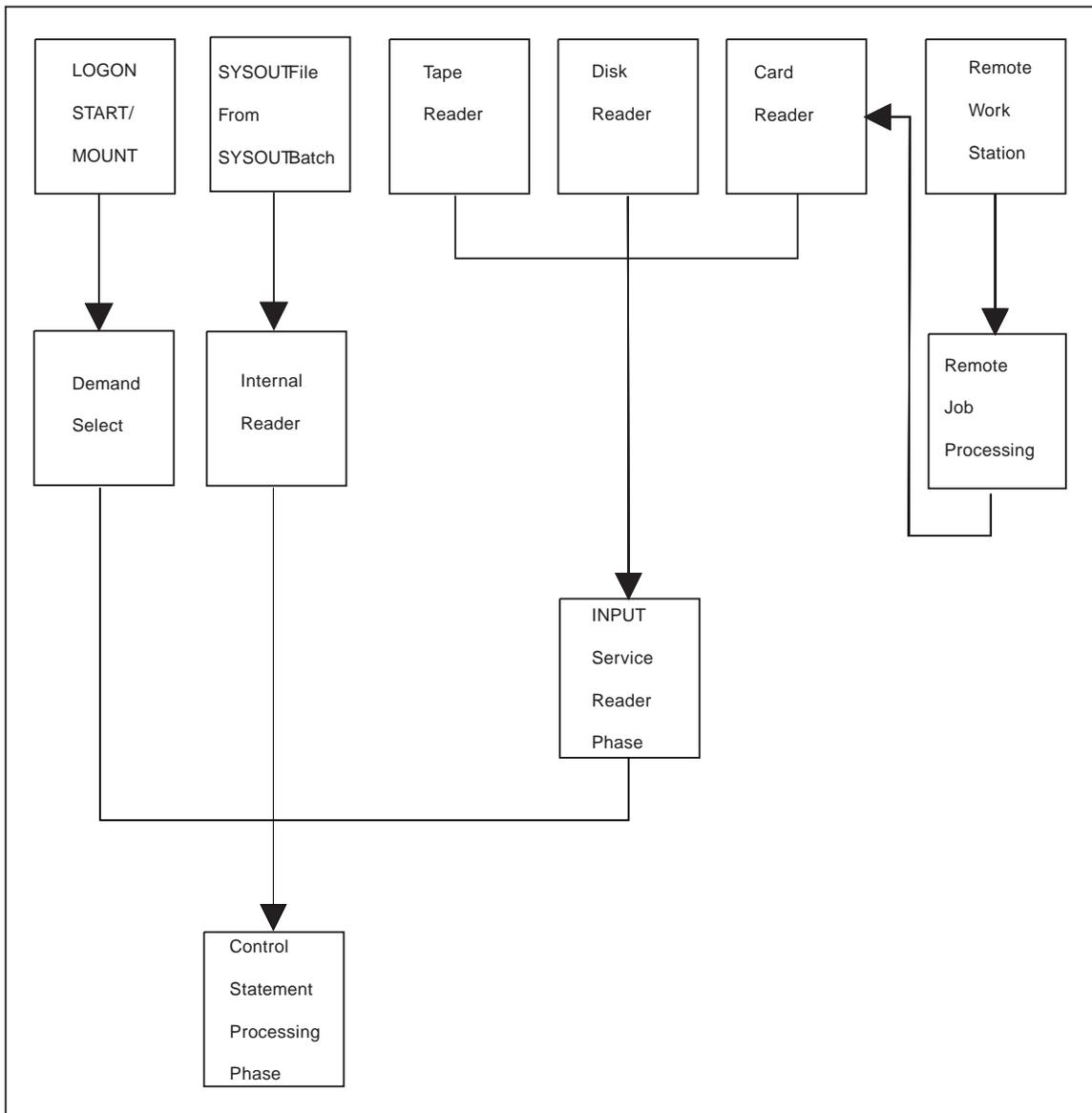


Figure 14. Overview of Job Flow to Input Service

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. You can specify a default procedure library on the JES3 STANDARDS initialization statement as follows:

- For an internal reader job, the INTPROC parameter
- For a started task, the STCPROC parameter
- For a TSO/E 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 Main for Job Execution

Sometimes a user may want to select the main on which to execute a particular job. For example, the job could need a feature or require an I/O device that is unavailable to some mains or not defined to JES3. To be sure the job executes on the right main, 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 main.
- Specify a scheduling environment by coding the SCHENV parameter on the job's JOB statement. This requires your installation to define the scheduling environment in the WLM policy (see *z/OS MVS Planning: Workload Management* for information about how to define resources and scheduling environments.).

If the user has specified a main by coding the SYSTEM parameter on the `//*MAIN` statement, the control statement processing phase assigns the job to execute on the specified main. Otherwise, the control statement processing phase assigns the job to execute on a main eligible for the job's job class. The SYSTEM= keyword on the JES3 CLASS initialization statement defines the eligible main(s) for a job class.

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

If the user specified a scheduling environment, then the status of the scheduling environment will be used in addition to the systems from the `//*MAIN` statement or JES3 CLASS initialization statement to determine where a job will execute.

For example, suppose you have three systems in your SYSPLEX: SY1, SY2, and SY3. Suppose that a job specifies a scheduling environment, IMSPROD, which is available on SY2 and SY3. The following table shows where the job will execute depending on what was specified on the `//*MAIN` JECL or CLASS initialization statements:

Table 20. Where a job will execute based on `//*MAIN` and CLASS statements

<code>//*MAIN</code> or CLASS Statement Information	Can Job Execute? On what system?
Nothing Specified	YES — SY2 and SY3
<code>//*MAIN SYSTEM=SY2</code>	YES — SY2
<code>//*MAIN SYSTEM=SY1</code>	NO
<code>//*MAIN SYSTEM=(SY1,SY3)</code>	YES — SY3
CLASS, ...SYSTEM=SY2	YES — SY2

Table 20. Where a job will execute based on `/*MAIN` and `CLASS` statements (continued)

<code>/*MAIN</code> or <code>CLASS</code> Statement Information	Can Job Execute? On what system?
<code>CLASS,...SYSTEM=SY1</code>	NO
<code>CLASS,...SYSTEM=(SY1,SY3)</code>	YES — SY3
<code>CLASS,...SYSTEM=ANY</code>	YES — SY2 and SY3

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 installation 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`
- JES3 control statements (statements that begin with `/*`) as allowed by specifying the `ALTJCL=ACCEPT` keyword value on the `STANDARDS` initialization statement.

JES3 control statements begin with `/*`. By default, JES3 also accepts JES3 control statements that begin with `/*`. If you wish to require that both slashes be used, specify `ALTJCL=ERROR`, `ALTJCL=IGNOREW`, or `ALTJCL=COMMENT` on the `STANDARDS` initialization statement, depending on whether you want messages issued and whether you want a job containing such statements to be flushed. See *z/OS JES3 Initialization and Tuning Reference* for how to specify `ALTJCL=` on the `STANDARDS` initialization statement.

To verify or change the `JOB` JCL statement, implement installation exit `IATUX28`. You can also use this exit to remove the job from the system.

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

To verify or change `DD` JCL statements, use installation exit `IATUX34`. This installation 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, implement installation exit `IATUX44`. This installation exit also allows you to request that JES3 ignore the statement.

Preventing a Job from Dominating JSAM Buffers

To prevent a job with many `JCL SYSIN DD` statements from dominating JSAM buffers, use the job `SYSIN DD` statement limit. If a job contains more `SYSIN DD` statements than the job `SYSIN DD` statement limit allows, JES3 cancels the job.

To select the job `SYSIN DD` statement limit, determine the number of `SYSIN DD` statements that are in the largest job you want to run at your installation.

Note: 12 `SYSIN DD` statements will fit in a 4K JSAM buffer.

Specify the number of `SYSIN DD` statements on the `MAXINDD=` parameter of the `STANDARDS` initialization statement. The next time you do a hot start with refresh, warm start, or cold start using the initialization stream containing that `STANDARDS` statement, JES3 uses the job `SYSIN DD` statement limit you specified.

Converter/Interpreter Service

Converter/Interpreter (C/I) service controls the conversion of JCL statements to Converter/Interpreter text and then into control blocks. This service comprises primarily the JES3 CI and POSTSCAN dynamic support programs, 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 JCL into scheduler control blocks. At this time, the scheduler control blocks are created in the scheduler work area (SWA).
- *Prescan phase:* Extracts job requirements 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 JES3 spool.
- *Postscan phase:* Locates data sets and gathers information for use in JES3 device setup.

You can define one or more C/I functional subsystem (FSS) address spaces to perform the converter/interpreter, prescan, and locate (catalog search) portion of the postscan phase for some or all jobs. A C/I FSS can operate on the global or any local main.

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 by implementing installation exit routine IATUX26.

For detailed information about how you can set up and influence C/I service, see Chapter 5, "Defining and Managing C/I Service".

Converter/Interpreter Phase

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

When JES3 has a job that can be scheduled for C/I processing, it 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 main 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 implementing installation exit IATUX46. If the initialization stream defines C/I FSS address spaces, you can override JES3's selection of an address space by implementing installation exit IATUX49. JES3 schedules the job to an available CI DSP based on the installation exit routines' responses.

If the number of JCL statements in a job exceeds the job JCL statement limit, the JES3 calls installation exit routine IATUX41 to see if it should cancel the job. If so, JES3 cancels the job 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, JES3 links to the MVS interpreter to create the scheduler control blocks from the Converter/Interpreter text.

At the end of interpretation, the Storage Management Subsystem (SMS) scans the SWA control blocks to determine the required catalogs for new data sets. Catalogs for jobs can be SMS-managed or JES3-managed. SMS uses the list of SMS-managed volumes to determine which mains are eligible for catalog searches. A list of catalogs that are not SMS-managed is returned to JES3 and becomes part of JES3's setup requirements.

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 extract each job's resource requirements from the SWA control blocks created in the interpreter phase.

On entry to the prescan phase, JES3 examines the SETUP parameter on the JES3 STANDARDS initialization statement:

- If you specify SETUP=NONE (that is, you do not want JES3 to perform pre-execution setup for the complex), JES3 does not build extract resource requirements.
- If you specify any value other than NONE on the SETUP= keyword, JES3 determines the job's resource requirements.

Installation 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.

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 express canceled 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 JESYSMSG data set, and the job is then canceled-with-print on completion of the CI DSP. For more information on JCLTEST and JSTTEST, see *z/OS JES3 Diagnosis*.

If the job is being processed in a C/I FSS, JES3 determines whether the job is eligible to have catalog searches performed on the main where the C/I FSS is executing. If catalog searches can be performed on that main, locate processing occurs in the C/I FSS and control then returns to the JES3 global address space for the remainder of postscan processing.

Postscan Phase

The functions of the postscan phase are to resolve cataloged data set references and prepare the job for the main device scheduler (MDS).

Cataloged Data Set Resolution

If SMS is active, JES3 calls it to search the system catalogs and collect scheduling information for the job. SMS searches the system catalogs to obtain information about a job's data sets when the job's JCL does not specify unit information.

If SMS fails to find a data set name, the postscan phase calls installation exit IATUX07. Through this installation exit, you 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. JES3 cancels jobs that request an unavailable catalog.

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. HSM derives an SMS storage class and a SMS storage group for SMS-managed data sets.

In addition, whenever the response of a LOCATE request has been received, you can use installation exit routine IATUX11 to inhibit printing of the LOCATE request/response in the JESYSMSG data set.

C/I Service Preparation for the Main Device Scheduler

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

You can have installation exit IATUX08 examine the setup requirements for each job that uses job setup. Once implemented, IATUX08 can modify the type of job setup or fail a job before the main device scheduler receives control. See *z/OS JES3 Customization* for a complete description of installation 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 the JES3 main device scheduler or use MVS (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).

Note: You need to be aware that if MDS is used to provide resource management, MDS takes into account the availability of a job's scheduling environment during resource allocation. This scheduling environment is a list of resource names and their required states that are used to help ensure that units of work are sent to systems that have the appropriate resources to handle them. Scheduling environments and resources are defined by the installation in the WLM policy (see *z/OS MVS Programming: Workload Management Services* for more information).

MVS Allocation Compared with JES3 MDS Allocation

MVS and MDS allocation consider a job's resource requirements at different levels. MVS 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 MVS allocation.

MVS Allocation

In systems that do not use MDS or the WLM functions of MVS, jobs are presented to MVS 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, MVS 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, MVS 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 MVS 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. Also, MDS cooperates with the workload management (WLM) component of MVS to ensure that the scheduling environments for jobs are honored.

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 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 JES3 `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 main, a combination of jobs that will execute without contention for both the sharable and nonsharable devices and data sets attached to that main. 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 main.

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 MVS

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. MVS 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.

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.

If a job is assigned a scheduling environment (for example, the SCHENV= parameter was specified on the JOB statement), the availability of the job's scheduling environment is taken into consideration when determining which systems have the resources for a job. If a job's scheduling environment is not available on a particular system, the job will not run on that system even if the other resources required by the job (e.g. DASD volumes, SMS storage groups) are available on that system. If the scheduling environment later becomes available on a system, the job will be able to run on that system provided that the resources required by the job are also available on that system.

MDS is divided into the following stages:

- Volume fetch
- System select
- Allocation
- Volume verification
- System verify
- 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 mains 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 which 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 JES3 SETPARAM initialization statement. When you select the fetch option, 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 you do not specify the volume fetch option, jobs go directly into the system select stage of MDS if the job requires SMS-managed resources, or into allocation if the job does not require SMS-managed resources.

System Select

The system select phase of MDS is performed when a job requires one or more resources managed by the storage management subsystem (SMS). If the job does not require any SMS-managed resources, the job proceeds directly to MDS allocation.

JES3 is not aware of the availability or connectivity of SMS-managed resources. If a job requires SMS-managed resources, JES3 requests SMS to determine the availability of those resources and to determine which mains have access to those resources.

If a required SMS-managed resource is temporarily unavailable, the job waits in system select until the resource becomes available. You can use an *INQUIRY,S,SS,J= command to determine why jobs are waiting. If all required SMS-managed resources are available, SMS provides JES3 with a list of mains that have access to those resources. JES3 uses the list of eligible mains passed by SMS to determine which mains have access to all of the required resources for the job (both SMS-managed and MDS-managed.)

If JES3 determines that one or more mains have access to all of the required resources, the job proceeds into the allocation phase. If no mains have access to all of the required resources, JES3 invokes user exit IATUX61 to determine whether the job should be placed on the MDS error queue or canceled. If you do not implement IATUX61, MDS places the job on the MDS error queue where an operator must either restart the job or cancel it using a *RESTART,SETUP or *CANCEL,SETUP command respectively. See *z/OS JES3 Commands* for information about how to restart or cancel jobs in setup. Refer to *z/OS JES3 Customization* for information about using IATUX61.

Allocation

This phase of MDS uses allocation algorithms to provide required devices. When allocation is successful, JES3 issues mount request messages for all required volumes except:

- Deferred mount requests - where no mount is as yet requested but the device that the volume is to be mounted on is allocated to the user.
- Permanently resident volumes - where the mount is unnecessary
- Multi-volume mount - where only the first volume of a multi-volume data set is mounted; secondary volumes are not mounted until required.

You can use the ALLOCATE= keyword on the JES3 SETPARAM initialization statement to control how you want jobs processed during MDS allocation. If you specify ALLOCATE=AUTO (default), MDS sends incoming jobs directly into allocation unless a job requires SMS-managed resources. If a job requires SMS-managed resources, the job is first sent to system select before proceeding into the allocation phase.

If you specify ALLOCATE=MANUAL, 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 that 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. Setup is performed for dynamic allocation requests at the time each request is made.

To start the allocation phase of MDS, a job is selected from the ALLOCATE queue. MDS examines the job's resource requirements and attempts to allocate the required devices, volumes, and data sets. If a job requires SMS-managed resources, only the data set(s) is allocated. JES3 is not aware of SMS-managed volumes and devices. When MDS initially tries to set up a job, it records the total device, volume, and data set requirements for the job. When allocation fails because needed devices, volumes, or data sets are unavailable, the job is queued for another attempt. However, MDS sends jobs back to the system select phase of MDS if all of the following conditions occur:

1. A job requires both SMS-managed resources and MDS-managed resources.
2. The list of eligible mains determined by the system select phase do not have access to both the MDS-managed resources and the SMS-managed resources.
3. One or more mains not in the original list of mains has access to all of the required resources, and the SMS-managed resources are temporarily unavailable to those mains.

In the latter case, the job is sent back to system select and waits until the required SMS-managed resources become available on a main that also has access to the required MDS-managed resources.

Device selection (through initialization parameters) limits the number of mains that can execute a job whenever a needed I/O device is not shared among all eligible mains. Any main 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 messages that instruct you to mount a job's required volumes. You can implement installation exit IATUX62 to validate, accept, or override JES3's decision about whether a volume has been successfully mounted. JES3 invokes IATUX62 after the volume verification phase of MDS.

The VERIFY function automatically obtains the volume serial number, label status, and other information for MDS after you mount the job's required volumes. You can install installation exit IATUX25 to validate any nonstandard labels used in the installation. When all volumes are properly mounted, the job is ready for execution. However, if the job requires SMS-managed resources, it proceeds to the system verify phase of MDS before execution. Device types other than tape or disk do not require operator action.

For more information about installation exits IATUX25 and IATUX62, see *z/OS JES3 Customization*.

System Verify

The system verify phase of MDS is performed when a job requires SMS-managed resources. This phase of MDS ensures that all of the SMS-managed resources required by a job are still available before execution. For example, if a job spends

too much time in MDS allocation or a long period of time elapses between a mount request and the actual mounting of a required volume, one or more of the SMS-managed resources could become unavailable.

If the status of SMS-managed resources required by the job has not changed, that is, all of the required SMS-managed resources are still available and are accessible by the eligible main(s), the job can proceed into execution.

However, if SMS-managed resources required by a job are no longer available, JES3 generates a new list of systems where SMS managed resources are available. JES3 then checks this list against the following:

- All jobs to see if they can access the SMS and JES3 resources
- Batch jobs to see if the class and group are available on the systems where SMS managed resources are available.
- Batch jobs with associated scheduling environments to see if there are any systems where the scheduling environment and the SMS managed resources are available.

If there are no systems where all of the above resources are available, the job is sent to the breakdown phase where MDS deallocates resources held by the job. MDS then sends the job back to the system select phase where MDS retries allocation.

Breakdown

JES3 automatically performs the breakdown phase of MDS when a job no longer needs a resource such as a data set, volume, or device. The resource is then available for use by other jobs. MDS does not deallocate SMS-managed resources other than data sets because JES3 is not aware of SMS-managed devices and volumes.

JES3 issues messages that indicate whether volumes should be retained for use by 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. You must restart or cancel these jobs using JES3 commands. You can prevent operator intervention by coding installation exit IATUX61. IATUX61 allows you to cancel jobs that would otherwise be placed on the MDS error queue. For more information about installation exit IATUX61, see *z/OS JES3 Customization* .

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 *z/OS MVS 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.

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.

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). If the SETUP 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 or when the specified setup is overridden by installation exit IATUX08.

Cataloged data sets may reference devices that are valid for use during HWS. You define these devices using the `HWSNAME` initialization statement. If a cataloged data set references a device that has been removed as a HWS device (that is, the device has been removed from the `HWSNAME` initialization statement), that data set is no longer eligible for HWS processing.

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 the routing information for all JES3 and MVS device-related messages. The `XUNIT` parameter accepts an MVS routing code or a JES3 destination class for messages that pertain to the device being defined. If you want a particular RJP console to receive messages for this device, the message destination must be assigned to the RJP console via the `DEST=` parameter of the `CONSOLE` statement. For efficiency, the console should be near the device. Use the `ROUTCODE` parameter on the `CONSOLE` statement in the `CONSOLxx` parmlib member to assign message destinations to MCS consoles.

As an example, suppose the following direct access and tape devices are defined:

```

DEVICE,XTYPE=(3390,DA),XUNIT=(130,SY1,S5,ON)
.
.
.
DEVICE,XTYPE=(3390,DA),XUNIT=(137,SY1,S5,ON)
DEVICE,XTYPE=(3390,DA),XUNIT=(230,SY1,S6,ON)
.
.
.
DEVICE,XTYPE=(3390,DA),XUNIT=(237,SY1,S6,ON)
DEVICE,XTYPE=3490,XUNIT=(180,SY1,S7,ON)

```

In order for RJP work station T0001 to receive messages associated with these devices, the following CONSOLE statement would have to be specified:

```

CONSOLE,TYPE=RJP,JNAME=T0001,DEST=(S5,S6,S7)

```

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=(3390,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=(3390,DA,PR),XUNIT=(230,SY1,S1,,132,SY2,S2)

```

You can also specify that a device is defined to all processors by using the processor name *ALL. For example:

```

DEVICE,XTYPE=(3390,DA,PR),XUNIT=(230,*ALL,S1)

```

When using *ALL, the same device number, message destination, and initial online/offline status apply to every processor. You can define a range of devices with a single device statement by using the NUMDEV keyword. For example:

```

DEVICE,XTYPE=(3390,DA,PR),XUNIT=(230,*ALL,S1),NUMDEV=5

```

indicates that devices 230, 231, 232, 233, and 234 are defined on all processors.

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.

Devices with 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 *z/OS 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 to run. JES3 periodically increases the job's selection priority in an attempt to run the job by the specified deadline. DJC allows you to create a network of related jobs.

Comparison of JES3 and WLM Initiator Management

The installation can have JES3 or Workload Manager (WLM) or both manage initiators for batch jobs. JES3 or WLM control of initiators is at the job class group level. To specify whether JES3 or WLM manages initiators for a job class group, the installation uses the MODE parameter on the GROUP initialization statement. If MODE=JES is specified or defaulted, the initiators are managed by JES3. If MODE=WLM is specified, the initiators are managed by WLM. The MODE parameter can also be changed dynamically using the *MODIFY,G command. A

group must be either WLM managed or JES3 managed; it cannot be WLM managed on one system and JES3 managed on another.

There are a number of reasons why you might choose WLM initiator management over JES3 initiator management:

- Fewer and simpler externals exist.

Less externals are needed in JES3 to control WLM-managed initiators and to perform workload balancing. Once the service administrator defines the performance goals and classification rules in the WLM policy, the system takes over the job of starting and stopping initiators.

- Externals reflect customer expectations.

With JES3 initiator management, it is the installation's responsibility to determine the number of initiators to be started on each system, the correct mix of jobs, and so forth. The externals do not reflect an actual performance goal, such as one hour turn around time for jobs in class X. How do you translate one hour turn around time into initialization statements?

With WLM initiator management, initiators are managed by WLM according to the service classes and performance goals specified in the WLM policy. The performance goals are typically expressed in terms that are found in service level agreements (for example, one hour turn around time).

- Dynamic, goal oriented initiator management exists.

The system adapts to changing conditions and how well the work is meeting its performance goals. The current JES3 initiator management puts the responsibility on the system programmer/operator to manage the work.

Workload balancing algorithms used by JES3 are difficult to define and static in nature; they require an operator command and sometimes a JES3 hot start to change. But even if they can be changed easily, why should human intervention be required? An operating system can better perform these tasks.

- Workload balancing across a SYSPLEX is automatic.

WLM decides when to start initiators and how many to start based on performance goals and the importance of batch work with respect to other work.

Note: WLM management of initiators does not necessarily imply that there will be an equal number of initiators on each system.

A comparison between JES3 and WLM initiator management results in the following.

Initiator Control

JES3-managed initiators (how many, when to start them, and where to start them) are controlled by JES3 based on the EXRESC parameter of the GROUP initialization statement and through the use of the *MODIFY,G command.

WLM-managed initiators are controlled entirely by WLM. WLM determines how many initiators to start, when to start them, and where to start them based on performance goals in the WLM policy, backlog of jobs, and system capacity.

Externals

JES3-managed initiators are expressed in terms of resources, not goals. For example, the EXRESC parameter on the GROUP initialization statement is used to specify how many initiators should be started, on what system, and when they should be started and stopped. The JOBMIX and CHOICE parameters on

the SELECT initialization statement are used to specify the correct mix of jobs and the best job to select based on the number of active initiators on a particular system.

WLM-managed initiators are expressed in terms of goals and their importance in the WLM policy. For example, batch work in class A should have one hour turn around time. Batch work in priority 0-2 has a discretionary goal and should be run only when there is excess capacity.

System Performance

JES3-managed initiators are started and stopped based on installation definitions and the backlog of jobs. How the system is performing, whether work is meeting its performance goals, or whether work is more important than other work is not considered. If you ask JES3 to start 100 initiators on a particular system, it will do so without regard to whether it is the correct thing to do. As a result, once jobs are selected to run, there is no guarantee that the jobs will actually make progress, because the jobs will be under SRM's control at that point and subject to the performance goals defined in the WLM policy.

WLM-managed initiators are started and stopped based on performance goals, available capacity, work importance, and information from JES3 about the backlog of jobs.

Association

JES3-managed initiators are associated with a job class group as defined in the GROUP initialization statement. A job class group consists of one or more job classes defined in the CLASS initialization statement. An initiator that is started for a particular job class group only selects jobs that belong to that group.

WLM-managed initiators are associated with a service class. An initiator started for a service class only selects work from that service class. A service class is a group of work that has the same performance characteristics, resource requirements, or importance. Service classes are defined in the WLM policy. Jobs are assigned a service using the classification rules in the WLM policy. See *z/OS MVS Planning: Workload Management* for more information on defining service classes and classification rules.

Note: Even through WLM management of initiators is at the job class group level, there is no one to one correspondence between job class groups and service classes (unless you define it that way in the classification rules in the WLM policy). Therefore, for a particular job class group there can be jobs with different service classes, but all jobs in that group will run under either a JES3-managed or WLM-managed initiator.

Job Selection

For JES3-managed initiators, jobs are selected within a job class group by priority. If there is more than one job with the same priority, the jobs are ordered by the time they are ready to be selected to run (that is, when they arrive on the GMS select queue). Changing a job's priority affects when the job is selected to run.

For WLM-managed initiators, jobs are selected within a service class by their *main service arrival time*. Main service arrival time is the elapse time for C/I processing of a job. Job priority does not influence whether a job is selected to run first; therefore changing the priority for a job has no effect unless such a change causes the job's service class to change to one having more aggressive goals.

Because main service arrival time is used to order jobs waiting for a WLM-managed initiator, a job in a WLM-managed group that requires setup can actually be selected to run before that it would if it was in a JES3-managed group. For example, suppose the following situation exists:

Table 21 illustrates how JES-managed and WLM-managed jobs are selected.

- Two jobs (JESTAPE and WLMTAPE) waiting in MDS processing for their tapes to be mounted. One job (JESTAPE) is in a JES-managed job class group and the other (WLMTAPE) is in a WLM-managed job class group.
- Assume that there are four jobs waiting to be selected to run in both the JES-managed and WLM-managed job class groups.
- Assume that two of the jobs in each group completed C/I processing after the jobs that require the tape mounts. That is, their main service arrival times are after the jobs that require the tape mounts.
- Assume that all jobs in both groups have the same priority and take the same amount of time to run.

Explanation

If the tape mounts are satisfied for both jobs at the same time, the job in the WLM-managed group will be selected to run first, because jobs in the JES-managed groups are ordered by priority and the job (JESTAPE) in the JES-managed group will be added to the end of the queue. Jobs in the WLM-managed groups are ordered by main service arrival time; therefore, the WLM-managed job (WLMTAPE) will be added to the middle of the queue; the amount of time the job actually holds the resource will be shorter compared to the JES-managed jobs, because it takes longer for the JES-managed job to be selected to run.

Table 21. Overview - Job Selection

GMS Select (before allocation complete)						MDS Allocation	
JES	JESJOB1 (Time=4)	JESJOB2 (Time=2)	JESJOB3 (Time=1)	JESJOB4 (Time=5)		JESTAPE (Time=5)	WLMTAPE (Time=3)
WLM	WLMJOB3 (Time=1)	WLMJOB2 (Time=2)	WLMJOB1 (Time=4)	WLMJOB4 (Time=5)			
GMS Select (after allocation complete)							
JES	JESJOB1 (Time=4)	JESJOB2 (Time=2)	JESJOB3 (Time=1)	JESJOB4 (Time=5)	JESTAPE (Time=3)		
WLM	WLMJOB3 (Time=1)	WLMJOB2 (Time=2)	WLMTAPE (Time=3)	WLMJOB1 (Time=4)	WLMJOB4 (Time=5)		

Workload Balancing

For JES3-managed initiators, some workload balancing exists but it is difficult to define and static in nature. JES3-managed initiators do not adapt to changing workloads and system conditions. There is not direct correlation between the externals and the performance goals that are desired.

The following initialization parameters can be used to perform workload balancing for JES3-managed initiators:

- JOBMIX on the SELECT initialization statement
- CHOICE on the SELECT initialization statement
- LSTOR on the SELECT initialization statement

- IORATE on the CLASS initialization statement
- LSTRR on the CLASS initialization statement

For WLM-managed initiators, JES3's workload balancing externals are ignored. WLM automatically makes adjustments based on performance goals, importance, and available capacity.

Class Limits

Class limits can be used for both JES3-managed and WLM-managed initiators to control the number of jobs in execution. Class limits are specified on the CLASS initialization statement and can be one of the following:

- TDEPTH to limit the number of jobs in this class that can run in the entire SYSPLEX (JESPLEX).
- MDEPTH to limit the number of jobs in this class that can run on a particular system.
- MLIMIT to limit the number of jobs in this class that can run on a particular system based on the number of jobs running on that system in another class.

Priority aging is performed for jobs in JES-managed job class groups. Aging causes a job's priority to be increased when an unsuccessful attempt is made to allocate the job's resource requirements or to select the job. The installation specifies priority aging using the SAGER/SAGEL and MAGER/MAGEL parameters on the SELECT initialization statement. SAGER and SAGEL are used for jobs waiting to have their resource allocated. MAGER and MAGER are used for jobs waiting to be selected for execution.

For WLM-managed job class groups, job selection related priority aging is not performed. Jobs in WLM-managed job class groups are queued for selection by their main service arrival time and not their priority. Priority, by itself, is not sufficient for job selection because it is only one of the many criteria used to classify a job. If priority is important to your installation, then it should be reflected in the WLM classification rules and the service classes assigned to jobs.

For WLM-managed job class groups, setup-related priority aging (SAGER and SAGEL) is still performed because jobs are setup and queued by priority; priority is still a factor in allocating a job's resources.

Queue Delay

If a velocity goal is defined for a service class that is used for batch jobs, queue delay (i.e. the amount of time the job spends waiting for an initiator) is not included in the velocity calculation for JES3-managed initiators.

For WLM-managed initiators, queue delay is included in the velocity calculations. If a service class with a velocity goal is not meeting its goals, and queue delay is a significant reason for not meeting the goal, WLM can start more initiators for the service class. See *z/OS MVS Planning: Workload Management* for more information on velocity goals.

Resource Use

JES3-managed initiators are started by JES3 and take up JES3 job numbers and spool space.

WLM-managed initiators are started by WLM and do not take up JES3 job numbers and spool space.

Modes of Operation

For JES3-managed initiators, WLM can run in either compatibility mode or goal mode.

For WLM-managed initiators, only goal mode is allowed on the JES3 global and any JES3 local system where you want WLM-managed initiators to be started. If any JES3 local is in WLM compatibility mode, then WLM-managed initiators cannot be started on that system.

When you activate WLM initiator management, JES3 does not check to see if WLM is in goal mode or not. Also, JES3 inquiry commands do not show that WLM is in compatibility mode. Your only indication of WLM running in compatibility mode will be that no initiators will be started on the systems that are in compatibility mode.

Release Levels

JES3 can be at any level to run JES3-managed initiators.

JES3 must be at least at the OS/390 Version 2 Release 8 level on the global and any local where you want WLM-managed initiators started. If a particular local is not at the JES3 OS/390 Version 2 Release 8 level, WLM-managed initiators cannot be started on that system.

Classifying Jobs

WLM classification is responsible for assigning a service class and optionally a report class to a job based on the classification rules in the WLM policy. A service class is a group of work which has the same performance goals, resource requirements, or business importance. For workload management, you assign a service goal and optionally a resource group to a service class. WLM starts initiators on a service class basis.

A report class is a group of work for which reporting information is collected separately. For example, you can have a report class for information combining two different service classes, or a report class for information on a single transaction. A report class is not used by WLM initiator management.

Prior to OS/390 JES3 Version 2 Release 8, jobs were classified during execution. With OS/390 JES3 Version 2 Release 8, jobs are classified at the end of C/I processing after user exit IATUX09 has been called. Classification is performed for all batch jobs, not just for jobs in WLM-managed groups.

When a job is classified, the following information is provided to you so that you can classify jobs:

- Job name (called transaction name in the classification rules)
- Job class (called transaction class in the classification rules)
- PERFORM parameter from the JOB statement. (The PERFORM parameter on the EXEC statement is not used and cannot be used to assign a new service class when a new step begins to execute.)
- Owner (RACF user id)
- Priority
- Accounting information
- Subsystem type - JES
- Subsystem name - JES3 subsystem name (typically "JES3")

See *z/OS MVS Planning: Workload Management* for more information on how to setup classification rules and service classes.

After a job is classified, it may be reclassified:

- When a job's priority is changed as a result of a *MODIFY,J=job,P=xx command or due to deadline scheduling. Because a job's priority is one of the inputs to classification, the job's service class may change when the priority is changed.
- When a job's class is changed with a *MODIFY,J=job,CLASS=class command. Because a job's class is one of the inputs to classification, the job's service class may change when the class is changed.
- When a new WLM service definition is installed and activated. The classification rules can be changed and service classes can be added or deleted only by installing and activating a new WLM service definition. When this occurs, jobs are reclassified to pick up the latest changes to the WLM service definition.

A job is not reclassified when its priority is changed through the use of the SAGER or MAGER parameters on the SELECT initialization statement. Unlike deadline scheduling, priority aging only updates a job's priority temporarily for the purposes of moving the job up in the queue ahead of other jobs. Once a job has completed setup or has been selected to run, the priority is restored to its original value.

Pre-Execution Delays

When a job enters the system, there are a number of circumstances that can affect when a job is selected to run. For example, the job could be placed into operator hold, the job's resources may not be available, the job's class and group may not be enabled, or all initiators may be in use.

For jobs in WLM-managed groups, WLM has control over only the amount of time a job waits for an initiator (which is known as queue delay). If a service class is not meeting its goals and queue delay is a significant part of why the goals are not being met, WLM can help the service class by adding more initiators. WLM cannot control how long a job is in operator hold, how long it takes to allocate resources, and so forth. These forms of delay are included in WLM's calculations by JES3. JES3 is responsible for keeping track of the types of delays and reporting them to WLM.

Delays fall into the following categories. Most of these delays are tracked by JES3 and appear in RMF reports and SMF reports.

User Delays

These delays are self-inflicted by the user. For example, if the user submits a job with TYPRUN=HOLD specified on the JOB statement or HOLD=YES on the /*MAIN statement, the job will not be scheduled for C/I until it is released. User delay is not tracked by JES3.

Conversion Delay

This is the amount of time it takes to complete Converter/Interpreter processing in JES3. This time includes the amount of time waiting to be scheduled for C/I, the amount of time (for DJC jobs) spent waiting for a predecessor job to complete, as well as the actual time spent in C/I processing. Conversion delay starts being tracked as soon as the job is added to the JES3 job queue during input service; the exception to this is for TYPRUN=HOLD or HOLD=YES jobs. In these cases, conversion delay begins the moment the job is released from hold.

Operational Delay

Delay that is counted after the job completes C/I processing and before the job is selected to run. Examples are as follows:

- Operator hold where the operator issued a *MODIFY,J=job,H command to hold the job.
- Priority hold where the operator issued a *MODIFY,P=prty,H command to hold a priority.
- Class/group unavailable where the class or group is disabled on one or more systems (for example, as a result of a *MODIFY,G command).
- System unavailable where the system is not connected to the JES3 global or is varied offline.

Note: In RMF reports and SMF records, operational delays are combined into what is called Ineligible Delay.

JES Scheduling Delay

These delays are caused by JES3 or by installation-defined limits or algorithms. These delays include duplicate job name delay and the amount of time waiting because the TDEPTH, MDEPTH, TLIMIT, or MLIMIT parameters on the CLASS initialization statement have been exceeded.

Note: In RMF reports and SMF records, JES scheduling delays are combined into what is called Ineligible Delay.

Resource Delay

Resource delays are caused by unavailable resource or waiting for resources to be allocated. For example, most of the time spent in MDS processing (for example, fetch, system select, and allocate) is tracked as resource delay. The amount of time spent waiting for a scheduling environment to become available is also tracked as resource delay.

Queue Delay

Queue delay is the amount of time spent waiting for an initiator. That is, it is the time specifically associated with the job waiting for the availability of an initiator to the job can run. Queue delay is not tracked by JES3 but is computed by WLM by using the amount of time since the job completed C/I processing (main service arrival time) and subtracting out the operational, JES scheduling, and resource delays.

The following table shows the different phases of JES3 processing and the delays that can occur.

Table 22. Types of Job Delays

JES3 Job State	Job Condition	Type of Delay
C/I	TYPRUN=HOLD, HOLD=YES	User Delay
	Priority Hold	Conversion Delay
	Job Hold	Conversion Delay
	Spool Hold	Conversion Delay
	DJC Hold for predecessor job completion	Conversion Delay
	No DSP Available	Conversion Delay
	Locate Main Not Available	Conversion Delay
	PROCLIB Not Available	Conversion Delay
	SMS Catalog Not Available	Conversion Delay

Table 22. Types of Job Delays (continued)

JES3 Job State	Job Condition	Type of Delay
Main-Job cannot be scheduled for main service	Priority Hold Job Hold Spool Hold No DSP Available Main/Class/Group Unavailable Duplicate Job Name	Operational Delay Operational Delay Operational Delay Operational Delay Operational Delay JES Scheduling Delay
Main-Job is in setup (MDS) Processing	Fetch Volume Wait System Select Allocate - Resources Unavailable - SCHENV Unavailable/Undefined - Device Fence Unavailable - Class/Group Disabled - System Unavailable - Job/DJC Hold - Class SDEPTH Exceeded - Main SDEPTH Exceeded Verify System Verify Volume Unavailable Error	Resource Delay Operational Delay Resource Delay Resource Delay Resource Delay Resource Delay Operational Delay Operational Delay Operational Delay Operational Delay JES Scheduling Delay JES Scheduling Delay Resource Delay Resource Delay Resource Delay Operational Delay
Main-Job is Waiting to be Selected for Execution	Job/DJC Hold Class/Group Disabled System Unavailable Spool Space Unavailable TDEPTH/MDEPTH/TLIMIT/MLIMIT Exceeded SCHENV Unavailable/Undefined Waiting for an Initiator	Operational Delay Operational Delay Operational Delay JES Scheduling Delay JES Scheduling Delay Resource Delay Queue Delay

Figure 15 shows an overview of pre-execution job delays.

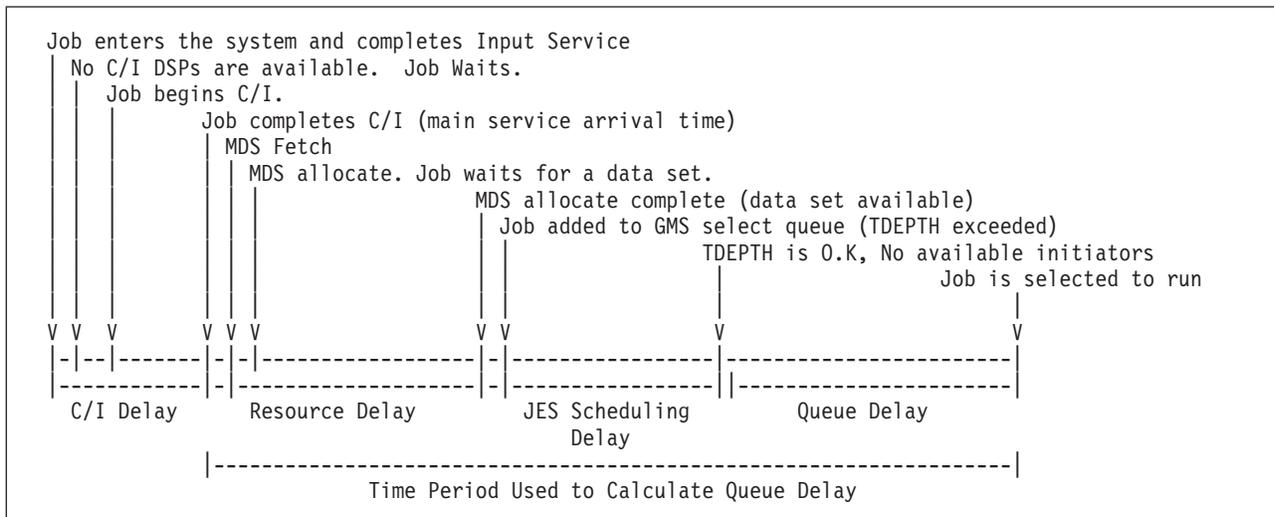


Figure 15. Overview of Pre-Execution Job Delays

Defining Service Classes and Performance Goals

A service class is a group of work with the same performance goals, resource requirements, or business importance. When you define a service class to WLM, you also assign performance goals to the service class.

Performance goals can be one of the following.

Discretionary

Work is run when the system resources are available.

Response Time Goal

A response time goal can be either an average response time or a response time and a percentile. Prior to OS/390 Version 2 Release 8, response time goals included the time the job entered the system until the time the job completed execution, specified TYPRUN=HOLD on the JOB statement or had to wait for a DJC predecessor job to complete before it could complete C/I processing. That time was included in the response time for the job. Since the amount of time a job is delayed is unpredictable in these cases, the installation often uses velocity goals and not response time goals.

In OS/390 Version 2 Release 8, response time goals are measured from the time the job completes C/I processing (main service arrival time) until it completes execution. Conversion delay is still tracked by JES3, but it is not included as part of the goal.

TYPRUN=HOLD time is not included in either the conversion delay or in the response time calculation. If you are currently using velocity goals for batch work, because TYPRUN=HOLD and conversion delay were included in the definition of response time, you may want to reevaluate this and see if a response time goal is more appropriate.

Velocity Goal

Velocity is a measure of how fast work should run when ready (without being delayed for WLM-managed resources). For jobs in JES-managed job class groups, the definition of velocity is unchanged. For jobs in WLM-managed job class groups, velocity includes queue delay (the amount of time waiting for an initiator). By including queue delay in the velocity calculation, WLM can determine whether adding another initiator will help a service class meet its performance goals. If you have velocity goals defined for batch service classes, the achieved velocity will decrease as a result of queue delay being included in velocity calculations. You may need to adjust your velocity goals.

Figure 16 on page 121 illustrates the differences between OS/390 Version 2 Release 8 and older releases with respect to response time/velocity goals.

* Input Service (Reader)		<*****		
- User Delay			*	
			*	
			*	
* Converter/Interpreter		Response	*	
- Conversion delay		Time	*	
		Pre-R8	*	
			*	
* Main Service			*	<*****
- Operational delay (job hold, class disabled)			*	
- JES scheduling delay (class limits, duplicate job name)			*	Response *
- Resource delay (resource affinity, JES3 setup)			*	Time *
- Queue delay (waiting for an initiator)			*	R8 *
		<*****	*	
			*	
			*	Velocity *
			*	R8 *
* Initiation	<*****		*	*
	Velocity *		*	*
	Pre-R8 *		*	*
* Execution	<*****	<*****	<*****	<*****
* Output Service				
* Purge				
Pre-R8 - indicates JES3 release prior to OS/390 Version 2 Release 8 or JES3 for OS/390 Version 2 Release 8 with JES-managed initiators.				
R8 - indicates JES3 for OS/390 Version 2 Release 8 with WLM-managed initiators.				

Figure 16. Response Time/Velocity Goals

Recommendations For Defining Service Classes

The following are general recommendations you should follow when defining service classes.

- Service classes should not include jobs in JES-managed and WLM-managed job class group. Doing so weakens the relationship between the number of WLM-managed initiators and the queue delay that is observed. For example, if a velocity goal is used, queue delay is included in the velocity calculations for WLM-managed initiators. If a service class contains jobs in JES-managed and WLM-managed job class groups, queue delay will be included in the velocity calculations for some jobs but not all of the jobs in the service class.
- Service classes should be unique across a JESPLEX. If you run multiple JESPLEXS within a SYSPLEX, keep in mind that WLM policies are SYSPLEX-wide and WLM manages the performance goals for a service class at a SYSPLEX level (for example, performance goals based on the SYSPLEX performance index). If a service class is shared between two JESPLEXS, it is possible for a service class to be meeting its goals from a SYSPLEX, but not meeting its goals in a particular JESPLEX within a SYSPLEX.
- If class limits are specified on the Class initialization statement, jobs in that class should map to a single service class. If jobs in the class map to multiple service classes, more initiators may be started than necessary because each service class will be treated independently with respect to the class limits. When JES3 reports to WLM the number of eligible jobs in the service class, it will apply the class limits to each service class individually.

Job Selection Algorithm for JES3-Managed Initiators

When a job in a JES-managed job class group 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 a job arrives from the JES-managed initiator, the job is ready to be selected for execution. Jobs that are requested by JES-managed initiators are still queued by priority, but jobs that are requested by WLM-managed initiators are queued by age to an appropriate service class queue.

JES-managed initiator job selection is driven by how you define job processing in the GROUP, CLASS, and SELECT initialization statements. All jobs classes are defined to JES3 by using the CLASS initialization statement. Each such class is associated with a job class group that is defined using the GROUP initialization statement. The GROUP initialization statement specifies the number of initiators to allocate on each system as well as how the initiators should be allocated and deallocated. A job class group can have one or more job classes associated with it.

After an initiator starts, it asks the JES3 global processor for a job. JES3 then examines the GMS select queue associated with the initiators job class group to see if there are jobs that can be selected to run. After a job is selected, JES3 reads the job's control blocks and sends the initiator all the information it needs to setup the job for execution. When the job completes execution, the initiator returns the job to JES3. JES3 then updates the job's control blocks to show that the job has completed execution.

For jobs requiring JES-managed initiators, JES3 controls the initiator selection process by first determining 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 needs to be increased the most.

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 the job has a scheduling environment (that is, the SCHENV= keyword was specified on the JOB statement), the job selection algorithm checks if the scheduling environment is available. If it is not available, the scan checks for the JSPAN and BAR parameters on the GROUP initialization statement. If these are not exceeded, it moves to the next job candidate in this group and starts again.

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.

Job Selection Algorithm for WLM-Managed Initiators

When a job in a WLM-managed job class group has completed processing in MDS, the job is placed in the queue of jobs waiting to be selected for execution by service class. This queue is ordered by main service arrival time (i.e. when the job completed C/I processing). Unlike jobs in JES-managed groups, priority is not used to order the jobs on the queue, since priority is one of the many criteria that can be used to assign a service class for a job. Because jobs are ordered by main

service arrival time and not priority, the time the job completes setup processing does not determine its place in the queue.

When a request for work arrives from a WLM-managed initiator, the source of the request narrows the choice in two ways:

1. The service class identification of the initiator limits the choice to jobs with that service class.
2. The processor on which the initiator is started limits the choice to jobs that can run on that processor.

Note: Even though WLM management operates at a job class group level, WLM-managed initiators select work by service class and not by job class group like JES-managed initiators. Therefore, the queue of work for a particular service class can have jobs from different job class groups. Unlike job selection for JES-managed groups, the following job selection parameters are ignored for selecting jobs in WLM-managed groups.

- IORATE on the CLASS statement
- CHOICE on the SELECT statement
- JOBMIX on the SELECT statement
- LSTOR on the SELECT statement
- JSPAN on the GROUP statement
- BAR on the GROUP statement

Note: Workload balancing parameters, such as CHOICE, JOBMIX, and LSTOR are ignored because workload balancing functions are left to WLM and SRM.

When a WLM-managed initiator requests a job, each job in the service class queue is examined until a job is found that is eligible to be selected for execution or until no jobs are left to run. JSPAN and BAR are not used to limit the number of jobs examined. The following job characteristics are examined when selecting a job:

- Is job selection suspended because a WLM service definition change has occurred, and JES3 is busy reclassifying jobs?
- Can the job run on the system on which the initiator is started?
- Is the system on which the initiator is started, connected and online?
- Is the job's group and class enabled on the system on which the initiator is started? Is the job held as a result of an operator command or because it is part of a DJC network?
- Is there available spool space in the spool partition that is assigned to the job? The spool partition that is examined is one of the following:
 - The spool partition specified by the SPART parameter
 - The spool partition assigned to the job class using the SPART parameter on the CLASS initialization statement
 - The spool partition assigned to the system using the SPART parameter on the MAINPROC initialization statement
 - The default spool partition
- Are any of the class limits exceeded as defined using the TDEPTH, MDEPTH, TLIMIT, and MLIMIT parameters on the CLASS initialization statement?
- If the job has a scheduling environment, is the scheduling environment available on the system on which the initiator is started?

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.

Note: As a result of deadline scheduling, a job will be reclassified when its priority is changed. Depending on the priority that is used in your installation's classification rules in the WLM policy, this may or may not cause a new service class to be assigned to the job. Changing the priority as a result of deadline scheduling does not affect when a job in a WLM-managed group is selected to run. Jobs waiting to be selected for execution by an WLM initiator are ordered by main service arrival time and not priority.

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.) Jobs belonging to a DJC network cannot be registered with automatic restart management.

For a complete description of the `//*NET` and `//*PROCESS` control statements, see *z/OS MVS 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".

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 *z/OS 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 *z/OS MVS JCL User's Guide*.

Controlling Job Selection

The JES3 initialization stream can be used by the system programmer to tailor JES3 job selection process 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` parameter of the `//*MAIN` control statement and the job class is analyzed.
2. JES3 MDS selects mains by location of non-shared 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 a job.
4. The availability of the job's scheduling environment can restrict the mains that are eligible to run a job.

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. `IORATE` is ignored during job selection for jobs in WLM-managed job class groups.

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 and I/O devices) as well as defining what type of initiator should process jobs in the group (JES-managed initiators or WLM-managed initiators). 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 17 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 non-shared 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.

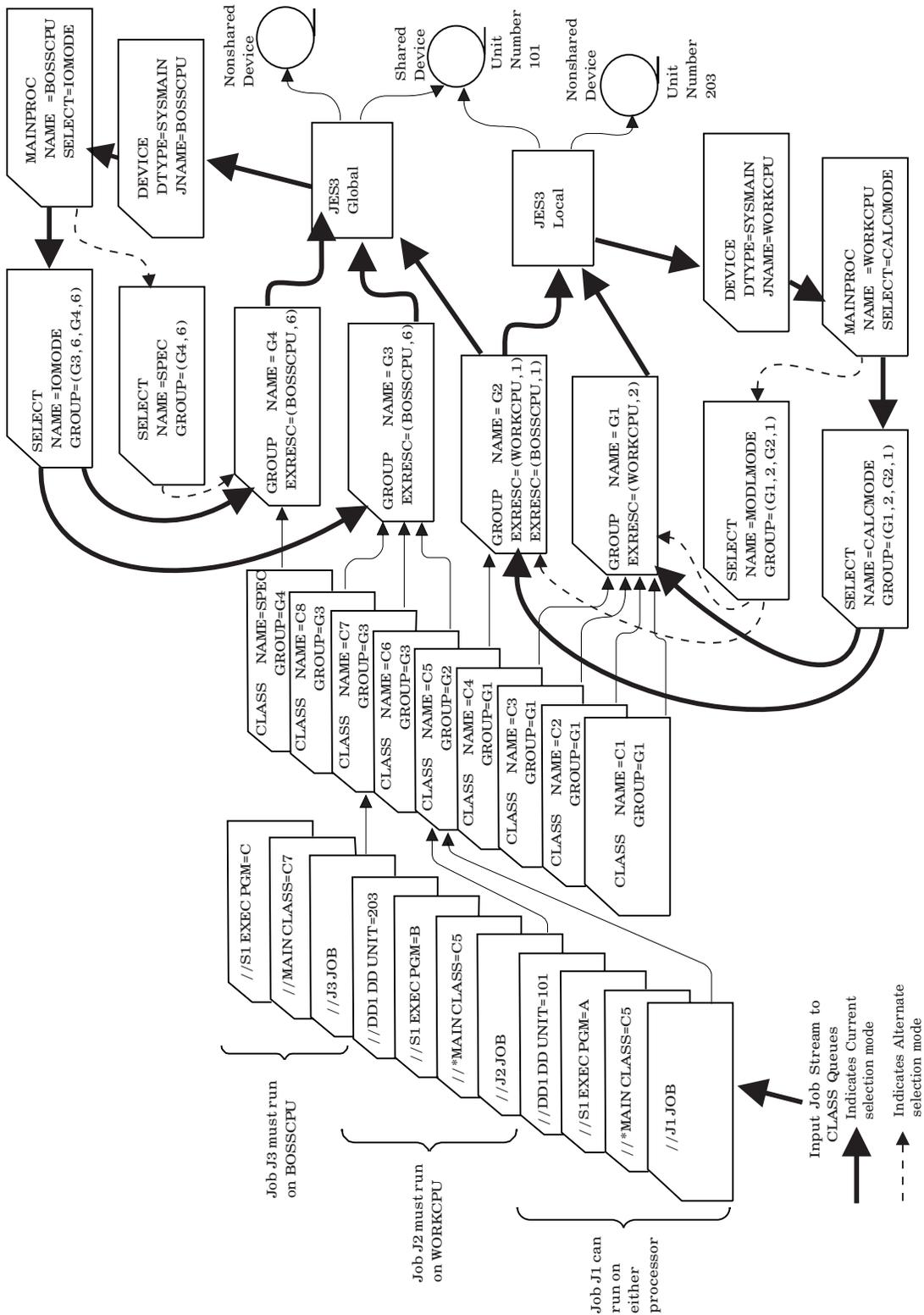


Figure 17. JES3 Job Selection Environment

Default Job Selection

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

JES initiator management is then used, and jobs are selected by priority only, in first-in-first-out order. In this case, two JES-managed 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. If you want WLM initiator management to be used for the default group, you can switch from JES to WLM initiator management with a *MODIFY command, or a GROUP initialization statement for JS3BATCH can be specified with MODE=WLM.

For JES-managed initiators, 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 only applies to JES-managed initiators (MODE=JES). These options control the conditions under which JES-managed initiators are started and stopped. For WLM-managed initiators, the starting and stopping of initiators is under control of WLM, and is based on information from the WLM policy, available system capacity, and information from JES3 related to the backlog of jobs. Initiator allocation and unallocation options can be defined for WLM-managed job class groups; however, these options will be ignored.

Demand or dynamic initiator allocation occurs when jobs are available for execution and when the job class group they are in becomes enabled. (Dynamic allocation is used to allocate JES-managed initiators when the first job of the group in the queue is eligible for main scheduling; demand allocation is used to allocate JES-managed initiators as they are needed.) Demand or dynamic deallocation for JES-managed initiators 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.

The overhead of starting and stopping initiators can also be avoided by using WLM-managed initiators (MODE=WLM specified on the GROUP initialization statement). When a new initiator needs to be assigned to a service class, a new initiator address space is not necessarily started. Instead, WLM maintains a pool of initiator address spaces and assigns them dynamically to the service classes when needed. Only if there are no available initiators is a new initiator address space started. When WLM decides that there are too many initiators assigned to a service

class, it does not stop the initiator address space; instead, it returns the initiator to the pool of idle initiator address spaces. After a period of time, the idle initiator address spaces will be stopped.

Manual or dynamic allocation and deallocation options are preferable in an JES initiator managed environment, where both of the following are true:

- Work is available for processing on a predefined schedule in a specific window; for example, work is available for processing at 4 PM to be completed by 6 AM the following morning.
- Work in the group is available in sufficient quantities to prevent unnecessary starting and stopping of initiators.

For work that arrives sporadically, DEMAND allocation and deallocation options are preferable.

Defining Logical Storage for Processors

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

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

Note: LSTOR parameter on the SELECT initialization statement is ignored in job selection for jobs that are in WLM-managed job class groups. For jobs in JES-managed job class groups the LSTOR parameters is useful in improving job selection efficiency.

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.

Attention: 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 central storage utilization can lead to one of the following operating conditions:

- Central storage is underutilized, and throughput is limited regardless of any other tuning.
- Central storage is fully utilized, and paging is optimum. This condition allows maximum throughput.
- Central 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 central 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 central storage utilization while the actual central 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 installation 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.

For jobs in JES-managed job class groups (MODE=WLM specified on the GROUP initialization statement), the CHOICE and JOBMIX parameters 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. For jobs in WLM-managed job class groups, JOBMIX and CHOICE are ignored for purposes of job selection.

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. (For jobs in WLM-managed job class groups, MAGER is ignored. Jobs destined for WLM-managed initiators are ordered on the queue by main service arrival time.)
- *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.(For jobs in WLM-managed job class groups, MAGEL is ignored. Jobs destined for WLM-managed initiators are ordered on the queue by main service arrival time.)
- *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. (SAGER is also used for jobs in WLM-managed job class groups because these jobs are in MDS allocation where jobs are ordered by priority.)
- *SAGEL*: Specifies an aging priority limit (0-15) beyond which a job cannot be aged during job setup. (SAGEL is also used for jobs in WLM-managed job class groups because these jobs are in MDS allocation where jobs are ordered by priority.)

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 queuing 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 *SDEPTH*s. 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.

Note: For jobs in WLM-managed job class groups, the *INCR* and *INCL* parameters will not expedite job selection once a job has been set up. Jobs in WLM-managed job class groups are ordered by main service arrival time, not by priority when waiting to be selected to run.

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 (or by using the **MODIFY,C=class* command to change the parameters):

- *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.

Notes:

1. TDEPTH, TLIMIT, MDEPTH, and MLIMIT are used for WLM-managed job class groups as well as JES-managed job class groups. However, keep in mind that JES3 applies the class limits to each service class separately during sampling. If you define class limits for a job class, and the jobs in the class are assigned to more than one service class, more initiators may be started than can actually run jobs. When class limits are used, it is recommended that all jobs in the class be assigned the same service class.
2. For started tasks, all CLASS initialization statement parameters are ignored except the following:
 - JESMSG= to allow or suppress writing to JESMSGGLG data sets for the started task.
 - SPIN= to allow or suppress the spin off of the JESMSGGLG and JESYSMSG data sets for a long running job.
 - Job class group name (GROUP parameter)
 - Device fencing (EXRESC and DEVPOOL parameters on the GROUP initialization statement)
 - Spool partitioning (SPART parameter)
 - Track group allocation (TRKGRPS parameter)

Grouping JES3 Job Classes

The GROUP initialization statement defines the resources available to a group of JES3 job classes. A maximum of 255 groups can be defined. The system programmer specifies whether jobs in the group will run under JES-managed initiators or WLM-managed initiators as specified on the MODE= parameter.

The EXRESC parameter defines the execution resources (initiators and devices) dedicated to the group, and also defines the systems where jobs in the group will run. 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. If the execution resource requirements are the same for all processors, the EXRESC parameter can be specified once on the GROUP statement with a processor name of *ALL.

The EXRESC parameter also defines the initiator allocation and deallocation options (see "Starting and Stopping Initiators" on page 131). For WLM-managed job class groups, the initiator related options (that is, the number of initiators and allocation and deallocation options) can be specified, but they will be ignored. If the job class group is later modified so that JES initiator management is used, the initiator related options will take effect.

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=PTY 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.

Note: A class can be assigned to a new job class group with the *MODIFY,C=*cls*,GROUP=*grp* command. This is useful if you have multiple job classes assigned to a group and you want to test WLM initiator management for one of the classes in the group. You can use the *MODIFY,C=*cls* command to switch one of the job classes to a WLM-managed job class group without changing initiator management for the other classes in the group.

Automatic Restart Management

Automatic restart management is an MVS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, or the system on which it is running fails, automatic restart management can restart the job or task without operator intervention. To be eligible for restart under automatic restart management, a batch job or started task must register with automatic restart management.

See *z/OS MVS Setting Up a Sysplex* for information about setting up and using the automatic restart manager.

The following are some JES3-specific items to consider when using automatic restart management:

- When a job or system fails and the job is restarted by automatic restart management, all non-spin sysout data sets created during the previous execution are deleted.

If you desire to make a sysout data set available for JES3 output service processing when the data set is closed, use the FREE=CLOSE JCL option (or the corresponding dynamic allocation option). Automatic restart management will not delete such data sets when the job is restarted by automatic restart management.

If you desire to retain a sysout data set that was still open and not yet closed at the time of the failure, use the SPIN=UNALLOC JCL option (or the corresponding dynamic allocation option). Using this option causes the data set to be unallocated and made available to JES3 at job termination time (that is, when the job or system initially fails and the job is restarted).

Table 23 on page 138 summarizes the use of the FREE=CLOSE and SPIN=UNALLOC options and the effect they have on sysout data created by automatic restart management-registered jobs:

Table 23. FREE=CLOSE and SPIN=UNALLOC Effect on SYSOUT Created by Automatic Restart Management

Options Used	SYSOUT Data Set Status at Time of Automatic Restart Management	JES3 Processing
FREE=CLOSE	Closed	The data set is unallocated and spun-off when closed by the job.
FREE=CLOSE	Open	The data set is deleted by JES3.
SPIN=UNALLOC	Closed	The data set is unallocated and spun-off at job end.
SPIN=UNALLOC	Open	The data set is unallocated and spun-off at job end.
FREE=CLOSE and SPIN=UNALLOC	Closed	The data set is unallocated and spun-off when closed by the job.
FREE=CLOSE and SPIN=UNALLOC	Open	The data set is unallocated and spun-off at job end.

Note: Sysout data sets created and maintained by JES3 (for example, JESMSG LG, JESYSMSG, JESJCL) are not deleted when a job is restarted by automatic restart management.

- You cannot use the JES3 Dump Job (DJ) facility to dump jobs that are waiting for restart by automatic restart management.
- You cannot register jobs that are a part of a dependent job control (DJC) network with automatic restart management.
- Failure options specified on the following parameters are ignored for jobs registered with automatic restart management.
 - FAILURE keyword on the CLASS initialization statement
 - FAILURE keyword on the STANDARDS initialization statement
 - FAILURE keyword on the /*MAIN JECL statement

If a job is registered with automatic restart management at the time of a system failure, automatic restart management determines whether the job is restarted regardless of the value specified on the FAILURE keyword.

- Any job registered with automatic restart management can be restarted only within a single JES3 complex. IBM recommends defining only one JES3 complex within a sysplex.

Output Service

Output service executes on the global and processes SYSOUT data sets destined for:

- JES3 managed devices
- PSF managed devices
- system application printer interface (SAPI) applications
- NJE (BSC and SNA)
- Process SYSOUT (PSO)

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 system output streams. MVS/BDT sends these job or system output streams to the proper node within a SNA/NJE network. You can use JES3 operator commands to hold, release, or cancel networking requests from the queue.
- *Output service writer queue (Q=WTR)*: This queue contains data sets waiting for output processing by JES3 managed devices. Output service automatically selects data sets for processing based on their selection characteristics such as output class, output priority, and output device-related requirements. You can use JES3 commands to place these data sets in operator-hold status. You can also use JES3 commands to modify a data set's selection characteristics or move the data set to the HOLD queue.

Note: Process SYSOUT and system application printer interface can also process work on the output service writer (WTR) queue.

- *Output service hold queue (Q=HOLD)*: This queue contains data sets that are awaiting output processing by other than JES3-managed devices. These data sets must be processed by the function for which they are held (system application printer interface (SAPI) application, external writer or TSO). The function that processes the data set can then change data set characteristics, 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 or issue a modify (*F) command to move the data set to the WTR for JES3 device processing.

You can use the JES3 *INQUIRY U and *MODIFY U commands to inquire about, modify, or delete data sets on the writer, hold, or BDT queue. For information about using these JES3 commands, see *z/OS JES3 Commands*.

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

installation exit routine IATUX34 for the SYSOUT DD statement and installation 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 installation exit routine IATUX33.)
- The SYSOUT class table (defined by SYSOUT initialization statements)

Note: The installation can change the characteristics in the OSE by coding the installation exit routine, IATUX19.

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, default and direct `//*FORMAT` statements are applied to the SYSOUT characteristics of the SYSOUT on the writer (WTR) 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. You override the OUTSERV initialization statement parameters with appropriate parameters on JCL or through particular DEVICE initialization statement parameters for those devices producing output. See *z/OS JES3 Initialization and Tuning Reference* for detailed descriptions of the OUTSERV and DEVICE initialization statements and their relationship to the JCL of jobs.

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 circumstances (explained below), JES3 creates separate OSEs from the two types of statements.

Table 24 shows the JES3 defined defaults that can be overridden.

Table 24. JES3 DEFINED DEFAULTS - Overriden by OUTSERV Initialization Statement

DEFAULT	PRINT	PUNCH
Carriage	6	0
CDSTOCK	N/A	5081
CHARS	GS10	0
FLASH	NONE,255	0,0
FORMS	1PRT	1PRT
MODIFY	NONE,'0'	0,'0'
STACKER	C	0
TRAIN	PN	PN (hard coded)
Notes:		
1. PROCESS MODE, which cannot be set in the INISH deck is set to 'LINE' for PRINT and to '0' for punch.		
2. '0' indicates the character 0 (hex 'F0')		

Override Sequence with “Direct” OUTPUT JCL Statements for WTR Queue and HOLD Queue: A “direct” OUTPUT JCL statement is one that is 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 (above).

EXAMPLE

This example shows the override process using the FORMS parameter on the OUTSERV initialization statement and “direct” OUTPUT JCL statement.

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT      ----> For WTR Q
SYSOUT,CLASS=F,FORMS,3PRT,HOLD=EXWTR ----> For HOLD Q
```

Under these conditions, the following job is run:

```
//MARIOA      JOB      MSGLEVEL=(1,1),MSGCLASS=F
//OUT1        OUTPUT   CHARS=GS14,FORMS=2PRT,DEFAULT=NO
//STEP1       EXEC     PGM=IEBDG
//SYSPRINT    DD       SYSOUT=(F,,4PRT),OUTPUT=*.OUT1
//SYSUT2      DD       SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80),
//             OUTPUT=*.OUT
//SYSIN       DD       *
                DSD OUTPUT=(SYSUT2)
                FD NAME=FIELD1,LENGTH=80,STARTLOC=1,PICTURE=5,'MARIO',FILL='x'40'
                CREATE QUANTITY=10,NAME=FIELD1
                END
/*
```

Table 25 illustrates the override process for this example.

Table 25. Output Parameter Overrides Using a Direct OUTPUT JCL Statement for WTR and HOLD Queue

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. OUTPUT FORMS=2PRT	3. DD for SYSPRINT		
SYSPRINT	1PRT	3PRT	2PRT	4PRT	4PRT	GS14
SYSUT1	1PRT	3PRT	2PRT	n/a	2PRT	GS14
JESJCL	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESMSGLG	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESYSMSG	1PRT	3PRT	n/a	n/a	3PRT	GS10

JES3 builds one OSE for each direct OUTPUT JCL statement referring to a data set. A copy of the output data set for each OUTPUT statement is made available for processing. 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 For WTR

Queue: 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. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement.
2. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 (above).
3. *`//*FORMAT xx,DDNAME=xxxx` JES3 control statement:* The values specified on the direct `//*FORMAT` statement override the values specified by sources 1 and 2 (above).

EXAMPLE

This example shows the override process using the FORMS parameter on the OUTSERV initialization statement and a direct `//*FORMAT` JES3 control statement.

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT ----->WTR Queue
```

Under these conditions, the following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
//*FORMAT     PR,DDNAME=SYSPRINT,FORMS=2PRT,CHARS=GS14
//STEP1       EXEC  PGM=IEBDG
//SYSPRINT    DD  SYSOUT=(F,,4PRT)
//SYSUT2      DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN       DD   *
DSD OUTPUT=(SYSUT2)
FD NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
CREATE QUANTITY=10,NAME=FIELD1
END
/*
```

Table 26 illustrates the override process for this example.

Table 26. Example of FORMS Override Using Direct `//*FORMAT` Statements for WTR Queue

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. DD for SYSPRINT	3. <code>//*FORMAT PR, DDNAME = SYSPRINT</code>		
SYSPRINT	1PRT	3PRT	4PRT	2PRT	2PRT	GS14
SYSUT2	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESMSGLG	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESJCL	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESYSMSG	1PRT	3PRT	n/a	n/a	3PRT	GS10

Override Sequence with Direct `/*FORMAT JES3 Control Statements for HOLD Queue SYSOUT:` 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, it is not used for SYSOUT on the HOLD queue.

1. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement.
2. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 (above). This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.
3. */*FORMAT xx,DDNAME=xxxx JES3 control statement:* The values specified here are not applied to the HOLD queue.

EXAMPLE

This example shows the override process using the FORMS parameter on the OUTSERV initialization statement and a direct `/*FORMAT JES3` control statement.

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT,HOLDE=EXTWTR ----->HOLD Queue
```

The following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
/*FORMAT      PR,DDNAME=SYSPRINT,FORMS=2PRT,CHARS=GS14
//STEP1       EXEC  PGM=IEBDG
//SYSPRINT    DD  SYSOUT=(F,4PRT)
//SYSUT2      DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN       DD   *
DSD OUTPUT=(SYSUT2)
FD NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
CREATE QUANTITY=10,NAME=FIELD1
END
/*
```

Table 27 illustrates the FORMS override process for this example.

Table 27. Example of Output Parameter Overrides Using Direct `/*FORMAT` Statements for HOLD Q

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. DD for SYSPRINT	3. /*FORMAT PR, DDNAME = SYSPRINT		
SYSPRINT	1PRT	3PRT	4PRT	n/a	4PRT	GS10
SYSUT2	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESMSGLG	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESJCL	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESYSMSG	1PRT	3PRT	n/a	n/a	3PRT	GS10

Note: If the SYSOUT were moved to the WTR Queue by issuing a `*MODIFY U,Q=HOLD,NQ=WTR,J=jobname/jobnumber` command, the SYSOUT characteristics would be the same as those in Table 26.

Override Sequence With Only Default OUTPUT JCL Statements for WTR Queue and HOLD Queue: 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 (above).
3. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 and 2 (above). 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, JESMSGLG, and JESYSMSG, unless explicitly specified. To learn how to use the OUTPUT JCL statement with system data sets, see *z/OS MVS JCL User’s Guide*.

EXAMPLE

This example shows this override process using the FORMS parameter on the OUTSERV initialization statement and a default OUTPUT JCL statement. The JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT          ----->WTR Queue
SYSOUT,CLASS=F,FORMS=3PRT,HOLD=EXTWTR ----->HOLD Queue
```

The following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
//OUT1        OUTPUT FORMS=2PRT,CHARS=GS14,DEFAULT=YES
//STEP1       EXEC   PGM=IEBDG
//SYSPRINT    DD  SYSOUT=(F,,4PRT)
//SYSUT2      DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN       DD    *
DSD OUTPUT=(SYSUT2)
FD NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
CREATE QUANTITY=10,NAME=FIELD1
END
/*
```

Table 28 illustrates the FORMS override process for this example.

Table 28. Example of Output Parameter Overrides Using Default OUTPUT Statements for WTR and HOLD Queue

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. OUTPUT DEFAULT=YES FORMS=2PRT	3. DD for SYSPRINT		
SYSPRINT	1PRT	3PRT	2PRT	4PRT	4PRT	GS14
SYSUT2	1PRT	3PRT	2PRT	n/a	2PRT	GS14
JESMSGLG	1PRT	3PRT	n/a	n/a	3PRT	GS10

Table 28. Example of Output Parameter Overrides Using Default OUTPUT Statements for WTR and HOLD Queue (continued)

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. OUTPUT DEFAULT=YES FORMS=2PRT	3. DD for SYSPRINT		
JESJCL	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESYSMSG	1PRT	3PRT	n/a	n/a	3PRT	GS10

Note: 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 for WTR Q: 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.
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 (above).
3. *SYSOUT DD JCL statements*: The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 and 2 (above).

EXAMPLE

The following example shows the override process using the FORMS parameter on the OUTSERV initialization statement and a default /*FORMAT JES3 control statement. The JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT ----->WTR Queue
```

The following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
//*FORMAT     PR,DDNAME=,FORMS=2PRT,CHARS=GS14
//STEP1      EXEC  PGM=IEBDG
//SYSPRINT    DD  SYSOUT=(F,,4PRT)
//SYSUT2     DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN      DD   *
DSD OUTPUT=(SYSUT2)
FD NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
CREATE QUANTITY=10,NAME=FIELD1
END
/*
```

Table 29 illustrates the override process for this example.

Table 29. Example of FORMS Overrides Using Default `//*FORMAT` Statements for WTR Q

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. <code>//*FORMAT PR, DDNAME=</code>	2. <code>SYSOUT, CLASS=F</code>	3. DD for <code>SYSPRINT</code>		
SYSPRINT	1PRT	2PRT	3PRT	4PRT	4PRT	GS14
SYSUT2	1PRT	2PRT	3PRT	n/a	3PRT	GS14
JESMSGLG	1PRT	2PRT	3PRT	n/a	3PRT	GS14
JESJCL	1PRT	2PRT	3PRT	n/a	3PRT	GS14
JESYSMSG	1PRT	2PRT	3PRT	n/a	3PRT	GS14

Override Sequence With Only Default `//*FORMAT` JES3 Control Statements for HOLD Queue: 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. *SYSOUT class table:* This table is constructed from the parameters on the SYSOUT initialization statement.
2. *SYSOUT DD JCL statements:* The values specified on the SYSOUT DD JCL statements override the values specified by sources 1 (above). This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.

Note: "Default" `//*FORMAT` JCL statements do not apply to SYSOUT on the HOLD Queue. SYSOUT characteristics are applied to data sets on the HOLD Queue when they are moved to the WTR Queue using a `*MODIFY U,NQ=WTR` command.

EXAMPLE

This example shows the override process using the FORMS parameter on the OUTSERV initialization statement and a default `//*FORMAT` JES3 control statement. The JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT,HOLD=EXTWTR ----->HOLD Queue
```

The following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
//*FORMAT     PR,DDNAME=,FORMS=2PRT,CHARS=GS14
//STEP1      EXEC  PGM=IEBDG
//SYSPRINT   DD  SYSOUT=(F,,4PRT)
//SYSUT2     DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN      DD   *
DSD OUTPUT=(SYSUT2)
FD NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
CREATE QUANTITY=10,NAME=FIELD1
END
/*
```

Table 30 illustrates the override process for this example.

Table 30. Example of Output Parameter Overrides Using Default `//*FORMAT` Statements for HOLD Queue

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides		FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. DD for SYSPRINT		
SYSPRINT	1PRT	3PRT	4PRT	4PRT	GS10
SYSUT2	1PRT	3PRT	n/a	3PRT	GS10
JESMSGLG	1PRT	3PRT	n/a	3PRT	GS10
JESJCL	1PRT	3PRT	n/a	3PRT	GS10
JESYSMSG	1PRT	3 PRT	n/a	3PRT	GS10

Note: After issuing the `*MODIFY U,NQ=WTR,Q=HOLD,J=jobname/jobnumber`, command for the above job, the SYSOUT characteristics would be the ones in Table 29.

Override Sequence with Default OUTPUT and `//*FORMAT` JES3 Control Statements for WTR Queue: If there is a default OUTPUT JCL statement and a default `//*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.
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 (above).
3. "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 and 2 (above).
4. *SYSOUT DD JCL statements*: The values specified on the SYSOUT DD JCL statements override the values specified by sources 1, 2, and 3 (above). This includes changes to the DD statements, such as changes made using TSO OUTPUT commands.

Notes:

1. "Default" OUTPUT JCL statements do not apply to system data sets, such as JESJCL, JESMSGLG, and JESYSMSG, unless explicitly specified. To learn how to use the OUTPUT JCL statement with system data sets, see *z/OS MVS JCL User's Guide*.
2. Unlike "Direct OUTPUT and `//*FORMAT` JCL statements, a separate OSE is not created for Default OUTPUT and `//*FORMAT` statements.

EXAMPLE

This example shows the override process using FORMS parameter on the OUTSERV initialization statement and a default OUTPUT and **//*FORMAT** JES3 control statement.

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT  ----->WTR Queue
```

The following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
//OUT1        OUTPUT FORMS=2PRT,CHARS=GS14,DEFAULT=YES
//*FORMAT     PR,DDNAME=,FORMS=6PRT,CHARS=GS12
//STEP1       EXEC  PGM=IEBDG
//SYSPRINT    DD  SYSOUT=(F,,4PRT)
//SYSUT2      DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN       DD  *
              DSD OUTPUT=(SYSUT2)
              FD  NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
              CREATE QUANTITY=10,NAME=FIELD1
              END
              /*
```

Table 31 illustrates the override process for this example.

Table 31. Example of FORMS Overrides Using Default OUTPUT and **//*FORMAT** Statements for WTR Queue

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides				FORMS Final Value	CHARS Final Value
		1. //*FORMAT PR, DDNAME=	2. SYSOUT, CLASS=F	3. OUTPUT DEFAULT=YES FORMS=2PRT	4. DD for SYSPRINT		
SYSPRINT	1PRT	6PRT	3PRT	2PRT	4PRT	4PRT	GS14
SYSUT2	1PRT	6PRT	3PRT	2PRT	n/a	2PRT	GS14
JESMSGLG	1PRT	6PRT	3PRT	n/a	n/a	3PRT	GS12
JESJCL	1PRT	6PRT	3PRT	n/a	n/a	3PRT	GS12
JESYSMSG	1PRT	6PRT	3PRT	n/a	n/a	3PRT	GS12

Override Sequence With Only Default OUTPUT and **//*FORMAT Statements for the HOLD Queue:** If there is a default OUTPUT JCL statement and a default **//*FORMAT** JCL statement, the **//*FORMAT** statement is **not** applied to the SYSOUT on the HOLD queue.

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

Notes:

1. "Default" OUTPUT JCL statements do not apply to system data sets, such as JESJCL, JESMSG LG, and JESYSMSG, unless explicitly specified. To learn how to use the OUTPUT JCL statement with system data sets, see z/OS MVS JCL User's Guide.
2. "Default" // *FORMAT JCL statements do not apply to SYSOUT on the HOLD Queue. SYSOUT characteristics are applied to data sets on the HOLD Queue when they are moved to the WTR Queue with a *MODIFY U,NQ=WTR command.

EXAMPLE

This example shows the override process using the FORMS parameter on the OUTSERV initialization statement and a "default" OUTPUT and // *FORMAT JES3 control statement.

The JES3 initialization statements include:

```
OUTSERV,FORMS=1PRT
SYSOUT,CLASS=F,FORMS=3PRT,HOLD=EXTWTR ----->HOLD Queue
```

The following job is run:

```
//MARIOA      JOB  MSGLEVEL=(1,1),MSGCLASS=F
//OUT1        OUTPUT FORMS=2PRT,CHARS=GS14,DEFAULT=YES
// *FORMAT    PR,DDNAME=,FORMS=6PRT,CHARS=GS12
//STEP1       EXEC  PGM=IEBDG
//SYSPRINT    DD  SYSOUT=(F,,4PRT)
//SYSUT2      DD  SYSOUT=F,DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN       DD   *
DSD OUTPUT=(SYSUT2)
FD NAME=FIELD1,LENGTH=80,STARLOC=1,PICTURE=5,'MARIO',FILL=X'40'
CREATE QUANTITY=10,NAME=FIELD1
END
/*
```

Table 32 illustrates the override process for this example.

Table 32. Example of Output Parameter Overrides Using Direct and Default // *FORMAT Statements for HOLD Q

Data Set	FORMS Initial Value in OUTSERV Statement	Order of FORMS Overrides			FORMS Final Value	CHARS Final Value
		1. SYSOUT, CLASS=F	2. OUTPUT DEFAULT=YES FORMS=2PRT	3. DD for SYSPRINT		
SYSPRINT	1PRT	3PRT	2PRT	4PRT	4PRT	GS14
SYSUT2	1PRT	3PRT	2PRT	n/a	2PRT	GS14
JESMSG LG	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESJCL	1PRT	3PRT	n/a	n/a	3PRT	GS10
JESYSMSG	1PRT	3PRT	n/a	n/a	3PRT	GS10

Note:

After issuing the *MODIFY U,NQ=WTR,Q=HOLD,J=jobname/jobnumber the SYSOUT characteristics would be the ones in Table 31.

OSE Processing: As each data set is selected, a temporary OSE is constructed containing all scheduling requirements. A call is then made to installation exit IATUX19 to allow the system programmer to examine and, if desired, to change the information before the OSE is spooled. For SYSOUT destined for a BSC/NJE or SNA/NJE destination, changes to characteristics must be made in IATUX39.

When control is returned from the installation 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 queuing 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 a SAPI application, an external writer, an output device or with an active writer that is waiting for more work.

Rules:

1. When a SYSOUT initialization statement uses an output descriptor (direct or default) that is defined through an OUTPUT JCL statement or a TSO OUTDES command, JES3 uses the values from that output descriptor, the SYSOUT initialization statement, and the OUTSERV initialization statement to establish the data set's output characteristics.

The characteristics defined in the output descriptor override any duplicate characteristics of the SYSOUT initialization statement. The characteristics of the SYSOUT initialization statement override any duplicate characteristics of the OUTSERV initialization statement.

2. When a TSO user issues the TSO OUTDES command to define a default output descriptor, any SYSOUT the TSO user creates is associated with that output descriptor.

- Given the following initialization statements:

```
OUTSERV,COPIES=1
SYSOUT,CLASS=B,TYPE=PUNCH,COPIES=2
```

If the TSO user issues the TSO TRANSMIT command, then only one copy of the data set will get sent.

If the TSO user issues the TSO OUTDES xxxx ... DEFAULT command followed by a TSO TRANSMIT command, two copies of the data set are sent.

TSO OUTDES Command:

```
OUTDES MYOUTDES DEFAULT NAME('My Name')
```

The COPIES=2 parameter from the SYSOUT initialization statement merges into the SYSOUT characteristics because the SYSOUT is associated with an output descriptor created by the OUTDES command.

- 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 to override those specified by earlier default `//*FORMAT` statements. The user receives the data set formatted only one way.

- 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 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.

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.

Note: For system application printer interface (SAPI) and process SYSOUT callers, JES3 does not schedule work to SAPI or process SYSOUT when searching the SAPI or WTR wait of work queues. If a match is made, the waiting application is posted for work and must issue an IEFSSREQ call requesting JES3 to schedule the work to it.

Determining Which Output Parameters Apply

The parameter values used for output vary according to the interaction between the device defaults and parameter values in the OSE based on writer-selection criteria. The following is a guideline for understanding which output parameters apply:

How device characteristics are set and changed: Parameters on the DEVICE statement act as the defaults when the device is initialized. Once the device is initialized, some of these defaults can be modified by the *CALL, *START, and *RESTART operator commands or by the type of output sent to the device. If you perform a warm or cold start on your system, the initial defaults are used and the modifications are lost.

How output characteristics are associated with a data set: Defaults specified on the OUTSERV and SYSOUT initialization statements and JCL or JES3 control statements are used to build the OSE for each instance of output. See “Override Sequences for Output Data Set Information” on page 140 for more detail.

How data set characteristics are associated with a device: For each parameter defined in the OSE (for example: CHARS or FORMS), one of the following is true:

- If the parameter is not one of the writer-selection criteria, the value in the OSE is ignored and the current value defined for the device is used for the output. See Example 1.

- If the parameter is one of the writer-selection criteria and is not changeable on the device, only an OSE whose parameter value matches the value on the device will be selected for output. See Example 2.
- If the parameter is one of the writer-selection criteria and is changeable on the device, the value on the device is changed to match the parameter value of the OSE to be selected for output. See Example 3.

Examples: The following statements apply to all of the examples.

In the initialization deck:

```
DEVICE ... WS=(D),CHARS=GS10,MODE=COMP
SYSOUT CLASS=A,TYPE=PRINT
```

In the job's JCL:

```
//TEST JOB ...
//KME DDSYSOUT=A,CHARS=GT15
```

- **Example 1:**

```
*CALL WTR OUT=00F
*START 00F
```

In the example, a writer is started for the first time. There is no writer-selection criteria for CHARS on the DEVICE statement, so the system uses the character defined by the device statement. The value specified by the user on the JCL is overridden. The CHARS=GS10 parameter is used.

- **Example 2:**

```
*CALL WTR OUT=00F,WS=(U),CH=(GT15,H)
*START 00F
```

In this example, a device is started with a writer-selection criteria of CHARS and the CHARS value is not changeable on the device. Because the device is selecting on CH=GT15, the SYSOUT data set name KME is selected. Note that the character set on the device matches the character set in the OSE.

- **Example 3:**

```
*CALL WTR OUT=00F,WS=(U),CH=(GT10,R)
*START 00F
```

In this example, a device is started with a writer-selection criteria of CHARS and the CHARS value is changeable on the device. The device is selecting on CH=GT10, but because the CHARS value is changeable, data set KME is selected and the character set on the device is changed to GT15.

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 8, “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.

There are several methods of controlling the starting and stopping of writing output:

- 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.
- 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.

- A *Timeout Value* can be defined for a writer. This value controls how long a writer remains idle before terminating.

The value is specified on the DYNAMIC= keyword of the DEVICE initialization statement (for example, DYNAMIC=(YES, *timeout value*)). If you specify DYNAMIC=(YES) (with no timeout value), the writer stops automatically when no more output is available for processing. If you specify DYNAMIC=(NO) (with no timeout value), the writer remains active indefinitely even when no output is available for processing.

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/SAPI Applications

External writer/SAPI application 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/SAPI application interacts with JES3, via the subsystem interface, to request data sets for processing. A subset of the output service scheduling function called PROCESS SYSOUT and system application printer interface are invoked as a result of this kind of request.

No attempt is made by output service to schedule external writers/SAPI applications as a result of constructing OSEs requiring their services; it is the responsibility of the operator to start external writers as required. JES3 posts started writers that are waiting for work when incoming jobs require processing.

For more information on external writers/SAPI applications, see *z/OS MVS Using the Subsystem Interface*.

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 using output data sets. When a job stream enters the system, data management 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.

If the submitter of an internal reader job is either a TSO user or has an associated TSO userid, the TSO userid will be propagated to the submitted job.

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 *z/OS JES3 Commands*.

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 *z/OS TSO/E Command Reference*.

Using 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).

Output Service Installation Exits

Output service provides eight installation exits, which are summarized in Table 33.

Table 33. Output Service Installation Exits

Installation Exit Routine	Purpose
IATUX19	Allows the user to access the contents of the temporary OSEs

Table 33. Output Service Installation Exits (continued)

Installation Exit Routine	Purpose
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
IATUX39	Allows the user to modify the data set header for a SYSOUT data set
IATUX43	Allows the user to modify a job header for a network stream containing SYSOUT data
IATUX45	Allows the user to modify the job information for data sets processed by output writer functional subsystems

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

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 *z/OS MVS System Management Facilities (SMF)* for a complete description of record type 26.

Part 5. Defining and Managing C/I Service

Chapter 5. Defining and Managing C/I Service	159
Setting Up C/I Service	160
Advantages to Using C/I FSS Address Spaces	160
How Many C/I FSS Address Spaces and Where to Put Them.	160
Storage Management Subsystem Considerations	161
Avoiding Common Storage Constraints.	161
MVS Performance Considerations	162
Defining a C/I FSS Address Space	162
Defining the Maximum Number of CI and POSTSCAN DSPs	163
Specifying the Number of CI DSPs in the JES3 Global Address Space	163
Specifying the Number of CI DSPs in a C/I FSS Address Space	164
Specifying the Number of POSTSCAN DSPs	165
Controlling Jobs Through C/I Service	166
Controlling Job Flow with Installation Exits	166
Assigning Jobs to the Appropriate Processor and Address Space for C/I Service.	167
Using System Symbols in Source JCL for Demand Select Jobs	169
Defining a Converter/Interpreter Options List	169
Managing the Scheduler Work Area	170
Creating SWA Space	170
Preventing a Job from Dominating the SWA	171
Selecting the Job JCL Statement Limit	171
Preventing Abnormal Termination of JES3 or a C/I FSS Address Space	171
Selecting the Address Space JCL Statement Limit	172
Monitoring and Modifying C/I Service.	174
Determining the Status of C/I Service	174
Displaying the Status of Jobs	174
Displaying the Status of CI and POSTSCAN DSPs	174
Modifying C/I Service	175
Managing Procedure Libraries.	176
Updating Procedure Libraries	177
Displaying the Status of Procedure Library Data Sets	178

Chapter 5. Defining and Managing C/I Service

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

- *Converter/interpreter phase:* Uses MVS services 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 main (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 gathers information 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 JES3 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 or any local main. Where the address space operates depends on what you specify for the FSS on its JES3 FSSDEF initialization statement. You specify the maximum number of CI DSPs that may be active in a C/I FSS address space using the JES3 FSSDEF initialization statement defining that address space.

The CI DSPs that run in a C/I FSS address space process jobs through the converter/interpreter and prescan phases of C/I service. JES3 can also perform the catalog search (locate) portion of the postscan phase in a C/I FSS if the main where the CI FSS runs has access to the necessary catalogs.

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 for the remainder of 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 JES3 STANDARDS initialization statement.

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.

Controlling Jobs Through C/I Service briefly describes the installation exit routines and other means available to influence a job's C/I processing.

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

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.

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

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

The primary advantage of using C/I FSS address spaces is to off-load much of the overhead caused by converter/interpreter processing from the global address space.

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.

How Many C/I FSS Address Spaces 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.

Storage Management Subsystem Considerations

If you are using the storage management subsystem (SMS), you should define C/I functional subsystems only on mains where SMS is enabled. JES3 will start C/I FSSs only on mains where SMS is active.

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 88 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 two C/I FSS address spaces are active in a given processor, that processor will have two BALJ data areas, each occupying 338 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 160 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 two 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 three data set status blocks (DSSs) in CSA. Each DSS occupies 328 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 central 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.

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.SIATSAMP in the members IATJ3CI and JES3CI. The JCL statements included in this procedure are:

```
//JES3CI EXEC PGM=IATINTKF,DPRTY=(15,14)
//STEPLIB DD DSN=SYS1.SIATLIB,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. If your JES3 load library is in the link list, you do not need the STEPLIB DD statement.

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.

For more information about coding the FSSDEF initialization statement and its selective parameters, see *z/OS JES3 Initialization and Tuning Reference*.

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 installation 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 576 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, JESJCLIN, and JESYSMSG data sets** require the following storage space for each data set to be accessed:
 - IATYDSB (DSB), the data set block: 208 bytes in subpool 230

- IATYDSS (DSS), the data set status block: 328 bytes in subpool 241
- IEZDEB (DEB), the data extent block: 68 bytes in subpool 0
- IATYDAT (DAT), the USAM spool data buffer block: two pages in subpool 229. To determine how many DATs fit in the two pages, divide 8192 bytes (two 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: 160 bytes in subpool 229 for every DAT (see above)
- **The RESQUEUE cell pool** requires 600 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 600, or 1800. 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 600 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 installation 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 installation exit routines and initialization statements available to you for controlling jobs through C/I service.

Controlling Job Flow with Installation Exits

The installation exit routines available for controlling job flow through C/I service are summarized in Table 34. 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 installation exits let you write routines to examine and change the results of C/I processing. One installation 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. Installation 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 installation exit routines, see *z/OS JES3 Customization*.

Table 34. Installation Exits for Monitoring JCL Interpretation

Installation Exit Routine	Purpose
IATUX03	Allows you to access JES3 Converter/Interpreter 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

Table 34. Installation Exits for Monitoring JCL Interpretation (continued)

Installation Exit Routine	Purpose
IATUX11	Allows you to inhibit the writing of the LOCATE request/response into a job's JESYSMSG 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)
IATUX29	Allows you to 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 <code>//*DATASET</code> and <code>//*ENDDATASET</code>)
IATUX34	Allows you to verify or change (during input service) DD JCL statements (except <code>DD *</code> or <code>DD DATA</code> 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

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

If you have defined C/I FSS address spaces, two installation 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 installation 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 installation 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 installation exit routines, see *z/OS JES3 Customization*.

You may also influence JES3's decision as to where to schedule C/I processing by the CIDEMAND (for demand select jobs) and CIBATCH (for batch jobs) initialization parameters on the CLASS and/or STANDARDS initialization statements. You may use these parameters to indicate that jobs of a certain class or all jobs must have C/I processing performed where the job's class is enabled or where the job is eligible to run.

Within the limits imposed by installation exits IATUX46, IATUX49, and the CIDEMAND and CIBATCH initialization parameters, 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 any C/I FSS on the global, provided that the job can execute on the global.
3. In any C/I FSS on a local, provided that the job can execute on the selected local.
4. In any C/I FSS on the global, regardless of whether the job can execute on the global.
5. In any C/I FSS on a local, regardless of whether the job can execute on the selected local.

Note: If more than one C/I FSS is active, JES3 selects the FSS with the greatest number of available demand select CI DSPs.

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

1. In any C/I FSS on the global, provided that the job can execute on the global.
2. In any C/I FSS on a local, provided that the job can execute on the selected local.
3. In any C/I FSS on the global, regardless of whether the job can execute on the global.
4. In any C/I FSS on a local, regardless of whether the job can execute on the selected local.
5. In the JES3 global address space.

Notes:

1. If more than one C/I FSS is active, JES3 selects the FSS with the greatest number of available batch CI DSPs.
2. Only the JES3 `//*MAIN SYSTEM = JECL` statement is considered for CI scheduling purposes when determining where the job is eligible to run. The job's scheduling environment does not apply.

Using System Symbols in Source JCL for Demand Select Jobs

JES3 allows you to specify system symbols in source JCL for demand select jobs. The symbols allow two or more systems in a JES3 complex to share the source JCL, while retaining unique values in the JCL.

System symbols represent unique values in source JCL for demand select jobs. Each system has its own values for system symbols. When shared demand select jobs are processed, each JES3 system replaces the system symbols with its own values.

When the source JCL for a demand select job is sent to another processor for conversion (see the section that describes the converter/interpreter phase in Chapter 5, “Defining and Managing C/I Service” on page 159), the converter uses the system symbols that are defined to the processor on which the started task was submitted (*not* the processor on which the conversion takes place).

For more information about system symbols, see:

- *z/OS MVS Migration* for an overview of system symbols
- *z/OS MVS Initialization and Tuning Reference* for details about planning to use system symbols
- *z/OS MVS JCL Reference* for information about using system symbols in JCL
- *z/OS JES3 Commands* for information about using system symbols in JES3 commands.

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 expanded 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 a Job from Dominating the SWA” on page 171 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 expanded 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 installation 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 hot start with refresh, 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 installation exit IATUX41. This installation exits 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 installation exit IATUX41, see *z/OS JES3 Customization*.

To set or change the job JCL statement limit without restarting JES3, issue the *FX,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 *IX,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 MAXASST= 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.

Note: 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.

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.

Selecting the Address Space JCL Statement Limit

Table 35 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.

Table 35. Procedure for Selecting and Setting an Address Space JCL Statement Limit

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.

Table 35. Procedure for Selecting and Setting an Address Space JCL Statement Limit (continued)

Procedure	Example
<p>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.)</p> <p>The quotient represents the maximum number of JCL statements whose scheduler control blocks can be stored in the SWA at one time.</p> <p>Ignore the remainder.</p>	<ul style="list-style-type: none"> • 60000/600 = 100 with a remainder of 0 • 100 JCL statements can be stored in scheduler control blocks in the SWA at one time
<p>3. To set (or change) the address space JCL statement limit using the initialization stream:</p> <ul style="list-style-type: none"> • 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. <p>(See Note 2.)</p> <p>Restart JES3 with a warm start or hot start with refresh.</p>	<p>STANDARDS,MAXASST=100,...</p> <p>FSSDEF,TYPE=CI,FSSNAME=fssname,MAXASST=100,...</p>
<p>4. To set (or change) the address space JCL statement limit without restarting JES3:</p> <ul style="list-style-type: none"> • For the JES3 global address space, issue the command *F,X,D=CI,MAXASST=100 • For a C/I FSS address space, issue the command *F,F,FSS=fssname,MAST=100 <p>(See Note 2.)</p>	<p>*F,X,D=CI,MAXASST=nnn.</p> <p>*F,F,FSS=fssname,MAST=nnn.</p>
<p>Notes:</p> <ol style="list-style-type: none"> 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. 2. Substitute the value you got in step 2 for nnn. 	

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 *z/OS JES3 Commands*.

Determining the Status of C/I Service

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

Displaying 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`

Displaying 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 information on why jobs are not being scheduled for the CI DSP and how long they are waiting:
`*I,Q,D=CI,W`
- 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 having a scheduling environment in a particular priority and waiting to be scheduled

- *I,P=12,W
- For jobs currently being processed by POSTSCAN DSPs:
 - *I,A,D=POSTSCAN
- For information on why jobs are not being scheduled for the POSTSCAN DSP and how long they are waiting.
 - *I,Q,D=POSTSCAN,W
- 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. Change is not remembered across a hot start with refresh (unless the FSS is currently active). 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 activate C/I debug processing while the job is in the queue:
*F,J=nnnn,CIDEBUG
- 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 hot start with refresh or 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 Chapter 2, "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 if you are using C/I FSSs. You must also catalog procedure libraries if you plan to use the JES3 procedure library update facility. See Updating Procedure Libraries for information about how to update procedure libraries.

Note: Converter/Interpreter functional subsystems (C/I FSS) and the PROCLIB update function will obtain unit and volume information for the procedure libraries from the catalog. For these functions, JES3 ignores unit and volume information that you specify 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 block size.

Attention: If a job that updates a procedure library is in a JES-managed job class group, and it is updating the procedure library used to start initiators, make sure that there is at least one initiator started before allowing the job to enter the system. Otherwise, a deadlock will occur; the procedure library used to start the initiator is disabled, the job is waiting for an initiator, and the initiator is waiting for the procedure library to be enabled. If this situation arises, the updating job must be cancelled and resubmitted or a `*RESTART,SETUP,jobno,CI` command can be used to enable the procedure libraries (and let the initiator start) and restart the job through C/I processing.

If a job that updates a procedure library is in a WLM-managed job class group, it is not necessary to check whether there is an initiator started before submitting the job. WLM-managed initiators are not started under JES3 and are not affected when a procedure library is disabled.

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.

Note: If you use the proclib update facility to move a procedure library to another volume, and the procedure libraries are allocated through the JCL statements in the JES3 cataloged start procedure, The JES3 local address spaces must be restarted in order to:

1. Reallocate the procedure library on the new volume. This is necessary if a JES3 local processor is a DSI candidate. Before a DSI is performed, all locals, which are DSI candidates must be restarted in order to pick up the change.
2. Vary offline the volume containing the old procedure library. During proclib update processing, the JES3 global and C/I FSS address spaces unallocate the procedure libraries being updated. However, the JES3 local address spaces do not unallocate the proclibs during proclib processing. This causes the VARY command for the proclib to fail when you attempt to vary the proclib offline to one of the local processors.

The JES3 local address spaces do not have to be restarted if the procedure libraries are defined via the DYNALLOC statements in the initialization stream.

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.

Part 6. Defining and Managing Spool Data Sets

Chapter 6. Defining and Managing Spool Data

Sets	181
Defining Spool Data Sets	181
Determining How Many Spool Data Sets You Should Allocate	181
Allocating Spool Data Sets	182
Formatting Spool Data Sets	182
Formatting During JES3 Initialization	182
Formatting with IEBDG	183
Reformatting a Spool Data Set	184
Using Spool Partitions	184
Advantages to Spool Partitioning	184
Isolating Different Spool Data Types	185
Defining Spool Partitions	185
Defining Spool Partition Overflow	186
Specifying Spool Data Sets as Members of Spool Partitions	187
Specifying a Spool Partition for Spool Data	188
Determining the Order of Spool Partition Overrides	188
How the User Can Request a Spool Partition Example of Spool Partitioning	189
Defining Spool Space Allocation Units	192
Defining a Track Group	192
Determining the Size of a Track Group	192
Determining Track Group Allocation Sizes	194
Track Group Allocation Overrides	195
Managing Spool Space	196
Adding or Deleting a Spool Data Set	196
Replacing a Spool Data Set	196
Moving a Spool Data Set to Another DASD Volume	197
Balancing the Work Load Across Spool Partitions	197
Deleting Output Data Sets	199
Freeing Spool Space Using the Dump Job Facility	199
How to Use Performance Measurement Tools to Tune JES3 Spool	199

Chapter 6. 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 either permanently or temporarily, using JES3 operator commands 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, 3380, 3390, or 9345 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).

See *z/OS JES3 Commands* for a discussion of each of these methods of starting JES3.

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 document *MVS/DFP Utilities* explains how to use IEBDG. Figure 18 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 18. 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 IEC0311. 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).

See *z/OS JES3 Commands* for a discussion of each of these methods of starting JES3.

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”.)
- 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” on page 196 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 do not define a spool partition (do not include SPART statements in the initialization stream), JES3 creates a minimum of three spool partitions. JES3 assigns spool data sets to one of the following partitions:

- DRAINED - result of an operator drain, hold or cancel command
- UNAVAIL - for data sets unavailable to JES3 during JES3 initialization
- DEFAULT - JES3 names the default partition 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.

SPART Initialization Statements — Example

```
SPART,NAME=TSODATA,OVRF=SMLBATCH
SPART,NAME=SMLBATCH,OVRF=YES
SPART,NAME=DEFPART,DEF=YES
SPART,NAME=BIGBATCH,OVRF=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.)

Attention:

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.**

Circular Overflow with SPART

```
SPART,NAME=TSODATA,OVRF= SMLBATCH
SPART,NAME=SMLBATCH,OVRF= TNYBATCH
SPART,NAME=TNYBATCH,OVRF= 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 OVRF=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.

Table 36 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

Table 36. Spool Partition Overrides

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	—	<ul style="list-style-type: none"> • Input data (SYSIN) is written to the default partition. • All other data is written to the partition specified on the MAINPROC statement.
—	—	X	<ul style="list-style-type: none"> • Input data (SYSIN) is written to the default partition. • All other data is written to the partition specified on the CLASS statement.
—	X	X	<ul style="list-style-type: none"> • Input data (SYSIN) is written to the default partition. • All other data is written to the partition specified on the CLASS statement.
X	X	—	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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.
X	X	X	<ul style="list-style-type: none"> • 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.

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 *z/OS MVS JCL User's Guide*.

The installation can override the spool partition specified by the user by coding installation 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
//
```

Table 37 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.

Table 37. Spool Partitions Used in Spool Partition Example

<i>PART 1</i>			
When this job runs on processor SY1	JES3 writes the job's spool data to this partition:		
	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
<p>Notes:</p> <ol style="list-style-type: none"> JES3 writes input (SYSIN) for all jobs to the default spool partition, PARTA, because JES3 always writes input data to the default partition. 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. 			
<p>JOB1:</p> <p>JES3 writes the data in SYSOUT class N to the default partition because:</p> <ul style="list-style-type: none"> The SYSOUT statement that defines class N specifies no spool partition. The MAINPROC statement that defines processor SY1 specifies no spool partition. 			
<p>JOB2:</p> <p>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 <code>/**MAIN</code> statement assigns this job to job class IMSBATCH.)</p>			
<p>JOB3:</p> <p>JES3 writes the data in SYSOUT class N to spool partition PARTE because the <code>/**MAIN</code> statement specifies partition PARTE. (A spool partition specified on a <code>/**MAIN</code> statement overrides a spool partition specified on a CLASS statement.)</p>			
<i>PART 2</i>			
When this job runs on processor SY2	JES3 writes the job's spool data to this partition:		
	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
<p>Notes:</p> <ol style="list-style-type: none"> JES3 writes input (SYSIN) for all jobs to the default spool partition, PARTA, because JES3 always writes input data to the default partition. 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. 			

Table 37. Spool Partitions Used in Spool Partition Example (continued)

<p>JOB1:</p> <p>JES3 writes the data in SYSOUT class N to spool partition PARTC because the MAINPROC statement that defines processor SY2 specifies PARTC.</p>
<p>JOB2:</p> <p>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.)</p>
<p>JOB3:</p> <p>JES3 writes the data in SYSOUT class N to spool partition PARTE because the <code>/*MAIN</code> statement specifies PARTE. (A spool partition specified on a <code>/*MAIN</code> statement overrides a spool partition specified on a CLASS statement.)</p>

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. 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.)

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; for example, 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; for example, 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 (for example, 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.

Attention: 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 *z/OS 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 Table 38.)

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.

Table 38. Record, Track, and Cylinder Characteristics for DASD Devices

Record	Record Size	Records/Track	Records/Cylinder
9345	4K	10	150
	2K	17	255
3390	4K	12	180
	2K	21	315
3380	4K	10	150
	2K	19	285
3375	4K	8	96
	2K	14	168
3350	4K	4	120
	2K	9	270
3340-70	4K	2	24
	2K	4	48
3340-35	4K	2	24
	2K	4	48
3330-11	4K	3	57
	2K	6	114
3330-1	4K	3	57
	2K	6	114
9345	4K	10	150
	2K	17	255

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 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: Installation 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 spool data sets, you can use the *MODIFY,U,Q= command to delete 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” on page 196 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*. 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. *z/OS 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.

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.

When you replace a spool data set, JES3 cancels all jobs with data on the replaced spool data set. If the replaced data set contains STT records, JES3 might lose

information that could result in the loss of jobs in the system. STT records include information such as the status of devices, DJC network data, deadline queue data, volume unavailable data, dynamic allocation checkpoint data, output service checkpoint data, JESNEWS, device fencing data, virtual unit status, GMS status, and FSS checkpoint data. If STT data is lost, JES3 issues messages that allow you to take the appropriate recovery 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.

If you replace spool data sets that contain STT information, subsequent spool `*INQUIRY` commands may indicate that the spool data sets have been drained. Because STTs are not actually drained, error messages may appear regarding the STT, the replaced volume, or jobs if any portion of the STT (which may span multiple extents) still contains records in use.

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.

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 or hot start with refresh, 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 Output Data Sets

You can delete output data sets from the output service hold, writer, or BDT queue using the *MODIFY,U operator command. The *MODIFY,U command allows you to specify the criteria by which JES3 selects data for deletion. For details about using the JES3 *MODIFY command see *z/OS JES3 Commands*.

You can code installation exit IATUX48 to override the operator command parameters that select the data sets for deletion. For information about using IATUX48 see *z/OS JES3 Customization*.

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 *z/OS JES3 Commands*.

How to Use Performance Measurement Tools to Tune JES3 Spool

JES3 performance is highly dependent upon efficient spool access. JES3 can perform poorly if your spool volumes or spool DASD channels are too busy or if you do not have sufficient spool space. Use the following reports to measure spool space and spool access:

- Check the **Spool Space Utilization Snapshot** portion of the JMF report or issue an *INQUIRY,Q,DD= command to display the amount of spool space being used. A reasonable limit is 35% - 40%. If the total spool space remaining is less than 60%, consider adding additional spool volumes.
- Check the **I/O Device Activity** portion of the RMF report to measure the amount of time spool is busy. A reasonable limit is 30% to 40%. You can reduce the amount of time that spool is busy by adding additional spool volumes.
- Check the **Channel Path Activity** portion of the RMF report to obtain the percentage of time a spool channel is busy. A reasonable limit is 25% - 30%. To improve spool availability, move your spool packs to the least busy channel.

The following table summarizes the actions you can perform to tune spool:

Table 39. Summary of Spool Tuning Techniques

Spool Tuning Task	Tuning Actions
Improve average seek time or avoid lockouts	Allocate only one spool data set per spool volume. When a volume contains more than one spool data set, the average seek time to access the data increases, which can degrade JES3 performance.
Improve spool space utilization for complexes that support mostly small jobs and TSO users	Specify a group size value that is less than half the size of a cylinder for the spool device type. Use the GRPSZ= keyword on the BUFFER or SPART initialization statements to specify the group size. A small value also reduces seek time on moveable head devices.
Reduce the amount of time that a job must wait to receive additional spool space	Specify a track group size value roughly the number of records per cylinder. Specifying a larger value can improve performance if your workload consists mostly of jobs that produce large amounts of output.

Table 39. Summary of Spool Tuning Techniques (continued)

Spool Tuning Task	Tuning Actions
Gain additional spool space without adding additional spool data sets	Delete unused output data sets using the JES3 *MODIFY,U command or replace existing spool data sets with larger data sets. Replacing spool data sets however, requires a warm start. Use the Dump Job DSP to release spool space by moving jobs to tape. You can restore those jobs when spool space becomes available.
Reduce spool I/O	Specify a buffer size of 4K. However, specifying a buffer size of 4K uses more virtual storage than a 2K buffer. Use the BUFSIZE= keyword of the BUFFER initialization statement to specify the size of JES3 buffers.
Reduce spool space usage	Specify a buffer size of 2K. Specifying a buffer size of 2K, however, increases spool I/O.
Improve spool access	<p>1) Allocate the JCT data set on its own spool volume or on a low-usage spool pack. Avoid placing the JCT data set on a catalog volume because of the reserve/release interference. Device busy for this data set should not exceed 10%.</p> <p>2) Center the single-track-table (STT) across all spool volumes or dedicate a separate spool extent to the STT using the STT parameter on the TRACK initialization statement. For fixed head devices, allocate cylinders under the fixed heads.</p> <p>3) If you can access your spool devices through an alternate control unit (such as string switching), enter yes for the ALTCTRL feature on HCD's Define Device Parameters / Features panel.</p> <p>4) Move one or more of your spool packs to the least busy DASD channel.</p>
Limit processor contention for spool volumes	Use spool partitioning.
Optimize spool space usage by using different track group sizes	Use spool partitioning.
Ensure spool space availability for critical work	Use spool partitioning.
Improve spool recovery and spool space management	Use spool partitioning.

Part 7. Defining Consoles and Message Routing

Chapter 7. Defining Consoles and Message

Routing	203
Defining Consoles	203
Defining MCS Consoles	203
Defining RJP Consoles	204
JES3 Console Management	204
Operator Communication	204
Input Processing	204
MCS Console Management	205
Entering Commands	205
Defining the Hardcopy Log	206
Recording Message Traffic	208
Rules	208
Defining Message Routing	209
Where and How Messages Originate	210
The JES3 MESSAGE Macro	210
The MVS WTO and WTOR Macros	210
Where Messages Can Go	211
Understanding the General Path of a Message	211
Path of Message Issued from a Local Processor	212
Path of a Message Issued from a Global Processor	213
JES3 Destination Classes and MVS Routing Codes	214
Two Types of Messages	215
Routing JES3 Messages to Consoles	216
Using MSGROUTE to Control Message Routing	216
Processing if SY1 Is either the Local or Global:	217
Coding Rules for the MSGROUTE Statement	217
Message Routing Exceptions	217
Message Routing Specified by MPF	218
Installation Exit.	218
Action Messages that Must Be Displayed	218
Hardcopy Only Messages	218
Deleted Messages	218
No Routing Specified.	218
Messages that Originate from Functional Subsystems	219
Broadcast Messages	219
Suppressed Messages.	219
Messages that Specify Only an MVS Routing Function	219
Job Status Messages Without Routing Codes	219
Special Messages	219
Diagnosing Misrouted Message Traffic	220
Suppressing the Display of Messages	220
Specifying Messages for Display Suppression	220
Controlling Message Suppression.	221
Examples of Message Suppression	221
Using MPF to Screen Messages for Automation	222
Automating Message Processing	223
Using NetView to Automatically Respond to Messages	223

How to Reduce Message Traffic Using NetView and MPF.	223
Order of Command Execution.	223

Chapter 7. Defining Consoles and Message Routing

This section 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 documents in which you can reference them are cited throughout this topic.

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:

- Multiple console support (MCS) consoles
- RJP (Remote job processing) consoles

MCS consoles are devices that you can physically attach to global or local processors. These consoles allow you to control the sysplex. Refer to *z/OS MVS Planning: Operations* for information about MCS consoles in a sysplex environment.

RJP consoles are devices that you attach to the JES3 global as part of a remote workstation using telecommunications lines. RJP permits you to submit jobs to and receive output at workstations that reside at some distance from your installation.

Defining MCS Consoles

The term *MCS consoles* refers to:

- MCS-managed consoles
- Extended MCS consoles

You must define MCS consoles in the CONSOLxx member of the MVS SYS1.PARMLIB data set. Extended MCS consoles are defined and activated through programs such as TSO/E and are authorized through RACF for use as operator consoles. MCS consoles accept MVS commands based on their MCS authority level or RACF authorization.

Any MCS console with master authority is the principal device for communication with MVS. Any MCS console with master authority has the highest MCS authority level assigned to it. MCS consoles with master authority can enter 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.

MCS consoles accept all JES3 commands except *FREE (non-directed) which affects console screen presentation.

JES3 on a local processor accepts only the following commands:

- *CALL, *START, *CANCEL,DSI
- *CALL, *CANCEL,JMF
- *CALL, *START, *CANCEL,VARYL

- *RETURN (requires a password)
- *DUMP (requires a password)
- *TRACE

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 consoles. Refer to “Using JES3 to Authorize Commands from MCS Consoles” on page 91 for information about defining console authority.

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 workstation to have the facilities of a JES3 console, you must code the workstation name and console options on the JES3 CONSOLE initialization statement for that console.

You can define simulated consoles for workstations 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

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.

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 entered. 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.

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.
- **Screen-oriented displays on display consoles:** At system initialization, the MCS console screen can be divided into multiple screen areas for receiving these displays in out-of-line or non-message 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. See "Using JES3 to Authorize Commands from MCS Consoles" on page 91 for information about defining console authority.
- **Console LOGON:** You can require that operators LOGON to MCS consoles by specifying options in the `CONSOLxx` member of the `MVS SYS1.PARMLIB` data set. You can also activate RACF to ensure that the person attempting to gain access to an operator console has the authority to do so. See *z/OS MVS Initialization and Tuning Guide* for additional information about `SYS1.PARMLIB`. Refer to "Authorizing Console Access" on page 76 for additional information about using RACF to restrict console access.
- **Log facilities:** You can use the operations log, systems log, or a real device to record the hardcopy log.
- **Enhanced display capability:** You can control several characteristics of MCS console such as reverse video, extended highlighting and seven color support.
- **NOJES3 Option:** If both JES2 and JES3 are installed on the same system and JES2 is running alone, then initialize with the `NOJES3` option to enable the short form reply. See *z/OS MVS Initialization and Tuning Reference* for more information on the `CON=` parameter.

Entering Commands

You can enter JES3 commands from all consoles.

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 consoles reject the JES3 `*FREE` command (however, you can issue `*FREE,con` which is also known as a directed `*FREE` command from MCS consoles).

JES3 on a local processor accepts only the following commands:

- *RETURN (with password)
- *DUMP (with password)
- *CALL, *CANCEL,JMF
- *CALL, *START, and *CANCEL,VARYL
- *CALL, *START, *CANCEL,DSI
- *TRACE

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.

Defining the Hardcopy Log

The hardcopy medium (also known as the hardcopy log) records command and message traffic for your systems. MVS and JES3 provide three forms of the hardcopy medium:

OPERLOG

centrally records command and message traffic for systems in a sysplex in Message Data Block (MDB) format.

JES3 DLOG

centrally records command and message traffic for systems in a JES3 complex in JES3 format. The JES3 DLOG is written to SYSLOG on the global processor.

SYSLOG

individually records command and message traffic for each system in MVS format.

IBM recommends use of OPERLOG on all systems in the sysplex as the only normally active hardcopy medium. The OPERLOG MDB records contain considerably more information than either the JES3 DLOG or SYSLOG formats. In addition, with OPERLOG each system writes its own command and message traffic to the common log, rather than all log activity taking place on the JES3 global processor as with DLOG.

If migration to OPERLOG cannot be achieved concurrent with the installation of JES3 SP5.2.1, the JES3 DLOG may be used as part of a staged migration to OPERLOG.

The JES3 DLOG, when active, contains command and message traffic for all systems in the JES3 complex. OPERLOG, on the other hand, may be activated on a system by system basis during the migration period. Thus, a staged migration to OPERLOG may be accomplished as illustrated in Figure 19 on page 207.

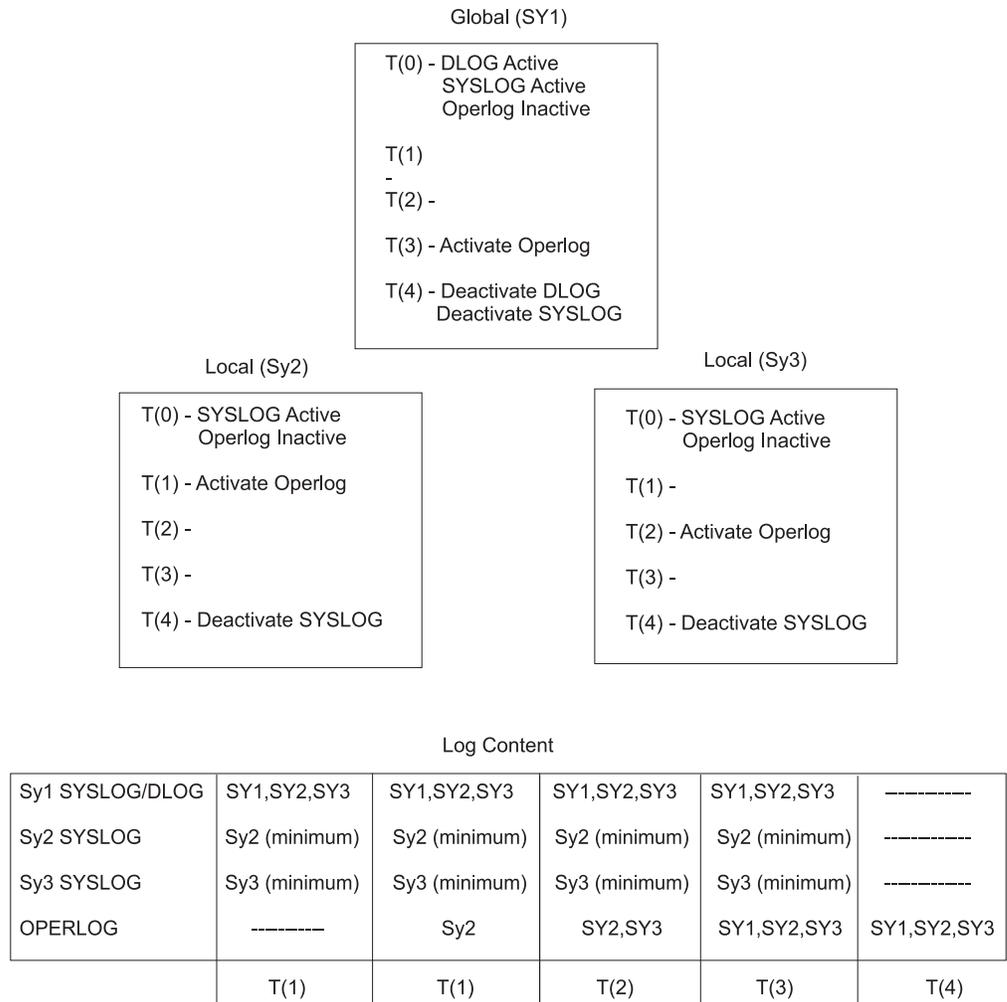


Figure 19. Hardcopy Log Migration Scenario

Initially, at time **t(0)**, DLOG is active and SYSLOG is active on all systems. In this example, we elected to activate the Operations log on local SY2 first. At this point forward, time **t(1)**, the hardcopy log generated by SY2 will be written to the Operations log.

At time **t(2)**, the Operations log is activated on SY3. Now the hardcopy log from both locals will be written to the Operations log.

At time **t(3)**, the Operations log is activated on SY1. Now the hardcopy log from each system will be written to the Operations log. The sysplex now has two sysplex-wide logs: a JES3 formatted log written by DLOG, and the operations log.

From time **t(0)** through **t(3)**, the JES3 SP5.2.1 DLOG facility will provide a JES3 formatted log containing messages from all systems in the JES3 complex.

At time **t(4)**, the DLOG facility and SYSLOG are deactivated on each system. The sysplex-wide log remaining is the Operations log.

Recording Message Traffic

JES3 and MVS record all message traffic in the system log. Each DLOG 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 console from which a command was entered
- The 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 20 shows sample log entries.

```
CONMSTR1 95178 1744241 -W A
MLG      95178 1744241 SY1 R= SYSLOG   IEF196I IEF237I JES3 ALLOCATED TO SYSLOG03
MLG      95178 1744242 SY1 R= SYSLOG   IEF196I IEF285I  +MASTER+.SYSLOG.JOB00001.D000000B.?      SYSOUT
          95178 1744242 SY1 R= SYSLOG   IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS A
MLG      95178 1744242 SY1 R= SYSLOG   0000000 SY1      95178 17:44:24.23 SYSLOG  00000000 IEE042I SYSTEM LOG
MLG      95178 1744242 SY1 R= SYSLOG   DATA SET INITIALIZED
SI       95178 1744244 IAT7001 JOB SYSLOG (JOB00001) IS ON WRITER PRT002(0002),RECORDS=147
LOG      95178 1744244 IAT7007 JOB SYSLOG (JOB00001) ON WRITER PRT002 (0002), DSN=
LOG      95178 1744244 IAT7007 +MASTER+.SYSLOG.JOB00001.D000000B.?, PURGED.
CONMSTR1 95178 1744429 +I Q
CONMSTR1 95178 1744429 IAT8674 JOB SYSLOG (JOB00001) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1 95178 1744429 IAT8674 JOB IRRDPTAB (JOB00003) P=15 CL=A      OUTSERV(PENDING WTR)
CONMSTR1 95178 1744429 IAT8674 JOB VTAMJ3 (JOB00009) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1 95178 1744429 IAT8674 JOB DC (JOB00010) P=15 DC(ACTIVE)
CONMSTR1 95178 1744430 IAT8674 JOB TCAS (JOB00011) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1 95178 1744430 IAT8674 JOB SUPERU (JOB00012) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1 95178 1744488 +I Q Q SP=ALL
CONMSTR1 95178 1744489 IAT8980 DRAINED HAS NO SPOOL DATA SETS
CONMSTR1 95178 1744489 IAT8980 UNAVAIL HAS NO SPOOL DATA SETS
CONMSTR1 95178 1744489 IAT8509 PART1 : 525 GRPS, 410 LEFT ( 78%); MIN 3%, MRG 5%, DEF, INIT
```

Figure 20. Sample Log Entries (with 2-digit-year dates)

Rules

- 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 LOGLMT 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 specify during initialization whether you want JES3 DLOG to be active. You can also use commands to dynamically changes the status of JES3 DLOG. Through the use of *MODIFY,O commands, you can enable or disable DLOG. However, JES3 rejects any command that would leave your installation without some form of hardcopy log.
- 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 *z/OS MVS Initialization and Tuning Guide*. Also, the HCFORMAT keyword on the

HARDCOPY parameter provides you the option of recording year dates in 2- or 4-digit format. Compare the sample sections of the log entries in Figure 20 on page 208 to Figure 21 for the differences in the fields that contain the console name and year as follows:

Year Date Format

Column

2-Digit

6-13 - Console Name 15-16 - 2-digit year

4-Digit

5-12 - Console Name 13-16 - 4-digit year

```

CONMSTR1995178 1744241 -W A
MLG      1995178 1744241 SY1 R= SYSLOG  IEF196I IEF237I JES3 ALLOCATED TO SYSLOG03
MLG      1995178 1744242 SY1 R= SYSLOG  IEF196I IEF285I  +MASTER+.SYSLOG.JOB00001.D000000B.?          SYSOUT
          1995178 1744242 SY1 R= SYSLOG  IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS A
MLG      1995178 1744242 SY1 R= SYSLOG  0000000 SY1   1995178 17:44:24.23 SYSLOG  00000000 IEE042I SYSTEM LOG
MLG      1995178 1744242 SY1 R= SYSLOG  DATA SET INITIALIZED
S1       1995178 1744244 IAT700I JOB SYSLOG (JOB00001) IS ON WRITER PRT002(0002),RECORDS=147
LOG      1995178 1744244 IAT700I JOB SYSLOG (JOB00001) ON WRITER PRT002 (0002), DSN=
LOG      1995178 1744244 IAT700I +MASTER+.SYSLOG.JOB00001.D000000B.?, PURGED.
CONMSTR1995178 1744429 +I Q
CONMSTR1995178 1744429 IAT8674 JOB SYSLOG (JOB00001) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1995178 1744429 IAT8674 JOB IRRDPTAB (JOB00003) P=15 CL=A      OUTSERV(PENDING WTR)
CONMSTR1995178 1744429 IAT8674 JOB VTAMJ3 (JOB00009) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1995178 1744429 IAT8674 JOB DC (JOB00010) P=15 DC(ACTIVE)
CONMSTR1995178 1744430 IAT8674 JOB TCAS (JOB00011) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1995178 1744430 IAT8674 JOB SUPERU (JOB00012) P=15 CL=A      MAIN(EXECUTING-SY1)
CONMSTR1995178 1744488 +I Q SP=ALL
CONMSTR1995178 1744489 IAT8980 DRAINED HAS NO SPOOL DATA SETS
CONMSTR1995178 1744489 IAT8980 UNAVAIL HAS NO SPOOL DATA SETS
CONMSTR1995178 1744489 IAT8509 PART1 : 525 GRPS, 410 LEFT ( 78%); MIN 3%, MRG 5%, DEF, INIT

```

Figure 21. Sample Log Entries (with 4-digit-year dates)

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 installation exits
- JES3 and MVS commands.

This topic 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 document for detailed information about using those facilities. The following documents also contain information about message routing:

- *z/OS JES3 Migration*
- *z/OS JES3 Customization*
- *z/OS JES3 Commands*
- *z/OS MVS Initialization and Tuning Guide*
- *z/OS MVS Planning: Operations*

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 installation 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 before message routing begins. The following sections generically refer to messages issued by a WTO or WTOR macro as WTOs. Refer to *z/OS JES3 Customization* for 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 *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO* for information about using the WTO or WTOR macro.

Where Messages Can Go

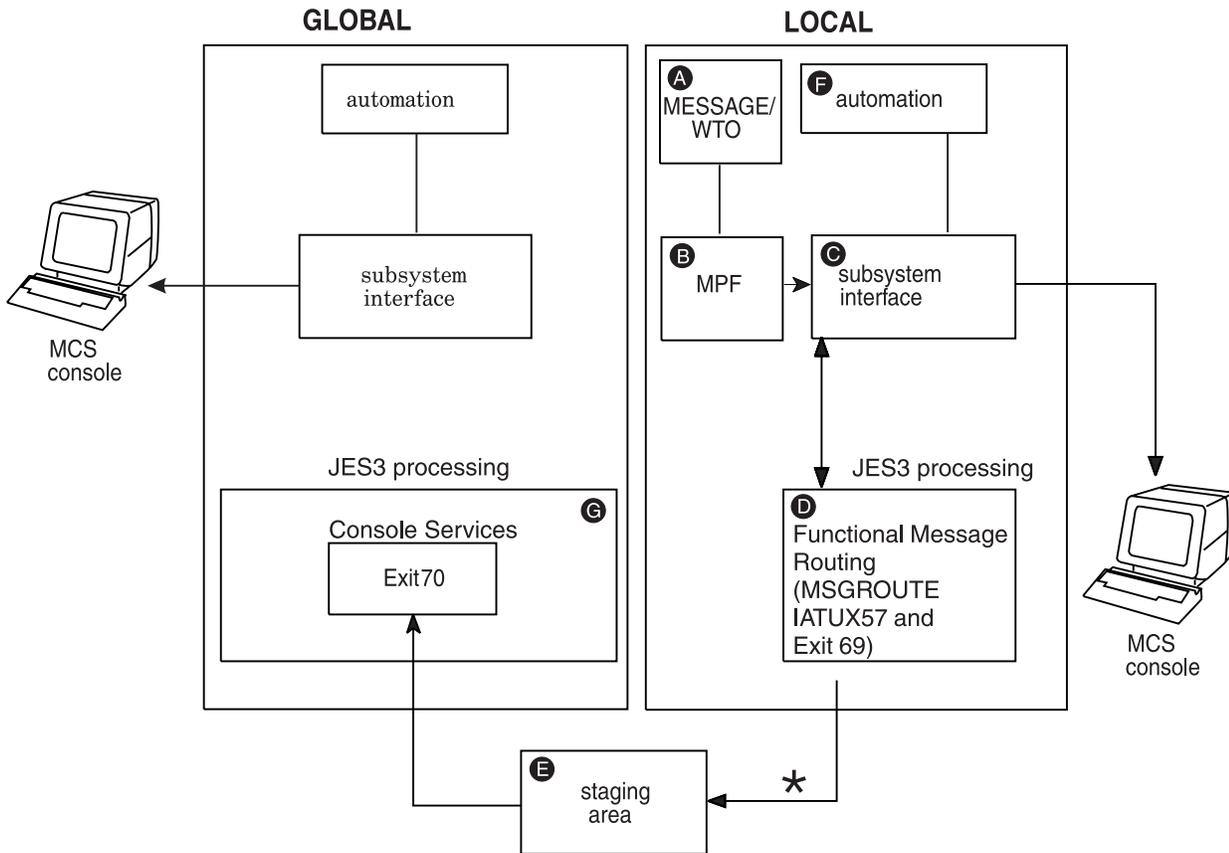
MVS and JES3 present messages in many places. Notice that the term *present* is used rather than *display*, because message destinations can be internal, such as disk logs and automation packages in addition to, or instead of external destinations, such as operator display consoles and hardcopy logs. Depending on the JES3/MVS routing algorithms and the routing decisions you make, the system may or may not present messages to:

- MCS consoles
- The hardcopy log
- The MVS message processing facility (MPF)
- The MVS 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. Figure 22 and Figure 23 illustrate the general path of a message. These figures include optional functions such as the MVS message processing facility (MPF), installation exits and automation packages. There are special exceptions to this path that are discussed in the following sections:

Path of Message Issued from a Local Processor



* This processing does not occur if the JES3 global and local are at a minimum release level of HJS7705.

Figure 22. 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**. JES3 calls exit 69 to process the message. The message is sent to the global processor in a staging area **E** if the JES3 global and local are below a minimum release level of HJS7705 and any one of the following conditions exist:
 - The message needs to be logged in the joblog data set (JESMSGLG).
 - The message requires special processing, for example, it is a message related to a JES-managed device.
 - Exit 69 indicates the message needs further processing.

Note: If the JES3 global and local are BELOW a minimum release level of HJS7705 and exit 69 indicates that the message should be sent to the global address space, the message is sent to exit 70. When the JES3

global and local are at a minimum release level of HJS7705, the message is NOT sent to the JES3 global, but processed on the user's address space.

4. The message is also placed back on the SSI for processing by an automation package **F** if present, or for display on MCS consoles.
5. In addition to JES3 processing **G**, MCS routes a message for presentation to consoles attached to other systems in the sysplex. MCS routes a message only if there is an active receiver (for example, a console eligible to receive the message) on the system.

The message is placed on the SSI on all systems that receive the message. MPF processing takes place only on the originating system. The GLOBMPF parameter on the JES3 CONSTD initialization statement can be used to indicate MPF should be invoked on the global processor for all messages routed to the global processor regardless of the originating system.

Path of a Message Issued from a Global Processor

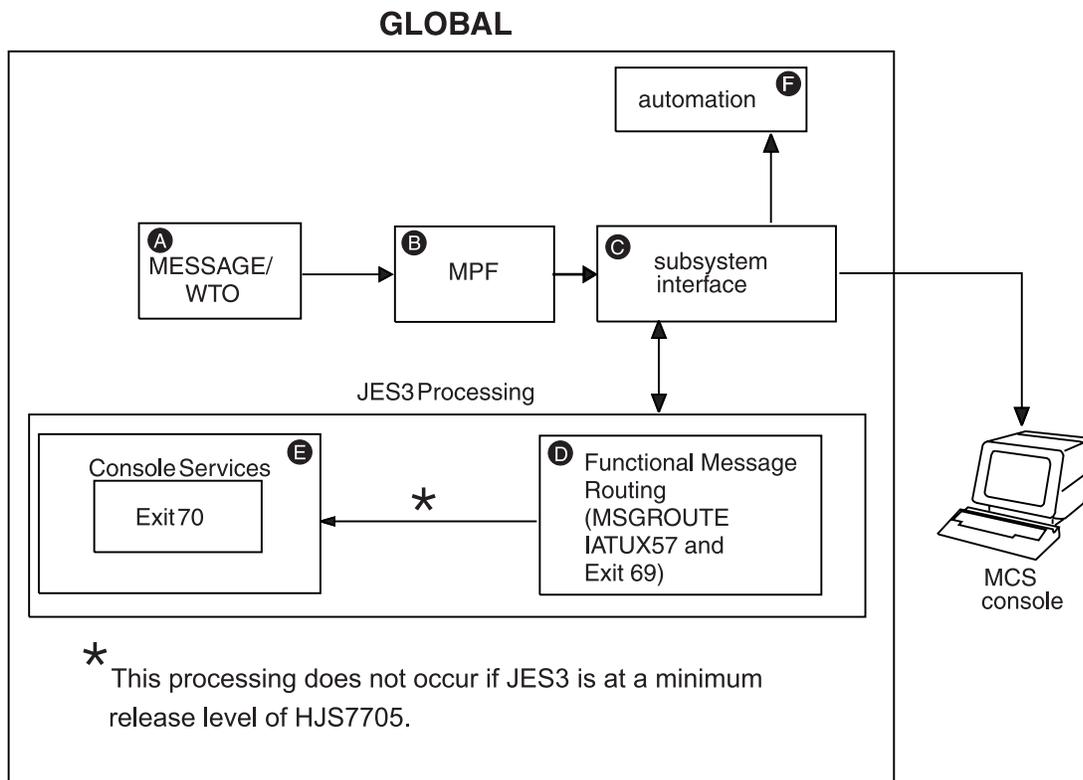


Figure 23. Simplified Path of a Message Issued from a Global Processor

Messages that originate on a global processor travel a path similar to that of a message issued on a local processor.

1. When a message originates **A** on the global 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**. JES3 calls exit 69 to

process the message. The message is sent to the JES3 global address space if the JES3 is at a release level below HJS7705 and any one of the following conditions exist:

- The message needs to be logged in the joblog data set (JESMSGLOG).
 - The message requires special processing, for example, it is a message related to a JES-managed device.
 - Exit 69 indicates the message needs further processing.
4. Depending upon the results of JES3's processing, the message can be packaged in a staging area and sent to the JES3 global address space **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 MCS consoles.
 5. If exit 69 indicates that the message should be sent to the global processor, the message is sent to exit 70.

Note: When the JES3 global and local are at a minimum release level of HJS7705, the message is NOT sent to the JES3 global, but processed on the user's address space.

6. In addition to JES3 processing, MCS routes a message for presentation to consoles attached to other systems in the sysplex. MCS routes a message only if there is an active receiver (for example, a console eligible to receive the message) on the system.

The message is placed on the SSI on all systems that receive the message. MPF processing takes place only on the originating system.

JES3 Destination Classes and MVS Routing Codes

JES3 uses 95 destination classes to route messages to MCS and RJP consoles. You can define which messages you want displayed on a JES3 RJP 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. Table 40 shows the 95 JES3 destination classes and their corresponding MVS routing codes.

Table 40. Valid Destination Classes and Their Corresponding Routing Codes

JES3 Destination Class	Equivalent MVS Routing Code/Function	Destination Class Purpose
ALL	Broadcast	Messages of general interest. These messages are received by MCS consoles receiving broadcast messages and JES3 RJP consoles which include DEST=ALL in their destinations.
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.

Table 40. Valid Destination Classes and Their Corresponding Routing Codes (continued)

JES3 Destination Class	Equivalent MVS Routing Code/Function	Destination Class Purpose
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-routing code mappings are: D1=43 D5=47 D9=51 D13=55 D17=59 D21=63 D2=44 D6=48 D10=52 D14=56 D18=60 D22=64 D3=45 D7=49 D11=53 D15=57 D19=61 D4=46 D8=50 D12=54 D16=58 D20=62
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-routing code mappings are: M1=65 M7=71 M13=77 M18=82 M23=87 M28=92 M2=66 M8=72 M14=78 M19=83 M24=88 M29=93 M3=67 M9=73 M15=79 M20=84 M25=89 M30=94 M4=68 M10=74 M16=80 M21=85 M26=90 M31=95 M5=69 M11=75 M17=81 M22=86 M27=91 M32=96 M6=70 M12=76
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 devicerelated messages. The exact JES3 destination class-routing code mappings are: S1=97 S8=104 S15=111 S22=118 S29=125 S2=98 S9=105 S16=112 S23=119 S30=126 S3=99 S10=106 S17=113 S24=120 S31=127 S4=100 S11=107 S18=114 S25=121 S32=128 S5=101 S12=108 S19=115 S26=122 S6=102 S13=109 S20=116 S27=123 S7=103 S14=110 S21=117 S28=124

The following destination classes are also valid for use on the RJP CONSOLE statement 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 IATxxx 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 routing information specified on the JES3 MESSAGE macro is converted to its MVS equivalent when the message becomes a WTO. MCS consoles on both global and local processors can display the message if you have defined them to receive the equivalent MVS routing parameters. JES3 converts that code back to its original destination class for display on JES3 RJP consoles.

If a message is issued that prohibits subsystem modification, then JES3 does not modify the message's routing information.

Refer to *z/OS JES3 Customization* for information about using macros to issue messages.

Using MSGROUTE to Control Message Routing

If an MVS-issued message has multiple routing codes (excluding MVS routing code 11), JES3 invokes installation exit IATUX57 to select a single code so that JES3 can translate that code to a single destination class for display on MCS consoles. 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 and 128.

For additional information about installation exit IATUX57 and the default selection algorithm, see *z/OS JES3 Customization*.

You can use the JES3 MSGROUTE initialization statement to control the routing of subsystem-modifiable messages.

- To local and global MCS consoles
- To only the hardcopy 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 consoles that receive JES3 security messages, you could use the MSGROUTE statement to map routing code 9 to destination class SEC. Afterwards, all consoles that receive JES3 security messages would also display MVS security messages.

You can define one MSGROUTE statement for each processor in your installation.

Use of MSGROUTE Statement

```
MSGROUTE,MAIN=SY1,1=(M1,CN1,J),2=(M28,CN1),3=(S1,,J),8=(TP),9=(, ,J)
```

The MSGROUTE statement is optional. If you don't define a MSGROUTE statement, JES3 will not perform message routing processing for the messages issued from that system. You can choose not to define a MSGROUTE statement on one or more processors.

Note: The message's original routing information is unchanged if its routing code(s) are omitted from the MSGROUTE statement.

Processing if SY1 Is either the Local or Global:

- Messages assigned routing code 1 will be displayed on console CN1 and all consoles defined to receive the routing code equivalent of M1 (which is 65). The J parameter causes the routing code equivalent of the destination class to be used in place of the message's original routing information.
- Messages assigned routing code 2 will be displayed on the console name CN1, all consoles receiving the original routing code(s), and on all console defined to receive the routing code equivalent of destination class M28 (routing code 92).
- Message assigned routing code 3 will be displayed on all consoles defined to receive the routing code equivalent of destination class S1 (routing code 97).
- Messages assigned routing code 8 will be displayed on all consoles receiving the original routing codes as well as consoles receiving the routing code equivalent of destination class TP (routing code 8).
- Messages assigned routing code 9 are sent to the system log and displayed on all consoles receiving the hardcopy destination.
- Messages assigned to routing codes that you omit from the MSGROUTE statement are not modified by JES3.

Coding Rules for the MSGROUTE Statement

- The routing information is unchanged for messages assigned routing codes which are omitted from the MSGROUTE statement.
- Specify the J parameter in order for the routing information specified on the MSGROUTE statement to be used instead of the message's original routing information.
- 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 the hardcopy log. 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 IATxxx 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:

Message Routing Specified by MPF Installation Exit

If an MPF exit has indicated that the primary subsystem cannot alter message routing for this message, then it is the users responsibility to ensure that the message will be processed. Message routing will be affected as follows:

- **On the Local or Global -**

If an MPF exit indicates JES3 cannot alter the routing of this message, routing codes on this processor will not be changed.

Destination information passed to JES3 will still be affected by JES3 routing functions.

- **Logging -**

The message will be put in the hardcopy log with the routing information used by JES.

Action Messages that Must Be Displayed

MCS consoles with the undeliverable (UD) attribute display all action messages that no other console displays. For example, if MVS issues an action message with a routing code that you haven't defined to any console, then these consoles display that message.

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 installation 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

IEAVMXIT, an MPF exit, or a statement in the MPFLSTxx member of SYS1.PARMLIB can be used to suppress the display and logging of a message. 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 for display or logging. This type of message is often referred to as a *deleted message*. These messages are also written to the JES3 JESMSGGLG data set. No additional message routing is performed.

No Routing Specified

JES3 considers a message to have no routing information if all the following conditions are true:

- No routing codes are specified.
- No console ID is specified.
- No MSGTYPE option is specified.
- BROADCAST is not specified.
- HARDCOPY is not specified.
- No job status descriptor code is specified.

If a message is issued without any routing information, or you use an MPF exit to remove all routing information, and no default routing is set by MCS, JES3 defaults the routing to the destination class or routing code that was specified (or defaulted) on the issuing system's MAINPROC initialization statement.

Messages that Originate from Functional Subsystems

Use the JUNIT= keyword of the JES3 DEVICE initialization statement to assign a message destination to messages that originate from an output writer functional subsystem (FSS). The message destination that you specify overrides the message destination selected by other message processing (such as the JES3 MSGROUTE initialization statement for modifiable messages, and default processing for non-modifiable messages).

For messages that originate from a C/I FSS address space, JES3 uses a default message destination of "JES" unless other routing information is specified. This default can be bypassed if you specify the CIFSS=MSGRROUTE parameter on the CONSTD initialization statement. See for more information on the CONSTD initialization statement.

Broadcast Messages

Messages assigned the MVS *broadcast* function are available for display on MCS consoles and all JES3 RJP consoles with destination class ALL.

Suppressed Messages

You can use the MVS message processing facility (MPF) to suppress the display of messages. These messages are routed only to the hardcopy log. 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 message destination 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 message destination 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. JES3 does not change the original routing for the message.

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 hardcopy log. Otherwise, the message is treated as a device-related message which is described below.

Device-Related Messages: The message destination 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 message destination, 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 hardcopy log. The original routing of messages about command failures is not affected.

Main Processor Messages: Messages about a specific main are sent to the message destination you specify on the MDEST= keyword of the JES3 MAINPROC

initialization statement. If you do not specify a message destination, JES3 routes messages to consoles receiving destination class M1 or its equivalent MVS routing code (routing code 65).

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 *z/OS JES3 Diagnosis* for information about invoking and using GTF in your JES3 installation. You can also refer to *z/OS MVS Diagnosis: Tools and Service Aids* for information about GTF.

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 two ways you can suppress message displays:

- Message processing facility (MPF)
- MVS installation exit IEAVMXIT

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 the MVS installation exit IEAVMXIT to suppress messages. This installation exit can run on any or all processors in the installation, and can be used to change or suppress message displays.

The system records each message that you suppress in the hardcopy log, and flags these messages to help you identify them.

Specifying Messages for Display Suppression

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 *z/OS MVS Initialization and Tuning Guide* and *z/OS MVS Initialization and Tuning Reference*.

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 JES3 DLOG. JES3 uses the flag defined in MPFLSTxx member on the global processor to flag suppressed messages on all systems. For instructions on how to specify a flag, see *z/OS MVS Initialization and Tuning Guide*. If you do not specify a flag, the system uses an ampersand sign (&) to flag the messages.

Controlling Message Suppression

To control the message processing facility, issue the MVS SET MPF=xx command. Thereafter, when that processor issues a message that you specified in MPFLSTxx, MPF suppresses the message display.

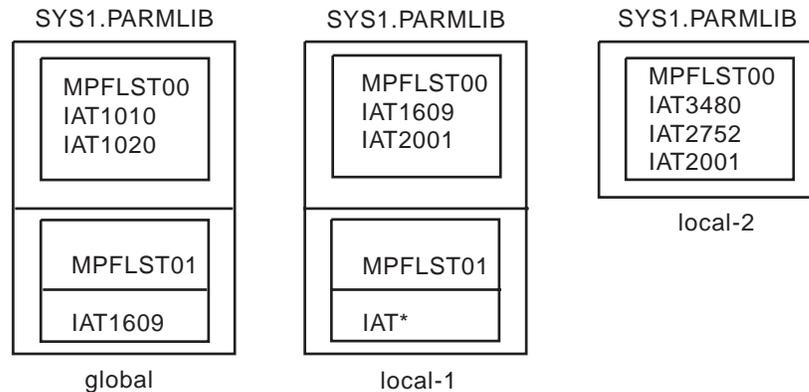
To terminate control of message suppression, enter the MVS SET MPF=NO command from an MCS console.

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 continue to display the message unless you use MPF on the originating processor to suppress the message.

Examples 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.

These examples assume that the JES3 GLOBMPF option is not being used and that MPF processing only occurs on the system from which a message is issued.

Example 1: Controlling message suppression from a local processor.

- | | | |
|-----------|------------|---|
| (local-1) | SET MPF=00 | <ul style="list-style-type: none"> • 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 | <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • When the global issues message IAT1609, the system suppresses the message display. However when local-2 issues message IAT1609, the message is eligible for display on all systems including the global since MPF processing only occurs on the originating system (local-2). • The system continues to suppress the display of all JES3 messages issued by local-1.
(local-1)	SET MPF=NO	<ul style="list-style-type: none"> • When the global issues message IAT1609, the system suppresses the message display. However, when local-1 or local-2 issues message IAT1609, the message is eligible for display on all systems including the global. The system displays all other messages.
(global)	SET MPF=NO	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • The system now displays all messages.
(local-1)	SET MPF=00	<ul style="list-style-type: none"> • 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 *z/OS MVS Initialization and Tuning Guide*. The contents of an MPFLSTxx member that controls which messages are sent to an automation program might be as follows:

```
.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, 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 enter all JES3 commands except console-specific commands (such as *FREE).

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, *CANCEL,JMF
- *CALL, *START, and *CANCEL,VARYL
- *DUMP
- *TRACE
- *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 220 or see *z/OS MVS Initialization and Tuning Guide* for information about using MPF.

Order of Command Execution

Many factors control the order of command execution. JES3 cannot guarantee the order in which commands are executed for the following reasons:

- Multiple commands entered via a single command issuance (eg. stacked) will be parsed from the end until a command delimiter (eg. semi-colon) is found and re-issued to be processed by automation and MPF/message exits individually.
- JES3 FCT dispatching priorities will cause some commands to be executed before others.
- Console appendages written for each Dynamic Support Program (DSP) may process commands in any order.
- Multiple FCTs may exist to process a certain type of command; this affects the order commands are executed. For example, multiple MODOSFCT FCTs can be active at the same time processing *MODIFY,U commands occurs.
- JES3 or MVS command exit processing may affect the time any single command takes to execute.
- Depending on if any JES3 spool I/O is required to process the command will affect the time any single command takes to execute. Many JES3 commands result in calls to the Work-To-Do-Driver (WTDDRVR) FCT, which processes commands which require JES3 spool I/O.
- Long running commands based on job queue searches and JES3 spool I/O which may be necessary to access job information should not delay other critical JES3 command processing.
- Non-JES3 commands processed by MVS or other subsystems will execute under other programs beyond JES3's control. JES3 is not designed to synchronize the order of command execution between JES3 and non-JES3 commands.

IBM recommends that customers use message-based automation (for example, MPF command exits) if feasible, or use automation to setup time delays as appropriate between critical commands, whenever order of command execution is important to their operations. In these cases, automation should be used to ensure any single command has finished execution before issuing the next command in any critical sequence.

Part 8. Defining and Managing JES3 Resources

Chapter 8. Defining and Managing JES3

Resources	227	Adding Devices	253
JES3 Data Sets	227	Changing Devices	253
Allocating JES3 Data Sets	227	Deleting Devices	253
Allocating the JES3 Checkpoint Data Set(s)	228	Allocating an I/O Device to JES3	254
Determining the Size and Placement of the		Grouping I/O Devices	254
Checkpoint Data Set(s)	228	Specifying Device Fencing	255
Defining the Maximum Number of Jobs for		Dynamically Reconfiguring I/O Devices	255
Your Complex	229	Volumes	256
Controlling the Number of Jobs Allowed in a		How Resource Definition Affects JES3 Resource	
JES3 Complex:	229	Management	257
Reasons for Increasing the Number of Jobs in		I/O Device Management	257
Your Complex	230	Data Set Access	258
How to Determine the Size of the JCT Data		Data Set Integrity	259
Set	232	Volume Management	262
Using Performance Measurement Tools to		Dynamic Allocation	263
Tune JCT Access	235		
JCT Utility (IATUTJCT)	237		
Introduction	237		
Migrating From Releases Lower Than			
HJS6606	237		
Falling Back from Releases HJS6606 (or			
Above) to Releases HJS6605 (or Below)	238		
Copying a JCT Data Set Using the IATUTJCT			
Utility	238		
Running the IATUTJCT Utility	238		
Testing IATUTJCT Processing	242		
Migrating Using IATUTJCT	242		
Identifying Resident Data Sets	242		
Using System Catalogs	243		
I/O Devices	243		
Defining I/O Devices to JES3	243		
JES3 Devices	244		
Execution Devices	245		
Shared Devices	245		
Defining Process Modes	245		
Using Process Mode Defaults	246		
Running a Printer Under an Output Writer			
Functional Subsystem	246		
Defining PSF FSSs and FSAs for Printers	246		
Defining an Output Writer Functional			
Subsystem	247		
Determining the Maximum Number of			
Printers for an Output Writer FSS	248		
Defining the 3800 Model 3 Printer as a Line			
Printer	249		
Defining Channel Attached Printers			
Controlled by FSSs	249		
Defining Non-Channel Attached Printers	250		
Using the Infoprint Server for z/OS Printer			
Inventory	251		
Defining a Device Configuration	251		
Defining a Range of Devices	251		
Defining a Device To All Processors			
Simultaneously	252		
Maintaining a Device Configuration Using HCD	253		

Chapter 8. Defining and Managing JES3 Resources

You are responsible for defining and allocating resources for JES3. Resources include data sets, libraries, catalogs, devices, processors, volumes, and main storage. To define and allocate resources use JES3 initialization statements and the JES3 cataloged procedure.

For a discussion about defining and managing mains and storage, see Chapter 9, “Defining and Managing JES3 Mains and Storage” on page 267.

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 you want JES3 to use these partitions. Chapter 6, “Defining and Managing Spool Data Sets” on page 181 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 the location of data sets, JES3 will not have to search for them. You can also improve performance by the way you configure system catalogs.

You should mark JES3 data sets 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:

- STEPLIB
- CHPNT or CHPNT2
- JES3IN

If you dynamically allocate a data set, do not include a DD statement for that data set. JES3 assigns a disposition of SHR to dynamically allocated data sets.

The JES3 global dynamically allocates all of the data sets specified on the DYNALLOC initialization statement. The JES3 locals dynamically allocate the following subset of data sets specified on the DYNALLOC statements:

- JES3JCT (JCT data set)
- JESABEND
- SYSABEND
- SYSUDUMP

If a local processor becomes the global processor through dynamic system interchange (DSI), JES3 dynamically allocates the remaining DYNALLOC statements:

- IATPLBxx (procedure library data sets)
- JES3DRDS (disk reader data set)
- JES3SNAP
- Any user 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.

One or both JES3 checkpoint data sets must be defined in the JES3 start procedure using ddnames of CHPNT and/or CHPNT2. 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” on page 162, in Chapter 5, “Defining and Managing C/I Service” on page 159.) 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 *z/OS JES3 Diagnosis* .

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 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.

Defining the Maximum Number of Jobs for Your Complex

You can allow as many as 999,999 jobs to exist in your JES3 complex at the same time. However, JES3 limits the maximum number of jobs that can be active concurrently by choosing the smallest of the following:

- The value you specify on the job limit parameter on the JOBNO= keyword of the OPTIONS initialization statement
- The range of job numbers that you define on the JOBNO= keyword of the OPTIONS initialization statement
- The number of entries in the job control table (JCT)

Use the JOBNO= keyword of the OPTIONS initialization statement to specify the maximum number of jobs for your system.

Controlling the Number of Jobs Allowed in a JES3 Complex:

The number of jobs concurrently allowed in a JES3 complex depends on the size of the JCT data set and JOBNO= keyword parameters specified on the OPTIONS statement in the JES3 initialization. The JOBNO= keyword has the following format.

```
OPTIONS, JOBNO=(low_job_number, high_job_number, max_number_jobs)
```

The job number range is specified by **low_job_number** and **high_job_number** values. The number of concurrent jobs allowed in the JES3 complex is the minimum of the following values:

- Number of records in the JCT data set
- Maximum number of jobs from the JOBNO= keyword
- Difference between low and high job numbers from the JOBNO= keyword

By specifying the OPTIONS initialization statement and the keyword JOBNO=, you can control the number of jobs in the JES3 complex. Below is a progressive example showing how you can decide on this number.

- To allow the entire range of job numbers from 1 to 32767 to be assigned to jobs, but limit the maximum number jobs allowed in the system to 10000, you specify the following:

```
OPTIONS, JOBNO=(1, 32767, 10000)
```

- To increase the range of job numbers assigned to the maximum. You do a JES3 warm start or hot start with refresh with the following specification:
OPTIONS,JOBNO=(1,999999,10000)

Note: The number of jobs allowed in the system has not increased, but the reuse of job numbers decreases in frequency.

- To support 25000 JCT entries, you allocate a JCT large enough to hold 25000 entries, raise the number of jobs allowed in the system to a number less than 25000 (ex: 15000) and increase the job number range to the maximum. You do a JES3 warm start or hot start with refresh with the following specification:
OPTIONS,JOBNO=(1,999999,15000)
- To support 40000 concurrent job in the system, you must increase the size of the JCT to be large enough to support 40000 jobs. You can cold start to use the larger JCT data set, in which case you must save jobs currently on JES3 spool and restore the jobs after the cold start, or you can avoid the cold start by running the IATUTJCT utility.

Reasons for Increasing the Number of Jobs in Your Complex

There are many reasons for increasing how many job numbers JES3 can process in a complex. The following discussion illustrates various installation reasons for increasing job numbers along with solutions.

Reason 1a

The number of jobs in the JES3 complex is currently limited by the size of the JCT data set. You do not need to support a larger number of jobs concurrently in the complex, but would like to increase the job number range.

Solution 1a

Increase the job number range by performing a JES3 warm start or hot start with refresh with an updated JES3 initialization stream. For example, if your "old" values span job number values from 1 to 30000, and you want to increase the range from 1 to 55000, you update the your initialization stream as follows:

Current Initialization Stream

```
OPTIONS,JOBNO=(1,30000),
```

New Initialization Stream

```
OPTIONS,JOBNO=(1,55000),
```

Notes:

1. Do not change to a job number range that does not include the current job number range. Jobs with numbers not in the new range will be discarded. During the warm start or hot start with refresh, JES3 prompts the operator for each and every job whose current job number is not in the new job number range. Furthermore, for each of these lost jobs that was active on the main, the processor that the job was running on must be IPLed.
2. JES3 saves 100 job numbers and 64 JCT entries for situations involving job number exhaustion and certain error recovery situations. The effective job number range is reduced by 164 and therefore the JES3 job number range should be at least 300.

Reason 1b

The number of jobs in the JES3 complex is currently limited by the third parameter on the JOBNO keyword, not the size of the JCT data set. You do not need to support a larger number of jobs concurrently in the complex but would like to increase the job number range.

Solution 1b

Increase the job number range by performing a JES3 warm start or hot start with refresh with the JES3 initialization stream updated as follows.

Current Initialization Stream

```
OPTIONS,JOBNO=(1,20000,10000),
```

New Initialization Stream

```
OPTIONS,JOBNO=(1,30000,10000),
```

The same rules apply for the job number range overlaps as in Situation 1a. The maximum number of jobs allowed in the system is controlled by the third JOBNO= keyword value.

Reason 2

The current JCT data set contains 999999 JCT entries (or some number that is adequate to meet the installation's needs), and you would like to increase the job number range and the number of jobs allowed in complex.

Solution 2

This is similar to Reason 1b. You only need to change the initialization stream to allow an increased number of job numbers to be used and to increase the concurrent number of jobs in the JES3 complex. You then perform a JES3 warm start or hot start with refresh with the OPTIONS statement updated to set the job number range appropriately. For example,

Current Initialization Stream

```
OPTIONS,JOBNO=(1,30000,10000),
```

New Initialization Stream

```
OPTIONS,JOBNO=(1,65000,65000),
```

Reason 3a:

The current JCT data set is not large enough to contain the number of desired JCT entries (concurrent jobs on the spool), and you would like to increase the job number range and the number of jobs allowed in complex. A cold start of the JES3 complex is acceptable.

Solution 3a

To increase the concurrent number of jobs and the job number range, you must replace the JCT data set with one of sufficient size to accommodate the desired number of concurrent jobs. You then change the OPTIONS initialization statement to allow an increased range and number of jobs.

1. Dump the jobs on spool to tape using the DUMP JOB (DJ) utility. Refer to *z/OS JES3 Commands* for the information on using DUMP JOB.

Note: This invocation of DUMP JOB does not involve translation as the spool control blocks are compatible. (You will not have to specify TRANS=YES on the *CALL,DJ command.)

2. Allocate a new JES3 JCT data set of the appropriate size. See "How to Determine the Size of the JCT Data Set" on page 232 to determine the size of the data set to be allocated for the new JCT data set.

Note: When allocating a new JCT data set, the allocation must be in cylinders and as a single extent data set.

3. Update the cataloged JES3 start procedure or the JES3 initialization stream to point to the new JCT data set.

- Restart the JES3 complex with a COLD START.

Note: No reformat of the spool data sets is necessary with this cold start. If the spool data sets are defined with a TRACK statement, they are not formatted. (When JES3 formats the spool data sets, it does so one data set at a time, which can be a lengthy process.)

- Use the DUMP JOB utility to reload the jobs to spool (unless Step #1 was omitted).

Using DUMP JOB to unload and reload the JES3 spool allows you to have a new job number range that does not include the old job number range. When jobs are reloaded to spool, the old job number is used if available otherwise a new job number is assigned. Unlike Reasons 1 and 2, there is no need for the new job number range to include the old job number range.

Reason 3b

The current JCT data set is not large enough to contain the number of desired JCT entries (concurrent jobs on spool) and you would like to increase the job number range and the number of jobs allowed in complex. A cold start of the JES3 complex is NOT acceptable.

Solution 3b

Create a new JCT data set large enough for the desired number of concurrent jobs and change the OPTIONS initialization statement to allow for an increase in the job range or number of jobs, while including the current job number range. Run IATUTJCT to copy the contents of the current JCT data set to the new one, change the JES3 procedure to point to the new JCT data set, and perform a warm start or hot start with refresh with the new job number range.

How to Determine the Size of the JCT Data Set

The JCT data set contains information about the status of jobs in the system. You must make the size of the JCT data set large enough to accommodate the maximum number of jobs that can be in your JES3 complex simultaneously. If the data set is larger than required, JES3 uses only that portion needed to hold the maximum number of jobs (the remainder of the JCT is used for write error recovery). However, the larger the JCT data set, the longer JES3 initialization takes. For optimum performance, minimize the size of the JCT data set as much as possible.

- Calculate the size, in bytes, of the JCT entry within the JCT data set:

$$xxx = (4 \times \text{max SEs}) + zzz + 28 \text{ bytes (prefix)}$$

where:

- xxx is the size, in bytes, of a single JCT.
- 4 is the size of one JCT scheduler element in the JCT.
- Max SEs is the maximum number of scheduler elements as specified on the SE parameter of the OPTIONS initialization statement (10 through 90 is the valid range - 10 is the default).
- zzz is the size, in bytes, of the fixed JCT. Field JCTFSIZE in the IATYJCT macro defines this value. If you have modified the size of your JCT by making changes to IATYJCT, IATYCNDB or IATYFDB, use the value (from an assembled listing) in JCTFSIZE as the zzz value in the formula. If you have not modified the size of the JCT, the value of JCTFSIZE is 668 bytes. If you are migrating from a JES3 release below HJS6606, see *z/OS JES3 Migration* and APAR OW30849 for a discussion on migrating the JCT data set.

- 28 is the size, in bytes, of the SRF header prefix mapped when using the IATYSRF macro.
2. Determine the number of JCTs per cylinder. For IBM 3380 and 3390 devices, use the following documents to determine the number of JCTs that will fit on a track and cylinder:
 - *Using IBM 3390 Direct Access Storage in an MVS Environment*
 - *3380 Direct Access Storage Introduction*

Each document contains a table matrix in an appendix that shows how many physical blocks fit on a track and a cylinder for various block sizes (since the JCT records are unblocked, the logical record size IS the block size).

Note: JCT records are without keys. A portion of the information from *Using IBM 3390 Direct Access Storage in an MVS Environment*, Appendix B - Figure 71 "Records without Keys" is relevant to JCT records and is included below:

Table 41. Physical Blocks per Track and Cylinder

Data Length Range		Percent	Max Track Capacity (*)		Max Cylinder Capacity (*)	
Min	Max	Used (*)	Records	Bytes	Records	Bytes
1019	1086	63.2	33	35838	495	537570
985	1018	61.1	34	34612	510	519180
951	984	60.8	35	34440	525	516600
889	950	60.4	36	34200	540	513000
855	888	58.0	37	32856	555	492840
821	854	57.3	38	32452	570	486780
787	820	56.4	39	31980	585	479700
753	786	55.5	40	31440	600	471600
>719	752	54.4	41	30832	615	462480
691	718	53.2	42	30156	630	452340
657	690	52.4	43	29670	645	445050
623	656	50.9	44	28864	660	432960
>589	622	49.4	45	27990	675	419850

Table 41. Physical Blocks per Track and Cylinder (continued)

Data Length Range		Percent	Max Track Capacity (*)		Max Cylinder Capacity (*)	
Min	Max	Used (*)	Records	Bytes	Records	Bytes
<p>Note: (*) Calculations are made using the maximum size record in a range.</p> <p>The column "Maximum Track Capacity Records" will give how many JCTs per track a 3390 can contain.</p> <p>The column "Maximum Cylinder Capacity Records" will give how many JCTs per cylinder a 3390 can contain.</p> <p>The two arrows ">" in the "Data Length Range" column above highlight the data length ranges that apply to the pre-HJS6606 JCT and the HJS6606 / post-HJS6606 IBM-standard size JCT records. For example,</p> <ul style="list-style-type: none"> • The standard pre-HJS6606 JCT size is 620 bytes (from the formula in step 1 above) and falls into the 589 - 622 data length range, so a 3390 track can contain 45 JCTs and a 3390 cylinder can contain 675 JCTs. • The standard HJS6606 / post-HJS6606 JCT size is 736 bytes (also derived from the formula in step 1) and falls into the 719 - 752 data length range, so a 3390 track / cylinder can contain "41" and "615" JCTs. • The size ranges shown in this table will cover most of the likely JCT record sizes. For other JCT sizes, 3380 DASD, and other devices, consult the appropriate documents. 						

3. Determine the required size of the JCT data set:

$$\text{JCT data set size in cylinders} = \frac{(64 + \text{max \# of jobs})}{\text{number of JCTs per cylinder}}$$

where:

- The JCT data set size in cylinders must be incremented to the next whole number if the result is not a whole number.
- 64 is the number of JCTs JES3 reserves for write error recovery.
- The max # of jobs is the maximum number of jobs that can simultaneously be in your JES3 complex.
- The number of JCTs per cylinder (as explained in the device specific table - of step #2 above).

Notes:

1. JES3 reserves 64 or more JCT records in the JCT data set for "write" error recovery unless the number of records in the JCT data set is less than or equal to 64. If your JCT contains 64000 records and you specify JOBNO=(1,64000,64000) on the OPTIONS statement, the number of jobs in the system is limited to $64000 - 64 = 63936$. If you wanted to have 64,000 jobs in the system and specified JOBNO=(1, 64000,64000), the JCT data set would have to contain 64064 records to allow for the 64 records needed for "write" error recovery. On the other hand, if your JCT contains 64000 records and you specify JOBNO=(1,63000,63000) on the OPTIONS statement, the job number range is limited to 63000, and 1000 JCT records are reserved for "write" error recovery ($64000 - 63000 = 1000$).
2. The JCT data set should be allocated on a cylinder boundary.
3. In order to change the size and/or placement of the JCT data set, you must either perform a cold start or use the IATUTJCT utility. Refer to "JCT Utility (IATUTJCT)" on page 237 for more information on changing your JCT data set.

4. If you have not modified the length of the JCT by changes to IATYJCT, IATYCNDB, or IATYFDB, and you use the default number of SE entries (10), then a JCT entry is 736 bytes long.
 - 38 entries per track or 570 entries per cylinder on a 3380 device
 - 41 entries per track or 615 entries per cylinder on a 3390 device

Example

A HJS6606 customer wants the capability to process 20,000 JES3 jobs concurrently. The size of the standard IBM-supplied JCT has not been modified by making changes to IATYJCT, IATYCNDB, or IATYFDB. The customer has specified SE=15 in the OPTIONS initialization statement. The customer uses IBM 3390 Direct Access Devices for his JCT data set.

1. Calculate the size, in bytes, of the JCT entry within the JCT data set:

$$\begin{aligned} \text{xxx} &= (4 \times \text{max SEs}) + \text{zzz} + 28 \text{ bytes (prefix)} \\ 756 &= (4 \times 15) + 668 + 28 \end{aligned}$$

2. Determine the number of JCTs per cylinder:

Using the 3390 capacity table above, a 756 byte record falls within the 753 - 786 data length range, and 600 of these records will fit in a 3390 cylinder.

3. Determine the required size of the JCT data set:

$$\begin{array}{rcl} \text{JCT data set} & & (64 + \text{max \# of jobs}) \\ \text{size} & = & \text{-----} \\ \text{in cylinders} & & \text{number of JCTs per cylinder} \end{array}$$

$$33.44 = (64 + 20,000) / 600$$

Since the result is not a whole number, it must be incremented to the next whole number, 34.

Conclusion

This customer will require a JCT data set size of 34 cylinders.

You can allocate the JCT data set by including the //JES3JCT DD statement in the JES3 cataloged start procedure or by using a DYNALLOC initialization statement (see Figure 1 on page 8 for a sample of the JES3 cataloged procedure).

Using Performance Measurement Tools to Tune JCT Access

During initialization, JES3 places a copy of the JES3JCT data set on both JES3 spool and into the JCT data space. A *data space* is a range of up to 2 gigabytes of contiguous virtual storage. Unlike an address space, a data space can hold only data; it does not allow code to execute. See *z/OS MVS Programming: Extended Addressability Guide* for a description of data spaces and information about how to use them.

During normal operation, JES3 updates the JCT in the data space and on spool. However, the data space is subject to paging and MVS may migrate portions of the JCT to MVS paging data sets.

Excessive paging of the JCT data space can impact JES3 performance because JES3 must wait while MVS moves pages of the JCT back into central storage. JES3 may also have to wait additional time if MVS is servicing other paging requests ahead of JES3's.

You can optimize performance and avoid performance bottlenecks by minimizing the amount of time it takes JES3 to read the JCT. Use the following formula to determine the amount of time it takes to read the JCT in your installation:

$$(5 * x) * ((1 + (y * .5)) * 25) = \text{Milliseconds to read JCT}$$

where:

- 'x' represents the percentage of time that a JCT page was not in central storage during a read request. See the **JQE/JCT Access Method Report** section of your JMF report to determine the percentage of time a page was not in central storage for a read request.
- 'y' represents the average number of page requests queued ahead of a JCT page request. See the **System Paging Space Data** portion of your RMF report to determine the average number of page requests queued.

The total amount of time required to read the JCT for each job should not exceed 125 milliseconds. You can take the following actions to reduce JCT access time:

- Reduce the JES3 paging rate.
- Increase the amount of central or expanded storage.
- Install additional paging devices to reduce I/O contention.
- Use SRM storage isolation for the JES3 address space. See *z/OS MVS Initialization and Tuning Guide* for information about how to use storage isolation.
- Increase the intervals at which a changed or unchanged page that is to be paged out will be sent to expanded storage. Use the ESCTPOC and ESCTPOU keywords in the IEAOPTxx member of the MVS SYS1.PARMLIB data set to modify these intervals. See *z/OS MVS Initialization and Tuning Guide* for information about updating the MVS SYS1.PARMLIB data set.

The acceptable percentage of read page faults varies from installation to installation. Use the following formula to determine an acceptable percentage of JCT read page faults for your installation:

$$\frac{125}{((1 + (y * .5)) * 25) * 5} = \text{Acceptable \% of read page faults}$$

Use the **JQE/JCT Access Method Report** section of your JMF report to determine the percentage of time that a page was not in central storage for a read request. See *z/OS JES3 Diagnosis* for an example of a JMF report.

Notes:

1. The percentage of read requests when a page is not in central storage does not always imply that MVS has migrated the page to an external storage device. The value also includes page faults where pages may have been migrated to expanded storage.
2. If you use storage isolation to reduce paging, SRM considers the JCT data space as part of the JES3 working set and may increase the working set size. See the **JES3 Working Set Size** section of your JMF report to determine the amount of virtual storage that JES3 uses in your installation. If the JES3 paging rate increases after the JCT data space is enabled, consider increasing the JES3 working set size and/or add additional central or expanded storage.
3. The percentage of time that the JCT is not in central storage for a write request should be very small. The largest user of the JCT is the job segment scheduler (JSS) which reads the JCT, performs a small amount of I/O, and writes the JCT to spool. A high percentage would indicate more serious problems, such as an unacceptable paging rate.

4. Storage fragmentation and excessive JES3 storage requirements can cause excessive paging of the JES3 address space. You can obtain the paging rate for your installation using the **JES3 Paging Counts During JMF Monitoring** portion of your JMF report. If your paging rate is very high, consider the following actions:
 - Reduce the size of the console buffer text pool using the CONSBUF= keyword of the JES3 CONSTD initialization statement.
 - Reduce the size of the JSAM buffer pool using the PAGES= keyword on the JES3 BUFFER initialization statement; however, make the size of the JSAM buffer pool large enough to accommodate the amount of work in your system during peak periods.
 - Release held output. Releasing held output allows JES3 to purge jobs, which can free large chunks of storage.

JCT Utility (IATUTJCT)

Introduction

For various reasons you might need to replace an existing JCT data set. In order to allow a JCT data set to be copied without a cold start, JES3 provides a program called the JCT utility, or IATUTJCT.

IATUTJCT can be used to satisfy the following requirements:

- You need to allocate a larger JCT data set in order to provide capacity for more jobs in the JES3 job queue.
- You need to move your JCT data set to a new volume. The new volume can be the same device type as the one where your JCT data set currently resides, or it can be a different one.
- You are migrating from a JES3 release prior to HJS6606 to HJS6606 or higher (see “Migrating From Releases Lower Than HJS6606”).
- You are falling back from a JES3 release of HJS6606 (or higher) to a JES3 release of HJS6605 (or lower) (see “Falling Back from Releases HJS6606 (or Above) to Releases HJS6605 (or Below)” on page 238

Because the contents of the JCT data set are dependent on the device type and the location of the JCT data set on its volume, an existing JCT data set cannot be copied by a utility such as IEBGENER. Furthermore, there is information about the JCT data set in the checkpoint data set which cannot be copied.

IATUTJCT copies JCT entries from one JCT data set to another, and updates the device-dependent information at the same time. IATUTJCT also copies the JCT information that is kept in the checkpoint data set.

Migrating From Releases Lower Than HJS6606

In release HJS6606, the size of the JCT entry within the JCT data set has increased in size. In order to perform this migration without a cold start, IATUTJCT provides the capability to increase the size of the JCT entry while performing the copy. This migration allows a release lower than HJS6606 to run with the larger JCT entry. The smaller JCT entry is referred to as being at a lower version and the larger one is referred to as being at a higher version.

For further information, *z/OS JES3 Migration* describes the processes you need to follow when using IATUTJCT to migrate between JES3 releases.

Falling Back from Releases HJS6606 (or Above) to Releases HJS6605 (or Below)

If you have migrated to JES3 release HJS6606 or higher and you need to fall back to a JES3 release below HJS6606 and do not have APAR OW30849 installed on the lower release, you must run IATUTJCT with the P=FALLBCK option to shrink the JCT entries to be compatible with the lower JES3 release.

Each time you run IATUTJCT with P=FALLBCK, the JCT shrinks by 116 bytes. A warning message is presented if falling back shrinks the JCTs below the size required by HJS6606 and above. This warning message gives the operator the opportunity to cancel the shrinking. IATUTJCT does not allow JCTs to shrink below the lowest supported level (that is, a size required by HJS6605 and below).

For further information, *z/OS JES3 Migration* describes the processes you need to follow when using IATUTJCT to migrate between JES3 releases.

Copying a JCT Data Set Using the IATUTJCT Utility

To perform the copy using IATUTJCT, you must:

- Allocate a new set of checkpoint data sets in addition to the new JCT data set.
- You must identify the data set names for the new data sets used in the JCL for the JCT utility.
- Arrange your system such that JES3 can be started with either the new or old data sets. You can use symbolic parameters on the JES3 procedure to allow use of either the new or old data sets.
- Bring JES3 and all Converter Interpreter Functional Subsystems (CIFSS) down on the global processor and all local processors.

CAUTION:

If you do not bring JES3 down, JES3 might update the old JCT and the checkpoint data sets after you run IATUTJCT. These updates will be lost when you switch to the new data sets and will cause corruption to the JES3 job queue. This corruption then could lead to conditions from which a cold start is required to recover.

- Restart JES3, after running IATUTJCT, with the DD definitions that point to the new JCT data sets. For the checkpoint data set, these DD statements are in the JES3 procedure. For the JCT data set, these DD statements are in the JES3 procedure or identified by the DYNALLOC statement in the JES3 initialization stream.
- Restart the C/I FSS procedures with the new checkpoint data sets.

Running the IATUTJCT Utility

You run IATUTJCT by invoking a started task that runs the IATUTJCT program. Below is a sample JCL procedure you might use to run IATUTJCT. Such a JCL procedure must reside in SYS1.PROCLIB. Member IATUTJCS of the OS/390 SIATSAMP data set also provides a sample JCL procedure for IATUTJCT. IATUTJCT must be started using SUB=MSTR.

Sample JCL for IATUTJCT Utility

```
//IATUTJCT PROC P=  
//*  
//IEFPROC EXEC PGM=IATUTJCT,   Invoke the IATUTJCT Utility  
//   TIME=1440,  
//   PARM='&P'  
//*  
//STEPLIB DD DISP=SHR,           Required if IATGRCK and IATUTJCT  
//   DSN=SYS1.SIATLIB           are not in LNKLSTxx  
//*  
//JES3JCT DD DISP=SHR,           Current JCT data set  
//   DSN=SYS1.JES3JCT  
//*  
//NJES3JCT DD DISP=SHR,          New JCT data set  
//   DSN=SYS1.NJES3JCT  
//*  
//CHKPNT DD DISP=SHR,            Current primary checkpoint data set  
//   DSN=SYS1.CHPKNT  
//*  
//CHKPNT2 DD DISP=SHR,           Current alternate checkpoint data set  
//   DSN=SYS1.CHPKNT2  
//*  
//NCHKPNT DD DISP=SHR,           New primary checkpoint data set  
//   DSN=SYS1.NCHKPNT  
//*
```

Setup and Initialization Considerations:

- When using IATUTJCT to move the JCT data set, you can pick up the new data set with any type of JES3 start; however, if you perform a hot start with refresh or a warm start, and you use the DYNALLOC statement to define the JES3JCT DD name, you must change the DYNALLOC statement to point to the new JCT data set.
- The JES3 installation must properly meet specific programming requirements for using IATUTJCT.
 - Place the load library containing IATUTJCT and IATGRCK in the LNKLST concatenation or ensure they are in STEPLIB. JES3 must find these load modules in a normal library search.
OS/390, by default, places modules IATUTJCT and IATGRCK in SYS1.VnRnMn.SIATLIB.
 - Ensure that the library containing the IATUTJCT and IATGRCK modules is APF authorized.
 - Ensure that the JES3JCT, NJES3JCT, CHPKNT, and NCHKPNT DD statements are specified. The CHPKNT2 DD statement is optional and should be omitted if you are not using an alternate checkpoint data set or if you will not be using one after changing JES3 to use the new data sets.
 - Ensure that proper CHPKNT and CHPKNT2 DD statements are specified. There is no NCHKPNT2 DD statement. If you are using an alternate checkpoint data set, allocate a new one and leave it empty. When you restart JES3 to pick up the new primary and alternate checkpoint data sets, JES3 will automatically synchronize the new alternate checkpoint data set to the new primary checkpoint data set. Do not use the old alternate checkpoint data set as the new alternate checkpoint data set.

- The PROC statement in the procedure must define the P= keyword parameter. This parameter controls the migration of a JCT data set from a lower version to a higher version in preparation for installing a new release of JES3 that requires the higher version.

Use Considerations:

- If you are migrating your JCT data set in preparation for the installation of a higher release of JES3, refer to “Migrating Using IATUTJCT” on page 242 for information performing this migration.
- Allocate your new JCT data set and primary checkpoint data sets, and (optionally) your new alternate checkpoint data set. In doing this:
 - Be certain to allocate these data sets in cylinders with only one extent.
 - Be certain that your new JCT data set is large enough for your planned job number range, plus 64 spare JCT entries needed for write-error recovery. Refer to “How to Determine the Size of the JCT Data Set” on page 232 for information on calculating the size of the JCT.

Notes:

1. If you migrate the JCT data set from a “lower version” to a “higher version”, you must allocate a larger JCT data set to accommodate the same number of jobs because the JCT entry size in the “higher version” is larger.
 2. Your new JCT data set can be smaller than your old JCT data set if the JCT contains sufficient job capacity for the job number range on the options initialization statement. If you discover later that you need more capacity, you will then have to allocate a larger JCT data set and run IATUTJCT again.
- In JES3 installations where SYS1.PROCLIB is not shared, install the IATUTJCT procedure separately on any processor where IATUTJCT is to run.
 - Define the procedure to the security product.
 - Define either a profile for the started task in the STARTED security class or an entry in the started task table (ICHRIN03).
 - Provide the userid under which the procedure is to run with at least read access to the old JCT and checkpoint data sets and update access to the new JCT and primary checkpoint data set.

Note

If your processors do not share the security product’s data base and the STARTED security class is used to define IATUTJCT’s security environment, define the STARTED class identically on any processor where IATUTJCT is to run.

If you use the started task table to define IATUTJCT’s security environment, define the entry for the started task table identically on any processor where IATUTJCT is to run.

- Either change the JES3 start-up procedure in SYS1.PROCLIB to allow names of your checkpoint data sets (and JCT data set, if not defined by the DYNALLOC initialization statement) to be specified as symbols, or install a new JES3 procedure in SYS1.PROCLIB to point to the new data sets.

For a new start-up procedure:

- Define an entry in the STARTED security class or an entry in the started task table (ICHRIN03)

- Create a new SYS1.PARMLIB(IEFSSNxx) member pointing to the new name of the procedure you are creating as a subsystem with the PRIMARY attribute.
- If you use the DYNALLOC statement to define the JES3JCT DD name, create a new initialization stream member that points to the new JCT data set.
- If you want to test IATUTJCT, you can start it with JES3 up or down.

Note: You might want to test IATUTJCT in order to make sure that the new data sets are acceptable and that the new JCT data set is large enough to contain all the jobs from the old JCT data set plus 64 spare JCT entries for error recovery.

See “Testing IATUTJCT Processing” on page 242 for more testing information.

- When you are ready to copy the checkpoint and JCT data sets to the new ones you intend to use, stop all CIFS address spaces and bring JES3 down on the global processor and all local processors. If you have any other started tasks or jobs that reference the JCT and/or checkpoint data sets, such as user-written spool browsers, bring them down also.
- Start IATUTJCT with the SUB=MSTR parameter. If you are migrating from a “lower version” to a “higher version,” use the P=MIGRATE parameter. If you are falling back from a higher JES3 release to a lower JES3 release, use the P=FALLBCK parameter. You need to run IATUTJCT only once in a JES3 complex, and you can run it on the global or any local processor.

IATUTJCT Utility Start Command

```
S IATUTJCT, SUB=MSTR
```

IATUTJCT Utility Migrating Start Command

```
S IATUTJCT, P=MIGRATE, SUB=MSTR
```

IATUTJCT Utility fallback Start Command

```
S IATUTJCT, P=FALLBCK, SUB=MSTR
```

Note: No parameter value for P= other than MIGRATE or FALLBCK is allowed.

- If IATUTJCT fails, or if you decide to not switch to the new data sets, you must restart JES3 with the old data sets. You can perform a hot start, hot start with refresh, or IPL with a warm start. If you have changed the initialization stream (the JES3JCT DYNALLOC statement to point to the new JCT data set) and you perform a hot start with refresh or a warm start, do not use the initialization stream member containing the change.
- If you defined an alternate JES3 procedure as a primary subsystem on the global, IPL the global pointing to the new IEFSSNxx parameter.
- If IATUTJCT is successful, restart JES3 on the global pointing to the new data sets. You must begin using the new checkpoint data sets and the new JCT data set at the same time. Do not mix old and new data sets.
 - If your JCT data set is dynamically allocated and you are changing its cataloged status from cataloged to uncataloged or from uncataloged to cataloged, you must perform a hot start with refresh or warm start.

- If you perform a warm start or hot start with refresh and you define the JES3JCT DD using the DYNALLOC statement, you must use the initialization stream that you previously created, pointing to the new JCT data set.
- All converter/interpreter (CIFSS) startup procedures must be updated to point to the new checkpoint data sets. Or, you could make new CIFSS procedures that point to the new checkpoint data sets and then use the *MODIFY,F command to change the FSS to use the new procedure. Because a CIFSS startup procedure cannot accept JCL symbolics, you must change the DD statements within the procedure before restarting the CIFSS address spaces.
- If you defined an alternate JES3 procedure as a primary subsystem on the locals, IPL all locals pointing to the new IEFSSNxx member.
- Restart JES3 on all locals pointing to the new data sets.
- Start all CIFSS procedures pointing to the new checkpoint data sets.
- If you have any other started tasks or jobs that reference the JCT and/or checkpoint data sets, such as user-written spool browsers, change their data set references to the new JCT data set, and restart them at your convenience.
- Rename or delete the old checkpoint and JCT data sets so that you do not accidentally use them again. Once you have started using the new data sets, you must not revert to the old data sets. If you do, you risk corruption of the JES3 job queue possibly leading to conditions from which a cold start is required to recover.

Testing IATUTJCT Processing

You can run the IATUTJCT utility as a test while JES3 is up and running. Do not change any definitions in the JES3 startup procedure or CIFSS procedures.

- If testing the migration from a lower version to a higher one, use the P=MIGRATE parameter.
- If testing fallback from a higher JES3 release to a lower JES3 release, use the P=FALLBCK parameter.
- You need to test IATUTJCT only once in a JES3 complex, and you can test it on the global or any local.
- You do not need to specify SUB=MSTR parameter when testing IATUTJCT with JES3 up and running.

Migrating Using IATUTJCT

Special considerations for restarts apply if you are migrating between JES3 releases. For further information, *z/OS JES3 Migration* describes the processes you need to follow when using IATUTJCT to migrate between JES3 releases.

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 catalog searches (also called LOCATE processing) during the postscan phase of C/I service. Thus, you improve JES3 performance. To specify the names of the resident data sets, use the RESDSN initialization statement.

JES3 does not perform data set integrity processing for resident data sets. Therefore, do not specify SMS-managed data sets as resident data sets unless the data set can be accessed from only one processor or unless the user provides data set integrity. MVS provides data set integrity when a resident data set can be accessed from only one processor.

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 the storage management subsystem (SMS) is not active and 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. If SMS is active, no action is required to ensure catalog availability. JES3 and SMS together determine the availability and connectivity of catalogs.

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 *Catalog Administrator's 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.

If you are using the storage management subsystem, the master catalog must be an ICF catalog on all mains where C/I processing is performed. Ensure that all master catalogs contain alias names for all user catalogs. If you are using ICF catalogs, users can omit the JOBCAT and STEPCAT DD statements from their JCL.

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.

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**.

To simplify the task of adding or changing multiple devices, use the `NUMDEV` keyword and the processor name `*ALL`. Alternatively, you can use the `IATDEVA` and `IATDEVC` edit macros, which are documented in *z/OS JES3 Customization*.

JES3 cannot use restricted devices. Therefore, do not define a restricted device on a JES3 `DEVICE` initialization statement.

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 processor. The only exception is a device driven by an output writer functional subsystem (FSS), which you may define as a JES3 device and attach to the global processor or a local processor.

Use unique device numbers for JES3 devices when you execute the MVS configuration program. When the same device is attached to more than one processor, the device should have the same device number on each processor. On the other hand, if different JES3 devices attached to different processors 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 operator commands that refer to the devices by device number. Operator commands for those devices must refer to them using the name specified on the `JNAME` parameter of the `DEVICE` statement.

You can define as JES3 devices:

- 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 *z/OS JES3 Commands*.

Execution Devices

To define an execution device, specify the XTYPE and XUNIT parameters on the DEVICE statement. When any execution device 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.

Note: 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.

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, you can switch between the initial set of process modes and an alternate set.

Note: You can switch between sets of process modes using the *MODIFY,FSS,D= command.

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

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.

Table 42. Default Value(s) for ALTPM= When MODE=COMP

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)

Table 43. Default Values for ALTPM= When MODE=FSS

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)

Running a Printer Under an Output Writer Functional Subsystem

Some printers run under the control of the JES3 global address space. Certain printers, such as AFP printers, must run under the control of a non-JES3 address space called an output writer functional subsystem (FSS).

To use a PSF printer as a system output device, you must first do the following:

1. Define one or more PSF FSSs.
2. Define FSAs for each PSF printer.
3. Make sure that you have a startup procedure for each PSF FSS.

After you complete Steps 1 through 3, the operator can enter commands to start each FSA. JES starts the FSS automatically when the first FSA it controls is started. For more information on using PSF, see *Print Services Facility/MVS Customization Guide*.

Defining PSF FSSs and FSAs for Printers

To use a PSF printer under JES, you must first code JES initialization statements to define PSF FSSs and FSAs for PSF printers. You use the FSSDEF initialization

statement to define the FSS, and you use the DEVICE initialization statement to define each printer FSA. The following number of FSSs and FSAs are supported:

- A maximum of 2000 FSSs
- A maximum of 128 FSAs per FSS

Note: The actual number of FSAs per FSS depends on several factors, including:

- The virtual region size below the 16-megabyte line
- The number and the size of form definitions and page definitions
- The number of buffers specified for channel-attached printers
- The block sizes of the font, page segment, overlay, form definition, and page definition libraries

See “Determining the Maximum Number of Printers for an Output Writer FSS” on page 248 for more detail.

You specify whether you want a printer to run under the control of the JES3 global address space or under the control of an FSS by specifying the FSSNAME= and MODE=FSS keywords on the DEVICE initialization statement. If you run a printer under the control of an FSS and you want to define the characteristics of that FSS, you must include an FSSDEF initialization statement. Otherwise, JES3 dynamically creates an FSS using default values.

The following sections describe how to define an output writer FSS and how to define specific AFP printers (such as, 3900 and 3827 printers), to run under the control of an output writer FSS.

Defining an Output Writer Functional Subsystem

You can define one or more printers to run under the control of a single output writer FSS or you can allow JES3 to define a unique FSS for each printer by default.

You can also select the main on which you want an output writer FSS to operate by coding the SYSTEM= keyword on the FSSDEF initialization statement.

Perform the following steps to define one or more page printers to run under the control of an output writer FSS:

1. Attach the printer(s) you want to run in FSS mode to the main on which you plan to define the output writer FSS. The data processed by the printer(s) can originate from any main within your JES3 complex or at any node in a job entry network.
2. Define a cataloged procedure for starting the output writer FSS in SYS1.PROCLIB and specify the name of the procedure on the PNAME= keyword of the FSSDEF initialization statement. To learn more about cataloged procedures and how to modify them, see *Print Services Facility/MVS System Programmer's Guide* and *Print Services Facility/MVS Customization Guide*. Parameters that you define on JES3 initialization statements override the corresponding parameters in a cataloged procedure. However, the PDEFAULT= parameter on the DEVICE (I/O) initialization statement can be used to direct JES3 to not send the CHARS and FCB defaults to the FSS.
3. Include an FSSDEF statement in your initialization stream and name the FSS using the FSSNAME= keyword.

4. Include a DEVICE statement for each printer that you want to run under this FSS. You must include the FSSNAME= keyword on the DEVICE statement that specifies the same name that you code on the FSSNAME= keyword of the FSSDEF initialization statement.
5. Perform a hot start with refresh, warm start, and cold start or issue a *MODIFY,CONFIG command to add the new FSS.

Note: A *MODIFY,CONFIG command can only be used to add an FSS for a non-channel attached printer.

If you define a printer to run in FSS mode (that is, you specify MODE=FSS on the device initialization statement), or you define a AFP printer that operates only in FSS mode (such as a 3900 or 3827), you must define an output writer FSS using the FSSDEF initialization statement. If you do not define an output writer FSS, JES3 creates a default FSS for each printer that requires one.

If you let JES3 create a default FSS, the name of the FSS will be the same as the name you specify on the JNAME parameter on the DEVICE statement. In this case, you can change the default FSS's characteristics using the *MODIFY command, but you cannot specify its initial characteristics.

To change the definition of an output writer FSS during operation, use the *F,F command. You can also call, start, restart, or cancel an output writer FSS using JES3 commands. You can use the same operator commands as for any output writer (for example, *X,WTR).

Determining the Maximum Number of Printers for an Output Writer FSS

You can improve performance and reduce the amount of storage that JES3 requires to operate devices by defining more than one printer to run under the control of a single output writer FSS. Defining multiple printers under a single FSS reduces the number of FSSs that JES3 must initialize and cancel.

The maximum number of printers that you have an FSS manage depends on the amount of available virtual storage below 16-megabytes on the main where the FSS is defined.

Because each installation is unique and you do not want to degrade JES3 performance, you must determine the maximum number of devices that can run under control of a single FSS for your installation. IBM recommends defining your installation so that no more than 9 printers can be active concurrently under a single FSS. If more than 9 printers are active concurrently, you may experience a performance degradation problem.

The amount of available private storage on the main where you define an FSS also affects the number of devices that can run under the control of a single FSS. To determine the maximum number of devices that can be assigned to a single FSS at your installation, you should:

1. Generate a Resource Management Facility (RMF) report for the main where an FSS address space is defined. Use the Virtual Storage Activity Report to determine the amount of available virtual storage in the private area that the you can access. You will use this number in step 2 to determine the maximum number of active devices that can run under the control of an FSS.
2. Use the following formula to determine the maximum number of active devices that can run under the control of an FSS running in 24-bit addressing mode:

$$\text{Number of active devices} = \frac{\text{Total from step one} - 1.25}{.75}$$

Defining the 3800 Model 3 Printer as a Line Printer

You can operate an IBM 3800 model 3 printer like any other 3800 under the control of 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 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.

Defining Channel Attached Printers Controlled by FSSs

To use the advanced printing functions of the 3800 model 3, you must run it under the control of an output writer FSS. An output writer FSS allows printers to process data containing control characters that format the data as a composed page. Running a printer in this way is known as running it in *FSS mode*.

You can explicitly define an output writer FSSs or you can let JES3 create a default FSS for each printer. If you want to run a printer under the control of a particular FSS, 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 you want the printer to operate. You define the name of the FSS using the `FSSDEF` initialization statement. If you choose to let JES3 define the output writer FSS, omit the `FSSNAME` parameter.
- Specify the `MODE=FSS` parameter.
- Use the `PM` (processing mode) parameter to specify the types of data sets that you want the printer to process. 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 process both types. You can change the processing mode if the printer is offline and not currently allocated. Use the `*F,F` operator command with the `M=` keyword. (See *z/OS JES3 Commands* for more information on the `*F` command.)

You should also specify any other required parameters on the `DEVICE` statement for the specific device.

Defining AFP1 Printers: AFP1 printers are IBM page printers that run under the control of an output writer functional subsystem (FSS). To define an AFP1 printer to JES3, you must code a `DEVICE` initialization statement that defines the printer to JES3, and optionally, an `FSSDEF` statement that defines the specific FSS under which you want the printer to run.

Defining Channel-Attached AFP Printer

A channel-attached device is one that you connect directly to a host processor's channel subsystem. A non-channel-attached device is one that you attach using an ACF/VTAM-SNA connection.

```
DEVICE,DTYPE=PRTAFP1,JNAME=AFPPRT1,FSSNAME=FSSAFP1,  
JUNIT=(803,SY1,S1,OFF,803,SY2,S2,ON),XTYPE=(PRTAFP1,UR),  
XUNIT=(803,SY1,S1,OFF,803,SY2,S2,OFF),WS=(D),  
DGROUP=ARM1,PM=(LINE,PAGE),MODE=FSS  
  
FSSDEF,TYPE=WTR,FSSNAME=FSSAFP1,SYSTEM=SY1,  
PNAME=APSWPROA,TERM=YES  
  
SETNAME,XTYPE=PRTAFP1,NAMES=AFP1
```

This example defines an AFP1 printer as a channel-attached device to mains SY1 and SY2 at device number 803, and the FSS is explicitly defined.

The following rules apply when defining AFP1 printers to JES3:

- AFP1 printers can run only under the control of an output writer FSS. You must define an FSS and its attributes using an FSSDEF initialization statement or JES3 will define a default FSS during JES3 initialization.
- You can define AFP1 printers as channel-attached or non-channel attached devices. If you specify a device number for the first main on the JUNIT= keyword of the DEVICE statement, then you must also specify a device number for all subsequent mains or JES3 propagates the first device number to all of the mains that do not have device numbers. If you omit the device number for the first main, JES3 does not assign a device number to any of the mains, regardless of what is specified for other mains on the statement.
- All parameters that you can specify for IBM 3800 and IBM 3820 printers, except for the FLASH= and CB= keywords, can also be specified for AFP1 printers.
- You can modify the forms-mark value that you specify on the BURST= keyword of the DEVICE statement after initialization using a *CALL, *START, or *RESTART,WTR command. When you specify the forms mark option, JES3 prints marks on the edges of a job's trailer page (also called a burst page) or three blank pages following the job's output.
- You can issue a *INQUIRY,D command after initialization to display the status of AFP1 printers. However, JES3 does not display information about clear buffer, character generation storage, or flash characteristics since these characteristics do not apply to AFP1 printers. See *z/OS JES3 Commands* for descriptions of these commands and their syntax.

Defining Non-Channel Attached Printers

Non-channel attached printers (also called SNA-attached printers) can run only under the control of an output writer FSS.

To define a non-channel attached printer, specify the DTYPE= keyword and MODE=FSS on the DEVICE initialization statement. You must also omit the address on the JUNIT= keyword; instead, code a comma. Also, do not code the XUNIT= or XTYPE= parameters. You can also use the FSSNAME= parameter to specify the name of the functional subsystem under which you want the printer to run.

Use of FSSNAME= Parameter

```
DEVICE,DTYPE=PRTAFP1,JNAME=PRABC1,JUNIT=(,SY1,S1,OFF),MODE=FSS,  
PM=(LINE,PAGE),FSSNAME=WTRFSS2
```

In this example, a SNA-attached AFP printer (such as an IBM 3827) 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. This device can print both *page* and *line* mode data.

Although SNA-attached printers are not directly connected to processor channels, the controlling functional subsystems may still require specific resources, such as VTAM, that are available only on certain processors. Therefore, carefully consider where you want functional subsystems to execute when coding the SYSTEM= parameter on the FSSDEF statement.

Using the Infoprint Server for z/OS Printer Inventory

The Printer Inventory component of the Infoprint Server for z/OS manages the inventory of printer information that is used by NetSpool, Print Interface, IP PrintWay(TM), and PSF for z/OS. These applications obtain information from the inventory about handling jobs on the JES spool and starting and stopping their processes. The Printer Inventory lets you define and update printer definitions and define the functional subsystem (FSS), the functional subsystem application (FSA), and printer definition pool parameters. In deferred-printing mode, PSF can use printer initialization, tracing, and execution option parameters defined in the Printer Inventory instead of those defined in the PSF startup procedure.

With the Printer Inventory, you can define a printer in the PSF startup procedure and JES initialization statements before the printer is actually used. You can also define a range of printers in the JES initialization statements if you support a large number of printers. When the new printer is added, you can assign variable parameters for it in the Printer Inventory. This saves you time because you do not need to restart all the printers in a startup procedure when you add a new printer or change parameters for an existing printer. The Printer Inventory also makes it much easier to define parameters.

Refer to *z/OS Infoprint Server Customization* for more information about the Printer Inventory

Defining a Device Configuration

Defining a Range of Devices

You can define a range of devices using a single DEVICE statement by specifying the NUMDEV parameter. JES3 defines the requested number of devices starting at the XUNIT and/or JUNIT specified on the DEVICE statement.

For example, the following DEVICE statement defines five channel attached printers at device numbers 803, 804, 805, 806, and 807:

Range of Devices

```
DEVICE, DTYPE=PRTAFP1, JNAME=AFPP, FSSNAME=FSSAFP1,  
JUNIT=(803, SY1, S1, OFF, 803, SY2, S2, ON), XTYPE=(PRTAFP1, UR),  
XUNIT=(803, SY1, S1, OFF, 803, SY2, S2, OFF), WS=(D),  
DGROUP=ARM1, PM=(LINE, PAGE), MODE=FSS, NUMDEV=5
```

The following rules apply when using NUMDEV:

1. When using NUMDEV for a global device, the JNAME specifies a prefix. JES3 combines the prefix with each JUNIT in the requested range as a four digit device number to construct JNAMEs for all devices defined by this statement. In the above example, the JNAMEs for the printers defined by the statement are AFPP0803, AFPP0804, AFPP0805, AFPP0806, and AFPP0807. The prefix cannot exceed four characters.
2. NUMDEV cannot be zero, and cannot cause the range to exceed FFFF. If NUMDEV is 1, the JNAME specifies a prefix just as it does for larger values.
3. If you define an FSS managed printer without using the FSSNAME= parameter, the FSS name is the same as the constructed JNAME. If you use the FSSNAME= parameter, the FSS name is the specified FSSNAME.
4. If a JNAME of a device defined using the NUMDEV statement is referenced anywhere else in the initialization stream (for example, on the DEST= keyword of a SYSOUT statement) make sure that the referenced name is the same as the constructed name.
5. At least one XUNIT or channel attached JUNIT is required. For example, NUMDEV cannot be used on a DEVICE statement for a 3820 printer.
6. If the device is defined with different device numbers on different processors, each processor gets a range of the same number of devices starting with the specified unit. The unit used to build the JNAME is the first channel attached unit that appears in the JUNIT parameter.

Defining a Device To All Processors Simultaneously

Instead of listing each processor and defining the unit to the individual processors, you can define it to all processors simultaneously by using a system name of *ALL.

- Specifying *ALL makes it convenient to define parallel MVS hardware configurations.
- *ALL can be specified on a single DEVICE definition or on a DEVICE statement that uses the NUMDEV parameter.
- *ALL can be specified on a channel attached, VTAM-attached, or SYSMAIN (processor status) device.
- *ALL can be specified on a VTAM-attached printer that is added using the *MODIFY,CONFIG command.

Using *ALL on a device is particularly useful if you will be adding MAINPROCs later, as you do not need to add the new main name to any DEVICE statement that uses *ALL.

For example, the following statement could be used to define multiple channel attached printers to all mains simultaneously:

```
DEVICE, DTYPE=PRTAFP1, JNAME=AFPP, FSSNAME=FSSAFP1,  
JUNIT=(803, *ALL, S1, ON), XTYPE=(PRTAFP1, UR),  
XUNIT=(803, *ALL, S1, ON), WS=(D),  
DGROUP=ARM1, PM=(LINE, PAGE), MODE=FSS, NUMDEV=5
```

The following rules apply when using *ALL:

1. Mixing *ALL and system names within the same JUNIT or XUNIT is not allowed.
2. *ALL must appear only once within the same JUNIT or XUNIT.
3. If *ALL is used on the JUNIT and the device has both a JUNIT and an XUNIT, *ALL also must be used on the XUNIT. If *ALL is used on the XUNIT, the JUNIT can list processors, but the use of *ALL on the JUNIT is recommended.
4. In addition to the device number, the message destination and initial online status are defined identically to all processors.

Maintaining a Device Configuration Using HCD

You can use the Hardware Configuration Definition (HCD) program to add, delete, or change JES3 defined devices. These can be JES3 global devices (JUNIT), execution devices (XUNIT), or shared devices (both JUNIT and XUNIT). When making these changes, be careful not to introduce subgeneric splits, define devices to JES3 but not to MVS, or define devices as one device type to JES3 but a different device type to MVS.

If you make a mistake, JES3 will tolerate the error and initialize (if there are no other errors with higher impact); however, you should correct the error and perform a hot start with refresh at your earliest convenience.

Adding Devices

Adding a device to HCD that is also to be defined to JES3 requires a corresponding device definition in the JES3 initialization stream. The device can be defined by an individual DEVICE statement or it can be included in a range of devices by using the NUMDEV parameter on a DEVICE statement, which defines the first device in the range. The device must be specified on the XUNIT and/or JUNIT parameter. The processor can be defined by name or it can be defined implicitly by using the system name *ALL. If you use *ALL, the HCD configuration for every processor must define the device the same way.

If you are adding a new device type or changing the esoteric groupings of devices, you may also need to add or change SETNAME statements.

To define the new device to JES3, you must perform a hot start with refresh using your modified initialization stream after activating the configuration in HCD. All local processors that are at least at the level of OS/390 Version 2 Release 9 JES3 will be automatically restarted when the hot start with refresh occurs, so you must activate the configuration on all processors before you perform the hot start with refresh.

A warm start or IPL is not required to make the JES3 changes, although you can do so if you prefer.

Changing Devices

When you use HCD to change the characteristics of an existing JES3 defined device in HCD, you do not need to IPL the processor or restart JES3.

Deleting Devices

Deleting a JES3 defined device from HCD does not require any immediate JES3 change; however, you should plan to delete the device from JES3 and perform a hot start with refresh with the modified initialization stream at your earliest convenience after activating the configuration.

Allocating an I/O Device to JES3

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

Grouping I/O Devices

You can group JES3 devices by using esoteric names. Generic names are provided by the unit information modules (UIMs) included in hardware configuration definition (HCD). Do the following to define esoteric names to the system (MVS only).

- Use the Define, modify or view configuration task from the HCD primary task selection panel to define the I/O device to the control units, processors and operating system (OS) configuration.
- After you have defined the OS configuration data, you will be prompted to specify the required data, and you can then assign the device to the esoteric groups. See *z/OS HCD User's Guide* for detailed information on the steps to define the device data.

MVS divides generic device types into subgeneric groups, as illustrated in Figure 24. The use of subgeneric groups allows MVS allocation to serialize a subset of units within a generic name. For example, using the figure, if 3390A is requested, MVS allocation needs to serialize only subgeneric group 5 rather than all 3390 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	3490				3380				3390				
Esoteric Names (Defined by the Installation)					SYSDA				SYSDA				
									3390A				
Device Numbers	131	132	133	134	181	182	183	184	191	192	193	194	195
Subgeneric Groups	1				2	3	2		4		5		

Figure 24. Subgeneric Groups

The guidelines by which MVS determines subgeneric groups are:

- If an esoteric name (for example, 3390A) includes only a subset of the units in a generic name, that subset is a subgeneric group. (For 3390A 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 3380 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, 3390A 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.

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 *z/OS MVS System Commands*. This document 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 *z/OS MVS Using the Subsystem Interface*.

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 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.

DEVICE Statement — Example

```
DEVICE,XTYPE=(3330,DA,PR),  
XUNIT=(230,SY1,S1,130,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 (that is, 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

I/O Device 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 25 highlights the questions asked when determining how a device will be managed.

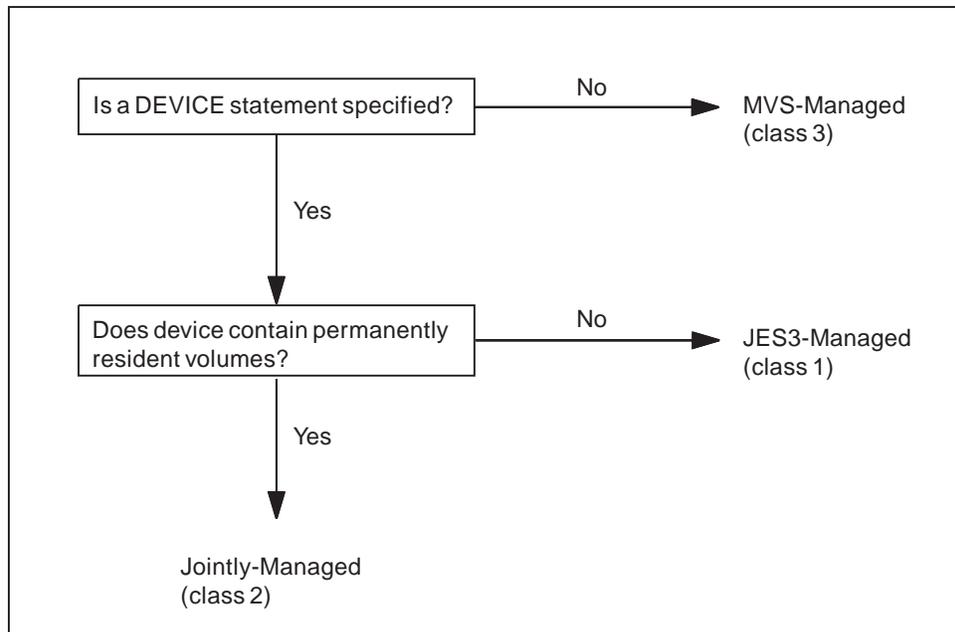


Figure 25. JES3-Managed, MVS-Managed, and Jointly Managed Device Determination

JES3 places each device type into one-of-three classes. The class identifies whether MVS or JES3 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 (these devices 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 during system installation. However, these names must not be specified in any SETNAME initialization statement.

Data Set Access

To determine whether JES3 or MVS will control 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
- Whether the data set is SMS-managed or not
- The type of request

Table 44 shows the decision making process that JES3 uses to determine who controls access to data sets.

Table 44. Who Controls Data Set Access

JES3 Decision Process	Yes or No	Action (if any)	Controls Access
(1) Data Set Name = RESDSN ?	Y		MVS
	N	Ask Question # 2	
(2) Data Set SMS-Managed ?	Y	Ask Question # 3	
	N	Ask Question # 4	
(3) SMSSETUP=NO Specified ?	Y		MVS
	N		JES3
(4) JES3 Unit Name Specified ?	N		MVS
	Y	Ask Question # 5	
(5) Device Type is DASD ?	N		JES3
	Y	Ask Question # 6	
(6) Is Request Specific ?	Y		JES3
	N	Ask Question # 7	
(7) Is Request Private ?	Y		JES3
	N		MVS

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 *z/OS MVS Programming: Authorized Assembler Services Guide*.

JES3 Data Set Integrity: What It Does: JES3 enforces data set integrity for:

- Data sets that are requested via DD statements that require JES3-managed devices
- Data sets that are requested dynamically that require JES3-managed devices
- Data sets that are SMS-managed, unless SMSSETUP=NO is specified on the SETPARAM statement.
- 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 using 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 do not require a JES3-managed device. For example, requests for new data sets on non-specific, non-private DASD volumes will be managed by MVS. JES3 will handle these type of requests only when they are SMS-managed. See "Data Set Management" 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 on a JES3 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.

JES3 does not perform data set integrity checking for resident data sets. Therefore, do not specify SMS-managed data sets as resident data sets unless the data set can be accessed from only one processor (in which case, MVS provides data set integrity), or unless the user provides data set integrity.

The following examples illustrate the effects of data set integrity on job processing. In the examples:

- The data set named DIRECT is data set that requires a JES3-managed device.
- The model 3490 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,...),...
/*

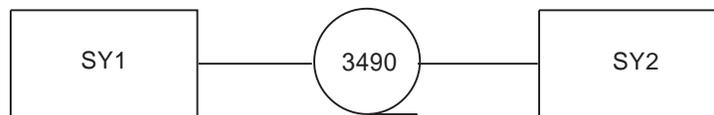
//JOB B JOB .....
//STEP1 EXEC PGM=JOB B
//DD1 DD DSN=DIRECT,DISP=(SHR,...),...
/*
```

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,...),...
/*

//JOB B JOB .....
//*MAIN SYSTEM=SY2
//STEP1 EXEC PGM=JOB B
//DD1 DD DSN=DIRECT,DISP=(OLD,...),...
/*
```

Example 3: There is one model 3490 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=3490
/*

//JOB B JOB .....
/*
```

```

//*MAIN  SYSTEM=SY2
//STEP1 EXEC PGM=JOB0
//DD1   DD  UNIT=3490
/*

```

Volume Management

Either JES3 or JES3 and MVS together manage volumes that are mounted on JES3-managed devices. Table 45 shows who manages a volume for different combinations of mount and use attributes.

Table 45. JES3 Volume Management

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

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 ADDR SORT 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.

Table 46 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.

Table 46. JES3 and System Handling of Allocation Requests

	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 <code>//*MAIN SYSTEM</code> or <code>JOB CLASS</code> .	JES3 setup assumes a volume is not mounted and schedules job on any processor unless <code>//*MAIN SYSTEM</code> or <code>JOB CLASS</code> is specified. JES3 asks the operator to mount; operator must cancel the job.

Table 46. JES3 and System Handling of Allocation Requests (continued)

	Allocated By:	
Mounted on MVS managed device	MVS allocation will find device on processor that must be defined by <code>//*MAIN SYSTEM</code> or <code>JOB CLASS</code> .	JES3 setup assumes a volume is not mounted and schedules job on any processor unless <code>//*MAIN SYSTEM</code> or <code>JOB CLASS</code> 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 <code>//*MAIN SYSTEM</code> or <code>JOB CLASS</code> .	JES3 setup locates required volume and processor.
Mounted on JES3-managed device	Allocation fails.	JES3 setup locates required volume and processor.
* 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.		

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 *z/OS MVS Programming: Authorized Assembler Services Guide*.

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.

JES3 manages all requests for SMS-managed data sets unless `SMSSETUP=NO` is specified on the `SETPARAM` initialization statement. However, JES3 is not aware of specific units for SMS-managed data sets. JES3 treats requests for these data sets as single requests and sends them through dynamic allocation - fast path. All

dynamic allocation requests for SMS-managed data sets that require more than one SIOT, such as GDG-all requests, will be sent through MDS for processing instead of dynamic allocation - fast path.

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 JESYSMSG file for TSO users. To suppress these messages, code the JESMSG parameter on the STANDARDS initialization statement.

Part 9. Defining and Managing JES3 Mains and Storage

Chapter 9. Defining and Managing JES3 Mains and Storage	267
Defining Mains	267
General Considerations	267
Defining a JES3 XCF Group	268
Running JES3 in a Sysplex Environment	270
Defining Storage	271
Defining Buffers	271
Defining the Size of JES3 Buffers	271
Defining the Size of the JSAM Buffer Pool	272
Defining USAM Unprotected Buffers (UBUFs)	273
Defining the Number of USAM Protected Buffers (PBUFs)	273
Reducing the Amount of Common Service Area Used by JES3	274
Determining How Many Buffers to Allocate in the JES3 Auxiliary Address Space	274
Using the Writer Output Multitasking Facility	275
Restart Considerations	275
DSI Considerations	276

Chapter 9. Defining and Managing JES3 Mains and Storage

Defining Mains

As part of JES3 initialization, the *mains* must be defined 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 run.

You must code a MAINPROC statement for every main in the JES3 complex. You may optionally code a DEVICE statement corresponding to some or all of your MAINPROC statements. The MAINPROC statement defines each main to JES3. The DEVICE statement, with DTYPE=SYSMAIN specified, defines the initial status of the main to JES3. If DTYPE= is not coded, the default is that the main is initially online to all mains. For a description of the parameters on these statements, see *z/OS JES3 Initialization and Tuning Reference*.

You are allowed to define a maximum of 32 mains in a single JES3 complex. Each main is defined on a separate MAINPROC statement. The name specified on the NAME parameter on the MAINPROC statement **must match** the MVS system name specified on the SYSNAME parameter in the IEASYSxx member of the MVS SYS1.PARMLIB data set.

For successful dynamic system interchange (DSI) processing, ensure that each processor containing a main that is eligible for DSI is properly defined.

General Considerations

Consider the following when planning your configuration:

- The total number of mains. A maximum of 32 MAINPROC statements is allowed.
- Which main will be the global. Establish DSI capability on other mains.
- 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 the job is to run. The user may also specify the job class or group in which the job is to be processed, which may be defined to a specific main.
- Adequate instructions on available facilities for operators and users. Procedures in run documents should detail how and when reconfigurations are to be performed 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.

The DEVICE statement defines the 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 every main in the configuration or the main name *ALL. Omitting the entire DEVICE statement is equivalent to defining the device with JUNIT=(,*ALL,,ON).

Note: Ensure that all mains in the JES3 complex are at compatible release and maintenance levels.

Example

The following configuration includes two processors, SY1 and SY2.



Each processor must be defined by a MAINPROC statement. The online/offline status of mains can optionally be defined by a DEVICE statement.

For SY1:

```
DEVICE,DTYPE=SYSMAIN,JNAME=SY1,  
JUNIT=(,SY1,,,,SY2,,ON)
```

```
,SY1,,,
```

Specifies that SY1 is initially online to SY1.

```
,SY2,,on
```

Specifies that SY2 is initially online to SY1.

```
MAINPROC,NAME=SY1,SYSTEM=JES3
```

For SY2:

```
DEVICE,DTYPE=SYSMAIN,JNAME=SY2,  
JUNIT=(,SY1,,,,SY2,,ON)
```

```
,SY1,,,
```

Specifies that SY1 is initially online to SY2.

```
,SY2,,ON
```

Specifies that SY2 is initially online to SY2.

```
MAINPROC,NAME=SY2,SYSTEM=JES3
```

Note: In the above example, the DEVICE statements could also be coded with `JUNIT=(,*ALL,,ON)` or they could be omitted completely, and the configuration would be equivalent.

Defining a JES3 XCF Group

JES3 uses the JES common coupling services (JES XCF) for communicating data amongst the JES3 XCF group members. Refer to *z/OS MVS Programming: JES Common Coupling Services* for a description of the JES common coupling services (JES XCF macros and exits). To allow both MVS and all the JES3 XCF members to know about each other, you need to define the group to JES3 in the JES3 initialization data stream as shown in Figure 26 on page 269. Refer to this figure as you study the following sections.

JES3 Initialization Stream

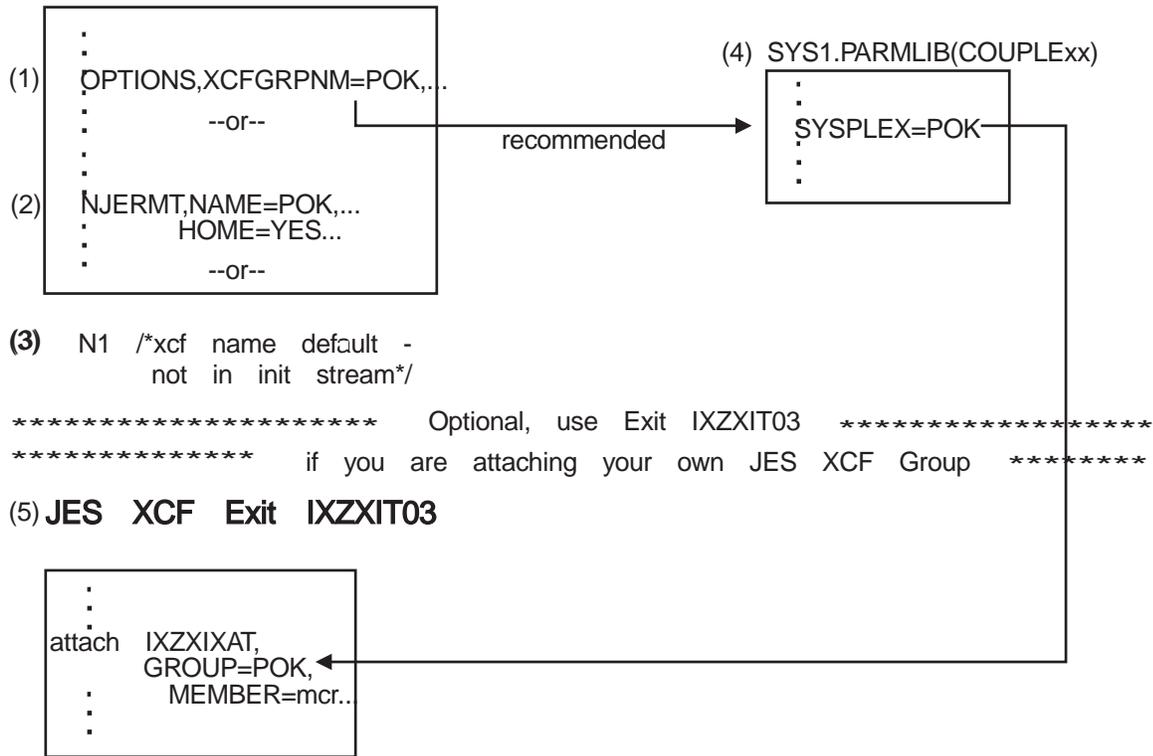


Figure 26. Defining a JES3 XCF Group - Putting the Pieces Together

A JES3 complex is assigned a default JESXCF group name. This default is the node name defined by the NAME= keyword on the NJERMT initialization statement for the home node (HOME=YES specified), if one exists. If you have not defined an NJERMT statement, the default name is N1.

If you use the default, your installation refers to both the node and the JESXCF group by the same name. In addition, if you need to refer to the sysplex by the same name, specify the same name on the SYSPLEX= keyword in the COUPLExx member of SYS1.PARMLIB [4](#). Use this approach if you want your sysplex name, your node name, and your JESXCF group to all be named the same (that is, there is a one-to-one relationship between each of these structures). This is the IBM recommended approach.

Note that there are no restrictions which require any of these names to be the same. In fact, you may wish to name these structures differently, for reasons such as the following:

- Your current node name does not meet naming restrictions for a JESXCF group name.
 XCF naming conventions require that your name must be 1-8 alphanumeric characters including \$, # and @. Changing your node name could be difficult and expensive and does not justify its change.
- You require multiple JES3 complexes within the same sysplex.
 While IBM does not recommend multiple JES3 complexes within a sysplex, your installation may choose to use this approach. If neither JES3 complex uses NJE

(and thus does not contain a nodename for the home node), both JES3 complexes would default to a JESXCF group name of N1 resulting in an abend condition.

In support of these scenarios and any others in which your installation may want a JESXCF group name that differs from the node name, use the XCFGRPNM= keyword on the JES3 OPTIONS initialization statement. (Use this optional keyword to define a name that must be 1-8 alphanumeric characters including \$, # and @.)

JES3 determines the XCF group name for your JES3 complex in the following order:

1. XCFGRPNM= keyword on the OPTIONS initialization statement, if specified - **1**
2. The node name defined by the NAME= keyword on the NJERMT initialization statement which specifies HOME=YES, if specified - **2**
3. N1 (the default node name) - **3**
4. Optionally, use the JES XCF attach/detach exit IXZXIT03 if you are attaching your own JESXCF group - **5**

Running JES3 in a Sysplex Environment

JES3 must run in a sysplex environment. A *sysplex* (system complex) is defined as a single MVS system or multiple MVS systems that use the MVS cross-system coupling facility (XCF). A JES3 complex then is considered to be a group of JES3 systems (or mains) with one global and multiple local processors, sharing a common spool, checkpoint, and XCF group.

The following lists considerations and recommendations for running JES3 in a sysplex environment:

- The systems in a JES3 complex may be at different levels. See *z/OS JES3 Migration* for the levels of JES3 that may coexist in the same JES3 complex.
- A JES3 complex must be contained entirely within one MVS sysplex.
- Although it is possible to have multiple JES3 complexes within a sysplex, IBM recommends a one-to-one relationship between the JES3 complex and the sysplex.
- All the systems in a JES3 complex must be within the same XCF group. In other words, one JES3 complex equates to one XCF group.
- A maximum of 32 mains can exist in one JES3 complex.
- The JES3 main names, as specified on the NAME parameter on the MAINPROC statement, **must match** the MVS system name, as specified on the SYSNAME parameter in the IEASYSxx parmlib member.
- A coupling facility is not required, but can be used if available. Communications between JES3 mains are via XCF signalling. (For XCF communications, JES3 uses whatever connections are supported by MVS/ESA, which could be the coupling facility or ESCON CTCs.)

For specific information and considerations, see *z/OS MVS Setting Up a Sysplex*, *z/OS and z/OS.e Planning for Installation*, and the *Parallel Sysplex Configuration Assistant*, available at <http://www.ibm.com/s390/pso/psotool>.

Defining Storage

You must also define portions of storage used by JES3, known as *buffers*. A JES3 buffer is an area of storage that JES3 uses to read and write data to and from spool.

Because JES3 supports 31-bit addressing, much virtual storage constraint imposed by the JES3 private area is minimized. Common area storage constraints are also minimized because most of the JES3 common area code and control blocks reside in the extended common area above 16 megabytes in storage. However, because all constraints are not relieved with 31-bit addressing capability (for example, C/I space), you may still wish to consider the tuning guidelines provided.

Defining Buffers

JES3 uses *buffers* to read and write data from and to spool. A buffer is an area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written.

JES3, FSS, and user address spaces require different types of buffers:

- A JES3 address space uses JES3 spool access method (JSAM) buffers
- A converter/interpreter FSS address space also uses JSAM buffers
- A user address space uses two types of user spool access method (USAM) buffers:
 - USAM protected buffers (PBUFs)
 - USAM unprotected buffers (UBUFs)
- An output writer FSS address space uses USAM PBUFs.

Data contained in JSAM buffers is read and written directly from and to spool. Data contained in UBUFs must be moved into PBUFs which are used to perform spool I/O.

The following sections describe how to determine the size, number, and placement of JSAM and USAM buffers in your JES3 complex.

Defining the Size of JES3 Buffers

You use the BUFSIZE parameter of the BUFFER initialization statement to set the size of:

- JSAM buffer
- USAM PBUF (protected buffer)
- USAM UBUF (unprotected buffer)

The BUFSIZE parameter value also sets the size of the spool records. You can specify between 1952 and 4084 bytes as the size of JES3 buffers. All buffers 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.
- Multiple record files use the first 44 bytes of a buffer for header information. You can determine the usable buffer space for a multiple record file by subtracting 44 bytes from the buffer size you select. If a DCB parameter is specified on a SYSOUT DD JCL statement, the LRECL= and BLKSIZE= keywords must not be larger than the usable buffer space.

- Some spool data is blocked when written to the JES3 buffers. Therefore, a larger buffer size can 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 a 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 algorithm to determine the minimum buffer size required to enable checkpointing the status of all groups defined:

Algorithm for Minimum Buffer Size

$$\text{BUFSIZE} = (10 \times \text{NUMBER OF EXRESC'S}) + 255 + 176$$

Where 10 is the size of each EXRESC entry to be checkpointed, NUMBER is the number of EXRESC keywords that specify 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.

Attention: You can change the buffer size only during a cold start.

Defining the Size of the JSAM Buffer Pool

You determine the primary allocation of JSAM buffers in your complex by defining the size of the JSAM buffer pool using the PAGES= keyword on the BUFFER initialization statement. The JSAM buffer pool is an area of storage in which all JSAM buffers are kept.

The size of the JSAM buffer pool is measured in terms of *pages of storage* that you define using the BUFFER initialization statement. A page of storage can accommodate two buffers if you specify a buffer size of 2036 or less, or one buffer if you specify a larger buffer size. After you select the buffer size, specify the size of the primary JSAM buffer pool in the JES3 global address space, JES3 local address spaces, and C/I FSS address spaces using the PAGES= keyword of the BUFFER initialization statement. Use the following algorithm to determine the size of the primary JSAM buffer pool in the JES3 global address space:

JSAM Buffer Pool Algorithm

$$\text{PAGES} = (a+b+c) / (4096 / (\text{BUFSIZE} + 12))$$

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. You can change the size of the JSAM buffer pool during a warm start.

To determine the size of the primary JSAM buffer pool in the CI FSS address space, start with one half the size of the JSAM 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 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.

When JES3 determines that all of the buffers in the primary allocation have been exhausted, it automatically expands the buffer pool by allocating additional buffers. Although JES3 provides a secondary allocation of buffers each time an out-of-buffer condition exists, JES3 will expand the buffer pool up to a maximum of 4 times or 32767 bytes, whichever occurs first. The number of buffers that JES3 allocates for each secondary allocation is one-half the number specified on the PAGES= keyword of the BUFFER initialization statement.

JES3 frees the secondary allocation(s) of buffers when the number of available buffers is greater than or equal to the number of buffers in the secondary allocation *plus* the acceptable minimum number of JSAM buffers. To find the minimum number for your installation, divide the total number of JSAM buffers (the primary allocation buffers from the PAGES= keyword plus all secondary buffer allocations) by the MINBUF= parameter.

You can use the JES3 *INQUIRY,C command to display the size of the primary buffer allocation, secondary buffer allocation, and the number of secondary buffer allocations allowed for your installation. You can also use the Spool Data Management section of your JMF report to monitor the size of the primary and secondary buffer allocations. See *z/OS JES3 Commands* for additional information about using the *INQUIRY command. Refer to *z/OS JES3 Diagnosis* for information about using JMF to diagnose spool.

Defining USAM Unprotected Buffers (UBUFs)

USAM unprotected buffers (UBUFs) are used for user address space I/O. Unlike JSAM and USAM protected buffers which are allocated from predefined buffer pools, UBUFs are allocated as pages of storage from the user's address space. User address spaces obtain UBUFs for each open SYSOUT data set as needed and release them when they are no longer required. As each buffer is filled, the data is moved into a USAM protected buffer which performs spool I/O. Buffer space for SYSIN data sets is fixed at one page. Use the USRPAGE keyword on the MAINPROC initialization statement to specify the number of pages available for each open SYSOUT data set.

Defining the Number of USAM Protected Buffers (PBUFs)

USAM protected buffers (PBUFs) are also used for output writer FSS and user address space I/O. JES3 moves user data from UBUFs to PBUFs before performing spool I/O. You can use the PRTPAGE keyword on the MAINPROC initialization statement to define the size of the PBUF buffer pool.

To determine the size of the PBUF buffer pool, start with parameter defaults and use the JES3 monitoring facility (JMF) to determine whether to increase or decrease the number of PBUFs, if necessary.

You can allocate PBUFs in the common service area (CSA) or in the JES3 auxiliary address space. If space in your CSA is constrained, allocate some of your PBUFs in the JES3 auxiliary address space. For performance considerations see “Determining How Many Buffers to Allocate in the JES3 Auxiliary Address Space” on page 274.

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 *z/OS HCD User's Guide*.

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 protected USAM buffers in the JES3 auxiliary address space to make more CSA available for application programs and for other subsystems.

To specify how you want JES3 to distribute protected USAM buffers (PBUFs) between CSA and the JES3 auxiliary address space, use the PRTPAGE keyword on the MAINPROC initialization statement.

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

Using the Writer Output Multitasking Facility

When the global 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. Only traditional (non-FSS) writers are eligible for writer output multitasking.

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`.

Part 10. Remote Job Processing

Chapter 10. JES3 Remote Job Processing . . .	279
Binary Synchronous Communication Remote Job Processing	279
Data Security Considerations	279
Data Compression	279
Operator Communications	280
Operator Commands Accepted from a BSC RJP Workstation	280
Debugging Facilities	280
BSC RJP Line Error Statistics	280
BSC RJP Line Snap Facility	280
BSC RJP Trace Facility	280
Initialization Statements that Affect BSC RJP	281
Systems Network Architecture Remote Job Processing	281
SNA RJP Implementation of SNA Concepts	281
Function Management Presentation Services (FMPS)	284
FM Inbound	284
FM Outbound	285
Presentation Service	285
Data Flow Control	286
JES3 to VTAM Interface	287
VTAM Considerations	287
Logon Mode Table	287
Initialization Statements that Affect SNA RJP	289
Maximum Record Lengths for SNA RJP Devices	289
Basic Exchange Support	289
Exchange Support	290
Exchange and Basic Exchange Initialization Considerations	290

Chapter 10. 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 workstation. 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 card reader (CR) DSP can be called to process input from a remote reader, and the writer (WTR) DSP can be called to send output to a remote printer or punch.

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 or the RACF command authority profiles defined when RACF is enabled. See “Using RACF to Authorize the Use of Operator Commands” on page 81 for additional information about using RACF to authorize the use of operator commands.

The following commands are used to initiate and cancel BSC RJP: *CALL,RJP; *START,RJP; *RESTART,RJP; and *CANCEL,RJP.

Operator Commands Accepted from a BSC RJP Workstation

You can enter several JES3 commands from BSC RJP terminals. You define a remote console’s authority using the LEVEL= keyword on a JES3 CONSOLE initialization statement. “Using JES3 to Authorize Commands from RJP Consoles” on page 89 describes how to define and use 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 *z/OS JES3 Diagnosis*.

BSC RJP Trace Facility

The BSC RJP event trace is always active when BSC RJP is active. The trace can be displayed in the hardcopy log by using the *MODIFY,T,L=lname,TRCEON command.

The hardcopy log 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 *z/OS 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.

You must specify at least one valid RJPLINE and RJPTERM statement in your JES3 initialization stream to activate BSC RJP. If you do not specify at least one valid RJPLINE and RJPTERM statement, JES3 issues error messages about initialization statements that refer to BSC RJP during initialization.

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 *VTAM General Information*.

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 27 shows the three SNA communication layers and the communication paths between the layers. Figure 28 shows the interface between JES3 and VTAM and the internal JES3 interfaces required to support SNA RJP.

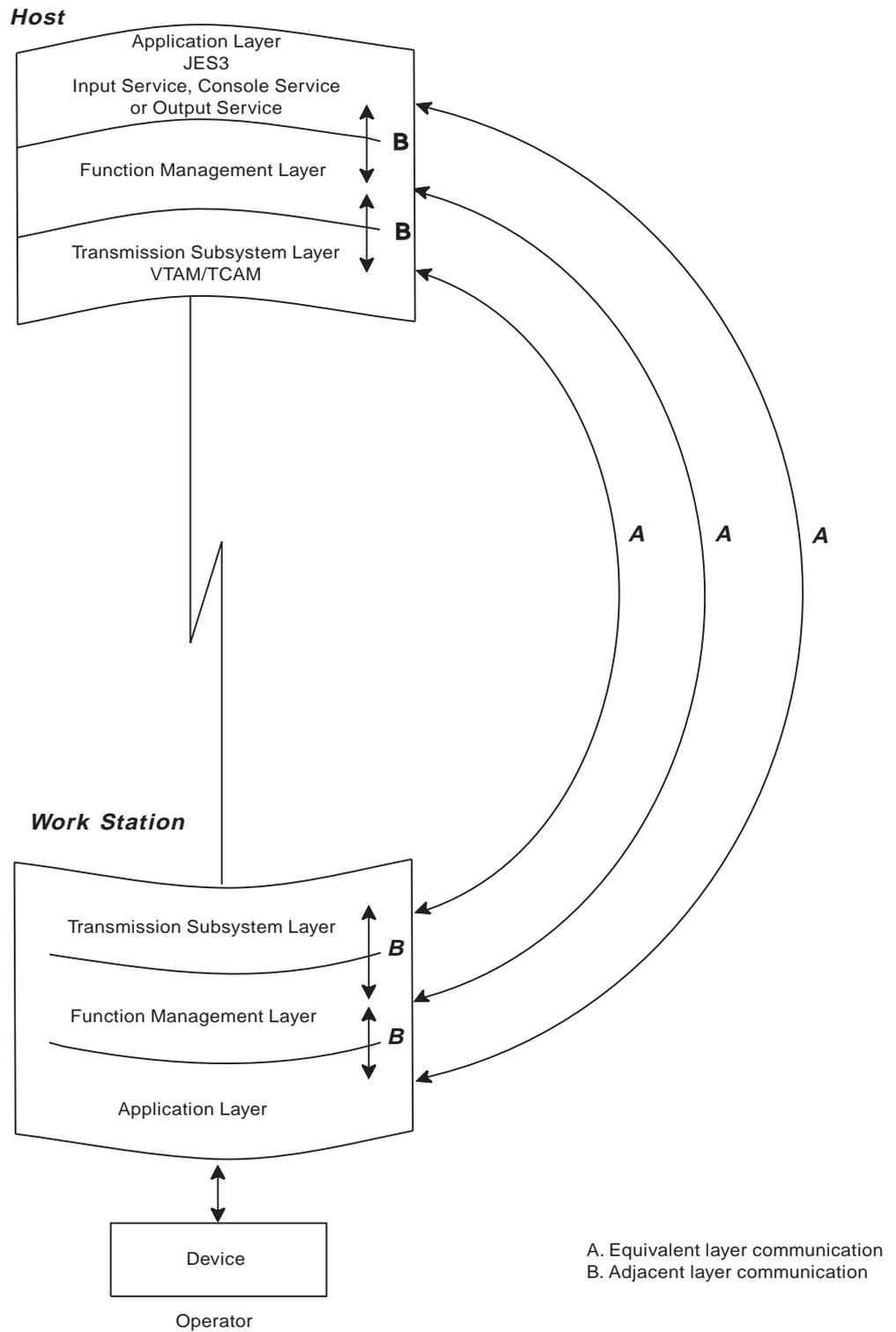


Figure 27. Overview of SNA Environment for JES3

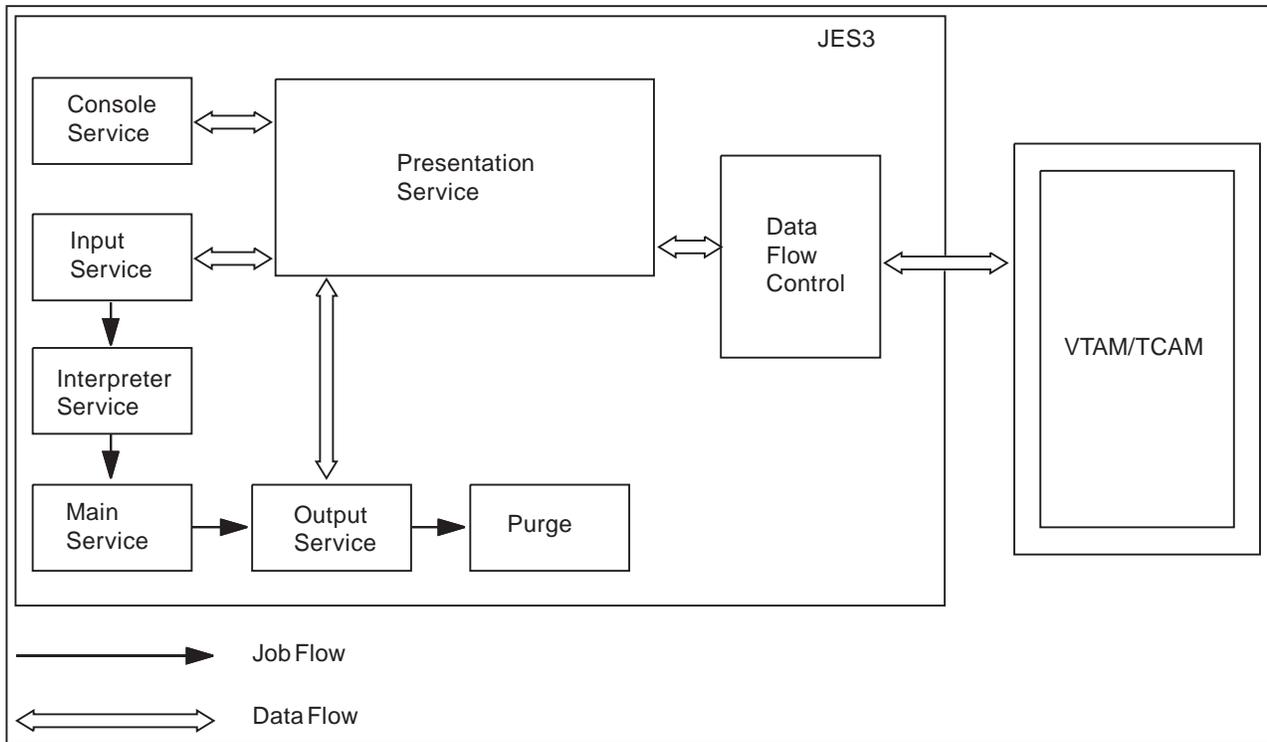


Figure 28. JES3-VTAM Interface

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 workstation 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. Table 47 shows the console and card reader control function characters that FM inbound routines support.

Table 47. SCS Function Characters Supported by FM Inbound Routines

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'

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 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 Table 48 for supported characters.

Table 48. SCS Function Characters Supported by FM Outbound Routines

Function	Source	Function Character
New Line (NL)	console	X'15'
Transparency (TRN)	reader	X'35'
Interchange Record Separator (IRS)		X'1E'
New Line (NL)	printer	X'15'
Carriage Return (CR)		X'0D'
Forms Feed (FF)		X'0C'
Select (SEL)		X'04'
Transparency (TRN)		X'35'
Set Vertical Format (SVF)		X'2BC2'

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. IBM recommends that the remote operator should enter a *RESTART command for the device to have the entire data set resent. Since no checkpoints are taken when sending a data set as one chain, the operator cannot start from the last checkpoint. This will ensure that the remote device receives all of the data. 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 *ACF/VTAM V3 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.

Logon Mode Table

A logon mode table contains one or more sets of session parameters representing session protocols to be used in a session. VTAM has an IBM-supplied logon mode table named ISTINCLM that provides generally accepted session protocols for a basic set of IBM device types.

The mode table entry contains information about:

- Alternate code (ASCII)
- Compression
- Compaction
- PDIR setup
- SLU pacing
- Card spanning
- Maximum RU size
- FM profile
- Data stream profile

The mode table entry contains a BIND image. JES3 optionally sets certain variable BIND parameters based on the setting of the corresponding parameter in the mode table entry specified for an LU.

You can modify or replace the IBM-supplied logon mode table, provided that the modified or replacement table has the same name as the IBM-supplied table and that you delete the IBM-supplied table. However, it is recommended that you create supplementary tables instead of deleting the IBM-supplied table, since the IBM-supplied table might be needed for problem determination.

You can create or modify logon mode tables using the MODETAB, MODEENT, and MODEEND macro instructions. See *VTAM Customization* for a description of these macros and their parameters. A logon mode table can be associated with a logical unit (LU) by specifying the table's name in the MODETAB operand of the LU's definition statement. If you do not designate a logon mode table for an LU, VTAM uses the IBM-supplied logon mode table.

You must assemble and link-edit each logon mode table into the MVS SYS1.VTAMLIB system library as an only-loadable data module. If you replace a logon mode table with a new one that has the same name, the devices that were using the old table will not automatically use the new table until you deactivate and subsequently reactivate the major nodes for these devices.

A MODEENT macro instruction associates a logon mode table with a set of parameters representing session protocols. See *ACF/VTAM V3 Programming* for a complete description of the bit settings for the parameter fields on the MODEENT macro.

The following list describes the parameters on the MODEENT macro which are significant to JES3:

- **FMPROF=X'03'**— This required parameter specifies the function management profile which is supported by SNA RJP to the work stations.
- **TSPROF=X'03'**— Represents the transmission profile which is supported by SNA RJP to the work station.
- **PRIPROT=X'Aa'**— Represents the application program-to-logical unit protocol.
 - A - Indicates multiple request unit chains. JES3 forces this field to A.
 - a - You must specify 1 or 3 as the value of this subparameter. Specifying 1 indicates primary transmission of an end bracket. Specifying 3 indicates primary transmission of end bracket and compression outbound (JES3 to work station).
- **SECPROT=X'Aa'** — Represents the secondary program-to-logical unit protocol.
 - A - Indicates multiple request unit chains. JES3 forces this field to A.
 - a - You must specify 1 or 3 as the value of this subparameter. Specifying 1 indicates primary transmission of an end bracket. Specifying 3 indicates primary transmission of end bracket and compression outbound (work station to JES3).
- **COMPROT=X'7b80'** — Represents the common logical unit protocols (regardless of direction).
 - 7 - Indicates function management headers, brackets, and conditional bracket termination are used. JES3 forces this field to 7.
 - b - Substituting 1 for the variable 'b' causes alternate code to be used. Substituting 0 causes alternate codes not to be used.
 - 80 - Indicated that half-duplex, flip-flop transmission with secondary winning contentions is desired. JES3 forces this field to 80.

- **RUSIZES=X'8585'** — Specifies a minimum request unit size of 256. If you specify a request unit size less than 256, JES3 defaults to 256.
 - 8585** - Specifies a maximum request unit size of 572.
 - meme** - Specifies a maximum request unit size of m*2**e. This number must be between 256 and 4096.
- **PSERVIC=X'cc1ed000B100'X'CO'0000X'01'00X'f0'**— Specifies device characteristics.
 - cc** - The value can be 00 or 01. This required subparameter specifies logical unit profile one.
 - e=0** - Indicates that basic control and cards can not span request units.
 - =1 indicates that basic control and cards span request units.
 - d=0** - Indicates that the primary will not compact or send PDIR.
 - =2 indicates that the primary will not compact but will send PDIR.
 - =4 indicates that the primary will compact but will not send PDIR.
 - =6 indicates that the primary will compact and send PDIR.
 - f=0** - Indicates that card media is not supported inbound.
 - =4 indicates that card media is supported inbound.

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.

Maximum Record Lengths for SNA RJP Devices

The following maximum record lengths apply to devices:

- Consoles and Printers - 132 bytes
- Card Readers - 80 bytes
- Card Punches - 255 bytes

Basic Exchange Support

Basic exchange allows a user to transmit print output from the host to a workstation and then within the workstation 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=BE 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 workstation. It then allows the workstation 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 workstation 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 workstation 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 workstation 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.

Part 11. JES3 Networking

Chapter 11. JES3 Networking	293
Networking Protocols	293
BSC Versus SNA	293
BSC	294
SNA	294
Converting Networking Protocols	294
Types of Nodes.	295
Defining the Home Node	296
Defining a Remote Node	297
Specifying a Communications Path for	
Indirectly-Connected Nodes	298
Defining an Alias	299
Defining a Spool Partition	299
BSC Considerations	300
Defining the Buffer Size	300
Specifying Passwords.	300
Defining BSC Communication Lines.	301
DEVICE Statement	302
NJERMT Statement	302
DESTDEF Statement	304
Logical Senders: How JES3 Names Them	304
SNA Considerations	306
Changing Node Definitions.	306
How Restarts Affect Networking Jobs	306
JES3 Hot, Hot Start with Refresh, or Warm	
Start	306
JES3 Cold Start.	306
MVS/BDT Hot or Warm Start.	306
MVS/BDT Cold Start.	307
Rerouting Jobs and SYSOUT	307
Networking Job Numbers	307
Defining a Network Message Destination	307
Monitoring the Job Entry Network with	
Installation Exits	307
Monitoring Files Sent via TSO/E TRANSMIT or	
CMS SENDFILE	310
Deleting Files from the Spool	311

Chapter 11. 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
- A spool partition defining the spool partition to be used to spool the incoming NJE stream

To monitor data sent to or from your node, you can use installation 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

You can use RACF to perform security checking on inbound work from remote nodes. See “Authorizing Network Jobs and SYSOUT (NJE)” on page 56 for information about using RACF to protect network jobs and data.

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 Versus SNA

There are two ways to send and receive data in NJE. One is binary synchronous communication (BSC) and the other is Systems Network Architecture (SNA).

A network can consist of any combination of SNA and BSC connections. Each of these has its advantages. Your choice of which to use depends on the available hardware and software.

BSC

BSC protocol allows data flow between nodes over physical links, which are typically channel-to-channel adapters, or leased or dial-up telephone lines.

BSC is a form of line control that uses a standard set of control characters and control character sequences for transmission of binary coded data between systems in a network. A BSC node uses:

- Remote terminal access method (RTAM) execute channel programming (EXCP) and emulation program (EP) or partitioned emulation programming extension (PEP) in the communications controller
- Channel-to-channel (CTC) connections are identical to BSC communications except CTC connections do not use EP or PEP. CTC connections are best suited for connections to nodes within the same computer facility.

NJE protocols support an ESCON Basic Mode CTC (defined to the hardware configuration dialog as BCTC) and a 3088 CTC, but do not support an ESCON CTC (defined to the hardware configuration dialog as SCTC).

SNA

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.

SNA is the description of logical structures, formats, protocols, and operational sequences, for transmitting information units through, and controlling the configuration of, networks. A SNA node uses:

- Virtual telecommunication access method (VTAM) to control the sessions between the nodes.
- The network control program (NCP) or partitioned emulation programming (PEP) extension to control lines.

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 to MVS/BDT in the MVS/BDT initialization stream and to JES3. You can define node names to JES3 using operator commands or in the JES3 initialization stream. 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*.

You can code the NJERMT initialization statement to define networking nodes to JES3 during initialization. See Table 49 on page 296 to determine which parameters you must code on the NJERMT initialization statement. If you have more than one MVS/BDT facility in your JES3 complex, you should also include a JES3 SYSID initialization statement to define the default MVS/BDT facility. You can also define networking nodes dynamically using the JES3 *MODIFY,NJE command. *z/OS JES3 Commands* describes how to use the the *MODIFY,NJE command.

Once the nodes are defined in both initialization streams, you can warm start or hot start with refresh JES3 and warm start 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.

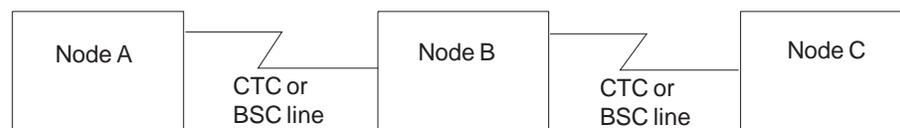
Types of Nodes

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 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.

Table 49 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.

Table 49. Parameter Requirements for the NJERMT Statement

NJERMT Parameter	Type of Node			
	Home Node	Alias	Directly Connected Remote Node	Indirectly Connected Remote Node
ALIAS		R		
AUTO			O (BSC)	
BDTID	O (SNA)			
BFSIZ			O (BSC)	
CTC			O (BSC)	
DEFCLASS	O	O		
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)	(See Note 6)	R (BSC,SNA)	R (BSC,SNA)
NETHOLD	O			
NJEPR	O (BSC)			
NJEPU	O (BSC)			
PATH			(See Note 5)	R (BSC,SNA)
PRTDEF	O (BSC,SNA)	O (BSC,SNA)		
PRTSO	O (BSC,SNA)	O (BSC,SNA)		
PRXWTR	O (BSC,SNA)	O (BSC,SNA)		
PUNDEF	O (BSC,SNA)	O (BSC,SNA)		
PWCNTL			O	O
PWD			O (BSC)	
RDLY			O (BSC)	
SIG			O (BSC)	
STREAM			O (BSC)	
TYPE			O (BSC,SNA)	
XNAMEREQ	O	O		

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.
6. NAME and ALIAS cannot be specified on the same NJERMT statement.

Legend:

R - Required parameter
O - Optional parameter

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.

To influence how the home node handles NETDATA, you code the NETHOLD= keyword parameter to indicated whether NETDATA is held at the node or released to JES3 writers.

To influence how the home node handles incoming SYSOUT whose characteristics, as specified by the sending node, do not match the characteristics of the SYSOUT class, you code the DEFCLASS or XNAMEREQ parameter. "Characteristics" includes whether the SYSOUT is print or punch output, and whether it is held for TSO, held for external writer, or not held.

DEFCLASS determines whether default SYSOUT classes may be assigned. PRTDEF, PUNDEF, PRTTSO, and PRTXWTR define the default classes. If default SYSOUT classes may be assigned, then XNAMEREQ determines how SYSOUT, that is defined as held for external writer, should be handled if no external writer name is present.

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. You can define a remote node dynamically using the JES3 *MODIFY,NJE command. *z/OS JES3 Commands* describes how to use the *MODIFY,NJE command.

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 dial-up 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 "BSC Considerations" on page 300 and "SNA Considerations" on page 306.

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,...
```

Defining an Alias

You can define an alias, that is, an alternate name for a node by which work received by the home node is recognized as being destined for the home node. You do this with the `ALIAS=aliasnam` parameter on the NJERMT initialization statement. The NJERMT initialization statement you use to define this alias name is the same one you use to define the complex on which you intend the alias name to be recognized. For example, if you define BROOKLYN as an alias of the home node NEWYORK, you code the following NJERMT initialization statements.

```
NJERMT,NAME=NEWYORK,HOME=YES  
NJERMT,ALIAS=BROOKLYN
```

To define the BROOKLYN alias to nodes, which define NEWYORK indirectly, you use the `PATH` parameter. For example, on an adjacent node to NEWYORK, you would code the following.

```
NJERMT,NAME=BROOKLYN,PATH=NEWYORK
```

These NJE connections between NEWYORK and the adjacent node can be either BSC or SNA.

Note: You define an alias on the receiving node only and not for the sending node or any intermediate store-and-forward nodes.

When defining an alias, you can also specify default SYSOUT classes for print and punch output that is received from other nodes. As with the home node, the default is A for print classes and B for punch classes.

To influence how the home node handles incoming SYSOUT whose characteristics, as specified by the sending node, do not match the characteristics of the SYSOUT class, you code the `DEFCLASS` or `XNAMEREQ` parameter. "Characteristics" includes whether the SYSOUT is print or punch output, and whether it is held for TSO, held for external writer, or not held.

`DEFCLASS` determines whether default SYSOUT classes may be assigned. `PRTDEF`, `PUNDEF`, `PRTTSO`, and `PRTXWTR` define the default classes. If default SYSOUT classes may be assigned, then `XNAMEREQ` determines how SYSOUT, that is defined as held for external writer, should be handled if no external writer name is present.

Defining a Spool Partition

The `SPART=` keyword on the NJERMT initialization statement can be used to define a spool partition for NJE streams (SYSIN and SYSOUT) received from the defined node. The `SPART=` keyword can be used for both BSC or SNA defined nodes. The `SPART` parameter specified on the NJERMT statement must correspond to the `NAME` parameter specified on a `SPART` statement.

When SYSIN or SYSOUT is received from a node, `SPART` information defined for the origin node will be used in the spooling of the SYSIN and SYSOUT stream.

Refer to *z/OS JES3 Commands* for NJE Inquiry (*I NJE) and Modify (*F NJE) commands for inquiring and changing the `SPART` definition for a NJE node.

BSC Considerations

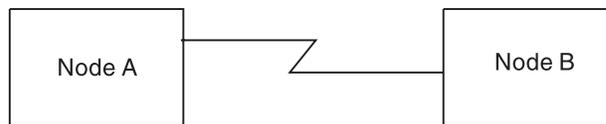
The following topics apply to defining nodes that are using the BSC networking protocol.

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.

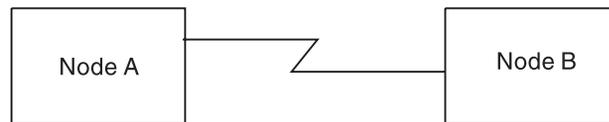
If you do not provide the correct passwords, your node and the remote node will be unable to communicate. The operator at your node will receive no messages to indicate that anything is wrong. The operator at the remote node will receive a message indicating that your node sent an incorrect password.

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.

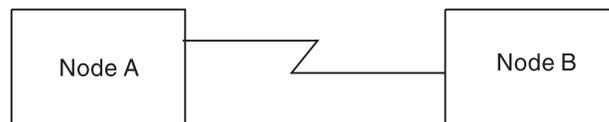
```
NJERMT,NAME=B,PWD=P1,SIG=P2,...
```



```
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 can use the NUMDEV parameter to define a range of lines. If you do, the JNAME parameter specifies a prefix of up to four characters, which, combined with the JUNIT as a four digit device, makes up 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
- You can use the *ALL system name within the JUNIT parameter to define simultaneously a line to all systems.
- 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
- Specify whether the incoming NJE SYSOUT data sets that appears to be NETDATA output should be held for a TSO user or released to JES3 writers.

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.

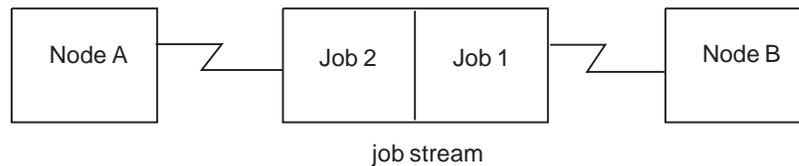
If you want JES3 to transmit one complete networking job at a time, code `STREAM=1`. If you want JES3 to interleave the transmission of networking jobs, code `STREAM=2`. These are the only valid subparameter values for the `STREAM=` parameter. (However, `STREAM=2` is not allowed for directly connected, remote BSC nodes, running RSCS under VM.)

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.

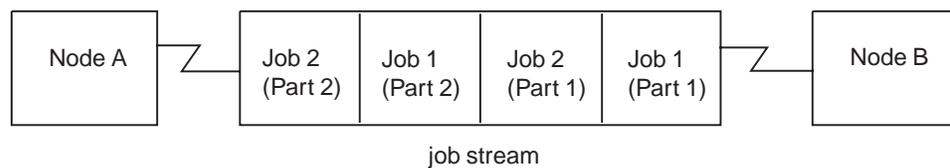
Example 1:

`NJERMT,NAME=B,STREAM=1,...`



Example 2:

`NJERMT,NAME=B,STREAM=2,...`



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.

To specify how incoming NJE SYSOUT data sets that appear to be NETDATA should be handled, code the NETHOLD= keyword parameter. Code NETHOLD=YES to indicate that the NJE SYSOUT is to be held for a TSO user. Code NETHOLD=NO or do not code it and the NJE SYSOUT data set will be released to JES3 writers.

Note: NETHOLD= can only be specified on the NJERMT initialization statement that defines the HOME node.

DESTDEF Statement

You use the DESTDEF initialization statement to define how inbound SYSOUT data sets from other NJE nodes are to be processed at this node.

If you want inbound NJE SYSOUT data sets that are to be processed by "named" devices, you code the DEVICE= keyword parameter to indicate the device. The inbound data sets are then placed on the writer queue (and not made available for TSO RECEIVE) for processing by JES3 writers.

If you have inbound NJE SYSOUT data sets destined for "named" users, you code the USERID= keyword parameter to indicate the users who are to receive the data sets. The inbound SYSOUT data sets are then placed on the hold queue and made available for the TSO RECEIVE user you have specified.

The following example illustrates DESTDEF use. This example specifies that inbound NETDATA destined for FRED or SUE that should be made available for TSO RECEIVE even if the inbound NETDATA matches the JNAME of a JES3 device. The inbound NETDATA destined for OEMPRTxx should be placed on the writer queue even if there is no JES3 device with a matching JNAME. The DESTDEF follows:

```
DESTDEF,USERID=(FRED,SUE),DEVICE=(OEMPRT*)
```

See *z/OS JES3 Initialization and Tuning Reference* for more information on using the DESTDEF initialization statement.

Logical Senders: How JES3 Names Them

To vary a node online or offline, you must specify the names of 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 29 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.

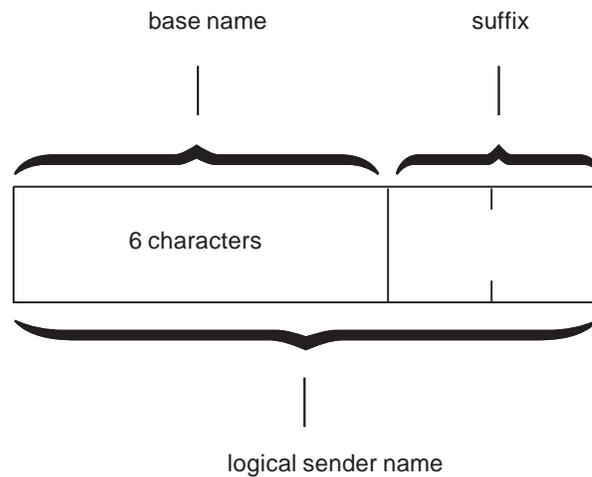


Figure 29. How JES3 Creates a Logical Sender Name

Table 50 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.

Table 50. The Number of Logical Senders Created and Suffixes Used for Valid Combinations of the MAXLINE and STREAM Parameters

	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

Table 50. The Number of Logical Senders Created and Suffixes Used for Valid Combinations of the MAXLINE and STREAM Parameters (continued)

	MAXLINE=	0	1	2	3
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.					

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, Hot Start with Refresh, or Warm Start

When JES3 requires a hot, hot start with refresh, or warm start, the networking jobs or the 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. Changes that you make to your network using the JES3 *MODIFY,NJE command are not saved across a restart.

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.

Note: If a node is using the BSC protocol, SYSOUT streams can be rerouted using the *F,U,Q=WTR,ND= command.

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.

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 Destination

You can define the message destination to which JES3 is to send network messages. To do this, code the message destination on the CLASS parameter on the NJECONS statement. The message destination can be specified as a JES3 destination class or as an MVS routing code. If you do not specify a message destination, JES3 sends all network messages to class S12 (the routing equivalent of S12 is 108).

A console receiving the routing code or the routing code equivalent of the destination class will receive network messages.

Monitoring the Job Entry Network with Installation Exits

Several installation 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

Table 51 lists each installation exit, states when JES3 passes control to the exit, and states what the exit routine can do.

Table 51. Network Installation Exit Summary

Installation Exit	Receives Control:	The exit routine can:
IATUX35	After your node receives commands from another node.	<ul style="list-style-type: none"> • 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".
IATUX43	Before your node sends a SYSOUT stream to another node.	Inspect or change the job header.

Figure 30, Figure 31, and Figure 32 group the installation exits according to the type of events that cause the exit routines to receive control.

- Figure 30 lists the installation exits that receive control when your node sends or receives a networking job.
- Figure 31 lists the installation exits that receive control when your node sends or receives a SYSOUT stream.
- Figure 32 lists the installation exit that receives control when your node receives a command from another node.

When a figure lists more than one installation exit, they are listed in the order they receive control.

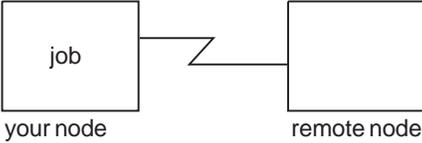
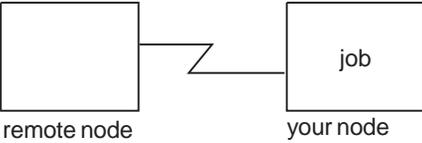
When	JES3 at your node invokes user exit
<p>a job that was submitted at your node is to be sent to a remote node.</p> 	IATUX40
<p>your node receives a job from a remote node and your node is to execute the job.</p> 	IATUX36

Figure 30. Job Related Installation Exits

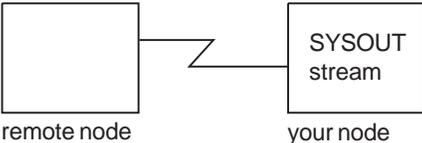
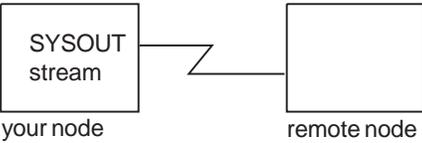
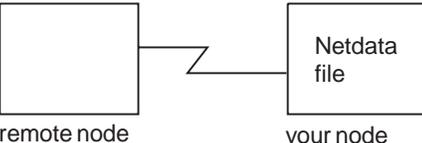
When	JES3 at your node invokes user exit
<p>your node receives a SYSOUT stream that it is to process.</p> 	IATUX36 IATUX37 IATUX38
<p>your node sends a SYSOUT stream to a remote node.</p> 	IATUX39 IATUX43
<p>your node receives a file that was transmitted from another node by means of the TSO/E TRANSMIT or CMS SENDFILE command.</p> 	IATUX36 IATUX37 IATUX38 IATUX42

Figure 31. SYSOUT Related Installation Exits

When	JES3 at your node invokes user exit
your node receives a command from another node and your node is to execute the command.	IATUX35

Figure 32. Command Related Installation 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 *z/OS TSO/E Command Reference*. 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 installation 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 *z/OS JES3 Customization*.

The system may not allow you to transmit a data set that it determines to be too large. The system compares the values you specify for the output limit on the TSO/E OUTLIM parameter with the value you specify on the CARDS= parameter of the JES3 STANDARDS statement. When SYSOUT is transmitted in the foreground for started tasks and TSO/E logons, the system uses the lower of these two limits. JES3 sets the following output limits for started tasks and TSO logons:

- 999,999 for lines, cards, and pages
- 2,147,483 (in thousands of bytes) for spool utilization.

An installation can change the limits for started tasks or TSO logons by using JES3 Exit 20 to change the limit for each particular started task or TSO logon. The limit for TSO/E transmits which are specified through the OUTLIM parameter, should not be greater than the limit JES3 sets for punches or a X'722' abend will occur. See *z/OS TSO/E Customization* for information about limiting the TSO/T transmit command.

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 installation 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

Part 12. JES3 IBM Tape Library Dataserver

Chapter 12. JES3 IBM Tape Library Dataservers	315
JES3 Support Overview	315
Allocation for 3495 Tape Data Sets	315
JES3 Initialization Statements	315
SETNAME Statement.	316
HWSNAME Statement	317
DEVICE Statement	317
3495 Configuration Example	318
MVS UNITNAME Definitions	318
SETNAME Statements	319
HWSNAME Statements	319
DEVICE Statements	320
Operational Considerations.	320
Unchanged Functions	321
Modified Functions	321
JES3 Limitations	322
Mount Failures	324
IBM 3590 Installed in IBM Tape Library	
Dataservers	324
Integration into Existing Tape Library	325
JES3	326
ATLDS JES3	327
Tape Verification Processing (IATLVVR	
Module)	327
General Considerations for Migration	328
IBM 3590 Support for JES3	328

Chapter 12. JES3 IBM Tape Library Dataservers

This section describes JES3 support for the 3494/3495 Tape Library Dataservers provided with DFSMS/MVS Version 1 Release 1 and JES3. This support enables a JES3 installation to manage tape drives within an IBM 3494/3495 Tape Library Dataserver. (This section references the IBM 3495 Tape Library Dataserver. However, all references to the IBM 3495 Tape Library Dataserver imply both the IBM 3494 and 3495 Tape Library Dataservers.)

JES3 Support Overview

JES3 supports the definition and usage of one or more IBM 3494/3495 Tape Library Dataservers within a JES3 complex. The tape drives within a given library are subsetted into library device groups (LDGs) based upon the device type. JES3 uses specific esoteric names to direct tape allocations to devices within the correct library when the tape resides in a library, and to drives outside of a library when the tape is not resident within a library.

The esoteric names for devices comprising a LDG are conveyed to JES3 through the SETNAME initialization statement. These esoteric names, in conjunction with the SETNAME statement, allow JES3 to associate the devices to a device group, the device groups to a library, and the libraries to a JES3 complex.

Allocation for 3495 Tape Data Sets

DFSMS/MVS directs new data sets to the IBM 3495 Tape Library Dataserver using the data class, storage class, and storage group information specified in JCL or provided by Automatic Class Selection (ACS). Storage class identifies the data set as DFSMS/MVS managed mountable. Data class is used to determine the required cartridge and device type. Storage group is used to determine the eligible 3495 libraries. Requests for old data sets are directed to a 3495 only when the volume or volumes containing the data set are located within a 3495. Eligible device types are determined from the cartridge type and recording format.

JES3 Initialization Statements

Your initialization deck must include JES3 SETNAME, DEVICE and HWSNAME statements for the 3495. These statements define the 3495 to JES3, ensuring that non-library requests are not allocated to 3495 tape drives.

Neither JES3 nor DFSMS/MVS verifies that a complete and accurate set of initialization statements are defined to the system. A 3495 definition that is either incomplete or inaccurate may result in jobs failing allocation or other unpredictable results.

Prior to defining the devices within a library to JES3, they must be properly defined to MVS using the hardware configuration definition (HCD) program. Unlike a JES2 environment, a JES3 operating environment requires the specification of esoteric unit names for the devices within a library. These unit names will be used in the required JES3 initialization statements.

Each device within a library must have exactly four (4) special esoteric names associated with it. These are:

1. The complex-wide name, this is always: "LDGW3495".
2. The library-specific name, this is an 8-character string composed of "LDG" prefixing the 5-digit library identification number.
3. The complex-wide device type, this is either: "LDG3490", "LDG3490E", or "LDG3591" depending on the model of the device.

Note: LDG3591 is the complex-wide device type name for 3590-1.

4. A library-specific device type name. For a 3490 device, this is the 8-character string "LDD" prefixing the 5-digit library device number of the library containing the device. For a 3490E device, this is the 8-character string "LDE" prefixing the 5-digit library device number. For a 3590-1 device, this is the 8-character string "LDB" prefixing the 5-digit library device number.

These esoteric names should be defined and the input for the JES3 initialization stream checker obtained prior to making changes to the JES3 initialization stream.

SETNAME Statement

Include a SETNAME statement for each XTYPE. The NAMES list for the SETNAME statement must include all required library device groups. It must not include any generic or esoteric names (such as 3490, 3480X, SYS3480R). The omission of these esoteric and generic names is required to prevent tapes from being mounted on unsuitable devices. The names specified must be defined to MVS as esoteric names.

The grouping of library devices into XTYPE groups is the same process for all other JES3 managed devices. The XTYPE groups are determined by the special esoteric names assigned during the HCD processing and associating the esoteric name to a device group. Use the following rules when composing your 3495 SETNAME statements:

1. The complex-wide library name used on all statements is "LDGW3495".
2. A library-specific name is included for all XTYPEs within a referenced library. The name has the prefix "LDG" followed by your 5-character 3495 library sequence number.

Note: Your library sequence number is printed just below the logo plate of the 3495, on the side containing the convenience I/O station.

3. A complex-wide device name is included for all XTYPEs associated with a specific device type used in the complex. The name has the prefix "LDG" followed by either "3490" or "3490E", or "3591".
4. A library-specific device name is included for all XTYPEs associated with the devices of the specified device type within a given library. For 3490E drives, the name has a prefix of "LDE" followed by the 5-character library sequence number. For 3590-1, the name has a prefix of "LDB" followed by the 5-character library sequence number.

The following guidelines will ensure correct generation of the SETNAME statements.

- There are always four esoteric names in the NAMES= list. These names are the special esoteric names that are used to identify library resident devices.
- There can never be esoteric names pertaining to different device types in the same list.
- There can never be two esoteric names for 3490, 3490E or 3590 device identifiers in the same list.

- There must be a LDEnnnnn value and vice versa if an LDG3490E exists. This is true for LDG3490 and LDDnnnnn, and LDG3591 and LDBnnnnn also.

Following is an example of a SETNAME statement. It uses a library sequence number of "123DE":

```
SETNAME, XTYPE=LB13490E, NAMES=(LDGW3495, LDG123DE, LDG3490E, LDE123DE)
```

•	•	•	•
•	•	•	•
•	•	•	•
Complex	Library	Complex	Library
Wide	Specific	Wide	Specific
Library	Library	Device	Device
Name	Name	Name	Name

HWSNAME Statement

Build the HWSNAME statements for your library device groups using the following rules:

1. The complex-wide library name includes all other LDGs as alternates.
2. The library-specific name includes all LDGs for the corresponding library as alternates. The complex-wide device type name is also included as an alternate when all tape devices of a specific type are housed within a single 3495.
3. The complex-wide device type name includes all library-specific device type names. When all devices of a specific type are contained within a single 3495, the complex-wide device type name is the same as the library-specific name. In this case, the library-specific name should also be specified as an alternate.
4. The library-specific device type name includes the following alternate names:
 - a. When all drives within the 3495 are of the same device type, the library-specific device type name is the same as the library name. The library-specific name should be specified as an alternate.
 - b. When these are the only drives of this type in the complex, the complex-wide device type name is the same as the library-specific device type name. Therefore, the complex-wide device type name is specified as an alternate.

The following guidelines will ensure correct generation of the HWSNAME statements.

- There is a one to one correlation to HWSNAME statements as to special esoteric names defined in HCD.
- There can never be esoteric names in the TYPE= lists that are not one of the special esoteric names. Similarly, your existing HWSNAME statements for tape devices must not have any of these special esoteric names in their TYPE= lists.

Following is an example of a HWSNAME statement:

```
HWSNAME, TYPE=(LDGW3495, LDG123DE, LDG3490E, LDE123DE)
```

DEVICE Statement

Include a DEVICE statement for each tape library drive in the complex. The XTYPE name must be unique for each device type within a library and cannot span libraries. You should not specify the 3495 as a JUNIT, or use the DTYPE or JNAME parameters. The absence of the DTYPE, JNAME and JUNIT parameters will prevent inadvertent and unsuccessful use of a library device by a JES3 dynamic support program (DSP).

Following is an example of a DEVICE statement:

```
DEVICE,XTYPE=(LB13490E,CA),  
XUNIT=(590,SYSTEM1,S3,OFF,590,SY2,S3,OFF,590,SY3,S3,OFF)
```

X

Note: You could also use NUMDEV and/or *ALL to more conveniently define your devices.

3495 Configuration Example

The configuration described in this section is a JES3duplex with two 3495s attached to it. An illustration of the configuration and associated JES3 initialization statements are provided as examples to guide you in the installation and initialization of the 3495 on your own system.

Figure 33 illustrates the two 3495 libraries defined for this configuration. Library 1 (sequence number 123DE) has only 3490E tape drives. Library 2 (sequence number 123DF) has both 3490 and 3490E drives.

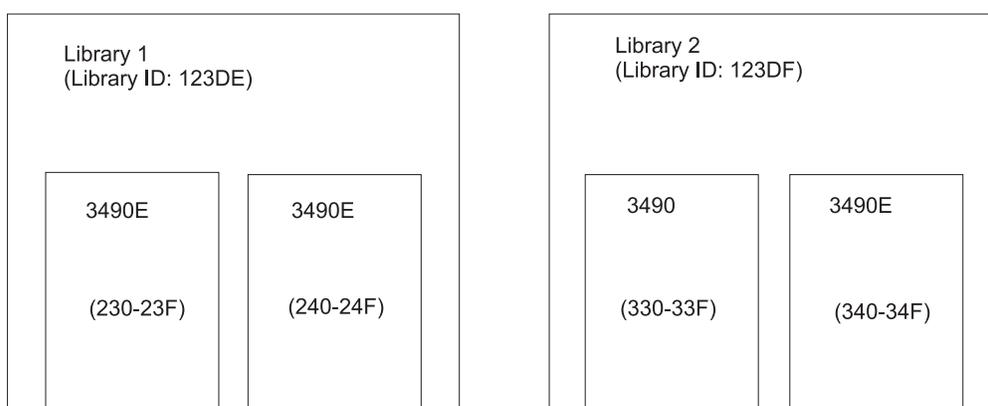


Figure 33. 3495 Libraries Defined for the Configuration Example

MVS UNITNAME Definitions

MVS UNITNAMES must be defined for all 3495 library device groups. Figure 34 shows the UNITNAMES defined for the library device groups in the configuration example. Note the following:

- The complex-wide name, LDGW3495, includes all devices in all libraries.
- The library-specific names, LDG123DE and LDG123DF, include devices from their respective libraries.
- The complex-wide device type names, LDG3490 or LDG3490E, include all devices of the same type, in all libraries.
- The library-specific device type names, LDD123DF, LDE123DF, and LDE123DE include all devices of the same type within their respective libraries.

Unit Name	Device Address Range
LDGW3495	230-23F,240-24F,330-33F,340-34F
LDG123DE	230-23F,240-24F
LDG123DF	330-33F,340-34F
LDG3490	330-33F
LDG3490E	230-23F,240-24F,340-34F
LDD123DF	330-33F
LDE123DF	340-34F
LDE123DE	230-23F,240-24F

Figure 34. MVS UNITNAMEs Defined for the Configuration Example

SETNAME Statements

SETNAME statements for the two libraries are shown in Figure 35. Note that:

- Three XTYPEs are required because of different unit names used for 3490E devices in the two libraries.
- Ordinary esoteric or generic unit names such as 3480, 3480X, 3490, SYS3480R and SYS348XR, are not specified.
- Installation specific esoteric names such as TAPE or CART are also not used to describe the library devices.

```

SETNAME,XTYPE=(LB13490E,TA),
      NAMES=(LDGW3495,LDG123DE,LDG3490E,LDE123DE)
      .
      .
SETNAME,XTYPE=(LB23490,TA),
      NAMES=(LDGW3495,LDG123DF,LDG3490,LDD123DF)
      .
      .
SETNAME,XTYPE=(LB23490E,TA),
      NAMES=(LDGW3495,LDG123DF,LDG3490E,LDE123DF)

```

Figure 35. SETNAME Statements for the Configuration Example

HWSNAME Statements

HWSNAME statements for the two libraries are shown in Figure 36. Note the alternate names used in the HWSNAME statements.

```

HWSNAME,TYPE=(LDGW3495,LDG123DE,LDG123DF,LDE123DE,LDE123DF,
              LDD123DF,LDG3490,LDG3490E)
.
HWSNAME,TYPE=(LDG123DE,LDE123DE)
.
HWSNAME,TYPE=(LDG123DF,LDE123DF,LDD123DF,LDG3490)1
.
HWSNAME,TYPE=(LDE123DE,LDG123DE)2
.
HWSNAME,TYPE=(LDE123DF)
.
HWSNAME,TYPE=(LDD123DF,LDG3490)1
.
HWSNAME,TYPE=(LDG3490,LDD123DF)
.
HWSNAME,TYPE=(LDG3490E,LDE123DE,LDE123DF,LDG123DE)2
.
.
.

```

Notes:

1. LDG3490 is a valid alternate for LDG123DF and LDD123DF because there are no 3490 drives in the other library.
2. LDG123DE is a valid alternate for LDG3490E and LDE123DE because that library contains only 3490E drives.

Figure 36. HWSNAME Statements for the Configuration Example

DEVICE Statements

DEVICE statements for the two libraries are shown in Figure 37. Note that none of the DEVICE statements have DTYPE, JNAME or JUNIT values. This prevents the devices from being used by JES3 dynamic support programs, which do not work with 3495 drives. These are strictly execution devices.

```

DEVICE,XTYPE=(LB13490E,CA),
XUNIT=(230,*ALL,S3,OFF),NUMDEV=32
*
DEVICE,XTYPE=(LB23490,CA),
XUNIT=(330,*ALL,S3,OFF),NUMDEV=16
*
DEVICE,XTYPE=(LB23490E,CA),
XUNIT=(340,*ALL,S3,OFF),NUMDEV=16

```

Figure 37. DEVICE Statements for the Configuration Example

Operational Considerations

Most JES3 processing capabilities are fully supported for the 3495 Tape Library Dataserver; however, certain commands and functions are modified. Although no JCL changes are required, there are a few processing restrictions and limitations associated with using the 3495 in the JES3 environment. These operational considerations are described in this section.

Unchanged Functions

The following JES3 functions and interfaces are fully supported without change for the 3495:

- Volume unavailable table
- Dynamic device reconfiguration (DDR)
- High water mark setup (HWS)
- Unit affinity
- Operator commands
- Drive sharing

As long as the SMS complex and JES3 complex are the same, the 3495 can be shared among all systems in a JES3 complex.

- Device fencing

Fencing capabilities include job group and dependent job control (DJC) net jobs.

Although you can fence library drives, you must ensure that the drives fenced are eligible to satisfy the volume request. For example, JES3 may fence two drives in a pool for a particular library. If the job requires a volume from a different library, the job may not be able to use the two drives.

Modified Functions

Some of the following JES3 functions and messages have been modified for 3495 processing; others have a different meaning in the 3495 environment:

- Pre-execution setup
- Dynamic allocation
- Verification
- Fetch messages
- Mount messages
- Breakdown Messages
- Online and Offline Status
- Dump Job DSP

Pre-execution setup

Drive, volume, and data set allocation are performed for pre-execution setup; however, volume mount and verification is deferred until job execution. Setup continues to select a device from the library device group that best meets the requirements of the data.

Dynamic allocation

Changes to dynamic allocation are the same as those described for pre-execution setup.

Verification

With 3495 mount processing deferred until execution, JES3 verification is no longer performed. JES3 verification is not needed because human error is no longer a factor and extensive verification is already performed by OPEN processing.

Note: If all mounts within a job are deferred, the job will not be included in the setup depth (SDEPTH) parameter. This could impact your ability to control the JES3 workload.

Fetch messages

The purpose of fetch messages is to alert operations of an impending need for specific tape cartridges so that the tapes can be pulled from library shelves and mounted before the requests are processed. Because cartridges in the 3495 are already located within arms' reach of a robotic accessor, fetch messages to the operator are no longer necessary.

All fetch messages (IAT5110) for 3495 requests are changed to the informational “USES” form of the message. Unlike the “GET” form, no action is required for these messages. The fetch messages are routed to the same console destination as other “USES” type fetch messages. This destination is specified on the MDSLOG parameter of the SETPARAM initialization statement.

Mount Messages

Mount messages are suppressed for the 3495.

Breakdown Messages

Breakdown messages issued for 3495 volumes are irrelevant and could confuse operations personnel. Therefore, breakdown messages IAT5410 and IAT5420 for 3495 requests are not issued to the operator’s console. They will continue to be written to the JES3 message log (JESMSGLOG) data set.

Online and Offline Status

Devices will automatically be varied offline to JES3 if the operator uses the VARY SMS,LIBRARY(libname),OFFLINE command. Devices will automatically be varied online to JES3 if the operator uses the VARY SMS,LIBRARY(libname),ONLINE command. Refer to *MVS/DFP Object Access Method Planning, Installation, and Storage Administration Guide* for more information concerning the VARY SMS command.

Note: Other components, such as OAM, can vary devices online or offline to JES3 by issuing the VARY SMS, LIBRARY (libname) command

Dump Job DSP

The dump job DSP runs in two modes: **non-server mode** and **server mode**. In non-server mode the installation uses the JUNIT and XUNIT initialization parameters of the DEVICE initialization statement to define the tape units the dump job will use. These tape units cannot be IBM 3494 and 3495 Tape Library Dataserver. In server mode the dump job process runs in its own address space (not in the JES3 address space) and can use any tape device in the MVS/JES3 system including the IBM 3494 and 3495 Tape Library Dataserver. In server mode, the JES3 installation can remove the JUNIT and XUNIT parameters from the DEVICE initialization statement through a hot start with refresh. Furthermore, if the JES3 installation does not use JES3 setup or does not use JES3 setup to manage tape devices, all tape DEVICE initialization statements and associated SETNAME statements can be removed from the initialization stream; this still requires a warm or cold start.

JES3 Limitations

The following functions have limited capability for the 3495 in the JES3 environment:

- Support units
- Device connectivity
- Storage group states
- Scratch integrated cartridge loader requests
- Mixed JES3-managed and non-managed devices
- Library device groups specified in JCL

Support Units

3495 tape drives cannot be used as support units by JES3 dynamic support programs. Therefore, you should not specify DTYPE, JUNIT, and JNAME parameters in your initialization deck.

JES3 does not prevent 3495 drives from being defined as support units. It also does not prevent them from being allocated to a dynamic support program. If this does occur, however, the dynamic support program must be cancelled and restarted with a non-3495 tape drive.

Device Connectivity

All 3490 and 3490E control units within a 3495 Tape Library Dataserver must be connected to every system in a JES3 complex. Otherwise, all jobs using the 3495 must complete converter/interpreter (CI) processing and locate processing on a system that has each of the 3495s attached to it.

The 3490/3490E device configuration within a 3495 is obtained by DFSMS/MVS during IPL or whenever a 3495 drive is varied online. This information is used during CI processing and locate processing to determine eligible JES3 devices.

If CI processing or locate processing are performed on a system that does not have connectivity to all tape drives within a 3495, those drives would be unknown and considered ineligible for JES3 setup. This could result in an allocation failure.

Storage Group States

Storage group states are not supported for the 3495 like they are for DASD:

- Because they are not used to determine the eligible JES3 mains, storage group states do not influence JES3 system selection. Therefore, you cannot use them to steer jobs to eligible systems. 3495 drives must remain offline to ineligible systems to prevent JES3 from scheduling jobs to them.

Note: If the SMS system selection SSI is called for DASD data sets, the tape storage groups states will not influence the setting of the JES3 main mask.

- Storage group states are not used to determine eligible devices for JES3 requests. Although JES3 selects the 3495 device, it must also be eligible for MVS allocation or the job will fail allocation. Because JES3 allocates 3495 drives prior to job processing, and MVS allocation determines eligible devices using the Storage Group state in effect when the job runs, changes between CI processing and job processing could invalidate the device selected by JES3.

To ensure that changes to storage group states do not affect job execution, you should define them as “enabled” to all 3495-eligible systems and not change their state while 3495-eligible jobs are queued to run.

Scratch Integrated Cartridge Loader Requests

Both Cartridge System Tape and Enhanced Capacity Cartridge System Tape can be used with 3490E drives. Enhanced Capacity Cartridge System Tape is automatically loaded into the Integrated Cartridge Loader (ICL), of the 3490E drives, if it is available to the 3495. Otherwise, Cartridge System Tape is used.

Scratch (new data set) requests for Enhanced Capacity Cartridge System Tape are allocated only to the 3490E and will benefit from reduced mount time provided by the ICL. Scratch requests for Cartridge System Tape are allocated to either 3490 or 3490E drives. When a 3490E is selected, the ICL cannot be used if it is loaded with Enhanced Capacity Cartridge System Tape. This results in a longer mount time. This condition exists only when there is mixed media within a library.

The LIBRARY SETCL operator command can be used to change from Enhanced Capacity Cartridge System Tape to Cartridge System Tape.

JES3 will not take the requested media type into account when choosing a device. Therefore, to increase the likelihood of JES3 choosing a drive whose ICL contains the requested media type, you should consider the number of anticipated requests for each media type when determining the media type of the tapes to be pre-loaded.

Mixed JES3-Managed and Non-Managed Devices

3495 tape drives within a complex should all be either JES3-managed or non-JES3-managed. Mixing JES3-managed and non-managed devices could prevent the non-managed devices from being used for new data set allocations and reduces device eligibility for existing data sets. This results in allocation failures or delays in job setup.

Library Device Groups Specified in JCL

LDG names are permitted in JCL but should not be used.

DFSMS/MVS determines an appropriate library device group name for JES3 setup from the data class and storage group assigned to the data set. It replaces the unitname found in the JCL. Because this is done after ACS processing, the JCL-specified unitname is still available to the ACS routines. Although an LDG name specified as a unitname in JCL cannot be used to direct allocations to a specific library or library device type, it can be used for ACS filtering or even to disallow the JCL-specified library device group names.

Mount Failures

Mount failures occur when the system is re-IPLed and tries to mount a tape on a drive other than the drive the tape was last mounted on.

To locate the volumes for which a mount failure occurred:

1. Go to the library manager console on the IBM 3495 Dataserver.
2. Select the database search function.
3. Locate all mounted volumes in the library manager database. The display will show the library manager tape drive identifier for each mounted volume.
4. Return to the MVS operator's console and issue DISPLAY U commands for each mounted tape drive. If the volume is not actually being used, for example, by a job running in a complex that shares the 3495, issue an MVS UNLOAD command for the device.

See *IBM 3495 Tape Library Dataserver Operator's Guide* for more information about the data base search function.

IBM 3590 Installed in IBM Tape Library Dataservers

You install IBM 3590 tape subsystems in IBM 3494 and 3495 Automated Tape Library Dataservers to allow tape operation to be fully automated. For the IBM 3495, four IBM 3590-B1A tape drives and an IBM 3590-A00 controller are supplied within an IBM 3590-A14 frame, and the frame is integrated into the IBM 3495 configuration. IBM 3590 tape subsystems within a IBM 3495 attach to host systems using ESCON channels only. With the IBM 3494, the IBM 3590 tape subsystems are installed in the library and drive frames. IBM 3590 tape subsystems within an IBM 3494 can attach to SCSI channels, but need the IBM 3590-A00 to attach to ESCON channels.

The IBM 3590 High Performance Tape Subsystem drive attaches either directly to a host through one of its SCSI interfaces or to the IBM 3590-A00 controller. The IBM 3590-A00 controller provides for ESCON attachment of the IBM 3590 tape drives. The IBM 3590 tape drives cannot be directly attached to a host through a SCSI interface and also attached to the IBM 3590-A00 controller.

When an IBM 3590 tape subsystem is directly attached to a host through its SCSI interface, an RS-232 or LAN interface is used to provide communication between the host and the Library Manager. When an IBM 3590 tape subsystem is attached through the model A00 controller, the communication path for the Library Manager is through the same ESCON channels as for the drive.

When an IBM 3590 tape subsystem is directly attached to a host through its SCSI interface, an RS-422 interface is used between the Library Manager and the drive to communicate drive status and control. When an IBM 3590 High Performance Tape Subsystem is attached through the model A00 controller, the drive status and control path is through the RS-422 interface between the model A00 controller and the Library Manager. The model A00 controller acts as the path between the Library Manager and each drive.

Integration into Existing Tape Library

To integrate the IBM 3590 into the IBM Automated Tape Library Dataserver, follow these steps:

- For an existing library, plan to make the library available to the IBM Customer Service Representative to allow the IBM 3590 frames and drives to be installed. Switch the IBM 3494 and 3495 Automated Tape Library Dataservers offline from all attached systems, one at a time, before the IBM Customer Service Representative starts any installation activity.
Ensure that the implementation causes a minimum of disruption, to automated tape services by warning all users well ahead of time.
- Shutdown the Library Manager and power off the IBM 3494 and IBM 3495.
- For an existing Library, the IBM Customer Service Representative
 - Installs a new level of Licensed Internal Code, if needed
 - Installs the new frames and new drives. The installation of each frame, including drives, will take approximately 5 hours.
 - Remove any old drives that are to be replaced. The IBM Customer Service Representative will perform a partial teach for the new frames. In an existing IBM 3494, the partial teaching process takes approximately 5 minutes to execute.
 - Insert a new IBM 3590 CE cartridge.
 - Perform a full inventory.
 - Hand back the IBM 3494 and IBM 3495.
- Insert the new IBM 3590 cleaner cartridges and the new IBM 3590 High Performance Tape Cartridges. Ensure that all IBM 3590 cartridges have a proper barcode, with a J media-type label.
- Set up the cleaning schedule to use the cleaner cartridges after every 100 tape uses.
- Vary the library and drives online, one at a time, and check that all paths and all drives come online with all attached hosts.
- Check that the information about the new cartridges has been uploaded to the hosts automatically; in some cases this requires operator action.

- If the host software environment has been set up properly, the new drives and cartridges are ready for testing.
- If the new drives are to replace existing equipment, you must manage the discontinuance of the old drives.

Note: If you are installing IBM 3590 tape subsystems as part of a new IBM Automated Tape Library Dataserver configuration, please refer to IBM 3494 Introduction and Planning Guide and IBM 3495 Introduction and Planning Guide.

If you are attaching to existing IBM Automated Tape Library Dataserver, the new frames need to be integrated into the library and attached to the Library Manager. The new drives and cartridges are ready for use.

JES3

For the most part, the use of the IBM 3590 tape subsystem as an execution device in a JES3 complex is transparent to JES3. You need to define the devices to JES3 in the Initialization stream, if JES3 is to manage the devices. Once you have done this, JES3 manages the IBM 3590 drives the same way other tape devices are managed.

To update the JES3 Initialization stream follow the steps described below:

1. Specify the generic unit name and any esoteric unit names by which the IBM 3590 drives may be requested; use one or more SETNAME statements.
2. Specify these unit names on HWSNAME statements, if you are using High Watermark Setup, or catalog overrides--for example, if a unit name specified on JCL is more restrictive than the IBM 3590 generic unit name of 3590-1. In these cases, the JCL unit name will override the unit name in the catalog.

The rules for specifying the IBM 3590 unit names on these statements are the same as for other device types. For example, each unit name must appear as the primary unit name on one HWSNAME statement.

Note: Since IBM 3590 is not compatible with other tape device types, the alternate unit names that follow the primary name must not contain any other device type.

3. Define the IBM 3590 drives through DEVICE statements. If the JUNIT parameter is specified, the DTYPE should be specified as TA435901.
To request an IBM 3590 drive for the DUMP Job DSP, you may specify TA4 or TA435901 or the device address on the IN= or OUT= parameter. You can request compaction using the existing MOD= key word. Do not use the DEN= key word: it is ignored except for a warning message because the IBM 3590 writes a single density tape.

Other JES3 Tape Dynamic Support Programs (DSPs) such as Tape Label or Tape-to-tape do not support the IBM 3590 devices.

4. If the IBM 3590 tape subsystems are installed in an IBM 3494 or 3495 Automated Tape Library Dataserver, you need to define the following esoteric names:
 - LDBsssss which includes all IBM 3590 devices in the library with library ID sssss.
 - LDG3591 which includes all IBM 3590 devices in any library.

Note: The name 3590-1 and any nonlibrary esoteric unit name must not include any library devices.

Figure 38 shows the sample JES3 definitions for an IBM 3494 or 3495 Automated Tape Library Dataserver with one string of four Magstar tape drives. The library name in this example is F4006.

```

SETNAME,XTYPE=LB135901,NAMES=(LDGW3495,LDGF4006,LDG3591,LDBF4006)
      |           |           |           |
      V           V           V           V
      Complex   Library   Complex   Library
      Wide      Specific  Wide      Specific
      Library   Library   Device   Device
      Name      Name      Name      Name

DEVICE,XTYPE=(LB135901,TA),XUNIT=(500,SY1,TAP,OFF)
DEVICE,XTYPE=(LB135901,TA),XUNIT=(501,SY1,TAP,OFF)
DEVICE,XTYPE=(LB135901,TA),XUNIT=(502,SY1,TAP,OFF)
DEVICE,XTYPE=(LB135901,TA),XUNIT=(503,SY1,TAP,OFF)
HWSNAME,TYPE=(LDGW3495,LDGF4006,LDG3591,LDBF4006)
HWSNAME,TYPE=(LDGF4006,LDG3591,LDBF4006,LDGW3495)
HWSNAME,TYPE=(LDG3591,LDBF4006,LDGW3495,LDGF4006)
HWSNAME,TYPE=(LDBF4006,LDGW3495,LDGF4006,LDG3591)

```

Figure 38. Sample JES3 Definitions for IBM 3590 with an IBM 3494 or 3495 Automated Tape Library Dataserver

You could also define the entire library with one DEVICE statement by defining it as follows: `DEVICE,XTYPE=(LB135901,TA),XUNIT=(500,*ALL,TAP,OFF),NUMDEV=4`

The complex-wide library name is always LDGW3495, no matter whether you have installed only an IBM 3494, only an IBM 3495, or both.

Note: IBM 3590 drives installed in IBM 3494 or 3495 Automated Tape Library Dataserver cannot be used to as support units for JES3 DSPs. Therefore, do not specify DTYPE, JUNIT, and JNAME parameters on the DEVICE statement. No check is made during initialization to prevent library drives from being defined as support units, and no check is made to prevent the drives from being allocated to a DSP if they are defined. Any attempt to call a tape DSP by requesting an IBM 3494 or 3495 fails because the DSP is unable to allocate an IBM 3494 or 3495 drive.

Job Entry Subsystem 3 (JES3), an OS/390 MVS subsystem, manages resources and workflow before and after jobs are processed. As documented in this chapter, JES3 is also modified to support Model E1x tape drives in an SMS-managed 3494 automated tape library dataserver.

ATLDS JES3

You can now enable JES3 to allocate Model E1x tape drives and tape volumes in an SMS-managed 3494 ATLDS. To do so, you need to specify one or both of the following new library device group (LDG) names in your JES3plex. An LDG is a predefined set of tape subsystems within a JES3plex. LDCsssss. The serial number ssss indicates the library within which a Model E1x tape drive emulates a 3590-1.

LDGdddd. dddd=359E indicates that a Model E1x tape drive emulates a 3590-1 in any library within the JES3plex.

Tape Verification Processing (IATLVVR Module)

The JES3 tape verification processing, namely, the IATLVVR module, is updated to extract VOLSER information from the volume's sense data if a 256TRACK scratch tape is mounted on a 128TRACK drive.

General Considerations for Migration

Add new esoteric names to JES3 initialization decks. For early JES3 releases, you need to check for all new esoteric names in order to suppress such operator messages as IAT5110 (fetch) and IAT5410 (keep/retain).

From Magstar and IBM 3590 High Performance Tape Subsystem: Multiplatform Implementation February 1997 Document Number SG24-2594-02.

From Magstar and IBM 3590 High Performance Tape Subsystem Technical Guide November 1996 Document Number SG24-2506-01.

IBM 3590 Support for JES3

In a z/OS JES3 environment, the IBM 3590 tape device is treated as one of the tape devices supported by JES3.

Library Device Group Name: To use the IBM 3590 devices in a JES3-managed IBM Automated Tape Library Dataserver, the following esoteric unit names must be defined as library device group (LDG) names in the HCD and SETNAME of the JES3 initialization parameter:

- LDBsssss - includes any IBM 3590 within the library indicated by library serial number, sssss.
- LDG3591 - includes any IBM 3590 in any library in the JES3plex.

DTYPE for JUNIT: When IBM 3590 tape devices are used as JES3-managed devices, they must be defined on DEVICE statements in the JES3 initialization parameter. If the JUNIT parameter is specified in the DEVICE statement, the DTYPE must be specified as TA435901.

JES3 Dump Job Dynamic Support Program: The JES3 Dump Job (DJ) Dynamic Support Program (DSP) supports the IBM 3590 tape device for dumping and restoring the JES3 spool. When the operator calls the Dump Job DSP, TA435901, TA4, or the device number can be specified on the IN= or OUT= parameter in the JES3 CALL,DJ command. The operator can also specify compaction.

Part 13. Appendixes

Appendix A. The External Writer

This appendix documents Intended Programming Interface and Associated Guidance Information.

An external writer allows an installation to perform output processing for data sets not eligible for processing by the primary job entry subsystem (JES2 or JES3). For example, an external writer processes data sets going to printers, plotters, diskettes or data sets that are to be stored on a device not supported by JES.

There are two major parts of the IBM-supplied external writer that can be modified or replaced by an installation: the output writing routine and the output separator routine.

This chapter describes:

- Overview of the IBM-supplied external writer
- How to set up and start the IBM-supplied external writer
- How the IBM-supplied output writing routine works
- How to add your own output writing routine
- How the IBM-supplied output separator routine works
- How to add your own output separator routine

See *z/OS MVS Using the Functional Subsystem Interface* for information in subsystem interface (SSI) function code 1 on how to write your own external writer.

Overview of the IBM-Supplied External Writer

IBM supplies an external writer, which contains two routines that can be modified or replaced by the installation:

- The output writing routine. This routine processes data sets.
- The output separator routine. This routine writes separator records within a continuous listing or deck to help the operator separate one job's output from another's.

Characteristics of the IBM-Supplied External Writer

The external writer has the following features:

- It runs as a started task, in its own address space, in 24-bit addressing mode
- It processes only those data sets that meet its selection criteria. You set some of these in a cataloged procedure (see "The External Writer Cataloged Procedure" on page 332) but you can override them with the START and/or the MODIFY operator commands (see *z/OS MVS System Commands* for the general description of these commands).

If you do not specify data set selection criteria, the external writer will be used to process **all** the output in the system.

- It removes data sets from the JES spool. That is, it dynamically allocates the data sets, reads them, writes them to output devices, and then dynamically deallocates them.
- It accesses data sets according to one or more of the following criteria:
 - Output class
 - Job ID

- Forms specification
- Destination (LOCAL, or remote workstation name)
- The name of your output writing routine

The usual technique however, for setting data set selection criteria, is to build a list of eligible SYSOUT classes for the devices that will use the external writer.

If no class is specified, then we will not start selecting any output and will wait for a MODIFY command to be issued. For example, if you have the following in process:

```
//IEFPROC EXEC PGM=IASXWR00,PARM='P',REGION=20K
```

No class is specified. An S XWTR will cause the external writer to be started, but idle. Then, a F XWTR,F=ABC may be specified to have the external writer select by form type. If another F XWTR,C= is issued, then everything on the output queue will be selected because a null class is specified.

How to Set Up and Start the External Writer

For an external writer to work in a JES environment, it must be defined to the system in a cataloged procedure residing in SYS1.PROCLIB; and it must be started by a START command, either from the system console or from within a program.

There are two ways to set up the criteria by which the external writer selects output. The PARM= field (described below) allows the specification of SYSOUT classes to select. Other criteria (such as forms, destination, writer name, jobid and also SYSOUT class) must be specified using the MVS MODIFY command. See *z/OS MVS System Commands* for a complete description of the modify command as it applies to the external writer.

The External Writer Cataloged Procedure

The IBM-supplied external writer is defined and invoked by the cataloged procedure named XWTR, which can serve as the base or a model for a procedure you would write for your own output writer.

XWTR contains one step and consists of two JCL statements:

- The EXEC statement specifies the name of the external writer program to be executed.
- The DD statement, which **must** be named IEFRDER as shown below, defines the output data set.

Following is the actual XWTR procedure:

```
//IEFPROC EXEC PGM=IASXWR00,REGION=20K, X
// PARM='PA,, ' X
//IEFRDER DD UNIT=TAPE,VOLUME=(,,35), X
// DSNAME=SYSOUT,DISP=(NEW,KEEP), X
// DCB=(BLKSIZE=133,LRECL=133,BUFL=133, X
// BUFNO=2,RECFM=FM)
```

The EXEC Statement

The generalized format for the EXEC statement is:

```
//IEFPROC EXEC PGM=IASXWR00[,REGION=nnnnnK,]
// [PARM='cxxxxxxxx[,seprname]] '
// [PARM='cxxxxxxxx[,seprname][,CERTIFY=Y|N]] '
```

The stepname must be IEFPROC, as shown. The parameter requirements are as follows:

PGM=IASXWR00

The name of the external writer load module. It must be IASXWR00, as shown.

REGION=nnnnnK

This parameter specifies the region size for the external writer program. The value nnnnn is a 1- to 5-digit number that is multiplied by 1K (1024 bytes) to designate the region size. The region size can vary according to the size of buffers and the size of your output writing routine. Insufficient region size will cause the external writer to abend.

[PARM='cxxxxxxx[,seprname][,CERTIFY=Y|N]]'

[PARM='cxxxxxxx[,seprname]]'

Specifies output writer routine characteristics.

- c An alphabetic character, either **P** (for printer) or **C** (for card punch), that specifies the control characters for the class of output the output writing routine will process.

xxxxxxx

From one to eight (no padding required) single-character SYSOUT class names. These characters specify the classes the output writing routine will process and establish the priority for those classes, with the highest priority at the high-order (leftmost) end of the character string.

Note: If the START command includes class name parameters, they override all of the class names coded here. If you do not code class names on the procedure EXEC statement or the START command, then the external writer will wait for a MODIFY command from the operator before processing any output.

seprname

This is the name of the output separator routine to run with the output writing routine. If you omit this subparameter, no output separator pages are produced.

IEFSD094 is the IBM-supplied output separator routine. If you write your own output separator routine, and name it here, put it in SYS1.LINKLIB (or a library concatenated to LINKLIB through a LNKSTxx member of parmlib).

Note: Do not use *CERTIFY* as the name of your output separator routine. For your separator routine to be invoked, you must either code its name on this subparameter or name it in the parameters on the START command.

CERTIFY=Y|N

Indicates whether (Y) or not (N) you want to ensure that all data is safely written to the output device. CERTIFY=Y indicates that data will **not** be lost even if the external writer abnormally terminates, such as would occur if the output data set becomes full.

Notes:

1. The use of the certify function increases processing overhead and decreases efficient space utilization by the output data set.
2. If you write your own output writing routine and require the CERTIFY function, you need to modify your code to be similar to that supplied

by IBM. Refer to “Using the CERTIFY Function” within section “Programming Considerations for the Output Writing Routine” on page 339 for further details.

3. IBM cannot guarantee data set integrity when writing multiple data sets to a PDS/PDSE member even if you specify CERTIFY=Y. Refer to “Functions of the Output Writer Routine” on page 336 for further details.

The DD Statement

In this statement, you must define the output data set that the external writer will use. The generalized format for the DD statement is:

```
//IEFRDER DD UNIT=device,LABEL=(,type), X
//          VOLUME=(,,volcount),DSNAME=aname, X
//          DISP=(NEW,KEEP),DCB=(list of attributes), X
//          UCS=(code[,FOLD][,VERIFY]), X
//          FCB=(image-id[,ALIGN][,VERIFY])
```

The ddname must be IEFRDER, as shown. The parameter requirements are as follows:

UNIT=device

This specifies the printer, tape, card punch, or DASD device on which the output data set is to be written.

LABEL=type

This describes a data set label, if one is needed (for tape data sets only). If this parameter is omitted, a standard tape label is used.

VOLUME=(,,volcount)

Needed for tape data sets only, this parameter limits the number of tape volumes that this external writer can use during its entire operation.

DSNAME=aname

This specifies a name for the output data set, so later steps in the procedure can refer to it. The data set name is required for the disposition of KEEP.

DISP=(NEW,KEEP)

The disposition of KEEP prevents deletion of the data set (tape and DASD only) at the end of the job step.

DCB=(list of attributes)

The DCB parameter specifies the characteristics of the output data set and the buffers. The BLKSIZE and LRECL subparameters are always required. The BUFL value, if you do not code it, is calculated from the BLKSIZE value. Other subparameter fields may be coded as needed; if they are not, the defaults are the QSAM default attributes. These are:

BUFNO—

Three buffers for the 2540 punch; two buffers for all other devices.

RECFM—

U-format, with no control characters.

TRTCH—

Odd parity, no data conversion, and no translation.

DEN—

Lowest density.

OPTCD—

Printer data checks are suppressed, and “select translate table”

characters are printed as data. The IBM external writer does not support OPTCD=J, a printer dependent specification.

UCS=(code[,FOLD][,VERIFY])

This specifies the code for a universal character set (UCS) image to be loaded into the UCS buffer.

FOLD causes bits 0 and 1 to be ignored when comparing characters between the UCS and print line buffers, thereby allowing lowercase alphabetic characters to be printed (in uppercase) by an uppercase print chain or train.

VERIFY causes the specified UCS image to be printed for verification by the operator.

The UCS parameter is optional, and is valid only when the output device is a 1403, a 3211, or a 3203-5 printer.

FCB=(image-id[,ALIGN][,VERIFY])

This causes the specified forms control buffer (FCB) image to be loaded into the FCB. ALIGN and VERIFY are optional subparameters that allow the operator to align forms. In addition, VERIFY causes the specified FCB image to be printed for visual verification. The FCB parameter is valid only for a 3203-5, 3211, or 3800 printer; otherwise, it is ignored.

See *z/OS MVS JCL Reference* for more information on the parameters mentioned here.

Special Printer Output Considerations: To process output jobs that require special chains for printing, you should have specific classes for each different print chain. You can specify the desired chain in your output writer procedure, and when that output writer is started, the chain will be loaded automatically. (Printers used with special chains should be named with esoteric group names as defined at system installation.) See *z/OS HCD Planning* for information on the eligible device table.

Following is an example of the JCL needed to define a special print chain in a cataloged procedure for an external writer.

```
//IEFPROC EXEC PGM=IASXWR00,REGION=20K,PARM='PDEG,IEFSD094'  
//IEFRDER DD UNIT=SYSPR,DSNAME=SYSOUT,FCB=(STD2,ALIGN), X  
// UCS=P11,DISP=(,KEEP), X  
// DCB=(BLKSIZE=133,LRECL=133,BUFL=133, X  
// BUFNO=2,RECFM=FM)
```

In this example, the UCS DD parameter requests the print chain alias for data sets in the SYSOUT classes D, E, and G.

If the output device is a 3211 or a 3203-5, a UCS or FCB image can be loaded dynamically between the printing of data sets. Therefore, you can specify a mixture of data sets using different images in a single output class for this device. This will probably require mounting trains and changing forms, however, so it might not be desirable.

When the output device is a 1403 or 3800, the UCS image or 3800 attributes are specified at START XWTR time; they cannot be changed until the writer is stopped. Therefore, all data sets within an output class must be printed using the same train.

The FCB image is ignored when the 1403 printer is the output device.

External writer output to an IBM 3800 Printing Subsystem can also make use of the CHARS, COPIES, FLASH, and MODIFY JCL parameters on the DD statement. For information about using these parameters, see *3800 Printing Subsystem Programmer's Guide*. The coding rules and defaults are documented in *z/OS MVS JCL Reference*.

How the IBM-Supplied Output Writer Routine Works

As stated previously, there are two main pieces to the external writer that can be modified or replaced by the installation: The output writer routine and the output separator routine. When the external writer is called, it gets control in module IASXWR00. The main function of IASXWR00 is to initialize a 7-word parameter list for the use of the other external writer routines.

The parameter list contains information about the output device and DCB addresses for each data set. The input data set to the external writer was a job's output data set which met the criteria for processing by the external writer. This input data set is not yet open.

The format of the external writer parameter list is as follows:

Table 52. External Writer Parameter List

Byte	Meaning										
0	Used to describe the type of output device										
	<table border="1"> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>011.</td> <td>2540 punch unit</td> </tr> <tr> <td>001.</td> <td>1403, 3203-5, 3211, or 3800 printer device</td> </tr> <tr> <td>010.</td> <td>tape device with punch-destined output</td> </tr> <tr> <td>000.</td> <td>tape device with printer-destined output</td> </tr> </tbody> </table>	Bit	Meaning	011.	2540 punch unit	001.	1403, 3203-5, 3211, or 3800 printer device	010.	tape device with punch-destined output	000.	tape device with printer-destined output
Bit	Meaning										
011.	2540 punch unit										
001.	1403, 3203-5, 3211, or 3800 printer device										
010.	tape device with punch-destined output										
000.	tape device with printer-destined output										
1-3	Reserved for IBM use										
4-7	Address of the DCB for the opened output data set where the external writer will put the input records										
8-11	Address of the DCB for the input data set, from which the external writer will obtain logical records.										
12-15	Address of the cancel ECB for writer routine subtask (IEFSD087 or user-written writer routine).										

The switches indicated by the three high-order bit settings in byte 0 can be used in translating control character information from the input records to the form required by the output device.

Functions of the Output Writer Routine

When you specify IEFSD087 as the output writer on the SYSOUT= parameter of the DD statement, or when you leave this parameter blank, control will pass to the IBM-supplied output writing routine in module IEFSD087, and it:

- Issues an OPEN (J type) for the input data set previously taken from the JES spool.
- Provides its own SYNAD error-handling routine on behalf of both the input and output data sets. See *z/OS DFSMSdfp Advanced Services* for more information about OPEN-J and SYNAD.
- Reads the input data set, using the locate mode of the GET macro.

- Calls a subroutine to handle ANSI and machine control character differences and to handle conversions between the input records and the output data set
- Calls a routine to write records to the output data set, using the locate mode of the PUT macro
- Closes the input data set after it has been read
- Provides accounting support by updating fields in the SMF type 6 record for the input data set
- Frees the buffer associated with the input data set (by issuing the FREEPOOL macro)
- Returns control to the main logic control module of the external writer, IASXWR00, using the RETURN macro and setting a return code.

Note: When writing multiple data sets to a PDS/PDSE member, if the 'end-of-output-data-set' condition arises, the output external writer cannot guarantee (even if you set CERTIFY=Y as a startup parameter) that all records will be saved. The current data set is saved, but all records written up to that point cannot be recovered in the case of writing to a PDS/PDSE member.

The following diagram shows the general flow of the IBM-supplied output writing routine:

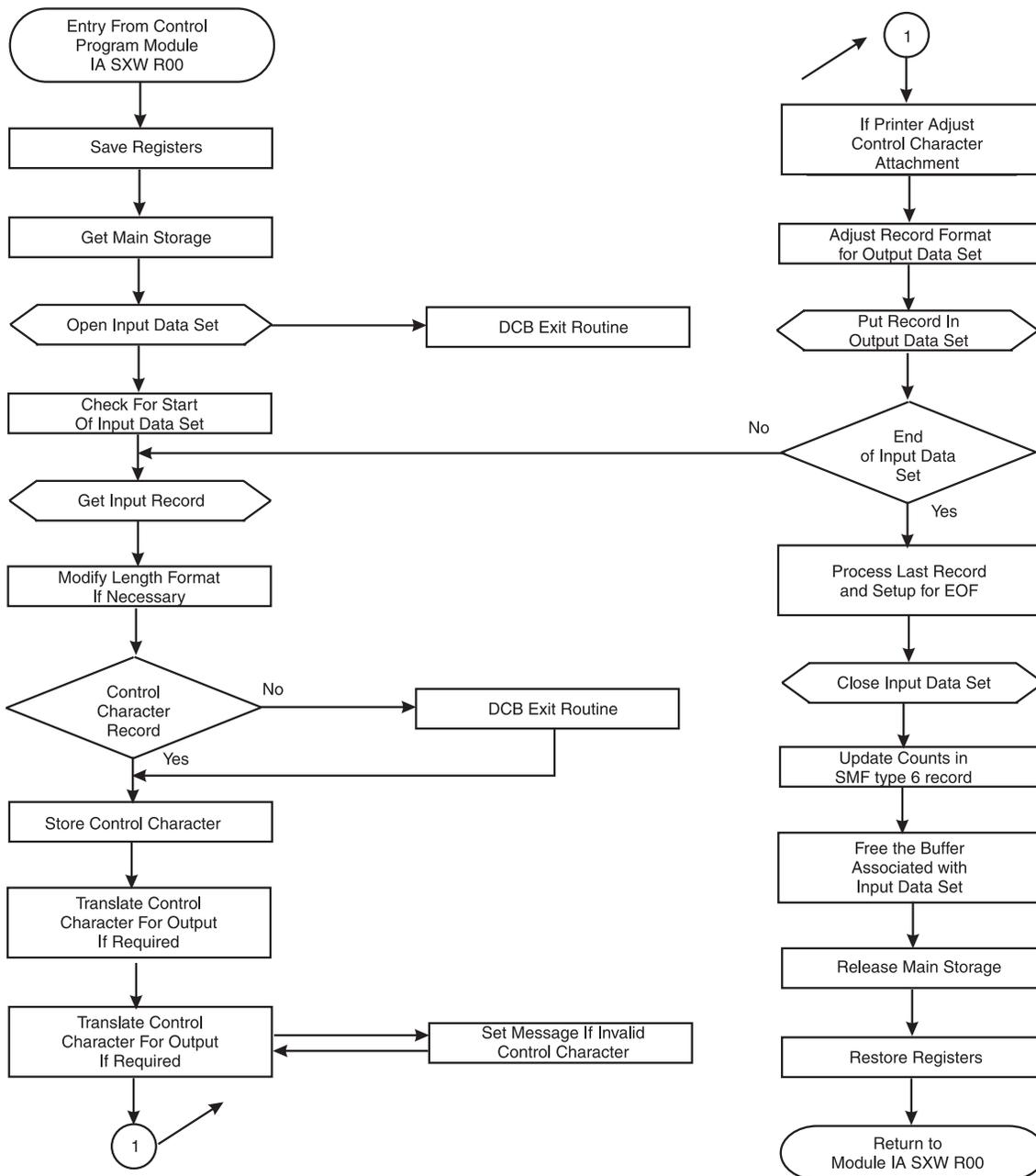


Figure 39. General Flow of IBM's Output Writing Routine

How to Add Your Own Output Writing Routine

If you want to add your own output writing routine to the external writer, consider whether your routine will be needed more often than the IBM-supplied routine. If your routine will be invoked to write **most** of the external writer's output, you might want to replace the IBM-supplied with your own routine, so that your routine will be called by default. You can retain the IBM-supplied routine by renaming it to an alias.

Replacing the IBM-Supplied Routine

You can replace the IBM-supplied routine with your routine by:

- Renaming the IBM-supplied routine (IEFSD087) to an alias.
- Naming your routine IEFSD087
- Installing your routine in SYS1.LINKLIB

The external writer will call your routine by default. You can request the IBM-supplied routine by coding its alias on the SYSOUT= parameter. For example:

```
//MYDATA DD SYSOUT=(H,IBMWRITR)
```

where IBMWRITR is the alias you've given to the IBM-supplied routine.

Do not code STDWTR as a writer name. STDWTR and INTRDR are reserved for JES used as a parameters in the MVS operator's MODIFY command.

In a JES3 system, do not code NJERDR as a writer name. NJERDR is reserved for JES3.

Using Your Routine for Only Certain Jobs

If you plan to have your routine invoked for only certain jobs, give it a unique name, and install it in SYS1.LINKLIB or an authorized library concatenated to LINKLIB through a LNKSTxx member of SYS1.PARMLIB. To have your routine invoked, the end-user must specify its name on the SYSOUT= parameter of the job's DD statement. When your routine is not specified, the system calls the IBM-supplied routine by default.

Coding Conventions for the Output Writing Routine

In order for your output writing routine to work in the external writer, it must observe the following conventions:

- If you put your routine in LPA so that you only use one copy for all external writers, then it must be reentrant. If you use a STEPLIB and each writer can have its own copy, then the code need not be reentrant.
- The routine must use standard entry and exit linkage, saving and restoring its caller's registers.

At entry, register 1 points to the parameter list and register 13 points to an 18-word save area.

- The routine must issue an OPEN-J macro to open the desired input data set. The data set to which the output writing routine will write records is opened before the routine is loaded.
- The routine must use the GETMAIN and FREEMAIN macros, or the STORAGE macro, to acquire and release any necessary storage.
- The routine must return control to its caller through the RETURN macro, in the same addressing mode it was called in. It must put a return code in register 15:
 - A return code of 0 indicates that the data set was processed successfully to the output device.
 - A return code of 8 indicates that the routine was unable to process the data set because of an output error.

Programming Considerations for the Output Writing Routine

In addition to the coding conventions, consider the following when writing your own output writing routine:

- **Obtaining Storage for Work Areas:** Using the GETMAIN or STORAGE macro, the output writing routine should obtain storage in which to set up switches and save record lengths and control characters.

- **Processing an Input Data Set:** To process a data set, the writing routine must get each record individually from the input data set, transform (if necessary) the record format and the control characters to conform to the output data set's requirements, and put the record in the output data set. Consider each of these tasks individually:

1. Before the routine actually obtains a record from an input data set, the routine must provide a way to handle the two forms of record control character that are allowed in an output data set, if the output device is a printer.

Most printers are designed so that, if the output data set records contain machine control characters, a record (line) is printed before the effect of its control character is considered. However, if ANSI control characters are used in the output records, the control character's effect is considered before the printer prints the line. Thus, if the input data sets do not all have the same type of control characters, you will need to avoid overprinting the last line of one data set with the first line of the next.

When the input records have machine control characters and the output records are to have ANSI control characters, the standard (IBM-supplied) output writing routine produces a control character that indicates one line should be skipped before printing the first line of output data.

When the input records have ANSI control characters and the output records are to have machine control characters, the standard writing routine prints a line of blanks before printing the first actual output data set record.

Following this line of blanks, the printer generates a one-line space before printing the first record.

Depending upon the characteristics of the printers in your installation, you will probably want your output writer to perform some kind of "printer initialization" like that outlined here.

2. After the output writing routine has properly opened the input data set and has completed any necessary printer initialization, it must obtain records from the input data set.

The standard output writing routine uses the locate mode of the GET macro. If you use this macro, you will need to check the MACRF field of the input data set's DCB to see if GET in locate mode is allowed. If not, you can override the MACRF parameter on the GET macro itself. See *z/OS DFSMSdfp Advanced Services* for information on coding and using all the QSAM macros.

In a JES2 system, the padding is done automatically.

In a JES3 system, if blank truncation is in effect (TRUNC=YES specified on the BUFFER initialization statement or the SYSOUT initialization statement), then fixed format records will not be padded with blanks. You need to do the padding manually.

3. Having obtained a record from the input data set, the output writing routine must now make sure that its format and control character are compatible with the requirements of the output data set.

Because the output data set is already opened when the output writing routine is entered, your routine will have to adhere to the established conventions.

The standard output writing routine uses the PUT macro in the locate mode to write records to the output data set. For fixed-length output, it obtains the record length for the output data set from the DCBLRECL field of the DCB.

If an input record is longer than the length specified for the output records, the standard output writing routine truncates the input record from the right.

If an input record is shorter than the length specified for the output records, the standard output writer left-justifies the input record and pads the field with blanks.

When the output record length is variable and the input record length is fixed, the standard output writer adds control character information (if necessary) and variable record control information to the output record. Control character information is one byte, and record control information is four bytes long. Both additions are at the high-order end of the record.

If the output record is not at least 18 bytes long, the standard output writer pads it on the right with blanks. The standard output writer also adjusts the length of the output record to match the length of the output buffer.

4. When the output writing routine has successfully adjusted the input and output records, it can read the input data set until end of data. At that point, you need to consider another aspect of input data set processing: what is going to happen to the last input record?

The standard output writer handles output to either card punch or printer, as required; your routine could also send output to an intermediate tape or DASD device. Depending upon the kind of device, the last few records obtained from the input data set will receive different treatment.

It might happen that all the records from a given data set are not available on the output device until the output of records from the next data set is started, or until the output data set is closed. However, if you specify CERTIFY=Y as a start-up parameter, the records can be made available on the output device after all the input records have been written by the output writing routine. When the output data set is closed, the standard output writer automatically puts out the last record of its last input data set.

For Punch Output: When a card punch is the output device, the last three output cards could still be in the machine when the input data set is closed.

To put out these three records with the rest of the data set, and with no breaks, the standard output writer provides for three blank records following the actual data set records.

For Printer Output: When a printer is the output device, the last record of the input data set is not normally put in the output data set at the time the input data set is closed.

To force out this last record, the standard output writer generates a blank record to follow the last record of the actual data set.

- **Using the CERTIFY Function:** The CERTIFY function can be specified as a parameter when starting the external writer. It guarantees that data will not be lost when an abend condition occurs. If the CERTIFY function is not enabled, it is possible that records written to the QSAM buffers, but not yet written to the output device, can be purged before they are recorded during an abend situation. This is possible, for example, when the output data set is not large enough to contain the data, resulting in an 'end-of-volume' (x37) abend.

The CERTIFY function flushes the buffers each time an input data set is processed, and makes that data available on the output device. However, while guaranteeing data integrity, this function might also increase processing overhead and decrease efficient output data set space utilization. Also, note that even with CERTIFY=Y, data integrity might be compromised in the case of writing multiple JES data sets to a partitioned dataset (PDS) or PDSE in an abend situation.

If you need to use the CERTIFY function, you need to modify your output writing routine to be similar to that supplied by IBM. Refer to module IEFSD087 for an example of how to enable the CERTIFY function. Basically, you must do the following:

1. In the external writer main task, force the number of buffers for the output data set to 1. That is, set DCBBUFNO=1.
 2. In the output writing routine after all records are PUT to the data set, perform a second PUT to write a blank line (a 'transition record') after the data to separate the data sets.
 3. In the output writing routine, commit the data to the output device.
- **Closing Input Data Set(s):** After the standard output writer finishes putting out the records of an input data set, it closes the data set before returning control to the calling module. All input data sets must be closed.

After closing the input data set, it is recommended that your output writer free the buffer pool associated with the input data set (by issuing the FREEPOOL macro).

- **Releasing Main Storage:** The output writing routine should release the storage it acquired, using the FREEMAIN or STORAGE macro, before returning to its caller.
- **Handling Errors:** The routine must put a return code into register 15 before returning to its caller using the RETURN macro.

The standard output writer sets a return code of 8 if it terminates because of an unrecoverable error on the output data set; otherwise, the return code is 0. The output writing routine must handle input errors itself.

How the IBM-Supplied Output Separator Routine Works

The second major part of the IBM-supplied external writer routine is the output separator routine. Any output processing to a punch or printer must include a means of separating one job from another within the continuous deck or listing. When you specify IEFSD094 on the PARM= parameter on the EXEC card of the external writer, the IBM-supplied output separator routine writes separation records to the output data set prior to the writing of each job's output.

You can modify this separator routine to suit your installation's needs, or you can create your own routine. IBM's version does the following:

- **For Punch-Destined Output:** The separator routine provides three specially-punched cards (deposited in stacker 1) prior to the punch card output of each job. Each of these cards is punched in the following format:
 - Columns 1 to 35**
blanks
 - Columns 36 to 43**
job name
 - Columns 44 to 45**
blanks
 - Column 46**
output classname
 - Columns 47 to 80**
blanks
- **For Printer-Destined Output:** The IBM-supplied separator routine provides three specially-printed pages prior to printing the output of each job. Each of these separator pages is printed in the following format:
 - Beginning at the channel 1 location (normally near the top of the page), the job name is printed in block characters over 12 consecutive lines. The first

block character of the 8-character job name begins in column 11. Each block character is separated by 2 blank columns.

- The next two lines are blank.
- The output classname is printed in block characters covering the next 12 lines. This is a 1-character name, and the block character begins in column 35.
- The remaining lines, to the bottom of the page, are blank.
- In addition to the block characters, a full line of asterisks (*) is printed twice (that is, overprinted) across the folds of the paper. These lines are printed on the fold *preceding* each of the three separator pages, and on the fold *following* the third page. This is to make it easy for the operator to separate the job output in a stack of printed pages.

To control the location of the lines of asterisks on the page, the IBM-supplied separator routine requires that a channel 9 punch be included (along with the channel 1 punch) on the carriage control tape or in the forms control buffer (FCB). The channel 9 punch should correspond to the bottom of the page. The printer registration should be offset to print the line of asterisks on the fold of the page.

The IBM-supplied separator routine makes no provision for the 3800 printing subsystem; if you use it on a 3800, the FCB must locate a channel 9 punch at least one-half inch from the paper perforation.

Separator Routine Parameter List: IBM's external writer provides its separator routine with a 7-word parameter list of necessary information. When the separator routine receives control, register 1 contains the address of the parameter list, which contains the following:

Table 53. Separator Routine Parameter List

Byte	Meaning										
0	Used to describe the type of output device <table border="0"> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>011.</td> <td>2540 punch unit</td> </tr> <tr> <td>001.</td> <td>1403, 3203-5, 3211, or 3800 printer device</td> </tr> <tr> <td>010.</td> <td>tape device with punch-destined output</td> </tr> <tr> <td>000.</td> <td>tape device with printer-destined output</td> </tr> </tbody> </table>	Bit	Meaning	011.	2540 punch unit	001.	1403, 3203-5, 3211, or 3800 printer device	010.	tape device with punch-destined output	000.	tape device with printer-destined output
Bit	Meaning										
011.	2540 punch unit										
001.	1403, 3203-5, 3211, or 3800 printer device										
010.	tape device with punch-destined output										
000.	tape device with printer-destined output										
1-3	Not used, but must be present										
4-7	Address of the DCB for the opened output data set where the external writer will put the input records										
8-11	Address of an 8-character field containing the job name										
12-15	Address of a 1-character field containing the output class name										
16-19	Address of the data set PROC name in the SSOB extension.										
20-23	Address of the data set STEP name in the SSOB extension.										
24-27	Address of the data set DD name in the SSOB extension.										

The parameter list points to a DCB; this DCB is established for the QSAM output data set, which is already open when the separator program receives control.

The address of the job name and the address of the output classname are provided in the parameter list so they can be used in the separation records the separator routine writes.

Output from the Separator Routine: A separator routine can write any kind of separation identification. IBM supplies a routine that constructs block characters. (See “Using the Block Character Routine” on page 345.) The separator routine can punch as many separator cards, or print as many separator pages, as necessary.

The output from the separator program must conform to the attributes of the output data set. To find out what these attributes are, examine the open output DCB pointed to by the parameter list. The attributes are:

- Record format (fixed, variable, or undefined length)
- Record length
- Type of carriage control characters (machine, ANSI, or none)

For printer-destined output, a separator routine can begin its separator records on the same page as the previous job output, or skip to any subsequent page. However, the separator routine should skip at least one line before writing any records because in some cases the printer is still positioned on the line last printed.

After completing the output of the separation records, the separator routine should write sufficient blank records to force out the last separation record. This also allows the error exit routine to obtain control if an uncorrectable output error occurs while writing the last record. One blank record, for printer-destined output, and three blank records, for punch-destined output, are sufficient to force out the last record.

How to Add Your Own Output Separator Routine: If you write your own separator routine, it must conform to the following requirements:

- The routine must have a unique name and this name must be specified in the PARM= parameter on the EXEC card for the external writer. If you do not specify the name of an output separator routine, no separator records are written.

Note: Do not use *CERTIFY* as the name of your output separator routine. Refer to “The EXEC Statement” on page 332 for details on specifying an output separator routine on the PARM= parameter.

- If you want to replace the standard IBM routine, then name your routine IEFSD094. This should reside in SYS1.LINKLIB or in a library concatenated to LINKLIB through a LNKLSTxx member of SYS1.PARMLIB.
- The routine must use standard entry and exit linkages, saving and restoring its caller’s registers, and returning to its caller through the RETURN macro, with a return code in register 15.
- The routine must use the QSAM PUT macro in locate mode to write separation records to the output data set.
- The routine must use the GETMAIN and FREEMAIN macros, or the STORAGE macro, to obtain and release the storage required for work areas.
- The routine must establish its own synchronous error exit routine, and place the exit address in the DCBSYNAD field of the output DCB. The error routine will receive control during output writing in case of an uncorrectable I/O error; it must set a return code of 8 (binary) in register 15 to indicate an unrecoverable output error.

If the separator routine completes processing successfully, it must set a return code of 0 in register 15, before returning to its caller.

Note: The separator routine receives control in problem-program state, but with a protection key of 0. Therefore, the routine must ensure data protection during its execution.

Using the Block Character Routine: For printer-destined output, the separator routine can use an IBM-supplied routine to construct separation records in a block character format. This routine is a reentrant module named IEFSD095 that resides in the module library SYS1.AOSB0.

The block character routine constructs block letters (A to Z), block numbers (0 to 9), and a blank. The separator routine furnishes the desired character string and the construction area. The block characters are constructed one line position at a time. Each complete character is contained in 12 lines and 12 columns; therefore, a block character area consists of 144 print positions. For each position, the routine provides either a space or the character itself.

The routine spaces 2 columns between each block character in the string. However, the routine does not enter blanks between or within the block characters. The separator routine must prepare the construction area with blanks or other desired background before entering the block character routine.

To invoke the IBM-supplied block character routine, the IBM-supplied separator routine executes the CALL macro with the entry point name of IEFSD095. Since the block characters are constructed one line position at a time, complete construction of a block character string requires 12 entries to the routine. Each time, the address of a 7-word parameter list is provided in register 1.

The parameter list contains the following:

Table 54. Block Character Routine Parameter List

Byte	Meaning
0-3	This fullword is the address of a field containing the desired character string in EBCDIC format.
4-7	This fullword is the address of a field containing the line count as a binary integer from 1 to 12. This represents the line position to be constructed on this call.
8-11	This word is the address of a construction area where the routine will build a line of the block character string. The required length in bytes of this construction area is $14n-2$, where n represents the number of characters in the string.
12-15	This word is the address of a fullword field containing, in binary, the number of characters in the string.
16-19	Address of the data set PROC name in the SSOB extension.
20-23	Address of the data set STEP name in the SSOB extension.
24-27	Address of the data set DD name in the SSOB extension.

Appendix B. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the document. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this document at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of z/OS.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

- Programming Interface Information
- End of Programming Interface Information

Trademarks

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

- ACF/VTAM
- BookManager
- CICS
- DFSMS
- DFSMS/MVS
- DFSMSdfp
- DFSMSdss
- DFSMShsm
- DFSMSrmm
- DFSORT
- ESCON
- FFST/MVS
- GDDM
- Hardware Configuration Definition (HCD)
- IBM
- IBMLink
- IMS
- Language Environment
- MVS/DFP
- MVS/ESA
- MVS/SP
- NetView
- OpenEdition
- OS/2
- OS/390
- Print Services Facility (PSF)
- RACF
- Resource Link
- RMF
- SOMobjects
- SP
- SystemView
- VisualLift
- VTAM
- z/OS
- z/OS.e

NetView is a trademark of International Business Machines Corporation or Tivoli Systems Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks and service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in JES3 documentation. If you do not find the term you are looking for, refer to the index of the appropriate JES3 manual or view *IBM Glossary of Computing Terms*, located at:

www.ibm.com/ibm/terminology

This glossary includes terms and definitions from:

American National Standard Dictionary for Information Systems, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by an asterisk (*) that appears between the term and the beginning of the definition; a single definition taken from ANSI is identified by an asterisk after the item number for that definition.

A

action message. A request for operator intervention from the operating system. In JES3, action messages are typically displayed on the operator's console.

address space. The virtual storage assigned to a job, TSO user, or a task initiated by the START command. Each address space consists of the same range of addresses.

Advanced Function Presentation (AFP). A set of licensed programs, together with user applications, that use the all-points-addressable concept to print on presentation devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information. See *presentation device*.

Advanced Program-to-Program Communication (APPC). A set of inter-program communication services that support distributed transaction processing in a SNA network. See also *logical unit type 6.2*.

AFP. See *Advanced Function Presentation*.

all points addressability. The ability to address, reference, and position text, overlays, and images at any defined position or pel on the printable area of the paper. This capability depends on the ability of the hardware to address and to display each picture element.

APA. See *all points addressability*.

APPC. See *Advanced Program-to-Program Communication*.

APPC/VTAM. The implementation of APPC on VTAM.

auxiliary task. A subtask under the JES3 primary task. Writer DSPs and the General Services DSP do some of their processing under this task.

auxiliary task control block (ATCB). A control block that JES3 uses to manage work done under the auxiliary task.

auxiliary task dispatching element (ATDE). A control block that JES3 uses to determine whether to dispatch a function control table (FCT) under the JES3 auxiliary task.

B

binary synchronous communication (BSC). (1) Communication using binary synchronous transmissions. (2) A uniform procedure, using a standardized set of control characters and control character sequences, for synchronous transmission of binary-coded data between stations.

binary synchronous communications remote job processing (BSC RJP). A facility that permits the input and output of jobs to and from BSC workstations.

Bulk Data Transfer (MVS/BDT). (Multiple Virtual Storage/Bulk Data Transfer) An IBM program product that uses SNA protocols to copy sequential or partitioned data sets within an SNA network.

C

call. See *communication call*.

called job. A job created by JES3 in response to a JES3 CALL command.

called DSP. A job created by JES3 in response to a JES3 *CALL command.

channel-to-channel (CTC) adapter. A device for connecting two channels on the same processor or on different processors.

cold start. For JES3, the first start after system generation and after some unrecoverable failures. Spool data sets are initialized during a cold start.

Glossary

common area. In MVS, an area of virtual storage that is addressable by all address spaces.

Common Programming Interface. Provides languages, commands and calls that allow the development of applications that are more easily integrated and moved across environments supported by Systems Application Architecture.

common service area (CSA). In MVS, a part of the common area that contains data areas accessible from all address spaces.

communication call. A conversation statement that transaction programs can issue to communicate through the LU 6.2 protocol boundary. The specific calls that a transaction program can issue are determined by the program's current conversation state. See also *verb*.

configuration. The arrangement of a computer system or network as defined by the nature, number, and chief characteristics of its functional units.

console authority level. A numeric value from 0-15 assigned to RJP consoles which governs the set of commands that can be issued from the console.

console destination classes. A set of named classes used by JES3 to direct messages to certain consoles. Also used in specifying the messages to be received at an RJP console.

control section (CSECT). The part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining main storage locations.

console service. A DSP that performs traffic management for consoles.

control statements. Statements placed into an input stream to identify special JES3 processing options for jobs.

converter/interpreter (C/I) DSP. A DSP that uses MVS converter/interpreter subroutines to process JCL statements. The C/I DSP creates internal JCL text for jobs being readied for MVS execution.

CPI. See *Common Programming Interface*.

CPU. Central processing unit (equivalent to the term **processor**).

CTC. Channel-to-channel.

D

data link. The physical connection and the connection protocols between a host and a communication controller nodes by using the host data channel.

DC. Dump core.

DDR. Dynamic device reconfiguration.

deadline scheduling. A method of scheduling jobs by time of day, or by week, month, or year.

deferred-printing mode. A printing mode that spools output through JES to a data set instead of printing it immediately. Output is controlled by JCL statements.

demand select job. A job created by MVS and passed to JES3 for processing. MVS creates demand select jobs in response to MVS START or MOUNT commands or the TSO LOGON command. (For processing of these commands, system resources are needed, hence JCL is used to define those resources. It is this JCL that JES3 processes.)

destination queue (DSQ). For JES3, a control block used by subsystem interface routines to route requests (represented by destination codes) to the JES3 routines responsible for servicing the requests.

dependent job control (DJC). The organizing of a collection of jobs that must execute in a specific order. DJC manages jobs that are dependent upon one another.

destination codes. For JES3, numeric codes used to represent information during communication between JES3 components on different processors by using the subsystem interface.

device fencing. Reserving devices for use only by jobs within a specified job group, or jobs with a specified job network.

DJ. Dump job.

DJC. Dependent job control.

DJC network. A set of jobs that JES3 must run in a predetermined order. Success or failure of one job can cause execution, holding, or cancelation of other jobs.

DR. Disk reader.

DSI. Dynamic system interchange.

dump job (DJ). A JES3 dynamic support program, invoked by operator command to write JES3 jobs to tape and later to restore them back to JES3 by reading them from tape back into the system.

dyadic. A multiprocessor that contains two CPUs (hardware term that is not normally used in software documentation).

dynamic destination queuing. The facility that allows the separate queueing of staging areas received by the JES3 global address space from the FSS address space.

dynamic allocation. For JES3, assignment of system resources to a job while it is executing rather than before it is executed.

dynamic device reconfiguration (DDR). A facility that allows a demountable volume to be moved, and repositioned if necessary, without abnormally terminating the job or repeating the initial program load procedure.

dynamic support program (DSP). Multiprogrammed JES3 system components that are scheduled by JSS and cause the implementation of some function of JES3. DSPs can be directly related to job execution (e.g., main service, output service) or can be a background utility such as card-to-tape.

dynamic system interchange (DSI). A JES3 recovery facility that allows the operator to switch the JES3 global functions to a local processor in case of global processor failure.

dynamic writer. An output service function that controls printing or punching of data sets with characteristics that are not assigned to a specific device but are assigned by JES3 to appropriate devices as they become available.

E

early resource release. The releasing of resources (devices, volumes, and data sets) after they are no longer needed.

explicit setup. The programmer's specification, on a JES3 control statement, of precisely which devices are to be set up.

external writer. An MVS routine that directs system output to unsupported devices such as unit record printers and punches, magnetic tape devices, DASD, and plotters. External writers must be started by the operator as required. Once started, an external writer requests output data sets from the JES3 output service DSP via the subsystem interface.

F

FCB. Forms control buffer.

full function mode. The state that permits a printer to produce page-mode output.

function codes. Numeric codes used by MVS when requesting a service or control information from JES3 by using the subsystem interface.

function control table (FCT). The master dispatching queue for JES3. Entries in the FCT are arranged in priority order and each represents a DSP to be dispatched.

functional subsystem (FSS). A functional subsystem performs JES3 functions on behalf of the JES3 global address space while residing in its own address space, which may be on any processor in the complex. The functional subsystem off-loads some of the work from the JES3 address space.

functional subsystem application (FSA). Contained within the functional subsystem address space, these routines handle a specific piece of JES3 work normally done by the JES3 global processor.

functional subsystem intercommunication (FSI). Provides formal communication between JES3 and the functional subsystem application or FSS.

G

generalized main scheduling (GMS). A set of algorithms that allow the JES3 system programmer to tailor job scheduling and selection to the specific needs of the installation.

global processor. The processor that controls job scheduling and device allocation for a complex of processors. See also **local processor**.

global main (and local mains). The **global main** controls job scheduling and device allocation for a complex of JES3 processors. Each **local main** in the complex exists under control of the JES3 global main and is connected to the global main by CTC adapters. The JES3 on the global main can perform centralized job input, job scheduling, and job output services. Only the global main performs scheduling functions, although scheduled work executes on the local mains. See also **local main**.

GMS. Generalized main scheduling.

H

high watermark setup (HWS). An attempt to allocate a minimum number of unique device types that fulfill the requirements for each job step. Devices used in one step can be released and used again in later steps.

hot start. A restart of the global processor using information obtained from the last set of initialization statements processed. Recovery is attempted for all jobs that were in execution at the time of the failure.

hot start with analysis. A special form of hot start where the JES3 job queue is examined and the operator is given the opportunity to delete any jobs that would cause another restart.

hot start with refresh. A special form of hot start where the JES3 initialization stream is read.

Glossary

hot writer. An output writer that must be started and stopped by the operator. Hot writers are typically used when operator intervention is anticipated (as for changing forms, etc.).

I

initialization. In JES3, the process that reads the JES3 initialization statements and creates the tables and control blocks used throughout the JES3 program.

input service. The function that accepts and queues all jobs, entering the JES system, except those invoked via the *CALL command.

input service driver (ISDRVR) DSP. A DSP that reads batches of jobs from the spool data set and constructs a separate JCT entry for each job.

input service job. A job created by the card, tape, or disk reader DSP for each batch job written on the spool data set. An input service job is represented by a JCT containing two scheduler elements: one for the ISDRVR DSP and one for the PURGE DSP.

installation exit. A part of JES3 specifically designed for replacement by user-written routines.

internal reader. A JES3 routine that processes input streams contained in SYSOUT data sets obtained from MVS.

IPL. Initial program load.

J

JCL. See *Job Control Language*.

JECL. See *Job Entry Control Language*.

JES control table (JESCT). A control block in the MVS nucleus that contains information used by subsystem interface routines.

JES managed. The system mode of operation where JES3 batch initiators are controlled by JES3.

JES2. A subsystem that receives jobs into the MVS system and processes all output produced by the jobs. In multiple-processor complexes, the JES2 program manages independently-operating processors via a common job queue.

JES3. A subsystem that receives jobs into the MVS system, optionally schedules resources for the jobs, and processes output data produced by the jobs. In multiple-processor complexes, the JES3 program manages processors so that one processor exercises centralized control over the others and distributes jobs to the others by a common job queue.

JES3 auxiliary address space. An address space used exclusively by JES3 for data areas that would otherwise be placed into the CSA. Parameters in JES3 initialization statements specify whether a JES3 auxiliary address space is desired and, if so, the size of each data area.

JES3 devices. The devices that JES3 uses to communicate with the operator, read jobs, store jobs awaiting execution, and write job output. See also *shared devices*.

JES3-managed devices. The devices that JES3 allocates to jobs. See also *MVS-managed devices*, *jointly-managed devices*, *shared devices*.

JES3 spool access method (JSAM). Data management routines that serve JES3 address space requests such as allocation and deallocation of JES3 buffers.

job class. A named collection of JES3 job processing and scheduling rules. Use of job class names on JES3 control statements is a way of specifying what job processing and scheduling rules JES3 should use for jobs.

job class group. A named collection of resources to be associated with a job class. Use of job class names on JES3 control statements is a way of specifying what resources will be needed for jobs.

job control table (JCT). A table into which one entry is placed for each job that JES3 is to process. Entries are arranged in the JCT in job priority order to facilitate later job selection by priority.

job control table (JCT) entry. A control block into which JES3 places the description of a job to be processed, and scheduler elements representing the DSPs needed to process the job.

Job Control Language (JCL). A problem-oriented language designed to express statements in a job that identify the job or describe its requirements to an operating system.

Job Entry Control Language (JECL). A problem-oriented language designed to express statements in a job that describe its requirements to an operating system's job entry subsystem.

job ID. An 8-character identifier used by JES3 to uniquely identify any job in a JES3 complex at any moment in time. The job identifier is of the form "JOBnnnnn" where nnnnn is the job *number* with the appropriate number of leading zeroes, if the job number is 99,999 or less. Otherwise, the job identifier is of the form "Jnnnnnnn", where nnnnnnn is the job number with the appropriate number of leading zeroes.

job number. A unique number assigned to a job by JES3. To create a job ID, JES3 adds the letters JOB in front of the job number if the job number is 99,999 or

less; otherwise the job number is left padded with zeroes up to seven digits and the letter J is added in front of this number.

job queue element (JQE). A control block containing a summary of information from a JCT entry. JQEs remain in storage and are used by JES3 instead of JCT entries for scheduling of work.

job segment scheduler (JSS) DSP. A DSP that scans the job control table (JCT) to locate scheduler elements eligible for processing, and then builds function control table (FCT) entries so the corresponding DSPs can be dispatched. JSS itself is represented by an FCT entry.

job summary table (JST). A table into which the converter/interpreter DSP places job setup requirements.

job validation. The process during JES3 initialization where JES3 examines the job-related spool control blocks to verify their validity. If JES3 finds incorrect control blocks, JES3 gives the system operator an opportunity to take corrective action to insure that JES3 initialization completes.

job volume table (JVT). A table into which the converter/interpreter DSP places the volume information it obtains from data definition (DD) statements.

jointly-managed devices. A special case where the same device is both a JES3-managed device and an MVS-managed device. Only direct-access devices with volumes that cannot be physically removed can be jointly-managed devices.

JSAM. See *JES3 Spool Access Method*.

L

line mode. A type of data with format controls that only allow a printer to format data as a line.

line mode data. A type of data that is formatted on a physical page by a printer only as a single line.

local console. Any console that is dedicated to a single main within a JES3 installation. A remote job processing console cannot be a local console.

local device. A device attached to a host processor by using a channel.

local main. In a complex of processors under control of JES3, a processor connected to the global main by a CTC adapter, for which JES3 performs centralized job input, job scheduling and job output services by the global main.

local start. A restart of a local processor. Initialization is unnecessary and user jobs are not affected.

logical storage. The amount of central storage required by a job or a job step to execute efficiently on a processor when running under JES3.

loosely-coupled multiprocessing. Two or more computing systems interconnected by an I/O channel-to-channel adapter. The processors can be of different types and have their own unique configurations.

logical unit. 1) a type of network addressable unit that enables end users to communicate with each other and gain access to network resources. 2) A port providing formatting, state synchronization, and other high-level services through which an end user communicates with another end user over an SNA network.

logical unit type 6.2. The SNA logical unit type that supports general communication between programs in a distributed printing environment; the SNA logical unit type on which CPI communications is built.

LU. See *logical unit*.

M

main. A processor named by a JES3 MAINPROC initialization statement, on which jobs can execute; represents a single instance of MVS. The two types of mains are (1) global main, and (2) local main.

MAINPROC. A JES3 initialization statement that defines a processor to JES3.

main device scheduler (MDS). Controls the setup of I/O devices associated with job execution.

main device scheduler (MDS). A phase of JES3 that controls the setup of I/O devices associated with job execution.

main DSP. A DSP that chooses jobs and supplies them to the MVS initiator(s).

main service. A dynamic support program that schedules problem programs for execution and manages the flow of data (system input, print, and punch) across the channel-to-channel adapter to and from the global processor.

MDS. Main device scheduler.

migration. The changing over from an installation's production operating system to an upgraded or entirely new operating system.

multifunction monitor (MFM). The master dispatcher for JES3. The MFM scans the function control table (FCT) for DSPs ready to be executed, and causes execution to begin.

Glossary

multiple console support (MCS). A feature of MVS that permits selective message routing of up to 99 operator's consoles.

multiple virtual storage (MVS). A virtual storage facility that allows each user a private address space.

multiprocessing system. A computing system employing two or more interconnected processing units to execute program simultaneously.

multiprocessor. A processor complex that consists of more than one CPU.

MVS. See *Multiple Virtual Storage*.

MVS/APPC. The implementation of APPC on an MVS system.

MVS-managed devices. The devices that MVS allocates to jobs. See also *JES3-managed devices*, *jointly-managed devices*.

N

network. For JES3, two or more systems and the connections over which jobs and data are distributed to the systems. One or more of the systems can be a JES3 global (and its local mains, if any). The other systems can be non-JES3 systems with compatible networking facilities. Connections can be established through communications paths using SNA or BSC protocol.

network job entry (NJE). The process in which a user at one installation can submit a job/output to be executed at or sent to a different installation (node to node). NJE is networking between installations using SNA or BSC protocol.

network job stream. A network job stream includes:

- a job header
- an MVS job comprised of JCL and/or SYSIN data
- a job trailer.

See also the definition of network SYSOUT stream.

network stream. A network stream contains either a network job stream or a network SYSOUT stream. See the respective definitions for each.

network job. Same as network stream.

network SYSOUT stream. A network SYSOUT stream includes:

- a job header
- a data set header (where there may be more than one data set header per SYSOUT data set transmitted)
- a SYSOUT data set
- a job trailer.

Note: There may be more than one SYSOUT data set-data set header pair. See also the definition of network job stream.

networking protocol. Rules for using communication lines. Protocols can identify the direction of data flow, where data begins and ends, how much data is being transmitted, and whether data or control information is being sent. The two protocols that JES3 uses to establish a networking environment are binary synchronous communication (BSC) and systems network architecture (SNA).

NJE. An installation to installation data communication network.

node. 1) An end point of a link, or a junction common to two or more links in a network. Nodes can be processors, controllers, or workstations. Nodes can vary in routing and other functional capabilities. 2) In JES3, one of the systems in a network of systems connected by communication lines. Each node defined to itself is the home node. All others are defined as remote nodes, directly or indirectly connected. The home node and the remote nodes are identified as such in the installation's initialization stream (NJERMT statement).

non-partitionable processor complex. A processor complex that cannot be partitioned.

non-standard job. A job for which JES3 defines processing from input received on `//*PROCESS` control statements.

normal job. A job received by JES3 in an input stream. Normal jobs can be standard jobs or nonstandard jobs. Contrast with "called job".

O

operating system. The software that controls the operation of a processor complex.

operator commands. Statements that system operators may use to get information, alter operations, initiate new operations, or terminate operations.

operator messages. A message from an operating system directing the operator to perform a specific function, such as mounting a tape reel; or informing the operator of specific conditions within the system, such as an error condition.

output scheduling element (OSE). A control block that describes the characteristics of one or more output data sets of the same job.

output service. The function that processes SYSOUT data sets. Processing includes printing, punching, or directing output to an external writer.

output service (OUTSERV) DSP. A DSP that schedules output writers for printers or punches, and

routes output data to TSO processor, MVS external writers, and the MVS internal reader.

output writer. A JES3 routine that transcribes output data sets to the printer or punch system output devices.

P

page mode. The mode of operation in which the AFP print (such as the 3800 Printing Subsystem) can accept a page of data from a host processor to be printed on an all points addressable output medium.

page mode data. A type of data that can be formatted anywhere on a physical page. This data requires specialized processing such as provided by the Print Services Facility for AFP printers, such as the 3800-3 and 3820.

page mode printer. An AFP printer, such as the 3800 model 3 and 3820, that can print page-mode data.

partition. Equivalent to the term **physical partition**.

partitionable processor complex. A processor complex that can be partitioned.

partitioned mode. Equivalent to the term **physically partitioned mode**.

partitioning. The process of forming multiple physical partitions from one processor complex.

physical partition. A set of hardware resources, formed by partitioning, that can support a single operating system.

pel. Picture element

physically partitioned mode. The state of a processor complex when its hardware resources are divided into multiple configurations.

pre-execution setup. That portion of setup performed by MDS prior to a job entering execution.

presentation device. A device that produces character shapes, graphics pictures, images, or bar code symbols on a physical medium. Examples of physical media are display screens, paper, foils, microfilm, and labels.

primary job entry subsystem. The active job entry subsystem. The primary job entry subsystem is determined during the system generation process.

primary task. The task under which most DSPs execute.

Print Services Facility (PSF). An IBM licensed program that produces printer commands from the data set to it. PSF programs run on the z/OS, OS/390, MVS, VM, VSE, OS/2, AIX, and OS/400 operating

platforms. For JES, PSF programs operate the 3800 model 3 and 3820 printers. PSF operates as a functional subsystem.

process SYSOUT (PSO). An interface to JES3 to allow access and control of SYSOUT data sets from other address spaces. It is used primarily by TSO OUTPUT and RECEIVE commands and external writers.

processor. A hardware unit that contains software to interpret and process instructions.

processor complex. The maximum set of hardware resources that support a single operating system.

protected buffer pool (PBUF). An area in the common storage area and JES3 auxiliary address space that has been divided into buffers.

protocol. The meaning of, and the sequencing rules for, requests and responses used for managing a network, transferring data, and synchronizing the states of network components.

purge DSP. A DSP that performs post-execution removal a job from the system, writes system management facilities (SMF) records, and frees spool space used by the job.

R

RACE. Resource Access Control Facility

reader DSP. A DSP that transfers a job's control statements and SYSIN data from an input device to the spool data set. Three types of readers exist: card reader, tape reader, and disk reader.

reader job. A called job created by JES3 each time the operator issues a CALL command for a card, tape, or disk reader.

reconfiguration. The process of adding hardware units to, or removing hardware units from, a configuration.

remote device. A device attached to a host processor by using a data link.

remote job entry (RJE). A process in which a user at a remote site is connected to the host system by a data link (telecommunication lines). RJE and RJP is networking between the user and the host system.

remote job processing (RJP). A facility that permits the input, processing, and output of jobs to and from terminals remote from the JES3 installation.

RJP. Remote job processing.

remote terminal processor (RTP). A programmable remote workstation.

Glossary

resident queue (RESQUEUE). A control block built in storage by the job segment scheduler to represent a scheduler element during the life of the scheduler element. It contains status information and queuing pointers.

Resource Access Control Facility (RACF). An IBM program product that provides for access control by identifying and verifying users to the system, authorizing and logging access to protected resources, and logging detected unauthorized attempts to enter the system.

RMT. Remote terminal processor program.

remote terminal processor (RMT). A self-loading object deck created as a result of an RMT generation. RTP programs allow JES3 to communicate with programmable remote workstations.

routing code. An MVS identifier that you use to route MVS messages to a specific console(s).

RTAM. Remote terminal access method.

RTP. Remote terminal processor.

S

SAA. See *Systems Application Architecture*.

scheduler element. A part of a job control table (JCT) entry. (Each JCT entry may contain multiple scheduler elements.) Each scheduler element represents one or more DSPs needed for JES3 processing of a job.

scheduling environment. A list of resource names along with their required states. If an MVS image satisfies all of the requirements in the scheduling environment associated with a given unit of work, then that unit of work can be assigned to that MVS image. If any of the requirements are not satisfied, then that unit of work cannot be assigned to that MVS image.

service class. A group of work which has the same performance goals, resource requirements, or business importance. For workload management, you assign a service goal and optionally a resource group to a service class.

server mode. A processing mode of the JES3 dump job function that runs in its own address space and can utilize any tape devices in the system.

session. A logical connection between two logical units that can be activated, tailored to provide various protocols, and deactivated as requested.

setup. The phase of JES3 processing that performs volume fetch, device, volume, and dataset allocation.

setup DSP. A DSP that performs volume fetch, job setup, high watermark setup, and explicit setup functions.

shared devices. (1) Devices that are connected to more than one processor. (2) Devices that are both JES3 devices and JES3-managed devices.

side. Equivalent to the term **physical partition**.

single-image mode. The state of a processor complex when all of its hardware resources are in a single configuration.

SNA. See *Systems Network Architecture*.

solicited message. A message that is a response to a command (also see unsolicited message).

spool data management. For JES3, the recording and retrieval of data on the spool data set and the management of space within the spool data set.

spool device. A direct-access device that JES3 uses for intermediate storage of control blocks and data needed for processing jobs. When JES3 is used for multiprocessing, the spool device becomes a collection point for job input data to be distributed to local processors, and for job output data coming from local processors enroute to I/O devices attached to the global processor.

spool device. A direct-access device that JES3 uses for intermediate storage of control blocks and data needed for processing jobs. When JES3 is used for multiprocessing, the spool device becomes a collection point for job input data to be distributed to local mains, and for job output data coming from local mains enroute to I/O devices attached to the global.

spool partition. A named collection of spool data sets.

staging area. An area into which subsystem interface routines store data to be transferred between address spaces. Staging areas can be contained in the common service area (CSA), or in an optional JES3 auxiliary address space. The staging areas are accessible from all address spaces.

staging drive group. A collection of staging drives for space management and recovery. It is created by the user with the Mass Storage Control Table Create program.

standard job. A job for which JES3 defines needed processing entirely from Input Service, Converter Interpreter, MAIN service, Output Service, and PURGE service.

statistics data area (SDA). A data area used to collect JES3 processing statistics by using the IATXSTAT macro.

Storage management subsystem (SMS). An MVS subsystem responsible for managing data sets and volumes. This subsystem supports JCL constructs such as storage class and storage group.

subsystem identification block (SSIB). The control block into which MVS places the name of the subsystem to which it is directing a request over the subsystem interface.

SSI. Subsystem interface.

subsystem interface (SSI). A set of program routines that allows two-way communication between a JES3 address space and other address spaces.

subsystem options block (SSOB). The control block into which MVS places a function code when communicating with JES3 over the subsystem interface. The function code identifies a requested service.

subsystem services common services. A term used to collectively identify JES3 routines that handle communication among JES3 modules running on separate processors. (For example, a subsystem interface service routine and a receiving DSP would be referred to as subsystem interface common services.)

system management facilities (SMF). An optional control program feature of MVS that provides the means for gathering and recording information that can be used to evaluate system usage.

systems application architecture (SAA). A set of software interfaces, conventions, and protocols that provide a framework for designing and developing applications with cross-system consistency.

systems network architecture (SNA). The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through a communication system.

systems network architecture remote job processing (SNA RJP). A facility that permits the input and output of jobs to and from SNA workstations.

systems network architecture/network job entry (SNA/NJE). A networking capability that works in combination with MVS/Bulk Data Transfer (MVS/BDT). Networking is established between nodes through MVS/BDT "sessions." Sessions can be established over telephone lines, microwave links, by satellite, or by channel-to-channel adapters.

T

TP. See *transaction program*.

transaction program. An application program that allows users to access resources in a SNA network.

U

uniprocessor. A processor complex that consists of only one CPU.

unsolicited message. A message that is not a response to a command (also see solicited message).

USAM. User spool access method.

user buffer pool (UBUF). An area in each user's address space that has been divided into buffers.

user spool access method (USAM). Data management routines that do not execute in the JES3 address space but provide the subsystem interface for allocation, deallocation, SYSIN/SYSOUT, OPEN, and CLOSE functions of user data sets.

V

volume. That portion of a single unit of storage that is accessible to a single read/write mechanism; for example, a drum, a disk pack, or part of a disk storage module.

VTAM. Virtual telecommunications access method.

W

warm start (W). For JES3, a restart where an IPL must be performed on all processors and there is a choice of using the last set of initialization statements processed or a new set of initialization statements.

warm start with analysis (WA). For JES3, a special form of warm start where the JES3 job queue is examined and any jobs that would cause another restart are automatically deleted.

warm start to replace a spool data set (WR). For JES3, a special form of warm start where a spool data set can be replaced by another data set with the same ddname; all jobs with data on the replaced spool data set are lost.

warm start with analysis to replace a spool data set (WAR). For JES3, a special form of warm start (W) combining warm start with analysis (WA) and warm start to replace a spool data set (WR) processing.

WLM managed. The system mode of operation where JES3 batch initiators are controlled by the workload management component of MVS.

Workload Management (WLM). WLM is a component of MVS that manages system resources.

workstation. A station at which an individual can send data to or receive data from a computer for the purpose of performing a job.

Glossary

writer. See *output writer*.

writer output multitasking. For JES3, a facility by which writer output processing can be performed concurrently with other JES3 functions on a multiprocessor global processor.

WTO. Write to operator.

WTO/R. Write to operator with a reply request.

Index

Special characters

???????? userID 68
//*FORMAT JES3 control statement
 changing 140
 OSE information and 140
 override sequence
 with "default" statement 145, 146,
 147, 148
 with "direct" statement 142, 143
//*MAIN control statement 101
//*MAIN JES3 control statement
 DEADLINE parameter 125
 LREGION parameter 132
 main selection for job execution 97
 PROC parameter 96, 176
 procedure library 96
 scheduler element 96
 SETUP parameter 107, 108
 SYSTEM parameter 97, 128, 243
 to specify the track group allocation
 size 194
 TYPE parameter 128
 UPDATE parameter 96, 177
//*NET JES3 control statement 125
 ABCMP parameter 127
 DEVPOOL parameter 126
 examining with a installation exit
 routine 126
 for a nonstandard DJC job 128
 NHOLD parameter 126
 RELSCHCT parameter 126
//*PAUSE JES3 control statement 204
//*PROCESS JES3 control statement 95
 effect on scheduler element 96
 for a nonstandard DJC job 126, 128
(HCD) hardware configuration
 definition 4
*MODIFY,CONFIG command 34
 add C/I FSS 34
 use 10
&RACLNDE profile
 creating 54, 70
 description 70
 used in WRITER profile 79
&RACLNDE variable
 recommended for JESJOBS
 profiles 52
 recommended for JESSPOOL
 profiles 73
 recommended for NODES
 profiles 57
&SUSER value
 on ADDMEM operand in NODES
 class
 description 49
 example showing simple NJE user
 translation 66
+++++++ userID 69

Numerics

3495 configuration example 318
 configuration illustration 318

A

access
 data set 258
access list 44
accessibility 347
acquire
 console 203
add device panel 254
adding a device 32
 FSS-managed printer 32
 printer 32
adding a FSS managed printer 32
address space JCL statement limit 170
 changing 172
 displaying 174
 selecting 172
ADDRSORT parameter
 on the SETPARAM initialization
 statement 111, 262
ALLOCATE parameter
 on the SETPARAM initialization
 statement 104
allocation
 of a resource 104
allocation for 3495 tape data set 315
 automatic class selection (ACS) 315
 data class 315
 storage class 315
 storage group 315
APAR cover letter
 contents 28
assign a security label 40
assign/unassign facility 244
assignable device 244
authority
 using RACF 81
authorization
 command 89
authorizing inbound work (JES) 56
authorizing outbound work 70
authorizing use of printers 80
automatic restart management
 JES3 considerations 137
AUX (auxiliary address space)
 staging area
 allocate 274
 USAM buffer
 allocate 274
 determine how many to
 allocate 274
auxiliary task
 description 275
available-devices queue 151
avoidance of cold start 24

B

BADTRACK initialization statement
 impact on checkpoint data set
 size 229
BADTRACK table
 impact on checkpoint data set
 size 229
BALJ data area
 storage requirements for 161
BAR parameter
 on the GROUP initialization
 statement 123, 137
batch job
 preventing unauthorized users from
 running
 with JES 3.1.3 or later installed 43
 scheduling to a CI DSP 168
benefit
 of security 39
binary synchronous communication 293
block character routine
 See output separator routine for an
 external writer
breakdown of a resource 106
BSC (binary synchronous
 communication) 300
 CTC (channel-to-channel)
 connection 294
 definition 293
 use of RTAM 294
BSC protocol 295
BSC RJP (binary synchronous
 communication remote job
 processing) 204
 cancel 280
 console
 authority level 280
 defining 204
 data compression 279
 data security consideration 279
 debug facility 280
 hardcopy log 280
 initialization statements that
 affect 281
 initiate 280
 job routing 279
 line
 error statistic top 280
 snap facility 280
 multi-leaving 279
 operator communication 280
 overview 279
 use 279
buffer
 defining for spool I/O 271
 JES3 buffer
 determining the size 271
 JSAM 271
 PBUF 271
 UBUF 271
 USAM 271

- BUFFER initialization statement
 - BUFSIZE parameter 271
 - GRPSZ parameter 192
 - PAGES parameter 272
 - SPLIM parameter 186, 198
 - buffer pool
 - JES3 buffer pool
 - determining size 272
 - PBUF (protected buffer pool)
 - determining size 273
 - BUFSIZE parameter
 - on the BUFFER initialization statement 271
- C**
- C/I (converter/interpreter) service
 - address space JCL statement
 - limit 170
 - changing 172
 - displaying 174
 - selecting 172
 - address space selection for a job 167
 - balancing the work load 170
 - C/I FSS (functional subsystem)
 - address space
 - overview for job management 99
 - changing 174, 175
 - CI DSPs, maximum number 163
 - converter/interpreter phase 99
 - dependent job control JCL scan 126
 - description 159
 - inquiring 174
 - job
 - scheduling to a CI DSP 168
 - job JCL statement limit 170
 - changing 171
 - overriding with IATUX41 171
 - selecting 171
 - jobs, controlling 166
 - with an installation exit 166
 - managing the SWA (scheduler work area) 170
 - maximum number in processing 163
 - modifying 174, 175
 - with an installation exit routine 99, 166
 - monitoring 174
 - operator command 174
 - options list 166, 169
 - overview of job management 99
 - POSTSCAN DSP 163, 165
 - postscan phase 100
 - cataloged data set resolution 100
 - where it takes place 160
 - prescan phase 100
 - procedure library 176
 - processor selection for a job 167
 - scheduler work area 170
 - setting up 160
 - status, displaying 174
 - tuning 174
 - C/I (converter/interpreter) subtask
 - for starting an C/I FSS address space 164
 - C/I FSS address space
 - abnormal termination,
 - preventing 171
 - adding 176
 - advantages to using 160
 - checkpoint data sets and 228
 - CI DSPs, maximum number 162, 163, 164
 - common storage constraints,
 - avoiding 161
 - common storage requirement 161
 - defining 162
 - description 159
 - ensuring access to a procedure
 - library 176
 - how many to use 160
 - impact on MVS performance 162
 - JCL statements, maximum
 - number 163
 - private virtual storage constraint,
 - avoiding 164
 - start procedure 162
 - starting automatically 162
 - storage requirement 161, 164
 - virtual storage requirement 164
 - where to put 160, 161
 - CANCEL command
 - controlling who can cancel jobs by job
 - name 53
 - cancelling jobs
 - controlling who can cancel jobs by job
 - name 53
 - catalog 243
 - cataloged data set resolution 100
 - cataloged start procedure 7
 - modifying for JES3 7
 - sample 8
 - cataloged start procedure (JES3) 7
 - statements 8
 - checkpoint data set
 - adding 228
 - allocate
 - risk 228
 - space requirement 228
 - where to place 228
 - allocating 228
 - ensuring against loss 228
 - I/O error
 - permanent 228
 - minimizing effect of losing one 229
 - replacing 228
 - size, determining 228
 - CHKPNT2 data set 7
 - CHOICE parameter
 - on the SELECT initialization
 - statement 122, 123, 134
 - CI DSP
 - effect on POSTSCAN DSP 163
 - ensuring access to a procedure
 - library 176
 - in C/I FSS address space
 - storage requirement 164
 - inquiring 174
 - number, maximum in an address
 - space 163
 - status, displaying 174
 - storage requirement 163
 - CIBATCH 168
 - CICNT parameter
 - on the STANDARDS initialization
 - statement 163
 - CIDEMAND 168
 - CIPARM initialization statement
 - PARMID parameter 169
 - CKPNT data set 7
 - class 97, 169, 258
 - security 40
 - class 1 device 258
 - class 2 device 258
 - class 3 device 258
 - CLASS initialization statement 135
 - and default job selection 131
 - IORATE parameter 123, 128
 - LSTRR parameter 132
 - main selection 97
 - MDEPTH parameter 123, 135
 - MLIMIT parameter 123, 136
 - SDEPTH parameter 108, 135
 - SPART parameter 188
 - SYSTEM parameter 97
 - TDEPTH parameter 123, 135
 - TLIMIT parameter 123, 135
 - to define the job selection
 - environment 129
 - TRKGRPS parameter 194
 - CLAUTH (class authority) attribute
 - example for JESSPOOL class 75
 - CLIST exec
 - RACF sample 40
 - cold start 24
 - Use of JCT Utility 24
 - command 174
 - *MODIFY,CONFIG 31
 - authorization 81, 89
 - command processing
 - order of execution 223
 - compatibility mode
 - definition 249
 - specifying 249
 - conditional step execution
 - consideration 103
 - configuration 3
 - dynamically changing 31
 - console 205
 - acquiring 203
 - authority checking 89
 - authority level 89
 - BSC RJP
 - defining 204
 - defining 203
 - description 203
 - destination code
 - on the DEVICE initialization
 - statement 108
 - entering a command
 - JES3 command 205
 - MVS command 205
 - JES3 203
 - LOGON command 88
 - management 204
 - MCS 203
 - MCS (multiple console support)
 - defining 203
 - description 203

- console (*continued*)
 - MCS LOGON 205
 - multiple console support 205
 - RJP (remote job processing)
 - description 203
 - SNA RJP
 - defining 204
- console service
 - input processing 204
 - operator communication 204
 - overview 204
- consoles
 - LOGON 76
- CONSTD initialization statement
 - HARDCOPY parameter 208
- controlling where output can be processed 79
- controlling who can cancel jobs by job name 53
- controlling who can submit jobs by job name 52
- converter/interpreter options list
 - defining 169
 - selecting
 - by operator command 169
 - for a started task 170
 - for a TSO LOGON job 170
 - for an internal reader job 170
- converter/interpreter subtask 169
- copying
 - JCTs 238
- CSA (common service area)
 - reduce the amount used by JES3 274
- CTC (channel-to-channel)
 - connections 294
- CVOL (control volume) 243

D

- DAFETCH parameter
 - on the SETPARAM initialization statement 103
- data compaction 285
- data compression
 - with BSC RJP 279
 - with SNA RJP 285
- data management control block 161
- data set 13
 - access 258
 - allocating 227
 - bypassing JES3 checking 260
 - cataloged
 - in a CVOL catalog 243
 - in a VSAM catalog 243
 - migrated by HSM 101
 - private 100
 - resolution 100
 - supplying data with a installation exit routine 100
 - unavailable 101
 - checkpoint 228
 - CHKPNT2 7
 - CKPNT 7
 - IATPLBnn 7
 - IATPLBST 7
 - integrity 259, 260

- data set (*continued*)
 - differences between MVS and JES3
 - checking 260
 - examples of effects on job processing 261
 - integrity checking 258
 - JES3DRDS 7
 - JES3IN 7
 - JES3JCT 7
 - JES3OUT 7, 10
 - JES3SNAP 7, 10
 - JESABEND 7, 10
 - management 258
 - multivolume
 - job setup processing 107
 - MVS
 - integrity 259
 - output
 - deleting 199
 - deleting held 139
 - held 139
 - monitoring in a network 307
 - providing with a TAT (track allocation table) 188
 - spool partitions and 188
 - output processing
 - nonstandard 139
 - override sequence for
 - information 140
 - waiting for 139
 - overview 227
 - partitioned use for INCLUDE 13
 - resident
 - identifying 242
 - spool1 7
 - spoolnn 7
 - STEPLIB 7
 - SYSABEND 7, 10
 - SYSOUT
 - monitoring in a network 307
 - providing TAT (track allocation table) 188
 - spool partitions and 188
- dataset
 - IATPLBST 17
 - IATPLBT1 17
 - JES3IN 17
 - JES3LIB 17
 - JES3OUT 17
 - JESABEND 17
 - STEPLIB 17
 - STG1CODE 17
- DD JCL statement
 - modifying with a installation exit routine 98
 - OSE information and 139
 - SYSOUT data set 139
 - UNIT parameter 110, 111
- DD statement 13
 - JES3IN 13
- DEADLINE DSP 125
 - reinitializing after system stopped or quiesced 125
- DEADLINE initialization statement
 - and deadline scheduling
 - algorithm 125

- deadline scheduling
 - description 125
 - internal clock
 - resetting after system is stopped or quiesced 125
- debug facility
 - for BSC RJP 280
- default
 - process mode 246
- default NJE userID
 - ownership of SYSOUT based on NODES profiles 63
 - specifying with SETROPTS
 - command 68
- default userIDs
 - concepts 68
 - description 68
- define
 - JES3 to RACF 47
 - maximum number of jobs 229
 - PSF/MVS printer 250
 - SNA-attached printer 250
- delete an output data set
 - output data set 199
- demand select job
 - definition 160
 - scheduling to a CI DSP 168
 - using system symbols in source JCL 169
- DESTDEF statement 304
- determine
 - maximum number of printers 248
- device 203, 254
 - allocation
 - DD statement request 110
 - dynamic 254
 - system standard 111
 - assignable 244
 - class
 - definition 258
 - class specification 109
- DASD
 - record, track, and cylinder characteristics 194
- dedicating 109
- dedicating a specific device 111
- defining
 - prerequisite 243
- DJC networks, reserving for 126
- dynamic allocation
 - restriction 254
- esoteric group 254
- execution
 - definition 244
- fencing 255
- for a spool data set 182
 - choosing a track group size for 193
- I/O
 - allocating to JES3 254
 - and MVS VARY command 243
 - defining to JES3 243
 - dynamically reconfiguring 255
 - grouping 254
 - layout 3
 - management 257
 - online/offline status 243

- device (*continued*)
 - I/O (*continued*)
 - reconfiguring 255
 - JES3
 - defining 244
 - definition 244
 - jointly-managed
 - definition 244
 - multiple access
 - specifying XUNIT parameter for 109
 - overview 243
 - permanently resident 110
 - pooling 255
 - pooling for a job-class group 109, 255
 - printer 246
 - restricted 244
 - selection
 - limiting effect on main eligibility 105
 - setup and 107
 - shared
 - definition 244
 - subgeneric group
 - example 254
 - volume 256
 - volume removability 109
 - DEVICE initialization statement 244, 245
 - adding statements 243
 - changing statements 243
 - to define a network BSC line or CTC connection 302
 - to define an I/O device
 - FSSNAME parameter 249
 - MODE parameter 249
 - PM parameter 249
 - XTYPE parameter 256
 - XUNIT parameter 108
 - to define processor status
 - DTYPE parameter 267
 - DEVICE statement 317, 320
 - example of Device statement 317
 - execution device 320
 - JES3 dynamic support program 320
 - DEVICES class
 - activating 81
 - DEVPOOL parameter
 - on the GROUP initialization statement 109, 255
 - DFC (data flow control) 282, 286
 - disability 347
 - DJ (dump job) 199
 - DJC (dependent job control)
 - completion option 127
 - description 125
 - DJC network
 - cancel a network or job 127
 - defining 125
 - hold a network or job 127
 - initializing 126
 - job pending count 127
 - modifying using an operator command 127
 - purging 127
 - release a network or job 127
 - reserving devices for 126
 - DJC (dependent job control) (*continued*)
 - DJC network (*continued*)
 - scheduling 126
 - terminating 127
 - installation exit to examine `//*NET` statement 126
 - JCL scan 126
 - nonstandard DJC job processing 128
 - to serialize catalog access 243
 - DLOG option 208
 - DMC (data management control block)
 - storage requirements for 161
 - documents, licensed x
 - DSI (dynamic system interchange)
 - writer output multitask facility and 276
 - DSPCNT parameter
 - on the FSSDEF initialization statement 162, 164
 - DTYPE parameter
 - on the DEVICE initialization statement 267
 - dump job facility 199
 - using to free spool space 199
 - DYNALLOC 10
 - DYNALLOC initialization statement
 - and procedure library 176
 - impact on checkpoint data set size 228
 - to allocate a spool data set 182
 - dynamic allocation 263
 - improving performance 264
 - reducing spool I/O requests 264
 - dynamic LPA facility 31
 - dynamic writer 153
- E**
- EARLYVERIFY operand
 - SETROPTS command 47
 - esoteric device type 254
 - example
 - of a RACF profile 40
 - EXEC JCL statement
 - interpretation 100
 - modifying with a installation exit routine 98
 - execution device
 - defining 245
 - definition 244
 - exit
 - that affect security 46, 88
 - explicit setup
 - advantage 108
 - description 108
 - EXRESC parameter
 - on the GROUP initialization statement 109, 136, 255
 - external writer 154, 331
 - access to spool data sets 73
 - block characters to separate jobs 345
 - characteristics 331
 - description 331
 - features of IBM-supplied version 331
 - overview 331
 - parameter list 336
 - format and content 336
 - external writer (*continued*)
 - selection criteria
 - data set 332
 - starting the external writer 332
 - using the 3800 Printing Subsystem 336
 - XWTR cataloged procedure 332
 - for the 3800 Printing Subsystem 336
 - modifying for special print chains 335
- F**
- FACILITY class
 - activating 78
 - FETCH parameter
 - on the SETPARAM initialization statement 103
 - FMPs (function management presentation services) 284
 - FORMAT initialization statement 182
 - impact on checkpoint data set size 228
 - FSA (functional subsystem application) 152
 - installation exit 156
 - FSS
 - adding a managed printer 32
 - FSS (functional subsystem)
 - defining maximum number of printers 248
 - FSS mode
 - definition 249
 - specifying 249
 - FSSDEF initialization statement
 - CI DSPs, defining the maximum number 163
 - DSPCNT parameter 162, 164
 - MAXASST parameter 163, 172
 - PNAME parameter 162
 - START parameter 162
 - SYSTEM parameter 162
 - to create a C/I FSS address space 162
 - FSSNAME parameter
 - on the DEVICE initialization statement 249
- G**
- generic profile
 - top profile in JESJOBS class 52
 - global access checking table
 - entries for JESNEWS 76
 - restored jobs
 - restored jobs 76
 - global processor
 - restart characteristics 25
 - glossary 353
 - GMS (generalized main scheduling) 111
 - controlling job scheduling 133
 - deadline scheduling
 - description 125
 - default job selection 131
 - determine job eligibility 128

- group IDs
 - translating 66
- GROUP initialization statement
 - and default job selection 131
 - BAR parameter 123, 137
 - DEVPOOL parameter 109, 255
 - EXRESC parameter 109, 131, 136, 255
 - JSPAN parameter 123
 - to define the job selection environment 129
- GROUPJ qualifier
 - on NODES profiles 58
- GROUPS qualifier
 - on NODES profiles 58
- GRPSZ parameter
 - on the BUFFER initialization statement 192
 - on the SPART initialization statement 192

H

- HARDCOPY parameter
 - on the CONSTD initialization statement 208
- hardware configuration definition using 16
- hardware configuration definition (HCD) 253
 - adding devices 253
 - changing devices 253
 - deleting devices 253
- Hardware configuration definition (HCD) 4
- hardware configuration program
 - grouping a JES3 device 254
- hierarchical storage manager 101
- HOME node
 - NETHOLD parameter 304
- hot start 19
- hot start with analysis 20
- hot start with refresh 9, 20
 - starting JES3 9
- hot start with refresh and analysis 22
- hot writer 153
- how JES and RACF work together 43
- HSM (hierarchical storage manager)
 - locate request 101
- HWS (high-watermark setup)
 - advantage 107
 - description 107
- HWSNAME statement 317, 319
 - alternate name 319
 - configuration example 319
 - example of HWSNAME statement 317
 - rules for building statement 317

I

- I/O device 243
- I/O rate 122, 123, 128
- IASXWR00 module 336
- IATPLBnn data set 7
- IATPLBST data set 7, 17
- IATPLBT1 data set 17

- IATUTJCT
 - copying JCTs 238
 - JCT Utility 237
 - migrating 242
 - migrating JCTs 237
 - running the utility 238
 - setup and initialization 239
 - testing the processing 242
 - using the utility 240
- IATUX03 installation exit 166
- IATUX04 installation exit 100, 166
- IATUX05 installation exit 100, 166
- IATUX06 installation exit 100, 166
- IATUX07 installation exit 100, 166
- IATUX08 installation exit 101, 166
- IATUX09 installation exit 166
- IATUX10 installation exit 166
- IATUX11 installation exit 101, 166
- IATUX15
 - *MODIFY,CONFIG processing 34
- IATUX17 installation exit 96
- IATUX18 installation exit 46, 84, 88, 89
- IATUX19 installation exit 88, 150, 155
- IATUX20 installation exit 155
- IATUX21 installation exit 155
- IATUX22 installation exit 155
- IATUX23 installation exit 155
- IATUX24 installation exit 126
- IATUX25 installation exit 105
- IATUX26 installation exit 99, 166
- IATUX28 installation exit 98, 166
- IATUX29 installation exit 166, 189
- IATUX30 installation exit 46, 91
- IATUX31 installation exit 220
- IATUX33 installation exit 98, 166, 189
- IATUX34 installation exit 98, 139
- IATUX35 installation exit 46, 84, 88, 308
- IATUX36 installation exit 308
- IATUX37 installation exit 308
- IATUX38 installation exit 308
- IATUX39 installation exit 155, 308
- IATUX40 installation exit 308
- IATUX41 installation exit 99, 166, 171
- IATUX42 installation exit 308, 310, 311
- IATUX43 installation exit 155, 308
- IATUX44 installation exit 98, 139
- IATUX45 installation exit 155
- IATUX46 installation exit 99, 166, 167
- IATUX48 installation exit 199
- IATUX49 installation exit 99, 166, 167
- IATUX58 installation exit 46, 84
- IATUX59 installation exit 46, 84
- IATUX60 installation exit 46
- IATUX61 installation exit 106
- IATUX62 installation exit 105
- IATUX67 installation exit 46
- id 176
- IEBUPDTE utility 7
- IEFPROC EXEC 7
- IEFSD094 output separator routine 342
- IEFSD095 output separator routine 345
- IKJEFF53 installation exit
 - and JESJOBS class 52, 53
- inbound SYSOUT definition
 - DESTDEF statement 304
- inbound work
 - authorizing 56

- INCL parameter
 - on the SELECT initialization statement 110, 135
- INCLUDE 10
- INCR parameter
 - on the SELECT initialization statement 110, 135
- INIT parameter
 - on the SPART initialization statement 187
- initialization data for JES3
 - isolating 184
 - spool partition for, specifying 187
- initialization of JES3 7
- initialization parameters
 - CIBATCH 168
 - CIDEMAND 168
- initialization statement
 - used for security 86
- initialization statements 13
 - DYNALLOC 10
 - INCLUDE 10, 13
 - protecting JES resources 39
 - RJPWS 35
 - RJPWS use 13
- initialization stream 13
 - checker 15
 - checker, data sets 17
 - checker, sample JCL 17
 - creating 11
 - modifying 11
 - organizing 12
 - structure of 13
 - testing 15
 - using segmented 13
- initialization stream (JES3) 11
- statements
 - manage 11
- initialization stream checker 15
 - abnormal termination 18
 - storage requirements 18
- initiator
 - adding 131
 - starting 131
 - stopping 131
- input data for a job
 - spool partition for 186
- input service
 - control statement processing phase 95
 - job flow overview 96
- internal reader 95
- main selection for job execution 97
- overview 95
- procedure library 96
- reader phase 95
- input sources
 - defining nodes as local sources 70
- installation
 - GENERATE command 4
 - hardware configuration definition 4
 - plan for JES3 3
 - RECEIVE/APPLY/ACCEPT 4
- installation exit
 - IKJEFF53 52
- installation exit routine 46, 88
- IATUX03 166

installation exit routine (*continued*)

IATUX04 100, 166
IATUX05 100, 166
IATUX06 100, 166
IATUX07 100, 166
IATUX08 101, 166
IATUX09 166
IATUX10 166
IATUX11 101, 166
IATUX17 96
IATUX18 46, 84, 88, 89
IATUX19 88, 150, 155
IATUX20 155
IATUX21 155
IATUX22 155
IATUX23 155
IATUX24 126
IATUX25 105
IATUX26 99, 166
IATUX28 98, 166
IATUX29 189
IATUX30 46, 91
IATUX31 220
IATUX33 98, 166, 189
IATUX34 98, 139, 166
IATUX35 46, 84, 88, 308
IATUX36 308
IATUX37 308
IATUX38 308
IATUX39 155, 308
IATUX40 308
IATUX41 99, 166, 171
IATUX42 308, 310, 311
IATUX43 155, 308
IATUX44 98, 139, 166
IATUX45 155
IATUX46 99, 166, 167
IATUX48 199
IATUX49 99, 166, 167
IATUX58 46, 84
IATUX59 46, 84
IATUX61 106
IATUX62 105

that affect security 46, 88

installation security plan 39

integrity checking

data set 258

internal reader

converter/interpreter options list
for 170

displaying the status of a job 154

jobs read during input service 95

output service and 154

procedure library, default 97

starting 154

stopping 154

INTPMID parameter

on the STANDARDS initialization
statement 170

INTPROC parameter

on the STANDARDS initialization
statement 97

IORATE parameter

on the CLASS initialization
statement 123, 128

IRRDC00 utility

for JESSPOOL profiles 73

J

JCL statement

address space JCL statement limit

changing 172

displaying 174

selecting 172

default specifications for the

installation 169

effect on SWA (scheduler work

area) 170

installation default 169

job JCL statement limit 99, 170

changing 171

overriding with IATUX41 171

selecting 171

maximum number processed

in a C/I FSS address space 163,
172

in the JES3 global address
space 172

modify 98

modifying 166

parameter default 169

scan for dependent job control 126

JCT (job control table)

calculating the size 229

JCT Utility 237

JDS (job data set)

output service 139

JES (job entry subsystem)

how JES and RACF work
together 43

RACF support for 39

security 39

started procedures table 43

JES-managed

job class group 121

job selection algorithm 121

JES3

command processing 223

terminology

glossary 353

JES3 access to resource

defining 47

JES3 auxiliary address space 274

JES3 auxiliary task

description 275

JES3 buffer

change the size 272

determine the size 271

JES3 buffer pool

determine the size 272

in FSS address space 273

JES3 control statement 96

modify 98

JES3 device

assignable 244

defining 244

definition 244

eligible device type 244

number 244

reserving 244

JES3 global address space

abnormal termination, preventing

when not enough SWA space 171

definition 159

JES3 global address space (*continued*)

POSTSCAN DSP 165

private virtual storage constraint,
relieving 160, 163

JES3 initialization statement 315

3495 definition 315

esoteric names 315

use of hardware configuration

definition (HCD) program 315

JES3 installation

GENERATE command 4

RECEIVE/APPLY/ACCEPT 4

JES3 maintenance

APAR cover letter 28

general 27

overview 27

philosophy 27

restart information 28

JES3 monitoring facility (JMF) 26

JES3 networking

See networking

JES3 support 315

IBM Tape Library Dataserver 315

JES3 support overview 315

definition and usage 315

esoteric names 315

JES3 system log 206

JES3DRDS data set 7

JES3IN data set 7, 17

JES3JCT data set 7

JES3LIB data set 17

JES3OUT data set 7, 17

JES3SNAP data set 7

JESABEND data set 7, 17

JESINPUT class

activating 55

JESJOBS class

activating 53, 54

JESNEWS data set

protecting 75

JESSPOOL class

activating 72

JESSPOOL profiles

allowing users to create 75

job 97

accessing output through TSO 155

address space selection for C/I
service 167

aging 134

assigning to spool partition 188

authorizing NJE 61

batch

scheduling to a CI DSP 168

C/I service status, displaying 174

catalog access, ensuring 243

controlling through C/I service 166

critical

ensuring spool space for 184

defining maximum number 229

demand select

definition 160

JCL statement limit 172

scheduling to a CI DSP 168

use of SWA space 163

using system symbols in source

JCL 169

device dedication 109

- job (*continued*)
 - device pooling for a job-class group 109
 - device selection
 - limiting effect on main eligibility 105
 - DJC (dependent job control) 125, 165
 - nonstandard 126, 128
 - DJC completion option 127
 - dummy partition, requesting 188
 - execution queueing process 134
 - group
 - selection span 122
 - I/O rate mix 122, 128
 - JCL statement
 - parameter defaults set by installation 169
 - JCL statement limit 99, 170
 - changing 171
 - selecting 171
 - main eligibility 128
 - main selection for execution 97
 - management of a job 95
 - maximum number in processing 163
 - modifying DD statement information
 - with a installation exit routine 100
 - modifying job information with a installation exit routine 100
 - modifying step information with a installation exit routine 100
 - monitoring in a network 307
 - primary spool space allocation 194
 - priority barrier 123
 - procedure libraries and
 - moving 177
 - updating 177
 - processor dependencies, avoiding 243
 - processor selection
 - for C/I service 167
 - purging 156
 - RAS, improving 184
 - rescheduled 165
 - resources, allocating for 101
 - scheduling 111
 - controlling 133
 - scheduling a processor for C/I service 99
 - scheduling to a CI DSP 168
 - secondary spool space allocation 194
 - selecting an address space for C/I service 99
 - selection 111
 - algorithm 121
 - controlling 128
 - defining the environment 128
 - setup type
 - description 106
 - explicit setup 108
 - high-watermark setup 107
 - job setup 107
 - spool data set, effect of deleting 196
 - spool partition selection 186
 - spool space allocation 194
 - spool space, effect on amount needed 181
 - status in C/I service, displaying 174
- job (*continued*)
 - submitting to internal reader 154
 - track group allocation size, specifying 194
 - transmitting using JES3 networking 302
- job class 123
 - catalog access, using to ensure 243
 - defining 135
 - device pooling for a job-class group 109
 - execution resources, assigning 136
 - grouping 136
 - mains, eligible 97
 - priority barrier for group 137
- job class group
 - JES-managed 121
- job control table 229
- job data sets
 - protecting 72
- job entry network
 - See* networking
- JOB JCL statement
 - modifying with a installation exit routine 98
- job JCL statement limit 99, 170
 - changing 171
 - overriding with IATUX41 171
 - selecting 171
- job management
 - C/I (converter/interpreter) service 99
 - deadline scheduling
 - description 125
 - GMS (generalized main scheduling) 111
 - controlling job scheduling 133
 - default job selection 131
 - determine job eligibility 128
 - input service 95
- JCL statement
 - modify 98
- JES3 control statement 98
- job class
 - defining 135
 - grouping 136
- job selection and scheduling 111
 - algorithm for selection 121
 - controlling scheduling 133
 - controlling selection 128
 - defining the selection environment 128
 - DJC (dependent job control) 125
 - main eligibility 128
 - main selection
 - for job execution 97
- MDS (main device scheduler) 101
- output service 138
- overview 95
- procedure library 96
 - default 97
- purge 156
 - type 26 SMF record 156
- resource allocation
 - initializing MDS 108
 - MDS (main device scheduler) 101
 - overview 101
- job management (*continued*)
 - resource allocation (*continued*)
 - setup type 106
 - scheduler element
 - modifying the sequence 96
 - standard sequence 95
 - with `/*MAIN` JES3 control statement 96
 - with `/*PROCESS` JES3 control statement 96
- job mix 122, 128
- job names
 - controlling the use of 51
- job priority barrier 123
- job queue 121
- job scheduling
 - controlling 133
- job selection
 - and scheduling 111
 - controlling 128
 - main eligibility 128
 - defining the environment 128
- job selection algorithm
 - JES-managed 121
 - WLM-managed 123
- job setup 101
 - `/*MAIN` JES3 control statement 107
 - advantages and disadvantages 107
 - description 107
- JOB statement
 - verifying that userID data is present on 47
- job step
 - default region size 169
 - maximum execution time 169
- JOB MIX parameter
 - on the SELECT initialization statement 122
- jointly-managed device
 - definition 244
- JSAM (JES3 spool access method)
 - buffer pool size
 - BALJ data area 161
 - data buffer block 161
 - determine 272
- JSPAN parameter
 - on the GROUP initialization statement 123

K

keyboard 347

L

label 105

- for security 45

licensed documents x

limitations JES3 322

- function limitation 322

- device connectivity 322

- library device groups 322

- mixed JES3 non-managed

- devices 322

- mixed JES3-managed devices 322

- limitations JES3 (*continued*)
 - function limitation (*continued*)
 - scratch integrated cartridge loader request 322
 - storage group state 322
 - support unit 322
- local nodes
 - treating some network nodes as local nodes 70
- local processor start 25
- local start 25
- LOCATE request
 - for a cataloged data set 100
 - for a migrated data set 101
 - inhibiting JESMSG printing with a installation exit routine 101
- logical sender
 - how JES3 names them 304
- logical storage 132
- LOGON command
 - for an MCS console 205
- long profile names
 - in JESSPOOL class 73
- LookAt message retrieval tool xi
- LPA
 - dynamic LPA facility 31
 - SETPROG command 31
- LSTOR parameter
 - on the SELECT initialization statement 132
- LSTRR parameter
 - on the CLASS initialization statement 132
- LU (logical unit) 282

M

- MAGEL parameter
 - on the SELECT initialization statement 134
- MAGER parameter
 - on the SELECT initialization statement 134
- main
 - define 267
 - limited by device selection 105
 - selection for job execution 97
 - //*MAIN JES3 control statement 97
 - CLASS initialization statement 97
- MAINPROC initialization statement
 - and default job selection 131
 - PRTPAGE parameter 153, 273
 - SELECT parameter 129, 133
 - SPART parameter 188
 - to define processor as main 267
 - TRKGRPS parameter 194
 - USRPAGE parameter 273
- maintaining JES3
 - APAR cover letter 28
 - general 27
 - overview 27
 - philosophy 27
 - restart information 28
- maintenance 26

- MAXASST parameter
 - on the FSSDEF initialization statement 163, 172
 - on the STANDARDS initialization statement 172
- maximum number of jobs 229
- MAXJOBST parameter
 - on the STANDARDS initialization statement 171
- MAXLINE parameter
 - on the NJERMT initialization statement 305
- MCS (multiple console support)
 - alternate console 205
 - authority level 205
 - backup console service 205
 - console
 - defining 203
 - guidelines for defining 203
 - log facility 205
 - MPF (message processing facility) 205
 - operator action message 205
- MCS console 203
- MDEPTH parameter
 - on the CLASS initialization statement 123, 135
- MDS (main device scheduler)
 - allocation 104
 - breakdown 106
 - compared with MVS system allocation 102
 - conditional step execution consideration 103
 - description of processing 102
 - device allocation
 - SETNAME initialization statement 110
 - initialization 108
 - listening for WLM changes 103
 - operator control 111
 - overview 101
 - preparation 101
 - SDEPTH parameter effect 134
 - system select 104
 - volume fetch 103
 - volume verification 105
 - WLM interaction 102
- message
 - action 205
 - example of message suppression 221
 - nonaction
 - suppressing display 220
 - RETAIN and KEEP 106
 - specifying for display suppression 220
 - suppressing 220
 - example 221
- message class
 - for JES3 networking 307
 - installation default 169
- message processing facility 220
- message retrieval tool, LookAt xi
- migrating
 - JCTs 237
 - using JCT Utility 242

- migration
 - defining JES as a started procedure with the trusted attribute 43
 - restructured RACF database and JESSPOOL profiles 73
- MLIMIT parameter
 - on the CLASS initialization statement 123, 136
- MODE parameter
 - on the DEVICE initialization statement 249
- modified functions 321
 - dynamic allocation 321
 - modify JES3 function 321
 - modify JES3 message 321
 - pre-execution setup 321
- MPF (message processing facility) 220
- MVS
 - resource monitoring facility (RMF) 26
 - system monitoring facility (SMF) 26
- MVS assign/unassign facility 244
- MVS configuration program
 - JES3 device number 244
- MVS converter/interpreter
 - options list for 169
- MVS data set
 - integrity 259
- MVS internal reader 153
- MVS system allocation
 - compared with MDS 102
 - description of processing 102
- MVS UNITNAME 318
 - definition for library device group 318
- MVS workload manager
 - WLM 101
- MVS/BDT 294

N

- NAME parameter
 - on the SELECT initialization statement 133
- NAMES parameter
 - on the SETNAME initialization statement 110, 111
- NAU (network addressable unit) 282
- NCP (network control program)
 - SNA (system network architecture) 294
- NETDATA output 302, 304
- NETHOLD parameter
 - HOME node 304
- network security
 - NJE network 56
- networking 293, 306
 - BSC communication line
 - defining 301
 - BSC protocol 293
 - buffer size
 - defining 300
 - example of defining 300
 - changing node definition
 - changing BDT initialization stream 306

- networking (*continued*)
 - changing node definition (*continued*)
 - changing JES3 initialization stream 306
 - continued data transfer 306
 - synchronizing 306
 - communications path
 - defining 298
 - example 298
 - defining networking protocol 294
 - directing SNA transaction 295
 - home node 295
 - defining 296
 - installation exit routines for 307
 - job
 - deciding how to transmit 302
 - interleaving transmission 303
 - logical sender
 - how JES3 names them 304
 - message class
 - defining 307
 - monitor with an installation exit routine 307
 - monitoring files sent using TSO/E TRANSMIT or CMS
 - SENDFILE 310
 - MVS/BDT work queue
 - hold-for queue 139
 - networking request 139
 - node
 - BSC (binary synchronous communication) 295
 - defining 295
 - definition 295
 - SNA (systems network architecture) 295
 - password
 - specifying 300
 - protocol 294
 - remote node 295
 - adjacent 295
 - defining 297
 - directly-connected 295
 - indirectly-connected 295
 - nonadjacent 295
 - required initialization statement 295
 - rerouting
 - job 307
 - SYSOUT data set 307
 - SNA protocol 293
 - spool file
 - deleting 311
 - networking job
 - submitting networking jobs to JES3 154
 - networking profiles
 - description 57
 - understanding 57
 - networking protocol
 - BSC (binary synchronous communication) 293
 - SNA (systems network architecture) 293
 - NJE
 - authorizing jobs 61
 - authorizing SYSOUT 63
 - validating SYSOUT 65
 - NJE (network job entry)
 - defining a network 293
 - NJE header
 - and security tokens 49
 - NJE jobs 61
 - where verified 48
 - NJE reader
 - output service 154
 - NJE security 56
 - NJE SYSOUT 63
 - NJERMT initialization statement
 - MAXLINE parameter 305
 - NETHOLD parameter 302
 - parameter requirements for node type 295
 - specifying a password 300
 - STREAM parameter 305
 - to define a networking communication line 302
 - NJEUSERID suboperand
 - SETROPTS command 68
 - nodes
 - treating some network nodes as local nodes 70
 - NODES class
 - activating 70
 - SETROPTS RACLIST processing 70
 - NODES profiles
 - description 57
 - understanding 57
 - non-JES output writing routine
 - See* external writer
 - Notices 349
 - number of jobs 229
- O**
- operational considerations 320
 - processing restrictions and limitations 320
 - operator
 - command authorization 89
 - operator command
 - accepted from BSC RJP work station 280
 - entering 204
 - for C/I service 174
 - for tuning a spool data set 198
 - input processing 204
 - monitoring in a network 307
 - to balance the spool work load 197
 - to communicate with a JES3 function 204
 - to display spool status 197
 - to modify a spool partition 198
 - TSO/E command 91
 - operators
 - assigning to RACF groups 44
 - defining to RACF 44
 - OPTIONS initialization statement
 - JOBNO parameter 229
 - SE parameter 229
 - OSE (output scheduling element)
 - content's source 139
 - modifying information with a installation exit routine 150
 - output service 139
 - OSE (output scheduling element) (*continued*)
 - override content's sequence 140
 - scheduling to a writer 151
 - OSS (output scheduling summary entry) 150
 - outbound work
 - authorizing 70
 - using security labels 72
 - output 155
 - controlling where output can be processed 79
 - providing with a TAT (track allocation table) 188
 - spool partition, specifying 188
 - spool partitions and 188
 - OUTPUT command (TSO/E)
 - operands allowed by JESSPOOL profiles 74
 - output data set
 - deleting 199
 - OUTPUT JCL statement
 - effect on output service 139
 - OSE information and 139
 - override sequence
 - with "direct" statement 141
 - output separator routine
 - writing 344
 - output separator routine for an external writer
 - block character routine by IBM 345
 - functions 345
 - invoking 345
 - module name IEFSD095 345
 - parameter list 347
 - description 342
 - IBM-supplied output separator routine 342
 - note on protection key 345
 - output from the routine 344
 - parameter list 343
 - requirements for writing 344
 - return codes 344
 - output service 154
 - accessing output through TSO 155
 - installation exit 155
 - internal reader 154
 - output parameters 151
 - output writer
 - available-devices queue 151
 - waiting for work queue 151
 - overrides 151
 - overview 138
 - queueing output 139
 - hold-for queue 139
 - output service hold queue 139
 - output service writer queue 139
 - scheduling output 151
 - determining which output parameters apply 151
 - writing output 152
 - output writer 151
 - as a functional subsystem 152
 - available-devices queue 151
 - dynamic writer 153
 - external writer 154

- output writer (*continued*)
 - FSS (functional subsystem application) 152
 - getting more work done 275
 - hot writer 153
 - MVS internal reader 153
 - print writer 153
 - punch writer 153
 - starting 153
 - stopping 153
 - support routine 152
 - timeout value 153
 - waiting for work queue 151
- output writer FSS
 - canceling 248
 - changing the definition 248
 - defining 247
 - defining maximum number of printers 248
- FSA (functional subsystem application) 152
 - installation exit 156
- letting JES3 define 247
- performance consideration 153
- restarting 248
- running a printer under 246
- running as dynamic writer 153
- spool data storage 153
- spool-access routine 153
- start procedure 247
- starting 248
- support routine 152
- output writing routine 336
 - coding conventions 339
 - description 336, 338
 - IBM-supplied routine 336
 - programming considerations 339
- OUTSERV initialization statement
 - OSE information and 140
 - WC parameter 151
 - WS parameter 151
- OVRFLL parameter
 - on the SPART initialization statement 186

P

- PAGES parameter
 - on the BUFFER initialization statement 272
- PARMID parameter
 - on the CIPARM initialization statement 169
- partition 184
- partitioned data set 13
 - alternate stream member 12
 - editing 32
 - JES3 disk reader facility 10
 - use on INCLUDE 13
 - use with initialization stream checker 17
- password
 - for JES3 networking 300
 - NJE job 71

- PASSWORD parameter
 - on JOB statement in JCL
 - consideration for inbound NJE jobs 59
- PEP (partitioned emulation programming) extension
 - SNA (system network architecture) 294
- performance
 - C/I FSS address space impact on MVS 162
 - effect on POSTSCAN DSP 165
 - improve
 - output writing capability 275
 - improving
 - by identifying a resident data set 242
 - by using a spool partition 184
 - by using spool partition 185
 - dynamic allocation 264
 - spool 188
 - output writer FSS considerations 153
 - overview 26
 - spool data sets and 182
- performance factors 26
- performance integrity
 - PROCLIB 11
- performance JES3
 - improve
 - fix PBUF page 274
- planning
 - JES system programmer 39
- planning for JES3
 - classifying jobs 3
 - configuration 3
 - I/O device layout 3
 - overview 3
- PM parameter
 - on the DEVICE initialization statement 249
- PNAME parameter
 - on the FSSDEF initialization statement 162
- POOLNAMS parameter
 - on the SETNAME initialization statement 111
- POSTSCAN DSP
 - effect on performance 165
 - for jobs processed in a C/I FSS address space 100
 - inquiring 174
 - limiting effect on job processing 163
 - number, specifying the maximum 163, 165
 - status, displaying 174
- print writer 153
- printer 246
 - 3800 model 3 249
 - maximum for an output writer FSS 248
 - output writer FSS, running under 246
 - processing mode 250
 - PSF/MVS 250
 - SNA-attached 250
- printer security 80
- priority barrier 123

- procedure library 176
 - access by C/I service, ensuring 176
 - assignment for a job 96
 - C/I service access, ensuring 176
 - default 176
 - internal reader 97
 - started task 97
 - TSO/E LOGON job 97
 - definition 176
 - ensuring access by C/I service 161
 - how to manage 176
 - IBM-supplied 176
 - input service 96
 - job assignment 96
 - managing 176
 - members, specifying 176
 - moving to another volume 177
 - preventing an update 178
 - standard procedure library 176
 - status, displaying 178
 - updating 177
 - preventing 178
- process mode
 - ALTPM keyword 246
 - default 246
 - defining 245
- processing order
 - commands 223
- processor 267
 - competition for spool data, limiting 184
 - logical storage, defining 132
 - message suppression, controlling from a local processor 221
 - sharing a system catalog 243
 - spool data, limiting competition for 184
 - uninitialized
 - identifying 257
- procid (procedure library id) 176
- PROCLIB
 - performance integrity 11
- profile
 - RACF sample 40
- propagation
 - across a network 51
 - across nodes in an NJE network 69
 - definition 51
- PROPCNTL class
 - activating 48
- protected data buffer 153
- protection
 - command 81
 - console access 88
 - NJE job password 71
- protocol
 - converting networking protocol 294
 - definition 293
- PRTPAGE parameter
 - on the MAINPROC initialization statement 153, 273
- PS (presentation service) 285
 - data compaction 285
 - data compression 285
 - string control bite 285
- pseudocommand 204

PSTCNT parameter
on the FSSDEF initialization
statement 165
PU (physical unit) 282
punch writer 153

R

RAB (record allocation block)
storage requirements for 161
RACF (Resource Access Control Facility)
command authority 81
define JES3 access to a resource 47
how security works 40
sample profile 40
security class 40
security label 40, 45
universal access authority 44
RACF database
restructuring for JESSPOOL
profiles 73
RACFVARS class
&RACLNDE profile 70
activating 70
local nodes 70
SETROPTS RACLIST processing 70
RAS (reliability, availability, and
serviceability)
improving for spool data 184
record allocation block 161
recovery
improving through spool
partition 185
reliability, availability, and
serviceability 184
remote job processing 279
overview 279
SNA RJP 281
remote workstation job
and input service 95
remote workstations
protecting 76
RESDSN initialization statement 242
resident job queue 139
resident volume allocation table 103
resource
console
management 204
data set
management 258
define JES3 access 47
defining 227
effect on resource
management 257
dynamic allocation 263
manage 267
managing 227
processor 267
system log 206
volume 256
management 262
writer output multitask facility 275
resource allocation 104
MDS (main device scheduler) 101
initialization 108
overview 101
setup type 106

resource breakdown 106
resource management
effect of resource definition 257
resource monitoring facility (RMF) 26
RESQUEUE (resident job queue)
output service and 139
restart
effect on networking job 306
job
effect of restart 306
networking job
effect of restart 306
restart
effect 306
warmstart 306
restart characteristics
global processor 25
restart information 28
restricted device 244
restructuring the RACF database
to allow long profile names 73
reverse MAC (mandatory access
checking)
for outbound work 72
RJE consoles
protecting 76
RJP consoles
protecting 76
RSCS node
control of inbound jobs or data 56
RU (request unit) 284
chain 286
conversion 286
running
JCT Utility 238
RUSER qualifier
on NODES profiles 57

S

SAF (system authorization facility) 40
SAGEL parameter
on the SELECT initialization
statement 110, 134
SAGER parameter
on the SELECT initialization
statement 110, 134
SAPI
access to spool data sets 73
SBAR parameter
on the SELECT initialization
statement 110, 133
SCB (string control bite) 285
scheduler control block
creation 99
modifying before written to SWA 99
SWA (scheduler work area) 170
scheduler element
modifying the sequence 96
standard sequence 95
with // *MAIN JES3 control
statement 96
with // *PROCESS JES3 control
statement 96
Scheduling Environments 102

SDEPTH parameter
on the CLASS initialization
statement 108, 135
on the SELECT initialization
statement 110, 134
SDSF (System Display and Search
Facility)
effect of JESSPOOL profiles on 74
effect of WRITER profiles on 80
SECLJ qualifier
on NODES profiles 58
SECLS qualifier
on NODES profiles 58
security
benefit 39
class 40
command 89
considerations with BSC RJP 279
installation exit 46
JES3 installation exit 46, 88
label 40, 45
protection
command 81
NJE job password 71
RJP console 88
RACF interface 40
SAF (system authorization
facility) 40
sample RACF profile 40
universal access authority 44
using an initialization statement 86
security labels
using to control outbound work 72
segmented initialization stream 13
using 13
SELECT initialization statement
CHOICE parameter 122, 123, 134
INCL parameter 110, 135
INCR parameter 110, 135
JOBMIX parameter 122
LSTOR parameter 132
MAGEL parameter 134
MAGER parameter 134
NAME parameter 133
SAGEL parameter 110, 134
SAGER parameter 110, 134
SBAR parameter 110, 133
SDEPTH parameter 110, 134
SELECT parameter
on the MAINPROC initialization
statement 133
sending security information
across nodes in an NJE network 69
separator routine
See also output separator routine for
an external writer
parameter list 343
server mode
using to free spool space 199
SETACC initialization statement
description 110
SETNAME initialization statement
NAMES parameter 110, 111
POOLNAMS parameter 111
SETNAME statement 316, 319
example of configuration 319
library device 319

- SETNAME statement (*continued*)
 - library-specific device 316
 - library-specific name 316
 - required library device group 316
 - sysplex-wide device 316
 - sysplex-wide library name 316
- SETPARAM initialization statement
 - ADDRSORT parameter 111, 262
 - ALLOCATE parameter 104
 - DAFETCH parameter 103
 - FETCH parameter 103
 - TAFETCH parameter 103
- SETPROG command
 - dynamic LPA facility 31
- SETRES initialization statement
 - description 111
 - to keep a volume mounted 256
- SETROPTS command
 - EARLYVERIFY operand 47
- SETUNIT table
 - entry order 111
- setup
 - JCT Utility 239
- SETUP parameter
 - on the STANDARDS initialization statement 100
- SETVOL (resident volume allocation table)
 - content 103
 - creation of entry 103
- shared device
 - defining 245
 - definition 244
- shortcut keys 347
- SMS consideration 161
- SNA (system network architecture)
 - definition 294
 - supporting software 294
- SNA (systems network architecture) 293, 306
- SNA protocol 295, 296
- SNA RJP (systems network architecture remote job processing)
 - APPL (application layer) 281
 - basic exchange support 289
 - initialization consideration 290
 - communications layer 281
 - concept 281
 - console
 - defining 204
 - data compaction 285
 - data compression 285
 - DFC (data flow control) 282, 286
 - exchange support 290
 - initialization consideration 290
 - FM (function management layer) 281
 - FMPS (function management presentation services) 284
 - initialization consideration 289
 - LU (logical unit) 282
 - NAU (network addressable unit) 282
 - overview 281
 - PS (presentation service) 285
 - PU (physical unit) 282
 - RU (request unit) 284
 - TS (transmission subsystem layer) 281
- SNA RJP (systems network architecture remote job processing) (*continued*)
 - VTAM interface 287
- SNA RJP Devices
 - maximum record lengths 289
- SNA/RJP 33
 - adding a work station 33
 - adding devices 33
- SPART initialization statement 185
 - GRPSZ parameter 192
 - impact on checkpoint data set size 228
 - INIT parameter 187
 - OVRFL parameter 186
 - SPLIM parameter 186, 198
- SPART parameter
 - on the CLASS initialization statement 188
 - on the MAINPROC initialization statement 188
 - on the SYSOUT initialization statement 188
 - override order 188
- SPLIM parameter
 - on the BUFFER initialization statement 186, 198
 - on the SPART initialization statement 186, 198
- spool
 - protecting
 - dumped jobs 76
 - job data sets 72
 - restored jobs 76
 - RJE consoles 76
 - SYSIN data sets 72
 - SYSOUT data sets 72
 - system data sets 50
 - spool data set
 - access time, minimizing 184
 - adding 196
 - allocating 182
 - advantages to dynamic allocation 182
 - how many 181
 - commands for tuning 198
 - data types, isolating 185
 - defining 181
 - delete a networking file 311
 - deleting 196
 - devices to use 182
 - formatting 182
 - during JES3 initialization 182
 - using IEBCDG 183
 - impact on checkpoint data set size 228
 - managing spool space 185
 - moving to another DASD volume 197
 - number of, maximum 182
 - operator commands for tuning 198
 - overview of defining and managing 181
 - partition 184
 - performance problems, avoiding 182
 - performance, improving 184
 - quantity 182
 - reformatting 184
- spool data set (*continued*)
 - relationship to number of processors 181
 - replacing 196
 - specifying spool partition as a member 187
 - spool space 192
 - track group size, changing 193
 - work load and 181
- spool data set replace 23
- spool partition
 - advantage 184
 - balancing the work load 197
 - command
 - for modifying 198
 - for monitoring 197
 - default partition 185
 - content 186
 - member 187
 - overflow attribute 186
 - space constraint 187
 - defining 185
 - description 184
 - dummy 187
 - and jobs requesting 188
 - example 190
 - for initialization data 187
 - how to request 189
 - JES3 actions when low on space 186
 - marginal space condition 198
 - minimal space condition 198
 - modifying with an operator command 198
 - monitoring with an operator command 197
 - overflow
 - circular, avoiding 187
 - defining 186
 - example 186
 - overrides for job assignment 188
 - overriding user request with an installation exit routine 189
 - performance consideration 184, 185
 - RAS consideration 185
 - requesting 189
 - space constraint 186
 - specifying a spool data set as a member 187
 - status, displaying 197
 - to control spool space allocation 195
 - tuning 197
 - user request 189
- spool space
 - allocating for a job 194
 - allocation unit 192
 - amount required by average job 195
 - capacity, increasing 196
 - freeing 199
 - managing 196
 - marginal space condition 198
 - minimal space condition 198
 - monitoring usage 196
 - primary allocation 194
 - releasing 199
 - secondary allocation 194, 195
 - shortages, overcoming 196
 - spool partitioning and 195

- spool space (*continued*)
 - spool record 192
 - track group
 - allocation size 194
 - defining 192
 - definition 192
 - override order 195
 - physical track size and 193
 - size, determining 192
 - tailoring example 193
 - tuning 196
 - unit of allocation 192
 - spool1 data set 7
 - spoolnn data set 7
 - SSCP (system services control point) 282
 - staging area
 - allocate an auxiliary address space 274
 - STANDARDS initialization statement and device allocation 111
 - CI DSPs, defining the maximum number 163
 - CICNT parameter 163
 - INTPMID parameter 170
 - INTPROC parameter 97
 - MAXASST parameter 172
 - MAXJOBST parameter 171
 - POSTSCAN DSPs, defining the maximum number 163
 - procedure library specification, default 97
 - PSTCNT parameter 165
 - SETUP parameter 100
 - STCPMID parameter 170
 - STCPROC parameter 97
 - TSOPMID parameter 170
 - TSOPROC parameter 97
 - start JES3 19
 - START parameter
 - on the FSSDEF initialization statement 162
 - start procedure
 - for C/I FSS address space 162
 - changing 162
 - default procedure JCL statement 162
 - started procedures table
 - JES (job entry subsystem) entry 43
 - started task
 - converter/interpreter options list for 170
 - procedure library, default 97
 - submitting jobs to JES3 154
 - STCPMID parameter
 - on the STANDARDS initialization statement 170
 - STCPROC parameter
 - on the STANDARDS initialization statement 97
 - STEPLIB data set 7, 17
 - STG1CODE data set 17
 - storage 160, 271
 - CSA (common service area)
 - reduce the amount used by JES3 274
 - JES3 buffer
 - change the size 272
 - storage (*continued*)
 - JES3 buffer (*continued*)
 - determine the size 271
 - JES3 buffer pool
 - determine the size 272
 - logical
 - defining for a processor 132
 - private virtual
 - in a C/I FSS address space 164
 - relieving constraint 164
 - reduce page fix/free for PBUF (protected buffer pool) 274
 - tune the buffer allocation 274
 - unprotected USAM buffer
 - determine the number 273
 - USAM protected buffer pool
 - determine the size 273
 - STREAM parameter
 - on the NJERMT initialization statement 305
 - STT (single track table)
 - redefining 196
 - subgeneric group 254
 - subgeneric splits 16
 - SUBMIT command
 - controlling who can submit jobs by job name 52
 - IKJEFF53 installation exit 52
 - submitter validation 65
 - submitting jobs
 - allowing another user to submit jobs for you 55
 - controlling who can submit jobs by job name 52
 - surrogate users 55
 - SURROGAT class
 - setting up surrogate users on MVS 55
 - surrogate job submission 55
 - across NJE networks 59
 - surrogate users on MVS 55
 - SWA (scheduler work area)
 - content 170
 - managing space 170
 - preventing a job from dominating 171
 - SYS1.PARMLIB 4
 - SYS1.PROCLIB
 - cataloged start procedure (JES3) 7
 - SYSABEND data set 7
 - SYSIN data sets
 - protecting 72, 73
 - SYSOUT
 - authorizing NJE 63
 - application programming interface
 - access to spool data sets 73
 - class
 - spool partition, specifying 188
 - class table
 - OSE information and 140
 - data set
 - monitoring in a network 307
 - providing with a TAT (track allocation table) 188
 - spool partitions and 188
 - SYSOUT data sets
 - protecting 72, 73
 - SYSOUT initialization statement
 - OSE information and 140
 - SPART parameter 188
 - TRKGRPS parameter 194
 - TYPE parameter 188
 - SYSOUT requests
 - &SUSER value 49
 - how verified 49
 - sysplex
 - MVS sysplex 270
 - running JES3 in sysplex 270
 - system allocation 102
 - system catalog
 - CVOL catalog 243
 - preventing a change 243
 - sharing 243
 - using 243
 - VSAM catalog 243
 - system log
 - content 206
 - DLOG option 208
 - system monitoring facility (SMF) 26
 - SYSTEM parameter
 - on the CLASS initialization statement 97
 - on the FSSDEF initialization statement 162
 - system select 104
 - systems network architecture 281
- ## T
- TAFETCH parameter
 - on the SETPARAM initialization statement 103
 - TDEPTH parameter
 - on the CLASS initialization statement 123, 135
 - test the initialization stream 15
 - testing
 - JCT Utility processing 242
 - timeout value
 - output writer 153
 - TLIMIT parameter
 - on the CLASS initialization statement 123, 135
 - token
 - submitter information propagated from trusted nodes 49
 - track group 194
 - TRACK initialization statement
 - impact on checkpoint data set size 228
 - translating
 - group IDs 66
 - security labels 66
 - userIDs 66
 - TRKGRPS parameter
 - on the CLASS initialization statement 194
 - on the MAINPROC initialization statement 194
 - on the SYSOUT initialization statement 194
 - override order 195

- TSO (Time Sharing Option)
 - command 91
 - LOGON job
 - converter/interpreter options list for 170
 - using to access output 155
- TSO/E (Time Sharing Option Extensions)
 - LOGON job
 - procedure library, default 97
- TSO/E CANCEL command
 - controlling who can cancel jobs by job name 53
- TSO/E OUTPUT command
 - operands allowed by JESSPOOL profiles 74
- TSO/E SUBMIT command
 - controlling who can submit jobs by job name 52
 - IKJEFF53 installation exit 52
- TSOPMID parameter
 - on the STANDARDS initialization statement 170
- TSOPROC parameter
 - on the STANDARDS initialization statement 97
- tune JES3
 - allocate a buffer 274
 - reduce page fix/free for PBUF (protected buffer pool) 274
- tuning JES3 182
 - balancing the spool partition work load 197
 - by using spool partitioning 184
 - data types, isolating 185
 - overview 26
 - spool data 185
 - spool data set size 188
- type 26 SMF record 156

U

- unchanged functions 321
 - drive sharing 321
 - JES3 sysplex 321
 - operator commands 321
 - support for 3495 321
- UNDEFINEDUSER operand
 - SETROPTS command 69
- unknown user token
 - for jobs submitted from a physical reader 49
- unprotected USAM buffer
 - determine the number 273
- USAM (user spool access method)
 - buffer
 - allocate an auxiliary address space 274
 - determine number 273
- USAM PBUF (protected buffer pool)
 - buffer pool size
 - determine 273
- USAM PBUF (protected data buffer)
 - output writer FSS and 153
 - reduce page fix/free 274
- USAM protected buffer pool
 - determine the size 273

- USAM unprotected buffer
 - size
 - determine 273
- user
 - forcing batch users to identify themselves to RACF 43
- user catalog
 - VSAM catalog 243
- userID
 - controlling propagation of 48
 - default userIDs 68
 - early verification of by JES 47
 - propagation for jobs that have no validated user ID 47
 - propagation, controlling 48
 - userID propagation
 - when jobs are submitted 47
- userIDs
 - security default 68
 - translating 66
- USERJ qualifier
 - on NODES profiles 57
- USERS qualifier
 - on NODES profiles 57
- using
 - JCT Utility 240
- USRPAGE parameter
 - on the MAINPROC initialization statement 273
- utility programs 11
 - IEBUPDTE 11

V

- validation
 - security 65
- virtual storage
 - common
 - requirements for a C/I FSS address space 161
 - private
 - constraint, relieving 160, 163
 - requirements for a C/I FSS address space 164
- volume
 - accessing from more than one processor 257
 - dynamically reconfiguring 255
 - fetch 103
 - keeping mounted 256
 - management 262
 - managing 256
 - mount attribute 256
 - mounting
 - handling by JES3 or MVS 262
 - multiple access 257
 - permanently resident 256
 - use attribute 256
- volume label
 - verifying with a installation exit routine 105
- volume verification 105
- VTAM (Virtual Telecommunications Access Method)
 - SNA (system network architecture) 294

- VTAM interface
 - with SNA RJP 287

W

- wait for work queue 151
- warm start 22
 - replace spool data set 23
- warm start with analysis 23
- WC parameter
 - on the OUTSERV initialization statement 151
- where NJE jobs are verified 48
- WLM 101
- WLM-managed
 - initiators 123
 - job selection algorithm 123
- Workload Manager
 - WLM 101
- writer 151
 - timeout value 153
- WRITER class
 - activating 80
- writer output multitask facility
 - description 275
 - disable 275
 - display status 276
 - DSI consideration 276
 - enable 275
 - on a multiprocessor 275
 - on a uniprocessor 275
 - restart consideration 275
 - turning off 275
 - turning on 275
 - use 275
- writer, external
 - access to spool data sets 73
- writing output 10, 152
 - ABDUMP task 10
 - JES3 task 10
- WS parameter
 - on the OUTSERV initialization statement 151

X

- XTYPE parameter
 - on the DEVICE initialization statement 256
- XUNIT parameter
 - and DEST parameter on the CONSOLE statement 108
 - on the DEVICE initialization statement 108
- XWTR cataloged procedure 332

Readers' Comments — We'd Like to Hear from You

z/OS
JES3 Initialization and Tuning Guide

Publication No. SA22-7549-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



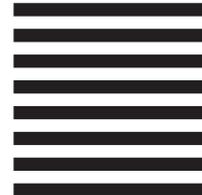
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5694-A01, 5655-G52

Printed in U.S.A.

SA22-7549-02

