z/OS

JES2 Initialization and Tuning Guide

Version 2 Release 1

SA32-0991-00

z/OS

JES2 Initialization and Tuning Guide

Version 2 Release 1

SA32-0991-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 419.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 5. Network job entry (NJE) 251

Chapter 6. Remote job entry (RJE) 325

Figures

Tables

About this document

This document is for installations running the JES2 element of z/OS (5650-ZOS). This document describes the general functions of JES2 and provides information describing JES2 initialization, JES2 processing, network job entry (NJE), remote job entry (RJE) for JES2, and factors affecting performance.

This document provides the information that you need to:

- Initialize JES2
- Tune JES2

z/OS JES2 Initialization and Tuning Reference describes JES2 initialization statements and their parameters and provides summary charts that highlight details of the statements.

Who should use this document

This document is intended for JES2 system programmers or for anyone responsible for initializing or tuning JES2. Installation procedures are release specific; therefore, they are provided in the Program Directory distributed with a specific product.

How to use this document

Use this document in conjunction with *z/OS JES2 Initialization and Tuning Reference*.

Most referenced publications are abbreviated throughout the text; their full titles appear in "Where to Find More Information," which follows. Additionally, be aware of how this document uses references to 3800 printers.

Where to Find More Information

This document references the following publications for further details about specific topics. Abbreviated forms of these titles are used throughout this document. The following table lists all abbreviated titles, full titles, and their order numbers that are not listed in the z/OS® Information Roadmap. See that document for all z/OS publications.

Short Title Used in This document	Title	Order Number
NJE Installation	*NJE Installation, Operation, and Use with JES2 and Other Systems*	GG22-9339
	ES/9000 and ES/3090 PR/SM Planning Guide	GA22-7123
	IBM 3704 and 3705 Communications Controllers	GC30-3008
	IBM 3725 Model 1 Communications Controller	GA33-0010
	System/370 Special Feature: Channel-to-Channel Adapter	GA22-6893

Short Title Used in This document	Title	Order Number
	OS/VS2 IBM 3540 Programmer's Reference	GC24-5111
	Advanced Function Presentation: Printer Information	G544-3290
DFP Advanced Services	*DFSMS/MVS DFSMSdfp Advanced Services*	SC26-4921
	z/OS DFSMS Macro Instructions for Data Sets	SC26-4913
	z/OS Communications Server: SNA Resource Definition Reference	*z/OS Communications Server: SNA Resource Definition Reference*
	z/OS Communications Server: SNA Programming	*z/OS Communications Server: SNA Programming*

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R1.0 JES2 Initialization and Tuning Guide
  SA32-0991-00
- The topic and page number that is related to your comment.
- The text of your comment.

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

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

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. JES2 initialization

JES2 initialization is the series of operations performed each time JES2 is started, in order to ready itself for processing.

Installation overview

The following steps should be followed to ensure correct JES2 operation on an MVS™ system:

1. Become thoroughly familiar with the System Modification Program/E(SMP/E) before attempting to install JES2. (See *SMP/E for z/OS User's Guide* for information concerning this procedure.)
2. Install JES2 using the System Modification Program/E (SMP/E) on an existing MVS system, as outlined in the Program Directory that is distributed with the component package. (The program directory and component package should be retained for future reference.)
3. Define the JES2 spool data sets (generally named SYS1.HASPACE) and the checkpoint data set(s) (generally named SYS1.JESCKPT1 and SYS1.JESCKPT2 when they are on DASD, or SYS1_JESCKPT1 and SYS1_JESCKPT2 when they are on a Coupling facility structure). (See Chapter 4, "Checkpoint data set definition and configuration," on page 191 for further information concerning the specifications of these data sets.)
4. Define the JES2 initialization statements. (See *z/OS JES2 Initialization and Tuning Reference* for full descriptions of these statements.)
5. Define a primary subsystem by specifying the PRIMARY keyword in the member IEFSSNxx in SYS1.PARMLIB. (See *z/OS MVS Initialization and Tuning Reference* for a description of subsystem definition.)
6. Define the cataloged JCL procedure for JES2. (See *z/OS MVS JCL User's Guide* for a description of cataloged procedures in general and *z/OS MVS Initialization and Tuning Reference* for a description of subsystem cataloged procedures in particular.)

Service considerations

The System Modifications Program/E (SMP/E) is required to install all IBM*-provided changes to the JES2 element of z/OS. It is highly recommended that any user modifications (both system and distribution library) be made using SMP/E.

JES2 initialization process overview

Each time you start a JES2 subsystem, MVS establishes JES2 as an authorized subsystem address space and gives control to JES2 to initialize itself through the following procedures:

- Begin the JES2 procedure through either an operator command (S JES2) or an IPL-time automatic process such as the COMMNDxx members of SYS1.PARMLIB. Then MVS:
  - locates the JES2 procedure in SYS1.PROCLIB.

  - allocates the libraries to which the DD statements in the JES2 procedure point, including those that the JES2 subsystem uses to locate the JES2 initialization statements.
  - gives control to the JES2 subsystem, which establishes the JES2 general ESTAE protection before calling a series of initialization routines (IRs), as follows, to process each phase of JES2 initialization.
- Process the start options specified and call installation Exit 0, allowing the installation to perform pre-initialization processing.
- Process the current status of the subsystem interface (SSI), establishing necessary control blocks and preparing for later SSI requests.
- Allocate various control blocks required during all JES2 processing or temporarily during initialization processing.
- Process the initialization statements specified through parmlib, console, and exit routine input, and call installation Exit 19 for each initialization statement.
- Process the initialization statements specified, verifying the validity of the parameters and calculated variable parameters. These post-processing functions include:
  - General processing and verifying the complete set of initialization statements and their parameters (specified and defaulted)
  - Calculating variable parameters based on other parameter values
  - Validating the parameters specified and the associated control blocks.
- Allocate the control blocks required to:
  - Define the devices specified through the initialization statements
  - Allow I/O activity for the defined devices
- Process the initialization and perform validation, as required for JES2 network job entry (NJE) activity.
- Process the initialization and perform validation, as required for JES2 remote job entry (RJE) activity.
- Validates the current JES2 direct access devices (DASD) that are defined through the initialization statements, analyzes the spool volumes and the checkpoint data sets (whether they reside on DASD volumes or Coupling facility structures), and performs any required related activities (such as checkpoint data set forwarding).
- Validates the multi-access spool (MAS) configuration specifications.
- Allocate common storage control blocks for command and message traffic, MVS initiator support, the subsystem interface (SSI), the JES2 trace facility, and other JES2-supported facilities.
- Connect the JES2 devices defined through the initialization statements to MVS device control blocks.
- Call miscellaneous MVS services (for example, to establish interfaces).
- Allocate the control blocks necessary to define those JES2 processors that are required immediately after initialization that cannot be dynamically added later.
- Generate the information string for the request subsystem (SSI) version call 54 routine.
- Complete JES2 initialization by building the required buffer pools and calling installation Exit 24 (post-initialization exit routines).

Modifying how JES2 performs initialization

The specific manner in which JES2 is initialized depends upon your choice of start options and initialization statements. You define how JES2 will perform initialization through the initialization options, which specify:

- Whether JES2 is to warm start or cold start.
- Whether JES2 should force the formatting of the spool volumes.
- The ddname for the DD statement in the JES2 cataloged procedure that defines the input data set(s) for the initialization statements.
- Whether JES2 should stop and give control to the operator for further statement input from the console after the parmlib input is exhausted.
- Which data set, as defined by the CKPT1= or CKPT2= parameter on the CKPTDEF statement, should be used to access JES2 warm-start information.
- Whether the checkpoint data set definitions should be reconfigured.
- Whether JES2 will print a log of the initialization statements specified and any related diagnostics to the data set referenced by the HASPLIST DD statement in the JES2 cataloged procedure.
- Whether JES2 will automatically start normal processing after initialization or wait for an operator command to start.
- Which start options will be overridden (by specifying the OPTSDEF initialization statement).

You define which JES2 functions and device defaults are to be overridden by specifying the initialization statements:

- Logical initiator characteristics
- Internal reader characteristics
- Local and remote device characteristics
- Default job and SYSOUT class characteristics
- NJE header and trailer buffers
- Multi-access spool (MAS) control statements
- Changes to certain JES2 default parameter values

You can control how JES2 schedules jobs by the way you specify these options and statements during JES2 initialization. Furthermore, you can respecify these options and statements to reflect changes in your system's configuration and workload any time JES2 is started.

Attention: Be aware, however, that some parameter changes require a JES2 cold start or all-member warm start and some parameters are ignored on a hot or warm start. These parameters are noted in the “Modification” heading for each parameter description in *z/OS JES2 Initialization and Tuning Reference*.

Carefully plan the initialization specification of each of these parameters to prevent having to revise them later through a warm start or cold start.

JES2 sample initialization data sets

IBM distributes the HASIxxxx members in the SYS1.SHASSAMP data set library that you can tailor to meet your installation's needs. To ensure that your SYS1.SHASSAMP is not overwritten, IBM suggests moving these data sets to an installation library before tailoring the HASIxxxx members. The following table lists these initialization data sets and their specific roles.

Table 1. JES2 Sample Initialization Data Sets

Member Name	Contents	Use
HASIPROC	Sample JCL procedure	Tailor, rename, and move to the production JCL procedure library for use as the JES2 subsystem JCL.
HASIPARM	Sample JES2 initialization parameter templates	Use to create production JES2 initialization parameters in other libraries.
HASIASM	Sample JCL procedure.	Use when assembling an IBM® JES2 source module or an installation JES2 exit routine (it saves the resulting object code). This member can be tailored, placed in a production procedure library, and used when the SMP/E system installation and maintenance tool is **not** managing JES2 code.
HASIBLD	Sample job stream that uses the HASIASM procedure.	Assembles IBM JES2 source-distributed modules (including sample exit routines) and link edits each of the production load modules. This member can be tailored, placed in another procedure library, and used when the SMP/E system installation and maintenance tool is **not** managing JES2 code.
HASISMPA	Sample job stream.	Tailor, then use to force the SMP/E system installation and maintenance tool to reassemble all modules for the JES2 release it manages. This two-step procedure might be necessary to assemble all modules at an installation with the same level of MVS system macros.

Initialization statements and parameters

This section explains how to specify the JES2 initialization statements and parameters and how JES2 performs initialization under different conditions.

Throughout this book, an initialization statement refers to a JES2 specification that can contain further parameter specifications. For example, PRINTDEF is a statement; LINECT, DBLBUFR, and UCS are parameters on that statement. Most initialization statements contain a set of parameters that relate to a single JES2 function. For example, the SPOOLDEF statement contains several parameters used to define the JES2 spool volume environment. Some statements, however, contain no parameters (for example, DEBUG: this statement is either specified as DEBUG=YES or DEBUG=NO).

Table 2 on page 5 provides an alphabetic list of the JES2 initialization statements and briefly describes the function of each. It is intended as a reference to help you decide which statements and parameters to use to initialize your JES2 member. *z/OS JES2 Initialization and Tuning Reference* describes each of these statements in detail and specifies their functions, formats, and default values.

Table 2. JES2 Initialization Statements

Initialization Statement	Function
APPL(avvvvvvv)	Defines an SNA NJE application to JES2.
BADTRACK	Specifies an address or range of addresses of defective spool volume tracks JES2 is not to use.
BUFDEF	Defines the local JES2 buffers to be created.
CKPTDEF	Defines the JES2 checkpoint data set(s) and the checkpointing mode.
CKPTSPACE	Defines how much additional space is available for expanding the JES2 checkpoint record.
COMPACT	Defines a compaction table for use in remote terminal communications.
CONDEF	Defines the JES2 console communication environment.
CONNECT	Specifies a static connection between the nodes identified.
DEBUG	Specifies whether debugging information is to be gathered by JES2 during its operation for use in testing.
DESTDEF	Defines how JES2 processing interprets and displays both job and SYSOUT destinations.
DESTID(jxxxxxxx)	Defines a destination name (mostly for end-user use as on a JCL statement) for a remote terminal, another NJE node or a local device.
D MODULE(jxxxxxxx)	Displays diagnostic information for specified JES2 assembly modules and installation exit assembly modules.
ESTBYTE	Specifies, in thousands of bytes, the default estimated output (SYSOUT) for a job at which the "BYTES EXCEEDED" message is issued, and the subsequent action taken.
ESTIME	Specifies, in minutes, the default elapsed estimated execution time for a job, the interval at which the "TIME EXCEEDED" message is issued, and whether the elapsed time job monitor feature is supported.
ESTLNCT	Specifies, in thousands of lines, the default estimated print line output for a job, the interval at which the "LINES EXCEEDED" message is issued, and the subsequent action taken.
ESTPAGE	Specifies the default estimated page output (in logical pages) for a job, the interval at which the "PAGES EXCEEDED" message is issued, and the subsequent action taken.
ESTPUN	Specifies, in number of cards, the default estimated punch card output for a job, the interval at which the "CARDS EXCEEDED" message is issued, and the subsequent action taken.
EXIT(nnn)	Associates the exit points defined in JES2 with installation exit routines.
FSS(acccccc)	Specifies the functional subsystem for printers that are supported by an FSS (for example Print Services Facility™).
INCLUDE	Allows new initialization data sets to be processed.
INIT(nnn)	Specifies the characteristics of a JES2 logical initiator.
INITDEF	Specifies the number of JES2 logical initiators to be defined.
INTRDR	Specifies the characteristics of all JES2 internal readers.

Table 2. JES2 Initialization Statements (continued)

Initialization Statement	Function
JOBCLASS(v)	Specifies the characteristics associated with job classes, started tasks, and time sharing users.
JOBDEF	Specifies the characteristics that are assigned to jobs that enter the JES2 member.
JOBPRTY(n)	Specifies the relationship between job scheduling priorities and job execution time.
LINE(nnnn)	Specifies the characteristics of one teleprocessing line or logical line (for SNA, or TCP/IP) to be used during remote job entry or network job entry.
L(nnnn).JT(m)	Specifies the characteristics for a job transmitter on an NJE line.
L(nnnn).ST(m)	Specifies the characteristics for a SYSOUT transmitter on a line defined for network job entry.
LOADMOD(jxxxxxxx)	Specifies the name of a load module of installation exit routines to be loaded.
LOGON(n)	Identifies JES2 as an application program to VTAM*.
MASDEF	Defines the JES2 multi-access spool configuration.
MEMBER(n)	Defines the members of a JES2 multi-access spool configuration.
NAME	Specifies the module or control section to be modified through subsequent VER and REP initialization statements.
NETACCT	Specifies a network account number and an associated local account number.
NETSRV(nnn)	Defines NJE over TCP/IP server address space.
NJEDEF	Defines the network job entry characteristics of this JES2 node.
NODE(nnnn)	Specifies the characteristics of the node to be defined.
OFF(n).JR	Specifies the characteristics of the offload job receiver associated with an individual offload device.
OFF(n).JT	Specifies the characteristics of the offload job transmitter associated with an individual offload device.
OFF(n).SR	Specifies the characteristics of the offload SYSOUT receiver associated with an individual offload device.
OFF(n).ST	Specifies the characteristics of the offload SYSOUT transmitter associated with an individual offload device.
OFFLOAD(n)	Specifies the characteristics of the logical offload device.
OPTSDEF	Defines the options that are currently in effect.
OUTCLASS(v)	Specifies the SYSOUT class characteristics for one or all output classes.
OUTDEF	Defines the job output characteristics of the JES2 member.
OUTPRTY(n)	Defines the association between the job output scheduling priorities and the quantity (records or pages) of output.
PCEDEF	Specifies the definition for various JES2 processes.
PRINTDEF	Defines the JES2 print environment.
PROCLIB	Ensures that data sets specified can be allocated.
PRT(nnnn)	Specifies the characteristics of a local printer.
PUNCHDEF	Defines the JES2 punch environment.

Table 2. JES2 Initialization Statements (continued)

Initialization Statement	Function
PUN(nn)	Specifies the characteristics of a local card punch.
R(nnnnn).PR(m)	Specifies the characteristics of a remote printer.
R(nnnnn).PU(m)	Specifies the characteristics of a remote punch.
R(nnnnn).RD(m)	Specifies the characteristics of a remote card reader.
RDR(nn)	Specifies the characteristics of a local card reader.
RECVOPTS(type)	Specifies the error rate below which the operator will not be involved in the recovery process.
REDIRect(vvvvvvvv)	Specifies where JES2 directs the response to certain display commands entered at a console.
REP	Specifies replacement patches for JES2 modules during initialization.
REQJOBID	Describes attributes to be assigned to Request Jobid address spaces.
RMT(nnnn)	Specifies the characteristics of a BSC or SNA remote terminal.
SMFDEF	Specifies the system management facilities (SMF) buffers to JES2.
SOCKET(vvvvvvvv)	Defines an IP address and port for NJE/TCP and the associated NJE node.
SPOOLDEF	Defines the JES2 spool environment.
SSI(nnn)	Specifies the characteristics associated with individual subsystem interface definitions.
SUBTDEF	Specifies the number of general purpose subtasks you want JES2 to attach during initialization.
TPDEF	Defines the JES2 teleprocessing environment.
TRACE(n)	Specifies whether a specific trace ID(s) is to be started.
TRACEDEF	Defines the JES2 trace environment.
VER	Specifies verification of replacement patches for JES2 modules during initialization.

Using compaction on SNA workstations

Compaction, which is available only for SNA NJE and RJE workstations, is used to increase data transmission efficiency by compacting master character pairs into a single 8-bit representation (4 bits per character).

Data transmissions to workstations and remote devices can be compacted if you specified compaction. The COMPACT initialization statement defines the characters to be compacted (master characters) and the characters to be recognized by the compaction routines but not compacted (nonmaster characters).

In the example below, the compaction table NEWONES specifies 15 master characters (the numbers 0-9 and the vowels A, E, I, O, and U) and 16 nonmaster characters. The NUMBER= parameter specifies a number by which both the installation and JES2 can refer to this table. If you do not specify a table name, NAME= defaults to the NUMBER= parameter value; therefore, you must specify the NUMBER= parameter.

```
COMPACT NAME=NEWONES,CHARS=(15,0,1,2,3,4,5,6,7,8,9,A,E,I,O,U,
        40,B,C,D,F,G,H,J,K,L,M,N,61,$,T,S),NUMBER=1
```

The COMPACT=YES parameter specification on the RMT(nnnn) statement, the $ADD RMT(nnnn) command, or the $T RMT(nnnn) operator command allows you to specify if an NJE or RJE workstation is to accept compacted data. The COMPACT parameter on the R(nnnnn).PR(m) and R(nnnnn).PU(m) statements is set equal to a valid compaction table identifier (which is specified on the COMPACT initialization statement) if compaction is required for the remote device. Also, the device must specify CMPCT=YES or JES2 does not compact the data.

The /*OUTPUT JES2 control statement or the JCL OUTPUT statement can be used to override the R(nnnnn).PR(m) and R(nnnnn).PU(m) statements. However, neither the /*OUTPUT statement nor the JCL OUTPUT statement has any effect on compaction for node-to-node transmissions (SNA NJE) because such compaction is done on a session basis. For further information, see *z/OS MVS JCL Reference*.

How compaction works

In the compaction algorithm, 3 to 16 user-specified master characters are assigned 4-bit representations to replace their conventional EBCDIC values. When master characters occur in pairs, in any combination, each pair can be compacted and represented by a single 8-bit value.

Because compaction allows the assignment of 3 to 16 master characters, as many as 244 nonmaster characters can be assigned. Nonmaster characters are not compacted, but when they occur in pairs with master characters or singly they do not require breaking the compaction string. When characters other than master or nonmaster are encountered, the compaction string is broken and the characters are transmitted in their standard representation.

To determine the best set of master and nonmaster characters, you must examine the nature of your installation's transmitted data. For example, if most data is numeric, you could compact the characters 0-9, +, -,., and =. You would still have 46 nonmaster characters in which to specify other frequently used special characters and alphabetics.

When establishing a set of master characters for compaction, remember that master characters are not compacted singly but in various combinations of pairs.

Defining JES2 to the cross-system coupling facility (XCF)

JES2 uses the JES common coupling services (JES XCF) for communicating JES2 member status and other data among the JES2 XCF group members in a multi-access spool (MAS) configuration. Optionally, you can modify this communication mechanism through the JES common coupling services. See *z/OS MVS Programming: JES Common Coupling Services* for a description of the JES common coupling services (JES XCF macros and exits).

All members of a MAS need to be contained within the same sysplex. If your installation is running a single MVS system, you can specify PLEXCFG=XCFLOCAL on the IEASYSxx parmlib member to avoid the need for formatting a sysplex couple data set. Also, you need to ensure that there is enough space on the sysplex couple data set to accommodate the JES2 XCF group and its member names.

How JES2 determines the JES2 XCF group name

To allow the JES2 XCF group members to know about each other, either:

- Accept the JES2 defaults

or

- Define the members to JES2 through the initialization data set.

XCF needs to know about the groups that use its services. It knows about these groups through their XCF group name. The JES2 XCF group name allows a JES2 MAS to be known to XCF. Define each member to JES2 by assigning each MAS a group name that is unique within its sysplex. This group name defaults to the JES2 local node name, which is defined by the NAME= parameter on the NODE(nnnn) initialization statement. (Note that if you use the JES2 network job entry (NJE) facility, this NAME= parameter matches the OWNNODE= parameter on the NJEDEF statement.)

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. For example, the sysplex specified below is named POK:

```
SYSPLEX=POK
```

This is the IBM recommended approach. It reduces system complexity by allowing the sysplex name, the node name, and the JES2 XCF group to share the same name.

You can assign a unique name through the XCFGRPNM= parameter on the MASDEF initialization statement if your:

- NAME= parameter on the NODE(nnnn) statement conflicts with an existing XCF group name.
- Current node name does not meet naming restrictions for a JES2 XCF group name
- Sysplex contains more than one MAS.

  While IBM does not recommend more than one JES2 MAS in a sysplex, your installation can choose this approach. To avoid the ABEND that results if multiple MAS names default to an XCF group name of N1, all but one MAS member must specify a nodename for the OWNNODE= parameter on their NJEDEF statements.

JES2 determines the XCF group name for your JES2 MAS in the following order:

1. XCFGRPNM= keyword on the MASDEF initialization statement, if specified:

   ```
   MASDEF ...XCFGRPNM=POK
   ```

2. The node name defined by the NAME= parameter on the NODE(nnnn) initialization statement, if specified:

   ```
   NODE(nnnn) NAME=POK
   ```

3. The node name defined by the nnnn subscript of the NODE(nnnn) initialization statement, if specified:

   ```
   NODE(POK)
   ```

4. N1, the JES2-assigned default node name if a node name or XCFGRPNM= is not specified in the initialization data set.

XCF group name restrictions

When coding the XCF group name on the XCFGRPNM parameter of the MASDEF statement, the name should follow the XCF naming conventions.

If the XCF group name is not coded on the MASDEF statement, the nodename on the NODE(xxxxxxxx) initialization statement should follow the XCF naming conventions.

Note: To avoid using the names IBM uses for its XCF groups, do not begin group names with the letters A through I or the character string SYS. Also, do not use the name UNDESIG, which is reserved for use by the system programmer in your installation.

How JES2 determines the JES2 XCF group member name

Within a group, each JES2 member is assigned an XCF member name. The member name that identifies a JES2 member to the JES2 XCF group is composed of the NAME= parameter on the NODE(nnnn) statement, the $ symbol, and the OWNMEMB= parameter on the MASDEF initialization statement. For example, using the following initialization statements:

```
NODE(1)  NAME=POKX
MEMBER(1)  NAME=HAS1
MASDEF  OWNMEMB=HAS1
```

the member name for MVS is **POKX$HAS1**.

For information about how to specify NAME= on the MEMBER(n) initialization statement and OWNMEMB= on the MASDEF initialization statement, see "Starting the multi-access SPOOL configuration" on page 60.

Because the member name includes a $, you might want to avoid using it on the OWNMEMB= parameter and NAME= parameter of the NODE(nnnn) initialization statement to eliminate potential confusion.

The JES2 XCF member names must be unique within the JES2 XCF group.

Defining JES2 structures in a CFRM policy

To specify that a checkpoint data set resides on a Coupling facility structure, you must have taken the following steps when planning your JES2 configuration.

1. Format a Coupling facility resource management (CFRM) couple data set for use at your installation. See *z/OS MVS Setting Up a Sysplex* for specific instructions on defining the coupling facility resource management administrative policies. These policies each provide a number of with structures that can contain checkpoint data sets. Since JES2 supports system managed processes, you should also review the Coupling facilitiessection "System-Managed Processes Considerations" in *z/OS MVS Setting Up a Sysplex* to enable system managed functions. We recommend that structure full monitoring be suppressed by setting a FULLTHRESHOLD value of 0 for the coupling facility structure where a checkpoint is to reside.
2. Ensure that you have allocated enough storage for the:
   - Coupling facilityStructure
   - JES2 checkpoint data set

   (To determine these sizes, see "Determining the size of your checkpoint data set" on page 197.)
3. Before starting the JES2 address space, specify the Coupling facility structure where a checkpoint is to reside on the CKPTDEF initialization statement as described in "Checkpoint data set specifications" on page 193 and *z/OS JES2 Initialization and Tuning Reference*.

How to control JES2 initialization

This section discusses initialization and configuration features of JES2. Specific considerations unique to NJE are discussed in Chapter 5, "Network job entry (NJE)," on page 251.

JES2 initialization is performed after JES2 is started and before JES2 begins to process jobs. To control JES2 initialization:

1. Create a data set containing the initialization statements.

   For a description of the different methods for specifying an initialization data set, see:

   - "JES2 initialization statements" on page 12.
   - "Specifying JES2 command and message prefixes" on page 12.
   - "The JES2 patching facility" on page 15.
   - "JES2 and MVS operator commands within the JES2 initialization data set" on page 15.
2. Select initialization statements to be entered at the console during JES2 initialization.

   For more information, see "JES2 initialization control statements" on page 13.
3. Update the JES2 procedure to include definitions of the initialization data set(s). IBM suggests using PROCLIB statements instead of JCL changes.

   For more information, see "Creating the JES2 cataloged procedure" on page 20.
4. Select start options to be used during JES2 initialization.

   For more information, see "Specifying the Start Options" on page 28.
5. Use the exit and scanning facilities to modify initialization processing and thus modifying or supplementing JES2 functions.

   For more information, see "Controlling initialization using JES2 exits and the $SCAN facility" on page 34.

Creating an initialization data set

The initialization data set can be access directly through a DD in the JES2 procedure or it can be a member of the MVS default PARMLIB concatenation. A sample initialization data set -- SYS1.SHASSAMP(HASIPARM) -- contains templates that you can modify for a particular installation. Before modifying the sample data set, see "Defining the data set for JES2 initialization parameters" on page 27.

You can mix the operator commands and the patching facility, AMASPZAP, and initialization control statements among the initialization statements without any special coding requirements. Figure 1 on page 17 shows an example of an initialization data set that contains operator commands and initialization control statements. New data sets or members of the default PARMLIB concatenation can be processed using the INCLUDE statement.

The initialization data set should be protected to prevent disclosing passwords contained on initialization statements, such as NODE, LINE, LOGON, and RMT statements. If a security product is used, ensure that no one can access the data set. If an installation uses RACF®, specifying UACC=NONE for the data set provides this protection.

JES2 initialization statements

The initialization statements allow installations to specify the functions and device characteristics JES2 will use during its current execution. You must define most devices (local and remote) through JES2 initialization statements; however, some can be defined dynamically through $ADD operator commands.

If you do not add these statements to the initialization stream, JES2 will not define them; no default values are provided. JES2 counts the devices you have thus defined, and based on these individual device statements, is able to provide default values for various other initialization parameters. These defaults are noted in the descriptions of the parameters thus affected in *z/OS JES2 Initialization and Tuning Reference*. If you choose not to specify initialization statements for many of the other available JES2 functions, default values are provided.

Use the SMFDEF initialization statement to define the system management facilities (SMF) buffers that JES2 requires for SMF to collect and record MVS system and job-related information. The BUFNUM= parameter specifies the number of buffers JES2 obtains. If your installation has 3800 printers, this number should be calculated by the formula presented with the description of the SMFDEF statement in *z/OS JES2 Initialization and Tuning Reference*.

JES2 can evaluate whether a particular SMF record should be written, by use of a routine in JES2 installation Exit 21. See *z/OS JES2 Installation Exits* for information about Exit 21.

Specifying JES2 command and message prefixes

The CONCHAR= parameter on the CONDEF initialization statement specifies the prefix:

- For all JES2-issued messages
- By which JES2 identifies commands destined for the particular member of a multi-access spool (MAS).

Normally, JES2-initiated messages are tagged with a "$" at the beginning of the text.However, the "$" character is taken from the value of CONCHAR=; therefore, a different specification of this value for each member allows the origin of the message to be uniquely identified.

You can determine whether the CONCHAR= value applies only to the MVS system on which the JES2 member is running, or to all MVS systems in the sysplex through the SCOPE= parameter on the CONDEF initialization statement.

If you specify CONDEF ...,SCOPE=SYSTEM...., the CONCHAR value is recognized on this MVS system alone. In order to issue a JES2 command that is to take effect on a specific JES2 member, an installation would have to direct the command to that system through one of the following:

- The MVS ROUTE command

  or
- A sysplex-scope command prefix (such as the system name, if the installation uses IEECMDPF SAMPLIB member).

If each member of a MAS specifies CONDEF ...,SCOPE=SYSTEM,...., and the same value for the CONCHAR= parameter, then a ROUTE *ALL command can be used to make global changes to parameters on all members in that MAS. Because the MVS ROUTE *ALL command provides a single-system image for all JES2 members and MVS systems in the sysplex, the SCOPE=SYSTEM specification would be more

valuable for a large MAS. However, with a single console displaying output from multiple MVS and JES2 images, it might be difficult to determine which member issued a message.

To identify the member that originated a command or message, you would have to notice the MVS system ID (**SYS1**) on each console display in response to a command:

```
SYS1        $d condef
               $HASP830 CONDEF 942
     $HASP830 CONDEF  AUTOCMD=20,CONCHAR=$,BUFNUM=200,CMDNUM=100,
     $HASP830         BUFFREE=200,BUFWARN=80,MASMSG=200,RDIRAREA=Z,
     $HASP830         RDRCHAR=$,SCOPE=SYSTEM,DISPLEN=64,DISPMAX=100
```

If you specify CONDEF ...,SCOPE=SYSPLEX.... on each member of a MAS, the CONCHAR= value is recognized as belonging to this JES2 member, no matter on which MVS system in the sysplex the command entered. The SCOPE=SYSPLEX specification might be valuable for a smaller MAS, where the CONCHAR value for each member can be remembered easily. Note that the JES2 limit of 22 CONCHAR= values indicates that a MAS of more than 22 members could not use this scheme.

If you use this method, you cannot enter the MVS ROUTE *ALL command to send messages to each member. SCOPE=SYSPLEX implies that JES2 still provides a multisystem image to installations.

JES2 initialization control statements

JES2 initialization control statements can be used to control the listing and entry of initialization statements while JES2 processes the initialization stream. Initialization control statements do not set JES2 processing values. Table 3 lists the JES2-supported control statements.

Table 3. JES2 Initialization Control Statements

Statement	Explanation
LIST=NO	Ends the listing of statements on devices defined by the HASPLIST DD card. LIST=NO remains in effect until a LIST=YES statement is encountered.
LOG=YES*	Copies the current and subsequent initialization statements to the HARDCPY console. Logging remains in effect until a LOG=NO statement is encountered.
LOG=NO	Ends the logging of initialization statements. LOG=NO is automatically in effect initially and remains in effect until a LOG=YES statement is encountered.
/*	Designates the beginning of a comment. Comments can appear anywhere a blank can be coded, except within vectors or subscripts. The comment is listed or logged as appropriate but is otherwise ignored.
*/	Designates the end of a comment. Comments can appear anywhere a blank can be coded, except within vectors or subscripts. The comment is listed or logged as appropriate but is otherwise ignored.

Table 3. JES2 Initialization Control Statements (continued)

Statement	Explanation

DISPLAY D	Displays the value of specified JES2 initialization statements or specific parameters on those statements. All parameter values are displayed unless specific parameters are listed. You can add multiple statements to one DISPLAY statement; use commas between statement names. (See the following syntax.)
	
	Your message can also be displayed by following the command with message text enclosed in single quotation marks as the following example shows.
	
	`&{DISPLAY&}'message text',stmt-name=(parm1,...parmn),`
	`&{D &} stmt-name=(parm1...)`
	
	Note: Place the DISPLAY statement following the initialization statement that you need to display in order for JES2 to read these statements before building the "DISPLAY" response.

CONSOLE	Causes JES2 to get subsequent initialization parameter statements from the operator through the system console. The operator's reply to the message issued as a result of the CONSOLE statement can be any valid statement. JES2 remains in console mode until the operator replies END in response to the $HASP469 (REPLY PARAMETER STATEMENT, CANCEL, OR END) message. Optionally, the operator can reply CANCEL in response to the $HASP469 message to end JES2 initialization. Two methods of adding additional initialization statements to the JES2 initialization data set are available to the operator through use of the CONSOLE option. The first method can be used only if the JES2 initialization data set initially includes the CONSOLE control statement. This causes JES2 to enter **console mode** at the time the CONSOLE statement is read. At this time the operator can add or override additional initialization statements. The second method is to use the CONSOLE option to stop JES2 initialization processing after all initialization statements have been read and processed before any subsequent initialization processing. The operator is permitted to add additional statements at this time. Consistent with JES2 initialization, if JES2 reads more than one initialization statement for a particular parameter, the last value read is used. This allows the operator either to add new statements to the initialization deck or to change any statements previously entered.
*	These statements are also controlled by options specified at the time you start JES2. You will only produce output if the start options, LIST and LOG, are specified. To display the current start options, use the DISPLAY OPTSDEF command (during JES2 initialization) and then override the options (using the OPTSDEF statement) from the console, if necessary. See *z/OS JES2 Initialization and Tuning Reference* for more information.

Table 3. JES2 Initialization Control Statements (continued)

Statement	Explanation
INCLUDE	The include initialization statement allows new initialization data sets or members of the default PARMLIB concatenation to be processed. JES2 will immediately start reading records from the new data set and processing them. When all statements in the included data set have been processed, JES2 initialization resumes processing records from the original data set. Include statements can be nested (included initialization data set can include other data set). There is code to detect a nested include loop (data set A includes data set B that then includes data set A). If a loop is detected, the include request will fail.

The JES2 patching facility

Patch and AMASPZAP statements can be used to make minor and temporary modifications to the JES2 object code until JES2 is restarted by directly replacing the changed code. The JES2 Patching Facility changes only the memory copy of data; the copy residing on DASD (for example, LPA) cannot be replaced.

The JES2 patching facility makes temporary patches to any module in JES2 (HASJES20 or installation exit load module) or to any absolute storage address in the address space into which JES2 is loaded. Because these patches are valid only until a module is reloaded, they must be applied every time that JES2 is started.

JES2 reloads the load modules at different stages in processing:

- Subsystem support modules are reloaded during all starts except a hot start.
- HASJES20 is reloaded in response to a JES2 START command any time the member is started.

These patches are applied at the time JES2 is initialized. The patching facility statements are submitted to the JES2 initialization data set.

There are two basic patching formats:

1. Using the NAME initialization statement to specify the module name
2. Specifying the module name in the VER initialization statement.

Note: If you specify the module name in both the NAME and the VER initialization statement, JES2 cannot patch the data.

For more directions on how to code the patch statements, see the NAME, REP, and VER initialization statements in *z/OS JES2 Initialization and Tuning Reference*. Directions for using the AMASPZAP program are provided in the "SPZAP" topic in *z/OS MVS Diagnosis: Tools and Service Aids*. JES2 processes the patch and AMASPZAP statements as they are read.

JES2 and MVS operator commands within the JES2 initialization data set

You can insert both JES2 and MVS commands into your JES2 initialization data set. This is convenient, for example, to "automatically" start devices when JES2 initializes. For instance, operator commands can be used to start RJE lines during initialization. (RJE lines, unlike other devices, cannot be started automatically by an initialization statement.)Figure 1 on page 17 contains a section called "Operator Commands" that shows a $S LINE command, a $T LINE command and a $VS command used to insert a MVS VARY command. IBM suggests that you add as

many such commands within your initialization date set as appropriate to your installation's needs. See JES2 operator commands and the $VS command in *z/OS JES2 Commands*.

During initialization, JES2 stores the operator commands in temporary message buffers. Then, when initialization is complete, JES2 processes the commands. To ensure that operator commands are completely processed before JES2 starts processing jobs, you should use the REQ start option, which lets the operator start JES2 processing. Alternatively, the $S command can be included as the last operator command in the initialization data set, eliminating the need for operator intervention.

Sample JES2 initialization data set and HASPLIST corrections

The following two figures provide examples of the JES2 initialization data set used to start JES2 and the HASPLIST data set that results from processing the initialization data set. The HASPLIST data set is generated through the LIST=YES parameter on the OPTSDEF initialization statement. Note that when an error occurs, the operator must retype the entire initialization statement correctly and follow it with an end statement to continue initialization processing.

You can preallocate a data set and indicate to JES2 to use it for HASPLIST. In this case, you need to perform the pre-allocation with DCB parameters of LRECL=121 and RECFM=FBA. You need to ensure that the BLKSIZE on the pre-allocated data set is an integer multiple of LRECL; otherwise an **013** abend occurs.

```
OPTSDEF   LIST=YES
/*********************************************************************/
/*                                                                   */
/*              SAMPLE JES2 PARAMETER LIBRARY LISTING                */
/*                                                                   */
/*********************************************************************/
CKPTDEF   CKPT1=(DSN=SYS1.JESCKPT1,VOL=CHECK1,INUSE=YES),
          CKPT2=(DSN=SYS1.JESCKPT2,VOL=CHECK2,INUSE=YES),
          MODE=DUPLEX,DUPLEX=ON
/*********************************************************************/
/*                                                                   */
/*               LOCAL DEVICES                                       */
/*                                                                   */
/*********************************************************************/
RDR(1)    UNIT=00C
RDR(2)    UNIT=011,PRIOLIM=9,CLASS=X,AUTH=(JOB=NO,SYSTEM=NO,DEVICE=NO),
          PRTDEST=R3
PRT(1)    UNIT=002,CLASS=AJH,UCS=PN
PRT(2)    UNIT=00E,CLASS=AJH,UCS=PN
PRT(3)    UNIT=00F,CLASS=A,ROUTECDE=22,UCS=PN
PRT(4)    UNIT=018,CLASS=NI,MARK=YES,BURST=NO,TRKCELL=YES
PUN(1)    UNIT=00D,PAUSE=YES
INTRDR    PRIOLIM=9,AUTH=(JOB=NO,SYSTEM=NO,DEVICE=NO)
INITDEF   PARTNUM=8
INIT(1)   CLASS=AFJKE                   /*INITIATOR 1*/
INIT(2)   CLASS=BCDEF                   /*INITIATOR 2*/
INIT(3)   CLASS=DEFGH                   /*INITIATOR 3*/
INIT(4)   CLASS=XKH                     /*INITIATOR 4*/
INIT(5)   CLASS=JKEBF                   /*INITIATOR 5*/
INIT(6)   DRAIN                         /*SPARE INITIATOR*/
INIT(7)   DRAIN                         /*SPARE INITIATOR*/
INIT(8)   DRAIN                         /*SPARE INITIATOR*/
JOBCLASS(STC) LOG=NO,OUTPUT=NO,CONDPURG=YES  /*STARTED TASK DEFINITIONS*/
JOBCLASS(TSU) REGION=50K,COMMAND=IGNORE,MSGLEVEL=(1,1),CONDPURG=YES
JOBCLASS(S)  PROCLIB=03,HOLD=YES        /*SYSTEM PROGRAMMER CLASS */
OUTCLASS(H)  OUTDISP=(HOLD,HOLD),TRKCEL=NO     /*SYSOUT CLASS HELD */
                                                /*FOR OUTPUT        */
OUTCLASS(N)  TRKCELL=YES
OUTCLASS(X)  OUTPUT=DUMMY,TRKCELL=NO     /*THROWAWAY CLASS*/
/*********************************************************************/
/*                                                                   */
/*               NJE/RJE INITIALIZATION PARAMETERS                   */
/*                                                                   */
/*********************************************************************/
NJEDEF    OWNNODE=1,NODENUM=10,LINENUM=15,RESTNODE=10
LINE(1)   UNIT=040,DUPLEX=FULL,TRANSPAR=YES,REST=10
LINE(2)   UNIT=041,TRANSPAR=YES,PASSWORD=SECRET,REST=20
LINE(3)   UNIT=042,TRANSPAR=YES,PASSWORD=SECRET,REST=15
LINE(4)   UNIT=043,TRANSPAR=YES,PASSWORD=SECRET,REST=50
LINE(5)   UNIT=044,TRANSPAR=YES,PASSWORD=SECRET,REST=10
LINE(6)   UNIT=SNA
LINE(7)   UNIT=SNA,PASSWORD=LINE4PW
RMT(1)    DEVTYPE=3780,LINE=1,NUMPUN=1,TRANSPAR=YES,BUFEXPEN=1,
          COMPRESS=YES
R1.PR1    PRWIDTH=144
RMT(2)    DEVTYPE=2922,NUMPU=1,CONS=YES,MULTILV=YES,TRANSPAR=YES
R2.PR1    PRWIDTH=132
RMT(3)    DEVTYPE=S/370,NUMPRT=2,CONS=YES,MULTILV=YES,TRANSPAR=YES
R3.PR1    PRWIDTH=150,FCBLOAD=YES
R3.PR2    PRWIDTH=132
RMT(4)    DEVTYPE=1130,CONS=YES,MULTILV=YES,NUMPUN=1
R4.PU1    START=NO
RMT(5)    DEVTYPE=SYSTEM/3,NUMRDR=3,NUMPUN=2,CONS=YES,MULTILV=YES
R5.PR1    PRWIDTH=132
RMT(6)    DEVTYPE=2780,NUMPUN=1,TRANSPAR=YES,MRF2780=YES,HTABS=YES
R6.PR1    PRWIDTH=144
RMT(7)    DEVTYPE=LUTYPE1,ROUTECDE=10,BUFSIZE=512,DISCINTV=8000,
          NUMPUN=1,COMPRESS=YES
RMT(8)    DEVTYPE=LUTYPE1,LINE=6,BUFSIZE=256,NUMPUN=1
OPTSDEF LIST=NO
/*********************************************************************/
/*                                                                   */
/*                REMOTE PASSWORDS                                   */
```

```
                    JES2 PARAMETER LIBRARY LISTING
                 CORRECTING INITIALIZATION PARAMETER ERRORS
                               PAGE 1

SHASPARM   STMT   1  OPTSDEF   LIST=YES
SHASPARM   STMT   2  /*********************************************/
SHASPARM   STMT   3  /*                                           */
SHASPARM   STMT   4  /*        SAMPLE JES2 PARAMETER LIBRARY LISTING  */
SHASPARM   STMT   5  /*                                           */
SHASPARM   STMT   6  /*********************************************/
SHASPARM   STMT   7  CKPTDEF   CKPT1=(DSN=SYS1.JESCKPT1,VOL=CHECK1,INUSE=YES),
                             CKPT2=(DSN=SYS1.JESCKPT2,VOL=CHECK2,INUSE=YES),
                     MODE=DUPLEX,DUPLEX=ON
SHASPARM   STMT   8  /*********************************************/
SHASPARM   STMT   9  /*                                           */
SHASPARM   STMT  10  /*               LOCAL DEVICES               */
SHASPARM   STMT  11  /*                                           */
SHASPARM   STMT  12  /*********************************************/
SHASPARM   STMT  13  RDR(1)    UNIT=00C
SHASPARM   STMT  14  RDR(2)    UNIT=011,PRIOLIM=9,CLASS=X,
                             AUTH=JOB=NO,SYSTEM=NO),PRTDEST=3
SHASPARM   STMT  15  PRT(1)    UNIT=002,CLASS=AJH,UCS=PN
SHASPARM   STMT  16  PRT(2)    UNIT=00E,CLASS=AJH,UCS=PN
SHASPARM   STMT  17  PRT(3)    UNIT=00F,CLASS=A,ROUTECDE=22,UCS=PN
DIAGNOSTIC STMT  17  RC=(18),PRT(3) ROUTECDE (ELEMENT 1)  - INVALID ROUTE
DIAGNOSTIC STMT  17  CODE
CONSOLE    STMT  18  PRT(3) UNIT=00F,CLASS=A,ROUTECDE=R22,UCS=PN
CONSOLE    STMT  19
                     END
SHASPARM   STMT  20  PRT(4)    UNIT=018,CLASS=NI,MARK=YES,BURST=NO,TRKCELL=YES
SHASPARM   STMT  21  PUN(1)    UNIT=00D,PAUSE=YES
SHASPARM   STMT  22  INTRDR    PRIOLIM=9,AUTH=(JOB=NO,SYSTEM=NO,DEVICE=NO)
DIAGNOSTIC STMT  22  RC=(03),INTRDR PRIOLIM  - INVALID PARAMETER STATEMENT
CONSOLE    STMT  23  INTRDR PRTYLIM=9,AUTH=(JOB=NO,SYSTEM=NO,DEVICE=NO)
CONSOLE    STMT  24  END

                    JES2 PARAMETER LIBRARY LISTING
                               PAGE 2

SHASPARM   STMT  25  INITDEF   PARTNUM=8
SHASPARM   STMT  26  INIT(1)   CLASS=AFJKE                /*INITIATOR 1      */
SHASPARM   STMT  27  INIT(2)   CLASS=BCDEF                /*INITIATOR 2      */
SHASPARM   STMT  28  INIT(3)   CLASS=DEFGH                /*INITIATOR 3      */
SHASPARM   STMT  29  INIT(4)   CLASS=XKH                  /*INITIATOR 4      */
SHASPARM   STMT  30  INIT(5)   CLASS=JKEBF                /*INITIATOR 5      */
SHASPARM   STMT  31  INIT(6)   DRAIN                      /*SPARE INITIATOR  */
SHASPARM   STMT  32  INIT(7)   DRAIN                      /*SPARE INITIATOR  */
SHASPARM   STMT  33  INIT(8)   DRAIN                      /*SPARE INITIATOR  */
SHASPARM   STMT  34  JOBCLASS(STC)  LOG=NO,OUTPUT=NO,     /*STARTED TASK     */
                     CONDPURG=YES                         /*DEFINITIONS      */
SHASPARM   STMT  35  JOBCLASS(TSU)  REGION=50K,COMMAND=IGNORE, /*TIME-SHARING */
                     MSGLEVEL=(1,1),CONDPURG=YES          /*USER DEFINITIONS */
SHASPARM   STMT  36  JOBCLASS(S)  PROCLIB=03,HOLD=YES     /*SYSTEM PROGRAMMER*/
                                                          /*CLASS            */
SHASPARM   STMT  37  OUTCLASS(H)  OUTDISP=(HOLD,HOLD),    /*SYSOUT CLASS HELD*/
                     TRKCEL=NO                            /*FOR OUTPUT       */
DIAGNOSTIC STMT  37  RC=(03),OUTCLASS(H) TRKCEL  - INVALID PARAMETER
DIAGNOSTIC STMT  37  STATEMENT
CONSOLE    STMT  38  OUTCLASS(H) OUTDISP=(HOLD,HOLD),TRKCELL=NO
CONSOLE    STMT  39  END
SHASPARM   STMT  40  OUTCLASS(N)  TRKCELL=YES
SHASPARM   STMT  41  OUTCLASS(X)  OUTPUT=DUMMY,TRKCELL=NO  /*THROWAWAY CLASS  */

                    JES2 PARAMETER LIBRARY LISTING
                               PAGE 3

SHASPARM   STMT  42  /*********************************************/
SHASPARM   STMT  43  /*                                           */
SHASPARM   STMT  44  /*          NJE/RJE INITIALIZATION           */
SHASPARM   STMT  45  /*                                           */
SHASPARM   STMT  46  /*********************************************/
SHASPARM   STMT  47  NJEDEF    OWNNODE=1,NODENUM=10,LINENUM=15,RESTNODE=10
SHASPARM   STMT  48  LINE(1)   UNIT=040,DUPLEX=FULL,TRANSPAR=YES,REST=10
SHASPARM   STMT  49  LINE(2)   UNIT=041,TRANSPAR=YES,PASSWORD=SECRET,REST=20
```

Creating the JES2 cataloged procedure

To start the JES2 component, you must provide a subsystem cataloged JCL procedure (PROC). The PROC name can be a maximum of four characters. For other naming restrictions, see *z/OS MVS JCL User's Guide*. Before IPL, define the PROC in the IEFSSNxx member(s) of SYS1.PARMLIB.

Instead of a distributed JES2 procedure, IBM provides the sample member 'HASIPROC' in the SYS1.SHASSAMP data set. 'HASIPROC' is a sample JCL procedure that should be tailored, renamed, and moved to the production JCL procedure library for use as the JES2 subsystem JCL (the procedure specified on a START command to start JES2).

The basic JES2 procedure (Figure 3) contains an EXEC statement and five data definition (DD) statements named PROC00, PROC01, HASPLIST, HASPPARM and STEPLIB. PROC00 defines a default procedure library to be used for converting the JCL of batch jobs, time-sharing logons, and system tasks. HASPLIST defines what is normally a dummy output data set, but you can modify this statement to permanently point to a specific device or data set. Also, you can use the START command facilities to generate a IEFRDER statement to override the HASPLIST statement.

The following parameters on the EXEC statement deserve mentioning:

PGM= Specify the name of the JES2 load module. (It must reside in an authorized library.)

TIME=
Specify 'NOLIMIT' or 1440 to prevent system ABEND code 322 (CPU time limit exceeded).

REGION=
Typically, do not code this parameter (or specify REGION=0M) to prevent any virtual storage restriction on the JES2 address space.

PERFORM=
Specify a unique performance group for tracking JES2 resources through RMF™.

PARM=
Specify 'Warm,NOREQ' for the start parameters in the JES2 procedure. If necessary you can override these parameter specifications on the 'S JES2' command. See "Specifying the Start Options" on page 28.

```
//JES2         PROC N=SYS1,L=SHASLNKE,
//                  M=JES2PARM,PN=SYS1,PL=PARMLIB,U=
//                  PROC00='SYS1.PROCLIB',PROC01='SYS1.PROCLIB'
//IEFPROC      EXEC PGM=HASJES20,TIME=1440,PARM=(WARM,NOREQ)
//HASPLIST  *  DD DDNAME=IEFRDER
//HASPPARM     DD UNIT=&U,DSN=&PN;.&PL(&M),DISP=SHR
//PROC00       DD DSN=&PROC00,DISP=SHR
//PROC01       DD DSN=&PROC01,DISP=SHR
//STEPLIB      DD DSN=&N;.&L,DISP=SHR
```

Figure 3. The Basic JES2 Procedure

* The MVS message, IEC130I HASPLIST DD STATEMENT MISSING, will be issued if this DD name is not included here.

You can change the JES2 procedure by entering updates with a TSO/E editor such as SPF or the IBM IEBUPDTE utility program. You can add DD statements to the JES2 procedure to define:

- Other cataloged procedure libraries that are associated with job classes by the JOBCLASS initialization statement or by the /*JOBPARM JES2 control statement.

  IBM suggests using PROCLIB statements instead of changing the JES2 PROC.

  Each library should be defined by a PROCnn DD statement. For instance, to specify a special library for TSO/E logons, code PROCLIB=nn (where nn corresponds to the associated PROCnn DD statement) in the JOBCLASS(TSU) initialization statement.
- One or more initialization data sets that define different conditions and types of workloads. See Figure 5 on page 23 for an example of an updated JES2 procedure.

IBM suggests that you specify DDNAMEs in the format of PARMxxxx when defining installation parameter libraries in the JES2 procedure. After initialization, JES2 automatically unallocates data sets that have a DDNAME of HASPLIST, HASPPARM, or PARMxxxx. By unallocating these data sets, installations can dynamically delete the devices on which these data sets reside.

Specifying a back-up JES2 cataloged procedure

IBM suggests that you define a back-up JES2 cataloged procedure. If JES2 fails to start because of either a JCL or an allocation error, you can use this member to start JES2 while you resolve the errors with the original procedure.

You must build a member in one of the data sets defined by the IEFPDSI statement in the MSTRJCLxx. In the example below, the data set SYS1.PROCLIB contains the JCL found for the JESBACK member.

Enter the following command to start this procedure:

```
START JESBACK,JOBNAME=JES2,SUB=MSTR
```

where *JESBACK* is an installation-defined name for the member of SYS1.PROCLIB that contains the backup JES2 procedure.

```
//IEFPROC      EXEC PGM=HASJES20,TIME=1440,PARM=(WARM,NOREQ)
//HASPPARM     DD UNIT=3390,DSN=SYS1.SECOND,DISP=SHR
//PROC00       DD DSN=SYS1.PROCLIB,DISP=SHR
```

Figure 4. The 'JESBACK' Back-up Procedure

When you have corrected the error in the original procedure, you can restart it by entering a $P JES2,ABEND operator command to bring down the JES2 member. Then restart the JES2 member through the MVS START command (S JES2) when you have replaced the back-up procedure with the original JES2 procedure.

Specifying user PROCLIBs

As part of the conversion processing for jobs, started tasks (STCs), and time sharing users (TSUs), you can specify that JES2 use an installation procedure library . This library is a set of common JCL procedures that the JCL being converted can reference. These procedure libraries, PROCLIBs, can be defined

either in the JES2 PROC or by specifying the JES2 PROCLIB initialization statement. Use operator commands without restarting JES2 to display, update, and delete PROCLIBs defined by JES2 initialization statements . In addition, an error in the PROCLIB initialization statement will not prevent JES2 from starting. For these reasons, the PROCLIB initialization statement is the recommended method for defining user PROCLIBs. For more information on the PROCLIB initialization statement, see *z/OS JES2 Initialization and Tuning Reference*.

If you choose to define your user PROCLIBs in the JES2 cataloged procedure, it is important to do so by using symbolic data set names that can be changed by the start command. Otherwise, a missing user PROCLIB would require IPLing a backup system to make any changes to the production system.

To ensure performance and integrity, **do not**:

- Allocate additional extents or release extents
- Compress a PROCLIB data set
- Defragment (move) a volume containing a PROCLIB.

The following example JES2 procedure allows an operator to change user PROCLIB data set names in the event of a failure. If a failure occurs during JES2 startup because of a missing user PROCLIB, restart JES2 and supply the missing SYS1.PROCLIB statement.

```
//*****************************************************************
//*                                                               *
//*            EXAMPLE OF AN UPDATED JES2 PROCEDURE               *
//*                                                               *
//*****************************************************************
//JES2    PROC DSN1='SYS1.PROCLIB',    * STANDARD PROCLIB.        *
//        DSN2='SYS1.PROCLIB2',        * SYSTEM SUPPORT PROCS.    *
//        DSN3='SYS1.PROCSPP',         * PROG. PRODUCT PROCS.     *
//        DSN4='SYS2.PROCLIB',         * USER PROCLIB.            *
//        STEPLIB='SYS1.JES2.SHASLNKE',   * JES2 PGM LIBRARY      *
//        PGN=20,                      * PERFORM FOR JES2         *
//        MBR=HASIPARM,TYPE=CTS        * DEFAULT NAME ALTERNATE   *
//*                                    * PARM, TYPE (HAS/NJE/CTS) *
//*****************************************************************
//* MODIFIED JES2 START PROCEDURE                                 *
//*  CHANGES -                                                    *
//*    ADDITIONAL PROCLIBS - 1) SYS1.PROCLIB  (SYSTEM PROCLIB)    *
//*                          2) SYS1.PROCLIB2 (SPECIAL PROCS)     *
//*                          3) SYS1.PROCSPP  (P.P. PROCS)        *
//*                          4) SYS2.PROCLIB  (USER PROC'S)       *
//*    SELECT VERSION OF JES VIA TYPE=HAS/NJE/CTS                 *
//*    CHANGED TO USE SYS1.PROCLIB FOR JES2 INIT. PARMS           *
//*    LAST CHANGE - 10/14/86.                                    *
//*                                                               *
//*****************************************************************
//*                                                               *
//IEFPROC EXEC PGM=&TYPE.JES20,TIME=1440,
//        PERFORM=&PGN,     DUMMY PERFORM FOR TRACKING
//        PARM=(WARM,NOREQ)  SPECIFY START OPTIONS
//STEPLIB  DD  DSN=&STEPLIB,DISP=SHR  (MUST BE AUTHORIZED)
//*                                                               *
//******** DEFAULT PROCEDURE LIBRARIES LIST  ***********************
//*                                                               *
//PROC00    DD DSN=&DSN1,DISP=SHR
//          DD DSN=&DSN2,DISP=SHR
//          DD DSN=&DSN3,DISP=SHR
//          DD DSN=&DSN4,DISP=SHR
//*                                                               *
//******** INDIVIDUAL PROCXX CARDS FOR EACH PROCLIB. ***************
//*                                                               *
//PROC01    DD DSN=&DSN1,DISP=SHR    SPECIFIC DECLARATIONS
//PROC02    DD DSN=&DSN2,DISP=SHR    FOR USE WITH JOBPARM PROC=
//PROC03    DD DSN=&DSN3,DISP=SHR    PARAMETERS.
//PROC04    DD DSN=&DSN4,DISP=SHR    *
//HASPPARM  DD DSN=SYS1.PROCLIB(&TYPE.PARM;),DISP=SHR
//PARMBACK  DD DSN=SYS1.JESPARM(&MBR),DISP=SHR          /*ALTERNATE*/
//HASPLIST  DD DDNAME=IEFRDER                           /*LISTING */
//*                                                               *
```

Figure 5. Updated JES2 procedure example

Note: Data sets specified in the JES2 procedure (such as all DD statements) must be reference in one of the following ways:

- Cataloged in the master catalog (or in a user catalog, provided the data set has a high-level qualifier other than SYS1)
- Specify UNIT= and VOL= on the JCL DD statement
- SMS managed data (do not specify UNIT or VOLUME) (UNIT= and VOL= on the JCL DD statement are ignored.)

Using dynamic PROCLIB allocation

JES2 can dynamically allocate the PROCLIB(s) it uses. By using an initialization statement and a set of operator commands, you can take one further step to

eliminate all-MAS member restarts. A restart is required for static PROCLIB changes. If you define your PROCLIB(s) by the static JCL statements in the JES2 PROC (as in Figure 1), and you accidentally introduce an error, such as imperfect syntax, or reference a damaged or missing data set, that error prevents JES2 initialization. You are then required to restart all MAS members to recognize the changes to the PROCLIB(s).

However, if you set up your JES2 system to dynamically allocate PROCLIB members by using the PROCLIB(xxxxxxxx) initialization statement (as in Figure 2) and a set of PROCLIB(xxxxxxxx) commands (as in Examples 1, 2, and 3) you eliminate your need to restart JES2 whenever you need to change PROCLIB definitions. When defined by initialization statement, PROCLIBs can then be added, changed, deleted, and displayed using JES2 commands.

The JES2 PROCLIB specifications are primed with the static PROCLIBs (from the JES2 PROC) during initialization processing. You can display or alter the static PROCLIBs using the PROCLIB statement. This creates a new logical PROCLIB concatenation which overrides the static PROCLIB. You can delete the logical PROCLIB concatenation using the $DEL PROCLIB(xxxxxxxx) command, but you cannot delete the static PROCLIB.

The following sections describe how you can define, change, and display PROCLIB(s) using this dynamic technique. Be aware of the following restrictions:

- The JES2 PROCLIB data sets are typically not protected by data set ENQs. As a result, PROCLIB data sets can be scratched or moved while JES2 is running. However, if the JES2 program property table is updated (using a SCHEDxx PARMLIB member), then a data set ENQ will be used for the PROCLIB data sets. If the data set ENQ is not obtained, IBM suggests setting up your own security procedure to prevent accidental loss. The $T PROCLIB command can be used to reallocate libraries if necessary.
- The $T PROCLIB and $DEL PROCLIB commands only delete the concatenation logically. As a result, when you use a PDSE in the concatenation, the PDSE might not be immediately available to scratch. The dynamic PROCLIB's USECOUNT must equal zero to successfully scratch the PDSE. When a PDSE is allocated as part of a dynamic PROCLIB concatenation, the PROCLIB remains open to a JES2 converter until the specific converter is assigned to convert a new job. A PDSE cannot be closed and released from JES2 allocation until all JES2 converter PCEs are submitted and converted, the dynamic PROCLIB's USECOUNT equals zero, and ample time has elapsed. You must perform this procedure on each member in the sysplex where the PDSE is part of a dynamic PROCLIB concatenation. $D PROCLIB(xxxxxxxx),DEBUG displays the USECOUNT. $D PCEDEF,CNVTNUM displays the number of JES2 converter PCEs.
- PROCLIB statement processing ensures that specified data sets can be allocated. It does not ensure that they actually exist, can be opened, or used as a PROCLIB data set.
- Because JES2 allocates PROCLIB during initialization, it does not become aware of extents you add at a later point. Therefore, using secondary extents can cause I/O errors. However, if JES2 encounters an I/O error reading the PROCLIB data set, JES2 closes and re-opens that data set to recognize and use any additional extents.

Note: When dynamic PROCLIBs are allocated, JES2 uses the DSI/NODSI program property attribute that is associated with the HASJES20 program. If DSI is

specified, JES2 requests allocation processing to obtain the appropriate data set ENQs for the dynamic PROCLIB data sets.

Table 4. Dynamic PROCLIB rules

Action	Existing PROCLIB	Results
$ADD PROCLIB	None	New dynamic PROCLIB added
$ADD PROCLIB	Dynamic PROCLIB	Error, a dynamic PROCLIB exists
$ADD PROCLIB	Static PROCLIB	New dynamic PROCLIB created, no data sets copied from the static PROCLIB
$T PROCLIB	None	Error, no selectable entries to modify
$T PROCLIB	Dynamic PROCLIB	Appropriate DD entries are modified (starts with existing DDs and makes updates to build a new concatenation)
$T PROCLIB	Static PROCLIB	A new dynamic PROCLIB is created by copying the existing DDs from the static concatenation to the new dynamic PROCLIB and then makes updates to build a new concatenation.
$DEL PROCLIB	None	Error, no selectable entry to delete
$DEL PROCLIB	Dynamic PROCLIB	Dynamic PROCLIB is deleted. If there is a static PROCLIB with the same subscript as the dynamic PROCLIB, it becomes active again
$DEL PROCLIB	Static PROCLIB	Error, cannot delete static PROCLIB

Using the PROCLIB(xxxxxxxx) initialization statement

Use the PROCLIB(xxxxxxxx) statement to define a dynamic PROCLIB concatenation to be used by JES2 during conversion processing on this JES2 member. Dynamic PROCLIBs can be used to override PROCxx DDs in the JES2 start PROC, but it cannot be used to alter or delete them.

You can add up to 255 DDs per PROCLIB member, and optionally specify VOLSER= and UNIT= definitions if you have not previously cataloged the PROCLIB data sets.

By adding UNCONDITIONAL to a PROCLIB statement, you are requesting that JES2 "ignore" data set(s) allocation failures. Therefore, the data set whose allocation failed is ignored, and JES2 continues statement processing, and creates the PROCLIB structure for all valid DD statement (DSNAMEs) if at least one allocation succeeds. Use a $D PROCLIB command to verify individual DSNAME allocations. JES2 returns $HASP319 PROCLIB(xxxxxxxx)... and flags all DD statements that failed allocations (DDs with "ALLOCATION FAILED"). You can then use $T

PROCLIB to change the definition, $DEL PROCLIB or $ADD PROCLIB to delete or add a new PROCLIB, respectively, all without a JES2 restart.

Figure 6 provides a static JCL PROCLIB definition. Compare it to Figure 7which defines the same simple definitions using the dynamic JES2 PROCLIB statement.

```
//PROC01  DD   DSN=USER.PROCLIB1,VOL=SER=J2COM1,UNIT=3390
//        DD   DSN=USER.PROCLIB2,VOL=SER=J2COM1,UNIT=3390
//        DD   DSN=SYS1.PROCLIB
```

Figure 6. Static JES2 PROCLIB definitions using JCL

```
PROCLIB(PROC01) DD(1)=(DSN=USER.PROCLIB1,VOLSER=J2COM1,UNIT=3390),
                DD(2)=(DSN=USER.PROCLIB2,VOLSER=J2COM1,UNIT=3390),
                DD(3)=(DSN=SYS1.PROCLIB)
```

Figure 7. Dynamic JES2 PROCLIB definitions using JES2 PROCLIB statement

See*z/OS JES2 Initialization and Tuning Reference*for complete PROCLIB(xxxxxxxx) syntax and parameter definition.

Displaying the PROCLIB(xxxxxxxx) initialization statement

Use the $D PROCLIB command to display your definitions as needed. This is particularly useful before adding a new or changed definition to your production system. Figure 8 and Figure 9reflect the definition specified in Figure 7. Note that Figure 9 includes the DEBUG operand which returns additional PROCLIB data: USECOUNT, DDNAME, and a creation timestamp.

```
$D PROCLIB(PROCLIB1)

      $HASP319 PROCLIB(PROC01)
$HASP319 PROCLIB(PROC01)  DD(1)=(DSNAME=USER.PROCLIB1,
$HASP319                  VOLSER=J2COM1,UNIT=3390),
$HASP319                  DD(2)=(DSNAME=USER.PROCLIB2,
$HASP319                  VOLSER=J2COM1,UNIT=3390),
$HASP319                  DD(3)=(SYS1.PROCLIB)
```

Figure 8. $D PROCLIB example

```
$D PROCLIB(PROCLIB1),DEBUG

       $HASP319 PROCLIB(PROC01)
$HASP319 PROCLIB(PROC01)  USECOUNT=0,DDNAME=SYS00006,
$HASP319                  CREATED=2003.145,14:33:22.36,
$HASP319                  DD(1)=(DSNAME=USER.PROCLIB1,
$HASP319                  VOLSER=J2COM1,UNIT=3390),
$HASP319                  DD(2)=(DSNAME=USER.PROCLIB2,
$HASP319                  VOLSER=J2COM1,UNIT=3390),
$HASP319                  DD(3)=(SYS1.PROCLIB)
```

Figure 9. $D PROCLIB example using the DEBUG feature

See *z/OS JES2 Commands* for complete $D PROCLIB(xxxxxxxx) syntax and parameter definition, and *z/OS JES2 Messages*for an explanation of $HASP319.

Modifying the PROCLIB(xxxxxxxx) initialization statement

There are several ways by which you can change your dynamic PROCLIB definitions. Here are three methods which you might find useful.

Example 1:

```
$T PROCLIB(PROC01),DD(1)=...,DD(2)=...  /* change production PROC dynamically without a restart      */
```

Always keep in mind how error prone such definitions can be. Repeated use of the $T command is acceptable, of course. If your PROCLIB is large, rather than using a single $T command, which might prove impossible because of command length limitations (consoles: 120 characters; card readers: 70 characters).

Example 2:

```
1. $ADD PROCLIB(TEMP01),DD(1)=...       /* copy and add PROC, but do not use                          */
2. $T PROCLIB(TEMP01),DD(2)=...         /* change definitions as required, still do not use           */
3. Test and update TEMP01 as required   /* test until correct                                         */
4. $T PROCLIB(TEMP01),NAME=PROC01       /* change NAME= to production name to override current PROC  */
```

Entering the correct subscripts on the DD statement can be error prone. If you are building up a concatenation, there is an easier way. The next example takes advantage of the fact that JES2 compresses null DDs out of the concatenation. Note that data sets B, C, and D all specify a high DD subscript (99) which JES2 compresses to the next (lowest) available DD subscript. JES2 renumbers them to 2, 3, and 4, respectively. The example uses operator commands, but you can use the same technique when coding initialization statements.

Example 3:

```
1. $ADD PROCLIB(TEMP01),DD(1)=DSN=A... /* Create new PROCLIB concatenation or copy an old one       */
2. $T PROCLIB(TEMP01),DD(99)=DSN=B...  /* but always add to DD number 99. Because JES2 compresses   */
3. $T PROCLIB(TEMP01),DD(99)=DSN=C...  /* null DDs, each $T adds to the end of the concatenation    */
4. $T PROCLIB(TEMP01),DD(99)=DSN=D...  /* (assuming less than 99 DDs)                               */
5. $D PROCLIB(TEMP01)
$HASP319 PROCLIB(TEMP01) DD(1)=DSN=A...
$HASP319                 DD(2)=DSN=B...
$HASP319                 DD(3)=DSN=C...
$HASP319                 DD(4)=DSN=D...
6. Test and update TEMP01 as required    /* test until correct                                        */
7. $T PROCLIB(TEMP01),NAME=PROC01        /* change NAME= to production name to override current PROC  */
```

See *z/OS JES2 Commands* for complete $T PROCLIB(xxxxxxxx) syntax and parameter definition.

Defining the data set for JES2 initialization parameters

The HASPPARM DD statement defines a member in SYS1.SHASSAMP that initially contains the sample initialization data set. The sample initialization data set is **only** an example and requires system-specific modification. You will find it to be a useful base for coding your initialization data set(s).

When creating the JES2 initialization data set for your installation, you must move it from SYS1.SHASSAMP to a data set comprised of 80 character records in a fixed record format. The data set can be blocked with a BLKSIZE that is a multiple of 80. If your installation's SYS1.SHASPARM data set is not blocked, you can improve system performance by moving this member to a blocked library and tailoring the HASPPARM DD statement in the JES2 cataloged procedure accordingly.

Use the HASPPARM DD statement to define your initialization data set to replace the IBM-supplied null data set. You can also create the initialization data set separately and define it by its own DD statement in the procedure.

Your installations can use resources more efficiently by dividing the entire initialization data set into multiple members. Multiple members allow you to:

- Separate JES2 parameters according to function.
- Allow multiple programmers to manage distinct subsets of parameters.
- Share a subset of parameters between the members of a JES2 MAS. (For the members of a JES2 MAS to share parameters, some HASPPARM data sets must be shared among all the members.)

To process multiple parameter members, you can either list them all in the JES2 procedure, using a concatenated DD for HASPPARM or use the JES2 INCLUDE initialization statement. To use the INCLUDE initialization statement, point the HASPPARM DD in the JES2 procedure to a single data set and member which contains the main JES2 initialization data set. This main data set can then INCLUDE each of the other initialization data sets. Use system symbolics to identify member–specific data sets that you need to include.

If JES2 cannot allocate a data sets specified on the HASPPARM DD statement, then you might need to start a backup JES2 procedure, or IPL a backup system to correct the problem. Using a concatenation of data sets for the HASPPARM DD increases the number of data sets that could cause a problem starting JES2. Using system symbolics in the JES2 procedure introduces another potential source of errors. Similar errors allocating data sets using the INCLUDE initialization statements will cause JES2 to enter console mode giving the operator an opportunity to correct the error without having to restart JES2. To ensure HASPPARM data set errors will not prevent JES2 from starting:

- Point the HASPPARM DD to a single, specific data set and member that is not likely be changed.
- Use INCLUDE statements in that initialization data set to add initialization statements from additional data sets.
- Use system symbolics in the INCLUDE statement to identify member–specific data sets or members to include.

The initialization of JES2 supports the concatenation of data sets of unlike attributes as the initialization statement data set. This allows the use of a card reader to introduce the statements to JES2 by concatenating it to the standard PARMLIB initialization member while they are being tested by the installation.

Specifying the Start Options

When JES2 is started, it uses the specified start options to determine how it will perform the current initialization. The start options described in this section are defined by JES2. However, you can define other start options for your installation by using the scanning facility (a $SCAN table is provided for initialization options) or an Exit 0 routine for this purpose. (See *z/OS JES2 Installation Exits* for more information about these procedures.) Unless these options are specified in the JCL procedure that is used for automatic starting of JES2, you must specify the start options from the console.

Starting Using a JCL Procedure

When the options are specified on the JCL procedure (on the EXEC statement) or by specifying the PARM= parameter on the MVS START command, JES2 suppresses the $HASP426 (SPECIFY OPTIONS) message and completes initialization without operator intervention unless CONSOLE control statements have been added to the JES2 initialization data set or an initialization statement is in error. *z/OS MVS JCL Reference* gives the rules about using the MVS START command. These rules govern the use of the PARM= parameter on the EXEC JCL

statement and apply to how you specify JES2 start options with a JCL procedure. The examples below illustrate both correct and incorrect use of these rules in starting JES2:

```
S JES2,PARM=(FORMAT,SPOOL=VALIDATE)   ...  is incorrect.
S JES2,PARM=(FORMAT,'SPOOL=VALIDATE') ...  is correct.
S JES2,PARM='FORMAT,SPOOL=VALIDATE'   ...  is correct.
S JES2,PARM=(FORMAT,SPOOL='VALIDATE') ...  is incorrect.
```

When the options are specified (from the JCL procedure or on the MVS START command), JES2 does some initial verification, and then issues the $HASP493 message indicating that initialization has started. After it has accepted the options, JES2 reads the specified initialization data set. Message $HASP492 indicates that initialization is complete. JES2 is then ready to start processing jobs. If the NOREQ options is specified, JES2 automatically starts processing. Otherwise, you will receive the following messages:

```
$HASP400 ENTER REQUESTS
```

This message requests the operator to use the $S command to start JES2 processing. JES2 issues the $HASP492 message to indicate whether initialization was performed for all members of the MAS configuration or only for the member indicated by **memname** (the member name). $HASP492 also indicates the type of start performed to start member activity.

Starting Using the Console

If the PARM= parameter is not specified on the MVS START command for JES2, you are choosing to start JES2 by specifying options from the console. JES2 issues message $HASP426:

```
*id $HASP426 SPECIFY OPTIONS - JES jeslevel SSNAME=ssname
```

You should respond by using the MVS REPLY command to specify the JES2 options determined by your installation procedures. (See *z/OS MVS System Commands* for a complete description of the REPLY command.)

```
REPLY id,options
```

For example, if the *id* for a reply is "003", the $HASP426 message is:

```
003 $HASP426 SPECIFY OPTIONS..... message
```

You would reply as follows:

```
REPLY 003,options..........
```

Note:

1. You are only allowed to type one line on the operator console to provide the JES2 options in your reply.
2. The reply to supply options must follow the rules for JES2 initialization statements.

The examples below illustrate correct responses:

```
REPLY 003,a,b,c,d,e
REPLY 003,'a,b,c,d,e'
REPLY 003,a,b,x=144,d,e
```

The option table below, Table 5 on page 30, lists the JES2 start options and an explanation of each. If you respond to message $HASP426 with the $PJES2 command, JES2 will terminate.

If this is the first start of JES2, the following sequence applies:

1. You should specify the COLD (or FORMAT) option. If COLD (or FORMAT) is not specified, JES2 issues the following messages:

   ```
   $HASP479 UNABLE TO OBTAIN CKPT DATA SET LOCK - I/O ERROR
   $HASP454 SHOULD JES2 BYPASS THE MULTI-MEMBER INTEGRITY LOCK?
            (Y OR N)
   ```

   Reply Y to message $HASP454. JES2 will then issue the following messages:

   ```
   $HASP478 INITIAL CHECKPOINT READ IS FROM CKPTn
            (dsname ON volser)
   $HASP434 INVALID CHECKPOINT RECORD ON CKPTn DATASET
   $HASP285 JES2 CHECKPOINT RECONFIGURATION IN PROGRESS
   $HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED.
   $HASP272 ENTER RESPONSE
   ```

   Reply TERM to the $HASP289 and $HASP272 message pair. Then you have to restart JES2 specifying the COLD (or FORMAT) option.
2. All options can be entered in uppercase or lowercase and must be separated by commas.
3. If two options are entered that are considered opposite, for example (WARM,COLD), the last option specified is the one JES uses.
4. If FORMAT is specified, the system will perform a cold start even though WARM is also specified.

The operator then enters the options using the standard reply format.

JES2 must be started and completely initialized before initiators accept work.

JES2 start options

Table 5. JES2 start options

Option	Explanation
FORMAT	FORMAT specifies that JES2 is to format all existing spool volumes. If you add unformatted spool volumes, JES2 automatically formats them whether FORMAT is specified or not. When you specify FORMAT, JES2 will automatically be cold started. **Note:** The FORMAT option is denied if this is a multi-access spool configuration, and JES2 is processing in one or more of the other members.
NOFMT	***Default***: NOFMT specifies that JES2 is not to format existing spool volumes unless JES2 determines that formatting is required.

Table 5. JES2 start options (continued)

Option	Explanation
COLD **WARM**	COLD specifies that JES2 is to be cold started. All jobs in the member are lost. **Note:** 1. The COLD option is denied if this is a multi-access spool configuration and JES2 is processing in one or more of the other members. 2. If the member to be warm started is in a multi-access spool configuration with any other active members, only this member is warm started. 3. When JES2 is cold started, it will by default start with z11 functions activated unless the UNACT parameter is also specified. ***Default***: WARM specifies that JES2 is to be warm started. JES2 will continue processing jobs from where they were stopped. If the FORMAT option was also coded, then JES2 will ignore the WARM specification and will cold start.
NOREQ **REQ**	NOREQ specifies that the $HASP400 (ENTER REQUESTS) message is to be suppressed and that JES2 is to automatically start processing when initialization is complete. ***Default***: REQ specifies that the $HASP400 (ENTER REQUESTS) message is to be written at the console. This message allows the operator to restore the MVS environment (VTAM, for example) to what it was before the preceding shutdown and start JES2 processing with the $S command. This can be overridden with OPTSDEF.
NOLIST **LIST**	NOLIST specifies that JES2 is not to print the contents of the initialization data set or any error flags that occur during initialization. If you specify NOLIST, JES2 ignores any LIST control statements in the initialization data set. The *z/OS JES2 Initialization and Tuning Reference* presents an example of an initialization data set listing produced by using the list option. ***Default***: LIST specifies that JES2 is to print all the statements in the initialization data set and any error flags that occur during initialization. (JES2 prints these statements if a printer is defined for that purpose when JES2 is started.) LIST will not print any statements that follow a NOLIST control statement in the initialization data set. This can be overridden with OPTSDEF.
NOLOG **LOG**	NOLOG specifies that JES2 is not to copy initialization statements or initialization errors to the HARDCPY console. If you specify NOLOG, JES2 ignores LOG control statements in the initialization data set. ***Default***: LOG specifies that JES2 is to honor any LOG statements in the initialization data set. This can be overridden with OPTSDEF.

Table 5. JES2 start options (continued)

Option	Explanation
SPOOL=VALIDATE SPOOL=**NOVALIDATE**	VALIDATE specifies that the track group map is validated on a JES2 all-member warm start in addition to the 7–day, ongoing track group validation cycle. This option when used in conjunction with job queue validation is useful for several special situations that require immediate track group map validation. Refer to the*z/OS JES2 Initialization and Tuning Guide* for a further discussion of using SPOOL=VALIDATE. ***Default***: SPOOL=NOVALIDATE specifies that the track group map is not validated when JES2 restarts (single-member and all-member warm start and hot start).
CKPT1 CKPT2	CKPT1 specifies that JES2 reads the CKPT1 data set as the source of the checkpoint data for building the JES2 work queues.CKPT2 specifies that JES2 reads the CKPT2 data set as the source of checkpoint data for building the JES2 work queues. **Note:** This option is valid only on an all-member warm start. ***Default***: If you do not specify this option, JES2 determines which checkpoint data set to initially read based on information in the checkpoint data set. This can be overridden with OPTSDEF or CKPTDEF.
RECONFIG	RECONFIG specifies that JES2 will use the checkpoint data set definitions as specified on the CKPTDEF statement in the initialization data set. JES2 overrides any modifications to the checkpoint data set definitions previously made either by the $T CKPTDEF command or through the use of the checkpoint reconfiguration dialog. Specifying RECONFIG will also cause JES2 to enter the reconfiguration dialog during initialization; and issue message $HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED. **The CKPTDEF statement definition cannot contain the most current checkpoint definition if you have previously reconfigured your checkpoint configuration through the checkpoint reconfiguration dialog because changes made through the dialog are not saved in the input stream data set.** **Note:** 1. This option is valid only on an all-member warm start. 2. Specifying PARM=RECONFIG has the same affect as specifying RECONFIG=YES on the CKPTDEF initialization statement.

Table 5. JES2 start options (continued)

Option	Explanation
HASPPARM=ddname HASPPARM=**HASPPARM**	HASPPARM=ddname specifies the name of the data definition (DD) statement that defines the data set containing the initialization statements that JES2 is to use for this initialization. ***Default***:HASPPARM=**HASPPARM** specifies that JES2 is to be initialized using the initialization statements in the data set defined by the HASPPARM DD statement in the JES2 procedure. Mutually exclusive with MEMBER=.
MEMBER=membname or PARMLIB_MEMBER= membname MEMBER=**HASjesx**	MEMBER=membname specifies the member of the default PARMLIB concatination containing the initialization statments that JES2 is to use for this initialization. PARMLIB_MEMBER= is a synonym for MEMBER=. MEMBER is mutually exclusive with HASPPARM=. The order of processing is: 1. MEMBER= if specified 2. HASPPARM= if specified 3. default HASPPARM=HASPPARM if DD can be opened 4. default MEMBER=HASjesx member of the default PARMLIB concatination
CONSOLE	Causes JES2 to simulate receiving a CONSOLE initialization statement after all initialization statements are processed. That is, if CONSOLE is specified, JES2 will divert to the operator console for further parameter input after all PARMLIB parameter input is exhausted. (Refer to the CONSOLE initialization control statement, above, for further information.) This can be overridden with OPTSDEF.
NONE U N	NONE, U, or N character specifies that JES2 is to use all the default start options. There is no difference between these three options. These options are equivalent. When NONE, U, or N is specified, JES2 uses the default start options which are: • NOFMT • WARM • REQ • LIST • LOG
UNACT	UNACT unactivates the new JES2 functions introduced in z/OS Version 1 Release 11. This option reverses the ACTIVATE level=z11 setting. It can be used on an all-member WARM or HOT start or a COLD start. ***Default***: On a WARM or HOT start, JES2 takes no action and the checkpoint release level stays at the previous setting. On a COLD start, JES2 starts in z/OS Version 1 Release 11 (z11) mode.
$P JES2	If specified, instructs JES2 to terminate immediately.

Table 5. JES2 start options (continued)

Option	Explanation

Controlling initialization using JES2 exits and the $SCAN facility

There are three exits taken by JES2 during initialization processing:

- Exit 0 for pre-initialization processing
- Exit 19 for initialization statement processing for each initialization statement
- Exit 24 for post-initialization processing

Each assists the JES2 initialization process by providing greater access to the initialization data set and increased checking and modifying capabilities.

If JES2 ends processing, Exit 26 can be taken for termination processing.

In addition, your installation can define new initialization statements, new parameters on existing statements, and JES2 options through the JES2 scanning facility. This facility scans input (JES2 initialization, control statements, and some operator commands), parses and validates the input, and sets or displays particular fields.

At each level of a scan, JES2 defines a table, and you can define an alternate table to override the JES2 table definition. These tables are used to determine the validity and content of each initialization statement scanned. The installation-defined table is used rather than the JES2 table. Thus, your installation can define your own statements, add parameters to JES2 statements, override the table definition for JES2 statements, or add messages. See *z/OS JES2 Installation Exits* for information concerning these exits and the scanning facility.

Controlling JES2 load modules

The following describes the structure of JES2 load modules and how they can be loaded onto your installation.

JES2 load modules

JES2 is comprised of three main load modules and a collection of subsystem support modules. In Table 6, the three mainload modules appear first, followed by the subsystem support modules.

Table 6. Subsystem Support Modules

Module	Purpose	Storage Location
HASCARMS	Automatic restart management SSI support.	common
HASCARSO	Support for automatic restart management services.	common
HASCDAU	Supports the allocation of multiple spin $IOTs to support multiple output characteristics.	common
HASCDSAL	Supports the allocation and deallocation of subsystem data sets.	common
HASCDSOC	Supports opening and closing subsystem data sets.	common

Table 6. Subsystem Support Modules (continued)

Module	Purpose	Storage Location
HASCDSS	Supports any data spaces JES2 creates.	common
HASCENF	Event Notification Facility (ENF) LISTEN Exit	common
HASCGGKY	Supports generic grouping keys.	common
HASCGGST	Supports generic grouping strings.	common
HASCHAM	Contains the JES2 access method support.	common
HASCJBST	Supports job selection and job end processing.	common
HASCJBTR	Supports end-of-memory and end-of-task processing.	common
HASCLINK	Supports linkage between JES2 modules.	common
HASCOFST	Ensures the addressability of installation- or JES2-modified code.	common
HASCPHAM	Contains the JES2 access method support.	common
HASCPOOL	Supports the JES2 cell pool services.	common
HASCRQUE	Request queue services.	common
HASCSAPI	Supports SYSOUT API (SAPI) requests.	common
HASCSIJD	Provides SSI support module.	common
HASCSIJI	Supports checkpoint versioning.	common
HASCSIJP	Supports JES properties subsystem interface (SSI) functions.	common
HASCSIRQ	Supports miscellaneous subsystem interface (SSI) functions such as write-to-operator, commands, and process SYSOUT.	common
HASCSJFA	Updates the $SJB with the output limits and the account number from the SWB.	common
HASCSJFS	Supplies SJF-related services.	common
HASCSRAX	JES2 AUX address space services.	common
HASCSRDS	Provides subsystem data set services.	common
HASCSRIC	Provides SSI control services.	common
HASCSRJB	Provides job-related services.	common
HASCTP	Provides APPC/MVS transaction program support.	common
HASCUBSR	Supports unwritten buffer processing.	common
HASCXJCT	Support for extensions to JCT.	common
HASCBLDM	Message building support and templates.	common
HASCINJR	Support for internal readers.	common
HASCNJAS	NETSRV address space creation support.	common
HASCNJE	NJE service routines.	common
HASCNJEX	NETSRV interface to IAZ services.	common
HASCNJGP	NETSRV address space general purpose subtasks.	common
HASCNJJR	NJE/TCP job receiver support.	common
HASCNJJT	NJE/TCP job transmitter support.	common
HASCNJRC	NETSRV recovery support.	common

Table 6. Subsystem Support Modules (continued)

Module	Purpose	Storage Location
HASCNJRQ	NETSRV interfaces to JES2 address space.	common
HASCNJSR	NJE/TCP SYSOUT receiver support.	common
HASCNJST	NJE/TCP SYSOUT transmitter support.	common
HASCSCAN	JES2 $SCAN support services.	common
HASCSRIP	Input processing service routines.	common
HASJES20	The main task module.	JES2 private
HASJ2MON	JES2 health monitor services.	JES2 monitor, JES2 private
HASPFSSM	The functional subsystem service module.	FSS private
HASPINIT	The initialization modules.	JES2 private

HASJES20 resides in the JES2 address space, HASPINIT resides in the JES2 address space during initialization processing, and HASPFSSM resides in the functional subsystem address space. HASJES20 is a split load module. Part of the module resides below the 16 megabytes in JES2 private storage, including the main entry point, and the remainder resides above the 16 megabytes in JES2 private storage.

The JES2 subsystem support modules reside in either the common service area (CSA, if the modules are found in the STEPLIB or LINKLST data set) or the pageable link pack area (LPA, if the modules are not found in the STEPLIB or LINKLST data set).

Loading JES2 subsystem support modules

You can control the area of storage into which JES2 loads these modules by placing copies of them in either the LNKLST or STEPLIB data set before you start JES2. Modules loaded into LPA are read-only; they cannot be replaced. The copy JES2 uses depends on where JES2 finds these modules. This gives you the flexibility to use copies that reside in CSA instead of PLPA. Using copies that reside in CSA eases migration and testing when your installation moves to a new level of JES2. If loaded into CSA, you do not need to IPL your member to obtain the new load modules following a JES2 restart.

Note: A hot start does not change the copies of subsystem support modules you are using.

See the STORAGE=CSA column of Table 7 on page 37 for details about which copy JES2 will use.

JES2 subsystem module placement and storage considerations

All JES2 subsystem support modules normally reside above 16 megabytes in storage. The only exception is HASCHAM, the load module containing the JES2 access method. HASCHAM must support direct calls from applications running AMODE 24. All JES2 subsystem support modules run in 31-bit mode addressing at all times, except for a small part of HASCHAM. MVS facilities and applications that invoke JES2 services through the subsystem interface (SSI) can run in either 24-bit or 31-bit addressing mode. The SSI bridges to the 31-bit JES2 subsystem modules.

Controlling the loading of installation-defined load modules

Use the LOADmod(jxxxxxxx) initialization statement to direct the loading of all installation-defined load modules (such as installation-defined exits). Exit routines should be loaded in this manner, rather than linking to JES2 load modules.

JES2 only searches for installation-defined exit routines in:

- User modules defined by the LOADmod(jxxxxxxx) statement
- HASPXIT0
- Modules with reserved names in the form HASPXJxx.

Attention: JES2 does not search for such routines in IBM-defined modules.

The STORAGE= parameter specifies the area of storage in which the load module is to be loaded. Table 7 presents a summary of the manner in which JES2 directs the load of a load module based on initial placement of that load module and the LOADmod(jxxxxxxx) STORAGE= specification.

Table 7. Placement of Load Modules Based on LOADmod(jxxxxxxx) STORAGE= Specification. LOADmod(x) is the first row header and spans all four table columns. The first table cell in the first table row is a row heading.

LOADmod(x)			
Module is Located In:	STORAGE=PVT	STORAGE=CSA	STORAGE=LPA
STEPLIB Only	PVT	CSA	$HASP003 RC=31
LPA Only	LPA	LPA	LPA
LNKLST Only	PVT	CSA	$HASP003 RC=31
STEPLIB and LPA	PVT (STEPLIB)	CSA (STEPLIB)	LPA version
STEPLIB and LNKLST	PVT (STEPLIB)	CSA (STEPLIB)	$HASP003 RC=31
LPA and LNKLST	LPA	LPA	LPA
STEPLIB, LPA and LNKLST	PVT (STEPLIB)	CSA (STEPLIB)	LPA version

Note the following restrictions:

- **You cannot use the LOADmod(jxxxxxxx) statement to direct the load of any IBM-supplied JES2 modules**. If you specify any IBM-supplied JES2 modules on this statement, JES2 issues the following message:

  ```
  $HASP003 RC=52, NO SELECTABLE ENTRIES FOUND MATCHING SPECIFICATIONS
  ```

  The placement of these modules depends on which data set contains the modules.(See the STORAGE=CSA column in Table 7 for the placement of the modules.)

- STORAGE=LPA is not valid if the load module is initially placed in STEPLIB only, LINKLST only, or both STEPLIB and LINKLST. JES2 issues message $HASP003 RC=31, MODULE COULD NOT BE LOADED.
- All other STORAGE= requests are valid, but you cannot receive the expected result (see Table 7 on page 37).
- You cannot load a module into the link pack area (LPA) following MVS initialization; you can only direct JES2 to use the LPA version.

Starting and stopping JES2

MVS and JES2 initialization are two distinct processes.This means you specify the processes associated with initialization of MVS (reloading the LPA, clearing the VIO data sets) and of JES2 (initializing the job queues) separately. You can start both MVS and JES2 through either a cold start or some type of warm start. For more information on start types, see "Restarting JES2" on page 43. Although for any given IPL, it is possible to cold start one (MVS or JES2) and warm start the other, the usual procedure is a warm start for both.

JES2 performs an initialization the first time it is started (as part of the IPL procedure) and every time it is restarted. In a multi-access spool (MAS) configuration, you must start JES2 in each member in the configuration. The following provides information on starting and stopping JES2:

- "JES2 program properties."

  This topic describes the JES2 entry for the program properties table (PPT). The PPT allows an installation to specify a list of programs that require special attributes to run as efficiently and securely as possible.
- "Starting JES2 for the first time (a cold start)" on page 39.

  This topic describes the procedure for starting a JES2 member on a cold start (for the first time). It provides a record of JES2 message displays and the accompanying operator responses.
- "Performing an orderly shutdown of a JES2 member" on page 41.

  This topic describes how you can bring down a JES2 member, providing a record of JES2 message displays and the accompanying operator responses.
- "SMF record summary: starting and stopping JES2" on page 42.

  This topic presents the SMF records generated whenever a JES2 member is started or stopped.
- "Using event notification signalling with JES2" on page 42.

  This topic describes how authorized programs can determine when JES2 starts or stops.

JES2 program properties

The JES2 entry of the program properties table (PPT) should indicate that JES2 is nonswappable because JES2 performs long-term page fixes. The program properties table should also indicate that JES2 is a started task and is exempt from data set name enqueue and cancellation. Further, to ensure appropriate security checks for data sets opened by JES2 on behalf of user-submitted jobs, the PPT entry for JES2 **must** include password protection (PASS). The entry for JES2 in the IBM-supplied program properties table looks like this:

```
PPT PGMNAME(HASJES20) NOCANCEL NOSWAP NODSI PASS SYST KEY(1) LPREF
```

For information about defining the JES2 entry in the program properties table, see *z/OS MVS Initialization and Tuning Reference*.

Starting JES2 for the first time (a cold start)

To start JES2 for the first time (a cold start) choose one of the following three methods:

1. Define your primary subsystem with the PRIMARY keyword in member IEFSSNxx on SYS1.PARMLIB.

   The IEFSSN00 member defines JES2 as the primary subsystem and does not specify the NOSTART option. This causes JES2 to start automatically.

   Because the IEFSSNxx member does not allow installations to pass parameters, you cannot use symbolic JCL if the START command is in the IEFSSNxx member of SYS1.PARMLIB. (See *z/OS MVS Initialization and Tuning Reference* for more information about using this parmlib member.)
2. Specify the MVS START command in the COMMNDxx member of SYS1.PARMLIB.
3. Enter the **S JES2** command through an operator command:
   In this example, the procedure that invokes the JES2 member is JES2.

```
S  JES2,PARM=(COLD),M=initialization deck,N=parmlib member
```

Regardless of the method you use, specifying the JES2 start option LOG=NO decreases JES2 initialization time. The default for this JES2 start option, LOG=YES, specifies that JES2 must write all initialization statements and initialization errors to the HARDCPY console and the SYSLOG. You should only use the LOG=YES start option when testing a new initialization stream.

JES2 console messages during initialization

As soon as JES2 is started, the SPECIFY OPTIONS message is issued to the operator. The operator then specifies the COLD parameter (or FORMAT) option.

A cold start results in the initialization of all JES2 job and output queues. During the spool processing for cold start, JES2 examines all spool volumes that are online to determine if each is formatted; any unformatted spool volume is formatted at this time.

If COLD (or FORMAT) is not specified, JES2 will issue the following series of messages:

```
$HASP479 UNABLE TO OBTAIN CKPT DATA SET LOCK
$HASP454 SHOULD JES2 BYPASS THE MULTI-MEMBER
INTEGRITY LOCK?
```

Reply 'Y' to this message, and then JES2 issues the following messages:
Reply 'TERM' to the $HASP289/272 message pair. You will then have to restart

```
$HASP478 CONFIRM INITIAL CHECKPOINT READ FROM CKPTn
$HASP434 INVALID CHECKPOINT RECORD ON CKPTn DATASET
$HASP285 JES2 CHECKPOINT RECONFIGURATION IN PROGRESS
$HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED
$HASP272 ENTER RESPONSE
```

JES2. Be certain to specify COLD (or FORMAT) when you again receive the $HASP426 SPECIFY OPTIONS message.

If any of your checkpoint data sets reside on Coupling facility structures rather than on DASD, JES2 issues the following message to verify the XCF connection, the JES2 XCF group name for your multi-access spool (MAS), and the name of the JES2 member that belongs to the JES2 XCF group:

```
 IXZ0001I CONNECTION TO JESXCF COMPONENT ESTABLISHED,
          GROUP WSC      MEMBER WSC$SY01
```

For more information about how JES2 XCF groups are defined, see "Defining JES2 to the cross-system coupling facility (XCF)" on page 8.

Regardless of where your checkpoint data sets reside, JES2 provides an informational message to display the size of the checkpoint that was determined from your initialization data set values. After receiving the checkpoint data set information, you must confirm that initialization should continue by responding 'Y' to the $HASP441 message. For example:

```
$HASP537 THE CURRENT CHECKPOINT USES 43 4K RECORDS
$HASP436 CONFIRM COLD START ON
 CKPT1 - STRNAME=J2WRKCKPT1
 CKPT2 - STRNAME=J2WRKCKPT2
 SPOOL - PREFIX=J2WRK  DSN=SYS2.HASPACE
 03 $HASP441 REPLY 'Y' TO CONTINUE INITIALIZATION OR 'N'
          TO TERMINATE IN RESPONSE
 $HASP478 INITIAL CHECKPOINT READ IS FROM CKPTn
          (STRNAME J2WRKCKPT1).
```

After you verify that JES2 should continue initialization, the member issues the following messages to start and format the JES2 address space:

```
$HASP493 JES2 COLD START IS IN PROGRESS
$HASP266 JES2 CKPT2 DATA SET IS BEING FORMATTED
$HASP267 JES2 CKPT2 DATA SET HAS BEEN SUCCESSFULLY FORMATTED
$HASP266 JES2 CKPT1 DATA SET IS BEING FORMATTED
$HASP267 JES2 CKPT1 DATA SET HAS BEEN SUCCESSFULLY FORMATTED
$HASP492 JES2 COLD START HAS COMPLETED
```

When initialization is complete, JES2 deletes the $HASP493 message and issues the $HASP492 COLD START HAS COMPLETED message.

When JES2 completes initialization, it either

- Begins to process jobs automatically after the $HASP492 message

  or
- Waits for the operator to enter the $S command in response to the $HASP400 ENTER REQUESTS message.

JES2 issues the $HASP400 ENTER REQUESTS message if the PARM= keyword has been omitted from the S JES2 command and EXEC statement of the JES2 procedure.

A problem can occur if a JES2 spool volume (specified by the VOLUME= parameter on the SPOOLDEF initialization statement) is not mounted and ready when the primary JES2 member is started. At least one spool volume must be mounted because JES2 must have spool space available in order to process an MVS MOUNT command. In this situation, the operator must ready a spool volume and IPL again.

To be certain you have at least one spool volume available at all times, include an entry in the VATLSTxx parameter library member specifying at least one spool volume without suppressing the mount option. VATLST processing will then request that a volume be mounted during the IPL process. An example would be:

```
SPOOL1,0,2,3380
```

See *z/OS MVS Initialization and Tuning Reference* for more information about VATLSTxx member.

Performing an orderly shutdown of a JES2 member

You can stop and restart JES2 in a system at any time by using operator commands. This capability allows you to:

- Quiesce job processing in preparation for an orderly system shutdown
- Restart JES2 to perform an initialization with different initialization parameter specifications.

The $DJES2 command lists all activity that would cause a $PJES2 command to be rejected. When all activity has been successfully quiesced, the response to the $DJES2 command will be $HASP608 ALL AVAILABLE FUNCTIONS COMPLETE. Then you can enter a $PJES2 command successfully. In each case, first enter the $P command to drain the JES2 queues. (Note that all printers supported by a functional subsystem need to be drained before entering the $P JES2 command). When all TSO/E users log off and all JES2 started tasks, logical initiators, printers, and punches complete their current activities and become inactive, JES2 notifies you with the following message:

```
$HASP099  ALL  AVAILABLE  FUNCTIONS  COMPLETE
```

You can then stop all started tasks and enter the $P JES2 command, which stops JES2 and removes it from the MVS system.

If an installation is running partially disabled because a PCE (processor controller element) has failed, you must enter the $P JES2,ABEND commandbecause the member cannot end without an abnormal termination. JES2 informs an installation if it is running with a disabled processor through the $HASP068 message, which identifies the processor that has failed. For more information about JES2 PCE recovery options see “Running with disabled JES2 processors.”

When you are halting the system at the end of the day or the end of the work shift, you can enter the MVS HALT EOD command to bring down the MVS system. If you halt the MVS system, see “Restarting JES2” on page 43 for instructions on how to restart the JES2 member.

Running with disabled JES2 processors

Processor control elements (PCEs) represent dispatchable work in JES2; there are PCEs for devices such as printers, lines, transmitters, and readers, and for functions such as conversion, execution, output, and purge.

JES2 recovery processing deactivates non-essential PCEs in the event of a failure. This procedure allows JES2 to continue processing work with all but the failing device or function. The installation can then decide when to restart JES2 and control the number of unexpected JES2 outages. Non-essential PCEs are not critical to the operation of the JES2 subsystem. However, system programmers must determine the relative importance of a particular PCE to their installation. See the complete list of JES2 PCEs in Appendix A in *z/OS JES2 Messages*.

Do not continue processing work for a long time without a PCE because JES2 might lose some function. For example, if a spool processor fails, an installation

could not issue commands to start, halt, or drain a spool volume. If a printer PCE fails, an installation would not be able to print output owned by that processor on another printer.

If a printer PCE fails while a SYSOUT transmitter is in the process of transmitting a large job across the network, an installation can run without the printer PCE until the SYSOUT transmitter finishes transmitting the data. When the SYSOUT transmitter has finished, an operator should bring JES2 down by entering a $P JES2,ABEND command, then hot start the subsystem to reacquire the failed PCE.

When a PCE fails:

- JES2 attempts normal recovery through $ESTAE processing. If $ESTAE recovery processing is successful, JES2 processing continues.
- If the PCE is not recoverable and is essential to the proper operation of JES2, then JES2 will end.
- If the $ESTAE routine cannot recover the PCE, and the device or function that the PCE represents is not essential to JES2, the PCE will never be given control again; JES2 will not attempt to recover the resources associated with the lost PCE and issues the $HASP070 message.

Note: The $HASP070 message occurs only if the number of ABENDs has exceeded the threshold set by the RECVOPTS(MAIN) initialization statement.

You can then decide if and when the lost PCE warrants a restart of JES2. See *z/OS JES2 Messages* for information about $HASP095 and $HASP068.

SMF record summary: starting and stopping JES2

The following lists the SMF records written when a JES2 subsystem is started and stopped. For a complete list of the record contents, see *z/OS MVS System Management Facilities (SMF)*.

Table 8. SMF Records Used When JES2 Starts or Stops a Member

Record Number	Action that Causes the Record to be Written
43	• An operator enters an S JES2 command. • An operator enters a $E MEM command.
45	• An operator brings down the JES2 address space ($P JES2 command). • JES2 ends abnormally and retains control long enough to write the record.

Using event notification signalling with JES2

Authorized programs can determine when JES2 is started by listening for ENF (event notification facility) signal 40. Any application program that relies on the JES2 member can tell when a primary or secondary JES2 begins or ends normally.

Authorized programs can determine when JES2 is ending normally by listening for ENF signal 40. ENF signal listeners are not notified if JES2 ends abnormally. For more information about how to specify the ENF signal, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Restarting JES2

You can restart JES2 by either selecting a cold start or a type of warm start. This selection is made when the operator responds to the SPECIFY OPTION request with either COLD or WARM. The restart type you select depends on the current environment within a single processor or multi-access spool (MAS) configuration. See "Modifying JES2 initialization statement specifications" on page 51 for notes on when a particular type of restart is required.

Cold start

On a restart of a JES2 member, you should implement a cold start with some caution, because all job data previously on the spool volumes is lost. No other member in a multi-access spool configuration can be active during a cold start, or JES2 will end with an error message.

If you request a cold start of JES2 or have specified the FORMAT start option, an IPL must precede the execution of the MVS START(S JES2) command. However, an IPL is not required if JES2 was stopped by a $P JES2 operator command.

Warm starts

There are five ways in which a warm start can be performed in a multi-access spool configuration. The following explains how you can use these options and the processing that JES2 performs.

All-member warm start

An all-member warm start is performed if a warm start is specified by the operator and JES2 determines that no other members of the configuration are active or there is only one member in the configuration. All in-process work in the MAS will be recovered. After an all-member warm start, other members entering the configuration for the first time will perform a quick start.

This is the only time that certain activities are performed within the MAS, because no other processors are currently using the checkpoint data set. These activities include:

- Rebuilding the track allocation status (track group map) if you specify SPOOL=VALIDATE as a start option.
- Validating (and rebuilding when necessary) the job and output queues and the job queue index. Damaged or corrupted job output elements (JOEs) and job queue elements (JQEs) are placed on the rebuild queue.
- Deleting spool volumes not previously removed through the $P or $Z commands that are no longer available to the member. See "Starting the multi-access SPOOL configuration" on page 60 for starting JES2 in a multi-access spool configuration.

Note: You can add members of a multi-access spool configuration at any JES2 warm start.

Single-member warm start

A single-member warm start is performed when WARM is specified and other members of the configuration are active. The warm-starting member joins the active configuration and recovers only work in process on that member when it failed or was stopped.

If JES2 experiences an abnormal termination (ABEND), and you want to request a JES2 warm start:

1. IPL the system.
2. Issue the MVS START JES2 COMMAND.
3. When a request for options (SPECIFY OPTIONS) is received at the console, respond with the keyword WARM.
4. In a multi-access spool configuration, all spool volumes that are online to the multi-access spool configuration must also be online to the member being started.

When warm starting, JES2:

- Only requeues jobs that were previously active on this processor.
- Validates and rebuilds when necessary the job and output queues and the job queue index. Damaged or corrupted job output elements (JOEs) and job queue elements (JQEs) are placed on the rebuild queue.

Quick start

You can perform a quick start when you respond with the keyword WARM to the request for options ($HASP426 SPECIFY OPTIONS) at the console and JES2 determines that the job queue and job output table do not need to be updated. In this case, the member being started is **not** the first member being started in the MAS.This can occur in these circumstances:

- After $P JES2 has been issued to quiesce the member. Because all work is quiesced, there is no need to update the job queue or job output table before restarting.
- After an all-member warm start has been performed and no work is waiting to be processed; therefore, the job and output queues are empty.
- A $E MEM command was entered at a processor within the multi-access spool configuration other than the member being started. This gives the issuing member the ability to:
  - Process any jobs that had a specific affinity for the member being reset.
  - Process any output for the reset member.
  - Restart any jobs that were interrupted on the reset member.
  - Dynamically add a new member to a multi-access spool configuration. A new member is defined as a member that has a NAME (on the MEMBER statement) that has not been previously defined to the MAS.

  Thus, there is no need to update the job queues or job output table on the reset member.

On a quick start, as on all other start types, JES2 validates (and rebuilds when necessary the job and output queues and the job queue index. Damaged or corrupted job output elements (JOEs) and job queue elements (JQEs) are placed on the rebuild queue.

Hot start

A hot start is performed when JES2 abnormally terminates, WARM is specified, and JES2 determines that an IPL has not occurred before restart.

You can use a hot start when JES2 has stopped, but other members have continued to function and have not experienced problems. When you hot start JES2, all address spaces continue to execute as if JES2 had never terminated. Functional subsystems will continue to process spooled output if possible before the hot start,

and JES2 will reconnect to the functional subsystem (FSS) during the hot start. Hot starts do not affect other members in a multi-access spool configuration.

JES2 validates (and rebuilds when necessary) the job and output queues and the job queue index. Damaged or corrupted job output elements (JOEs) and job queue elements (JQEs) are placed on the rebuild queue.

Also, many initialization statements and parameter specifications are ignored on a hot start of JES2. (*z/OS JES2 Initialization and Tuning Reference* notes these parameters.)

$E MEM command start

In addition to the above methods of restarting JES2, a restart of work on the failed member can be performed on any other active member in the configuration by the AUTOEMEM facility or by entering the $E MEM command. In-process work on the specified member is recovered and made available for selection by other members of the configuration, subject to system affinity for execution restart.

Table 9 shows which type of warm start is appropriate, depending on

- How the JES2 member was shut down
- Whether other JES2 members are active.

Table 9. Warm Start Types

Warm Start Type	Other Members Active	Circumstances for Doing a JES2 Warm Start
All Members*	No	IPL or $P JES2
Single member	Yes	Abend followed by MVS IPL
Quick	Yes	For any of the following: • $P JES2 (with or without IPL) • After first member in an all-member warm start • After a $E MEMBER command or the AUTOEMEM option performed from another member of the MAS • Dynamically add a new member • When adding an already defined member to a MAS
Hot	N/A	Abend without MVS IPL
Note: If there are no other members active in the MAS, JES2 performs an all member warm start.		

Warm start considerations

If HARDCPY=SYSLOG (in the IEASYSxx parameter list) is specified and MVS system WTO buffers in use exceeds the limit specified in the CONSOLxx MVS statement, JES2 will be placed in a wait until the buffers in use again falls below that limit.

During a warm-start initialization, JES2 reads through its job queues and handles each job according to its status:

1. Jobs in input readers are lost and must be reentered.
2. Jobs in output (print/punch) are requeued and later restarted.

The checkpoint for jobs on the 3800 printer points to the end of the page being stacked at the time of the checkpoint. Jobs that were sent to the 3800 printer but that did not reach the stacker are reprinted from the checkpoint. If no checkpoint exists, then it is reprinted from the beginning.

3. Jobs in execution are either requeued for execution processing or are queued for output processing.
4. All other jobs remain on their current queues.

If a job in execution was journaled, it is updated to indicate warm start, and the job is queued for re-execution. If a job in execution has no journal, it is tested to determine whether restart was indicated for the job. If *restart* was indicated, the job is updated to remove any warm-start indications, and the job is queued for re-execution. If restart was not indicated, the job is queued for output processing.

Following JES2 warm-start processing, jobs queued for re-execution are again selected by initiators.

- A nonjournaled job that was requeued because RESTART was indicated is handled as a new job.
- A journaled job that indicates warm-start processing receives special attention from an initiator.

The initiator/terminator purges existing job tables and messages for the job and then checks whether the job is authorized for warm-start processing:

- The job must have a valid restart definition (RD=) code in the RD parameter on its JOB statement.
- The operator must authorize the job to be restarted. (Each eligible job is presented to the operator asking whether the job is to be restarted.)

If either of these two authorizations is not made, the job is tested to determine whether RESTART was indicated. If it was, the job's JES2 job control table control block is updated to remove any warm-start indications, and the job is queued for re-execution. If restart was not indicated, the job is queued for output processing.

Jobs can be presented to the JES2 member for warm start at any time and in any order. This allows important jobs and member functions to be executed ahead of less important jobs scheduled for execution or termination. New jobs can be presented to the JES2 member and processed according to their priority, ahead of lower-priority warm-start jobs.

During warm-start initialization, JES2 lists the activity in process for each job at the time JES2 was stopped. Then, when the ENTER REQUESTS message is issued (unless the NOREQ start option was specified), the operator can enter requests to modify or delete each activity. This message also allows the operator to examine the status of output devices (such as UCS and FCB settings) in order to:

1. Determine what action should be taken before restarting the output process
2. Change the status of logical initiators and devices
3. Modify the status of jobs on the JES2 queues.

The status and activity of each device will revert to the specifications of the initialization parameters. If you specify NOREQ, JES2 begins processing each job according to the specifications of the initialization parameters as soon as resources to process each job become available.

Warm Starting with multiple JES2 levels in a MAS: JES2 supports up to three consecutive releases of JES2 running in the same MAS. JES2 enforces this rule and ships service to down level releases to allow them to exist with the newer releases. Failure to apply this maintenance or trying to go outside the supported release will result in a $HASP710 message being issued to indicate the problem. $HASP710 indicates active members in the MAS are incompatible with the initializing member. See the *z/OS JES2 Messages* book for more information.

JES2 also supports a more liberal migration policy. This migration policy covers going from one release of JES2 to a new release with a warm start. IBM guarantees that this migration policy will span a *minimum* of 5 releases of the operating systems. That is, the data generated by the JES2 provided with release x of the operating system is compatible with the data generated by the release of JES2 provided with release x-5 and release x+5 of the operating system. This may include a release that cannot coexist in a MAS. The requirements for migration are documented in the *z/OS Migration* manual for each release. Failure to abide by the migration requirements can result in a $HASP446 message being issued. For more information about $HASP446 see *z/OS JES2 Messages*.

Note: IBM supports data compatibility for only supported releases. Data compatibility is not guaranteed with unsupported releases.

Verifying Checkpoint Time Stamps: During warm start processing, JES2 performs some checks on the last written time stamps of the checkpoint data sets it is going to use. These tests try to detect cases where JES2 is reading from the wrong checkpoint data set. Ultimately it is the installations responsibility to ensure that the correct data sets are used to start JES2. However, some common problems are detected by JES2. Normally, the $HASP478 message is not highlighted and is issued for informational purposes only. However, if JES2 detects a potential problem with the last written times of the checkpoint data set(s), then the message is highlighted and followed by the $HASP441 WTOR. When this happens, the installation should verify that the time stamps associated with the data set compared with the time the MAS was last active and would have last written the checkpoint. If not, then reply 'N' to the $HASP441 message to terminate in response to $HASP478.

Rebuilding JES2 job queues: Errors that are not detected and corrected by other portions of JES2 can damage the JES2 job or output queues. These errors can occur in both software and hardware. You can correct these errors through any type of warm start. Because the original in-storage order of jobs might change as a result of the job queue rebuild, JES2 performs the queue rebuild only with operator permission. To indicate a job queue error and request operator instruction, JES2 issues $HASP483 JES2 JOB QUEUE ERROR -- REPLY ‘Y’ TO REBUILD OR ‘N’ to TERMINATE JES2.

When job queues are rebuilt, JES2 invokes the REQ initialization option, allowing jobs to be requeued or canceled at your discretion.

The success of the queue rebuild procedure depends upon how badly the queues are damaged. If there is insufficient information to permit rebuilding the queues for a job, that job is purged and must be resubmitted for processing.

When you warm start JES2, JES2 verifies data that includes the following:

- For JQEs:
  - Job number is non-zero and not used by any other job

  - Valid offset to next JQE
- For all JOEs:
  - The JOE type compared to the queue type
- For work JOEs:
  - The class type
  - The offset of the job queue element
  - The offset of the characteristics JOE.

If JES2 detects that the JQE or JOE has been corrupted and JES2 cannot correct the problem, JES2 places that JQE or JOE on the **rebuild queue** until it can be moved to the free queue. (To be moved to the free queue, a JQE must not be locked, busy, or retain JOEs.) JES2 moves the JQE or JOE when the member is again restarted, at the next all-member warm start, or when that JQE or JOE is no longer busy (that is, when this phase of job processing ends).

If in response to a $D JOBDEF or $D OUTDEF command, JES2 indicates that there are jobs or JOEs, respectively, on the rebuild queues, use the $D REBLD command to display all jobs and output elements on that rebuild queue. This might prove useful when trying to locate jobs that appear lost; a situation that might occur if in response to a $D J, $L J, or a $D OJ command JES2 does not find all the jobs you expected.

Typically, the rebuild queue is empty, but if JES2 places a job or JOE on the rebuild queue, it provides diagnostic information through messages $HASP440 and $HASP517 when it completes rebuilding the JOE queue and JQE queue, respectively. These messages also include the number of elements placed on the rebuild queue.

Validation of JES2 track group map: On an all-member warm start, you can use the SPOOL=VALIDATE start option to request that JES2 validate the track group map. This is typically not needed unless you receive indications from JES2 such as the following:

- Persistent JES2 disastrous errors ($HASP095, error code $DIS)
- LOGREC symptoms records (SYMRECs) that point to track group allocation or purge problems.

  Not all SYMRECs warrant your use of SPOOL=VALIDATE. Some of the symptom records are considered informational only. See *z/OS JES2 Diagnosis* for a list of the SYMRECs and an indication of their severity.

Immediate spool validation can then be useful as an immediate validation of the track group map in place of the ongoing track group validation cycle wherein all track groups are validated once every 7 days.

If JES2 rebuilds the job queues, JES2 forces track group validation on the next all-member warm start. If that rebuild occurs during an all-member warm start, the track group validation occurs during that warm start; it does not wait until the next all-member warm start.

Restarting JES2 with missing SPOOL volumes: During all-member warm-start processing, the operator is notified of any spool volumes not mounted, and is requested to provide information concerning their status.If a spool volume cannot be recovered, the operator's response (PURGE) will force the purging of jobs and SYSOUT data sets allocated to the volume. If the operator indicates that the volume is only temporarily unavailable (GO), no jobs allocated to that volume will

be processed until the volume is made available through a $S SPL command. If the required volumes have not been mounted at this point, the operator can end (QUIT) JES2 initialization.

Note: If JES2 is interrupted during spool validation (through either an abnormal end or a re-IPL), JES2 does not provide the GO option to the operator.

For information on the relationship between restarting and use of the checkpoint data set, see "Checkpoint reconfiguration: An overview" on page 222.

Hot start considerations

Initialization statements: A hot start will cause certain initialization statements and initialization statement parameters to be ignored. For the complete set of ignored parameters, consult the parameter description for each JES2 initialization statement in *z/OS JES2 Initialization and Tuning Reference*.

Exit facility: If JES2 is hot starting, it is possible that some jobs have not finished execution; these jobs could still be calling installation exit routines from one of the JES2 subsystem support load modules. Therefore, the names, number, and order of the exit routines for each exit are required to remain unchanged, and all EXIT(nnn) initialization statements encountered during a hot start are ignored.

You can change the exit routine addresses through the LOADmod(jxxxxxxx) statement, with the restriction that routines previously loaded into the private area remain there and those in commonly addressable storage (PLPA or CSA) remain there.

The only exit facility initialization processing that takes place during a hot start is the resolution of exit routine addresses from installation load modules reloaded into the private area.

Functional subsystem reconnection: Functional subsystem reconnection during a hot start requires considerations similar to those of exits. Because some jobs might have not finished printing, these jobs might still require the support of previously-defined functional subsystems. Therefore, the names and number of any defined functional subsystems must not change across a hot start; if they are changed, those changes are ignored.

If a previously started functional subsystem printer successfully reconnects to JES2 during a hot start, the settings for the following PRT(nnnn) initialization statementparameters are retained across the hot start and the values specified during initialization are ignored:

- BURST=
- COPYMARK=
- FCB|C=
- FLASH|O=
- Forms=
- MARK=
- PRESELCT=
- SEPDS=
- SEP=
- SETUP=
- TRace=

- TRKCELL=
- UCS|T
- UNIT=

JES2 honors all other PRT(nnnn) initialization statement parameters during a hot start.

Restarting JES2 after an orderly shutdown

If you do not halt the MVS system after stopping JES2, JES2 can be restarted with the S JES2 command. As part of this command, you can specify your start options as parameters; for example:

```
S  JES2,PARM='WARM,HASPPARM=PARMRJE,NOREQ'
```

When you specify the options on the START command in this manner, JES2 suppresses the $HASP426 SPECIFY OPTIONS message, just as it does when you specify the options as parameters on the EXEC statement within the JES2 procedure.

You can also use the START command to specify a printer address for the HASPLIST DD statement in the JES2 procedure; for example:

```
S JES2,00E
```

If you specify both a printer address and initialization options, the printer address must be specified first:

```
S  JES2,00E,PARM='WARM'
```

Restarting JES2 after a system failure

JES2 is restarted automatically in an MVS system whenever that system IPL is repeated following a system failure. The WARM start option allows you to warm start JES2 and continue job processing from the point of the last checkpoint before the member failure.

Specifying job journaling to ease restart processing

The job journal is a temporary sequential data set that resides on a JES2 spool volume.It preserves a set of selected job-related control blocks that are necessary for restart processing for the following situations: automatic step (using the RD= parameter on the JCL JOB statement), automatic checkpoint, continue, and JES2 member warm starts.

The job journal is necessary because the scheduler control blocks, maintained in the scheduler work area (SWA), are otherwise lost when a job abnormally ends. Reconstruction of the SWA is possible because the job journal preserves up-to-date copies of essential control blocks.

Unless you specify no job journal by defining JOURNAL=NO on the JOBCLASS initialization statement, each job is provided with a job journal. If you are using checkpoint restart or need to restart a job step, you need a job journal or automatic restarting will not be possible (and jobs that are executing during member failure need to be resubmitted).

However, if you are using automatic restart management to restart a job, you should use non-journaled classes.

Note that job journaling can cause some performance degradation. For further information regarding the JOURNAL parameter, see the JOBCLASS initialization statements in *z/OS JES2 Initialization and Tuning Reference*.

Modifying JES2 initialization statement specifications

If you are restarting JES2 for the purpose of changing initializationstatements or parameter values, a cold start is only required for a limited number of JES2 parameters. A warm start can often be used for most parameters, or a specific $T operator command might suffice. The level of minimal JES2 disruption and operator intervention is listed for each JES2 initialization parameter under the heading 'Modification:' within the full parameter descriptions in *z/OS JES2 Initialization and Tuning Reference*. Study these before restarting JES2 for the purpose of initialization parameter specification modification. You might be able to avoid a JES2 warm or cold start. The modification heading is listed for all parameters on all statements; it is specified as follows, with the minimal intervention level:

Table 10. Modification Heading

Minimal Modification	Meaning
COLD START	Only a cold start can be used to modify this parameter.
WARM START	A warm start is required to modify this parameter; a cold start can be used. (To understand the different types of warm starts, see "Restarting JES2" on page 43.)
Operator Command	This parameter can be modified by a specific $T operator command. See *z/OS JES2 Commands* for a full description of the $T command. This parameter might, or might not, also be modifiable by any type of WARM start, or a COLD start. (Some parameters can be modified only by $T command or cold start; these are noted within the parameter descriptions in *z/OS JES2 Initialization and Tuning Reference*.)

Most JES2 initialization parameters can be displayed by the $D operator command. Particularly for tuning purposes, the operator can ask JES2 for the current setting of many JES2 parameters. In this manner, both the operator and the system programmers can be aware of the JES2 environment.

Dynamic display and modification of the JES2 initialization parameters can be used to ease the tuning of your MVS system without unnecessarily disrupting system processing and decrease your installation's need for planned system outages.

How to correct initialization errors

During JES2 initialization, two kinds of errors can occur. The first can occur when JES2 cannot open the data set containing the initialization statements. This happens, for instance, when a DD statement is named in the HASPPARM=ddname start option, but the named DD statement is not defined in the JES procedure. JES2 acknowledges the error with the following messages:

```
$HASP450 OPEN FAILED FOR JES2 PARAMETER LIBRARY
$HASP441 REPLY Y TO CONTINUE INITIALIZATION OR N TO TERMINATE
```

The second message allows the operator to restart initialization processing with a correctly defined data set.

The second error can occur when JES2 encounters an error in the statements in the initialization data set. Initially, JES2 goes into CONSOLE mode, and the operator is

informed of the statement in error. Also, JES2 flags each statement in error and reads the next one in the data set. If the data set is printed out, each statement that was in error will have an error message ($HASP467) printed beside it. (See “Specifying the Start Options” on page 28 for a description of the LIST option and Figure 1 on page 17 for an example of these diagnostic messages.) The text of the $HASP267 diagnostic messages and an explanation of each is listed in *z/OS JES2 Messages*.

If any errors were encountered (when not in console mode) after all parameter statements have been read, JES2 issues the following messages:

```
$HASP451  ERROR ON JES2 PARAMETER LIBRARY
$HASP441  REPLY Y TO CONTINUE INITIALIZATION OR N TO TERMINATE
```

The operator's reply should be based on a careful examination of the initialization statement. If the reply is N, JES2 issues the following message:

```
$HASP428 CORRECT THE ABOVE PROBLEMS AND RESTART JES2
```

When you have included a JES2 initialization statement that contains a parameter specification that is in error, JES2 will ignore the entire statement and do one of the following:

- If the statement was previously specified, obtain whatever explicitly-defined parameter values were coded or defaulted and use these values during the initialization
- If the statement was not previously specified, provide default values for all the parameters on that statement that have JES2 defaults.

Poly-JES

The term “poly-JES” refers to the concurrent operation of multiple copies of JES2. MVS allows more than one JES2 subsystem to operate at a time, if one subsystem is designated as the primary subsystem and others are identified as secondary subsystems. Secondary JES2s can be useful in testing user modifications while the primary JES2 is being used for production.

You might also find a secondary JES2 system to be useful when you need to test a new initialization data set, that is as an "init-deck checker" before running that data set on your production system.

You should replace a few specifications in the initialization deck so that the secondary JES2 does not conflict with the production JES2.

- CKPTDEF - use different data set names or volumes
- SPOOLDEF - use different volumes
- CONDEF - use a different CONCHAR so that you can communicate with the secondary JES2

Certain operational considerations and restrictions apply to the secondary JES2:

- You can start primary and secondary subsystems in any order.
- Started tasks (STCs) can be directed to either a primary or secondary JES. However, when you identify a subsystem as a secondary, you cannot run started tasks (STCs) under that secondary unless you have initialized the primary at least once during this IPL. Attempting to run started tasks on a secondary without starting a primary at some time during this IPL can cause JES2 to become hung, forcing an installation to re-IPL.

- Time-sharing users (TSUs) can only interface with the primary JES.
- The MVS I/O attention table can only be associated with the primary JES. Therefore, secondary JESs cannot receive the "unsolicited interrupt" required to support pause-mode for print and punch devices and "hot readers" (that is, readers started by the physical start button without the $S RDR(n) JES2 command).
- The MVS log console (SYSLOG) can only be associated with the primary JES.
- Secondary subsystems are started individually rather than automatically during IPL using the IEFSSNxx parmlib member.
- If the primary and secondary JESs are in the same MAS, a job restarted by automatic restart management will have affinity to both the primary and the secondary JESs. A job can run in the primary and rerun in the secondary, if there are initiators on both JESs capable of selecting the job. If you don't want jobs to have affinity to both JESs, you should segregate the job classes between the primary and the secondary JESs.

When running more than one JES2 at a time, it is necessary to assign (by use of the CONCHAR= parameter on the CONDEF initialization statement) a unique operator command character to each JES2. If the secondary subsystem is not part of the same multi-access spool complex, each JES2 member requires both a unique spool data set (specified on the VOLUME= parameter of the SPOOLDEF initialization statement) and a unique checkpoint direct access device (specified on the CKPTn=(...,VOL=) parameter of the CKPTDEF initialization statement). Also, the keyword "JES2" must be used to stop whatever subsystem is running, regardless of the name on the START command; for example:

```
S   JES2         START  JES2
$P  JES2         STOP   JES2       (default CONCHAR)
START  JSS4      START  JSS4
/P  JES2         STOP   JSS4       (CONCHAR=/)
```

Many of the considerations for running a primary JES are also appropriate for running a secondary JES; note the following:

- The subsystem support module required by the secondary subsystem, must be placed in either the link pack area (LPA) or the common service area (CSA). See Table 7 on page 37, for more information.

  Note:

  1. It is recommended that a STEPLIB data set be used for the secondary subsystem if the subsystem support module is different than the one in use by the primary system.
  2. When the same copies of the JES2 subsystem support modules are in use by both the primary and secondary subsystems, minimize your storage use by making sure that the JES2 subsystem support modules are loaded into LPA.
- If the secondary subsystem is placed in a library other than SYS1.SHASLNKE or a LNKLST library, be sure to add that data set name as a member to the APF (authorized program facility) specification list in order to authorize it. (See *z/OS MVS Initialization and Tuning Guide* for a complete description of this procedure.)
- If a name other than HASJES20 is used for the main module (this is not recommended) that name must be added to the program properties table such that the program runs in protect key 1 and is nonswappable.

Defining a secondary JES2 member to MVS

Define and add secondary subsystems to MVS by defining the subsystems in the SYS1.PARMLIB member IEFSSNxx. These members contain lists of subsystems to be defined to MVS during master scheduler initialization. The SSN option can be specified when the system parameters are initialized or in IEASYSxx. (IEFSSN00 is reserved for IBM use.) The syntax for the SSN option is as follows:

```
SSN=(aa,bb,...)
```

This is the recommended method for defining and modifying a secondary subsystem. See *z/OS MVS Initialization and Tuning Guide* for further information.

Installing a JES2 secondary subsystem

The following steps show how an installation might install a secondary subsystem. The assemblies and link edits can be accomplished by using the System Modification Program/E (SMP/E) (see *SMP/E for z/OS User's Guide* for a description of this technique) or by using your own JCL. Examples of assembly and link edit procedures are provided in the HASIBLD member of the JES2 distribution library.

Note that JESA (the secondary member) must be added to a IEFSSNxx member of SYS1.PARMLIB.

1. Create a JESA PROC, which defines the secondary JES2's parmlib as 'JESPARM'. For example:

```
//JESA      PROC   M=JESAPARM,U=3380,L=LINKLIB,N=SEC,
//                 V=SYSRES
//IEFPROC   EXEC   PGM=HASJES20,TIME=1440
//STEPLIB   DD     UNIT=3380,VOL=SER=&V,DISP=SHR,
//                 DSN=&N;.&L;
//PROC00    DD     DSN=SYS1.PROCLIB,DISP=SHR
//HASPPARM  DD     DSN=&SYS1;&PARMLIB;(&M),DISP=SHR,UNIT=&U *
//HASPLIST  DD     DDNAME=IEFRDER
```

Note: Libraries specified in the JES2 procedure must be referenced in one of the following ways:

- Cataloged in the master catalog (or in a user catalog, provided the data set has a high-level qualifier other than SYS1)
- Specify UNIT= and VOL= on the JCL DD statement
- Managed by SMS (UNIT= and VOL= on the JCL DD statement are ignored.)

Figure 10. Example JESA PROC

2. Create load modules in a separate, APF-authorized library. This library should be cataloged in the master catalog. For more information on how to perform this task, see *z/OS MVS Initialization and Tuning Reference*.
3. If the checkpoint data set(s) resides on a direct access storage device (DASD), specify them through the DSNAME= and VOLSER= subparameters on the CKPT*n*= parameters of the CKPTDEF initialization statement. If the checkpoint data set(s) resides on a Coupling facility structure, specify the structures through the STRNAME= subparameter on the CKPT*n*= parameters of the CKPTDEF initialization statement.

   When the checkpoint data sets reside on DASD, they are often named SYSn.JESCKPT1 and SYSn.JESCKPT2, where 'n' represents any number from 0-9.

A checkpoint data set on a Coupling facility structure must refer to a predefined structure. For an example of the definitions necessary for JES2 checkpoints on Coupling facilities, see “Placement of the checkpoint data set” on page 192. For checkpoints residing on a Coupling facility structure, IBM suggests using a name without a period, such as SYSn_J2WRKCKPT1 and SYSn_J2WRKCKPT2, to differentiate from DASD.

4. Allocate a spool data set, using the name SYSn.HASPACE, and place it on a particular volume (many use SECJ2, but the volume name is optional).
5. Specify options to the primary JES2 member when it enters initialization.
6. If you intend to use a functional subsystem (FSS), add or modify any cataloged procedures for functional subsystems thatrun under the secondary subsystem. Ensure that the correct version of the HASPFSSM load module is used.

   The internal START command that JES2 uses to start a functional subsystem (FSS) cannot contain variable operands to specify STEPLIB information. Therefore, if multiple versions of the JES2 are maintained in separate libraries, you might need to maintain multiple versions of the FSS cataloged procedure(s) and vary the PROC= keyword of the FSS initialization statement.
7. Issue an 'S JESA' MVS command when the primary member is active.

The secondary JES2 proclib member (referred to as 'JESA' on the MVS START command) points to the JES2 initialization data set that defines the primary checkpoint data set to a Coupling facility structure and allocates the secondary checkpoint to a DASD volume. The following is an example of the initialization statements:

```
CKPTDEF    CKPT1=(STR=SYS1_WRKCKPT1,INUSE=YES)
           CKPT2=(DSN=SYS1.JESCKPT2,VOL=SECJ2,INUSE=YES)
SPOOLDEF   VOLUME=SECJ2
CONDEF     CONCHAR=/
.
.
.
```

To execute this procedure and obtain a listing of the initialization deck for secondary subsystem JESA on printer 00E, enter the following command:

```
S   JESA,UNIT=00E,
```

Enabling the AUTOEMEM option on a secondary subsystem: If both the primary and the secondary JES2 subsystem use the AUTOEMEM option to make jobs from failed members eligible for restart on another member, (AUTOEMEM=ON on the MASDEF initialization statement), you must ensure that they have distinct member names. Because the node name is from the NAME= parameter on the NODE(nnnn) initialization statement, ensure that the primary and secondary JES2 member have different variables for at least one of the parameters.

Starting an alternate JES2 subsystem as a primary subsystem

You can run an alternate job entry subsystem (JES) as a primary JES in place of the primary JES. To be run as the primary, start the alternate by using the name of the primary, because only the subsystem known to MVS as the “primary JES” can interface with TSO/E. To start the alternate, the procedure used to start the primary must have STEPLIB and HASPPARM DD statements that can be overridden by the START command. If these DD statements cannot be overridden, the required JES cannot be loaded to replace the primary JES.

To execute a different copy of JES2 as a primary subsystem, one of the following procedures should be used. If the primary member specified at system installation is automatically started, it must be stopped before initialization. You can stop the primary member either by specifying a CONSOLE statement in the JES2 initialization data set or on the MVS START command. Then reply:

```
Rnn,$PJES2
```

when the member prompts you with the $HASP426 message.

If the primary was not automatically started or was prevented from initializing by the previous procedure, then enter the command S JES2, overriding the data set name and the volume serial on the STEPLIB DD statement to point to the load library containing the alternate subsystem.

The following is a sample procedure that will run either the normal JES2 or any alternate version of JES2 as a primary job entry subsystem:

```
  //JES2      PROC  M=JESAPARM,U=3380,L=LINKLIB,N=SYS1,
  //                V=SYSRES
  //IEFPROC   EXEC  PGM=HASJES20,TIME=1440
  //STEPLIB   DD    UNIT=3380,VOL=SER=&V,DISP=SHR,
  //                DSN=&N;.&L;
  //PROC00    DD    DSN=SYS1.PROCLIB,DISP=SHR
1 //HASPPARM  DD   DSN=&SYS1;PARMLIB.(&M),DISP=SHR,UNIT=&U;
  //HASPLIST  DD    DDNAME=IEFRDER
```

Note: Libraries specified in the JES2 procedure must be referenced in one of the following ways:

- Cataloged in the master catalog (or in a user catalog, provided the data set has a high-level qualifier other than SYS1)
- Specify UNIT= and VOL= on the JCL DD statement
- Managed by SMS (UNIT= and VOL= on the JCL DD statement are ignored.)

Figure 11. Sample procedure

To cause JES2 to be loaded from an alternate library (TEST.LINKLIB on TSTPAK), and to cause an alternate member of SYS1.SHASPARM to be used for JES2 initialization, create a member in SYS1.SHASPARM. Then, enter the following command:

```
S  JES2,N=TEST,V=TSTPAK,M=TESTPARM
```

To make the secondary test JES2 the primary JES2 member when you have finished testing, issue the following MVS command:

```
S  JESA,JOBNAME=JES2
```

This command starts the 'JES2' job as a started task. For more information about this procedure, see “Specifying a back-up JES2 cataloged procedure” on page 21.

Note: JES2 starts each functional subsystem (FSS) through an internal STARTcommand. This command cannot contain variable operands to specify STEPLIB information. If multiple versions of JES2 are maintained in separate libraries, you might need to maintain multiple versions of the FSS cataloged procedure(s) and vary the PROC= keyword of the FSS initialization statements to ensure that the correct version of the HASPFSSM load module is used.

How to initialize JES2 in a multi-access SPOOL configuration

It is possible to operate from 2 to 32 members in a MAS configuration, as shown in Figure 12. MAS configurations can also participate as nodes in an NJE network. For sample definitions of such configurations, see "Multi-access SPOOL configuration considerations for NJE" on page 320.

The operation of each member in the configuration is independent and includes all functions previously described for single JES2 members. Each JES2 member can:

- Read jobs from local and remote card readers
- Schedule jobs for conversion and execution under MVS initiators
- Print and punch results at local and remote output devices
- Communicate with operators and time-sharing users.

However, all spool volumes are used by all members in the configuration.

The checkpoint data sets in use within a MAS must be shared by all JES2 members. If a checkpoint data set resides on a coupling facility structure, all members in the configuration must be connected to that coupling facility through the active Coupling Facility Resource Management (CFRM) policy. If the checkpoint data set resides on DASD, the volume containing the checkpoint must be accessible by all members in the configuration.

The members logically share common JES2 job and output queues. The workload can be balanced among members by allowing jobs to execute on whatever member has an idle initiator with the correct setup and required output devices.

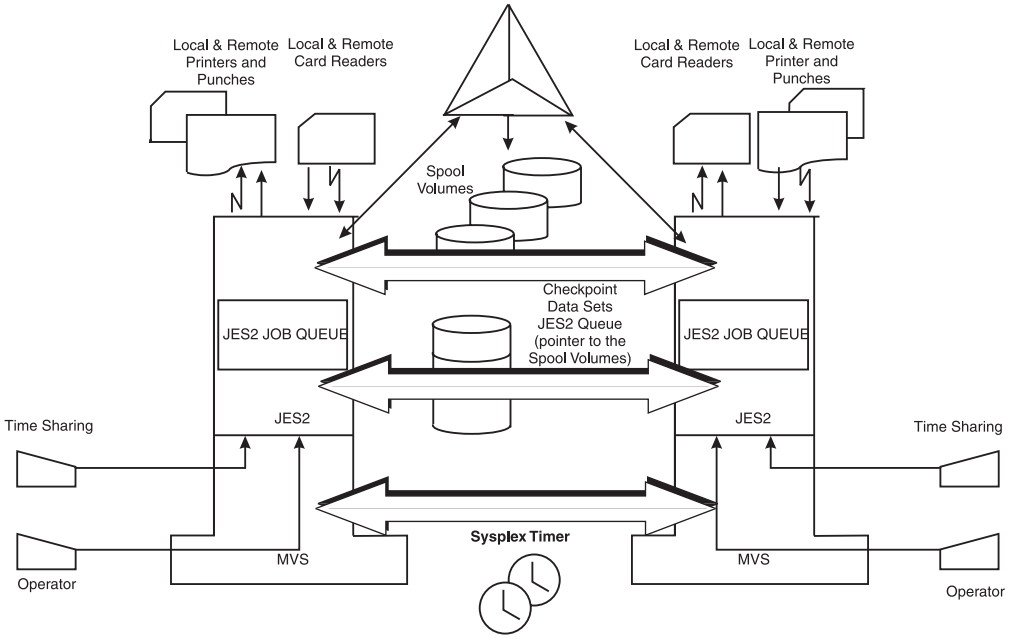


Figure 12. Example of a two-member shared JES2 configuration

Because all members are functionally the same, if one member in the configuration fails, the others can continue processing from the common queue. Only work in process on the failing member is interrupted; this work can be recovered by a warm start of the failing member while other members continue processing. This

work can also be recovered by an operator command ($E MEMBER) from one of the other members, or through the AUTOEMEM option.

automatic restart management can also be used to restart batch jobs and started tasks (STCs). However, in order for automatic restart management to restart jobs, all JES's must be connected in the same MAS. Although there might be more than one MAS in a sysplex, automatic restart management cannot restart jobs across MASes. For a description of both recovery methods, see "Recovering from member failures on other JES2 members" on page 241. For more information about using automatic restart management, see *z/OS MVS Setting Up a Sysplex*.

Accessing JES2 SPOOL information in a MAS

JES2 uses shared direct access storage device (DASD) hardware featuresto access data on all spool volumes. A copy of the JES2 queues and other status information (for example, spool space allocation maps) is written to the checkpoint data set to facilitate a warm start, as with a single JES2 member. This information is available to all members, one at a time, as needed.

Each member in the configuration must have at least one channel path to each spool volume, and these devices must be specified as shared during MVS system installation.

Accessing JES2 checkpoint data sets in a MAS

All members that use the checkpoint data set(s) specified by the CKPTn parameters on the CKPTDEF statement must at least have access to the shared volume on which CKPT1 exists. This shared access ensures adequate workload communication from member to member. JES2 checkpoint access in a MAS configuration depends on whetherthe checkpoint data set(s) reside on a Coupling facility structure or on a DASD volume.

For the checkpoint data set(s) that reside on a structure, the locking capabilities of the Coupling facility prevent JES2 members fromsimultaneously referencing and updating information kept in the checkpoint data set.

For the checkpoint data set(s) that reside on a DASD volume, JES2 processing uses RESERVE and RELEASE logic with a software checkpoint lock to prevent JES2 members from simultaneously referencing and updating information kept in thecheckpoint data set. Because of the use of RESERVE, it is recommended that the checkpoint data set not be placed on a spool volume.

For checkpoint data sets residing on DASD, IBM suggests that you add the checkpoint data set(s) to the RESERVE conversion resource exclusion name list (RNL) to prevent global resource serialization (GRS) from limiting access to that data set and degrading performance. For more information about specifying the RESERVE conversion RNL list, see *z/OS MVS Planning: Global Resource Serialization*.

Initializing the multi-access spool configuration

When setting up a multi-access spool configuration, you must first determine whether your checkpoint data set(s) reside on a Coupling facility structure or on a DASD volume. If any of your checkpoint data sets reside on DASD, the VOLUME parameter on the SPOOLDEF statement and the CKPT1=(...,VOL=) and CKPT2=(...,VOL=) parameter specifications on the CKPTDEF statement must specify the same volumes for all members in the configuration. Note that within each member, the spool volume and the two checkpoint volumes must all be

separate to reduce any potential loss in the case of volume failure. If checkpoint data sets reside on a Coupling facility structure, the CKPT1=(STRNAME=) and CKPT2=(STRNAME=) parameter specifications must be the same for all members in the configuration. In addition, the statements and parameters specified in Table 11 must match across the MAS.

Because the default values calculated for the JOENUM= parameter on the OUTDEF statement are configuration dependent, an inconsistency might inadvertently be introduced if this parameter is not specified explicitly at initialization. Although many of the parameters on the SPOOLDEF, OUTDEF, and NJEDEF statements can be dynamically changed by operator command, several require a warm start or cold start to modify. Review the description of each in *z/OS JES2 Initialization and Tuning Reference* to become familiar with these requirements.

Table 11. Multi-access spool and checkpoint data set compatibility

Keywords	**The definition of each of these keywords must be the same on each member in your configuration to allow multi-access spool and checkpoint data compatibility.**
CKPTDEF	CKPTn=({(DSN=,VOL=)|(STRNAME=)},INUSE=YES)
	LOGSIZE=,MODE=
CKPTSPACE	BERTNUM=
DESTDEF	LOCALNUM=
	Ndest=
	R|RM|RMTdest=
	Udest=
JOBCLASS	(all parameters)
JOBDEF	JOBNUM=
MASDEF	XCFGRPNM
NJEDEF	NODENUM=
	OWNNODE=
OUTDEF	JOENUM=
SPOOLDEF	BUFSIZE=
	DSNAME=
	TGSPACE=(MAX=)
	TRKCELL=
	SPOOLNUM=
	VOLUME=

For operational consistency, it is strongly recommended that the installation exit routines be the same in all members. Similarly, values specified on DESTID(jxxxxxxx) and NODE(nnnn) initialization statements should be the same across the configuration, although JES2 will not detect inconsistent values.

IBM suggests that local unit record devices and RJE lines be given unique JES2 device names over the whole configuration. These devices (printers, punches, lines, readers) are defined by JES2 initialization statements; therefore, be certain to define each device in the same manner to each member in your configuration. This allows all devices to be attached to one member (with appropriate manual switching) if other members are not operational.

Similarly, the LINE(nnnn), PRT(nnnn), PUN(nn), and RDR(nn) initialization statements should be set so that a physical device has the same JES2 device number no matter to which member it is attached. For example, if a 3211 printer is one of four local printers on a two-member configuration, it could be initialized as PRT(4) UNIT=102 for one member and PRT(4) UNIT=302 for the other, if it were attached to different channel paths on the two members.

A local unit record device or RJE line can only be attached to one member at any one time. JES2 initialization will detect devices that are not online and place them in a drained state. Later, the device can be activated by entering the $P device and VARY device OFFLINE commands on the member to which it is attached, performing hardware switching, then entering the VARY device ONLINE and $S device commands on the new member. The $S command will fail if no hardware path exists.

The algorithm for using the common JES2 queues and other information in the checkpoint data set is determined by the DORMANCY= and HOLD= parameters on the MASDEF initialization statement. These need not be the same for all members in the configuration and should be set according to characteristics such as the number of members in the configuration, relative processor speeds, and response requirements. (See Chapter 4, “Checkpoint data set definition and configuration,” on page 191 for further information on these parameter specifications.)

Starting the multi-access SPOOL configuration

Multi-access spool configurations **must** be started within a sysplex and as such, are synchronized through the sysplex timer (such as, an IBM Sysplex Timer®, 9037). **Each JES2 MAS must be entirely contained within one sysplex; configuring more than one MAS in a sysplex is not recommended.** In a sysplex, the sysplex timer synchronizes the time-of-day (TOD) clocks. The MASDEF SYNCTOL= parameter value is not used for clock tolerance synchronization. However, JES2 considers the SYNCTOL= value when determining if other members are active. JES2 examines the time other systems last accessed the checkpoint and determines if the time is within the SYNCTOL value of the current time. JES2 considers systems within SYNCTOL tolerance as active and attempts to detect if a system is:

- the only (first) member active in the sysplex
- active outside the sysplex
- within the sysplex that was started with an incorrect XCF group name.

In the second two cases, JES2 determines if such a system's access to the checkpoint was recent (within the SYNCTOL= value), If either case is detected, JES2 cannot determine if the current (initializing) member is correctly configured or if the current member is configured in error. To protect the MAS, JES2 might fail the start or request operator intervention to determine whether JES2 should continue to initialize. In this way, the MAS is protected from access by an improperly configured member.

Initially, one member of the MAS must do a cold start, other members are added, deleted, or changed on a warm start. As each new member is warm started and initialized, all other members of the MAS are made aware of its existence because each member shares the same spool volumes and checkpoint data sets and communicates through the JES common coupling services (JES XCF).

The individual members (up to 32) of a multi-access spool configuration are specified on the MEMBER initialization statement by the NAME= parameter. Initially, you can define all 32 members of the MAS by defining MEMBER(1) NAME= through MEMBER(32) NAME=. As you require these members within your configuration you can make them available and active by specifying their MASDEF OWNMEMB= parameter when you start each of the systems.Or you can define each individually at any time on a warm start. On a warm start, members can be added, deleted, or their names can be changed. The following series of examples is provided to illustrate the control you have over your MAS complex with the MEMBER and MASDEF statements. Systematically, the examples are provided to show member definition (**A** and **B**), addition (**C**), change (**D**), and deletion (**E**).

A Defining multi-access spool configuration members:

```
MEMBER(1) NAME=SY01           /*   Define member SY01 */
MEMBER(2) NAME=SY02           /*   Define member SY02 */
MEMBER(3) NAME=SY03           /*   Define member SY03 */
MEMBER(4) NAME=SY04           /*   Define member SY04 */
MEMBER(5) NAME=SY05           /*   Define member SY05 */
MEMBER(6) NAME=SY06           /*   Define member SY06 */
MEMBER(7) NAME=SY07           /*   Define member SY07 */
MEMBER(8) NAME=SY08           /*   Define member SY08 */
        .
        .
        .
MEMBER(32) NAME=SY32          /*   Define member SY32 */
MASDEF OWNMEMB=SY01           /*   Define own member  */
```

In example **A** , member MEMBER(1) NAME=SY01, defines the first member of a multi-access spool configuration. Each member has been assigned a member name (SY01 through SY32, respectively). If you have defined all 32 possible members to your multi-access spool configuration with MEMBER statements, all available slots in the member ID table will be allotted. You must activate each member by adding that member's name on the OWNMEMB= parameter of its MASDEF statement or the member will not be able to perform any work. When an OWNMEMB= parameter was added to example **A** , only MEMBER(1) was assigned at that time. Therefore, as you start each new member in your MAS, you will need to provide an OWNMEMB= parameter for that member within its initialization stream.

If the OWNMEMB= parameter was not specified, JES2 uses the SMF ID as the default for the OWNMEMB value; JES2 provides a message to inform you that the name will be added. JES2 determines the member number by scanning the MEMBER initialization statements for the OWNMEMB name. If the name is not found, JES2 assigns OWNMEMB to the first available slot in the member ID table.

In the example above, if the OWNMEMB= parameter had not been coded and allowed to default, JES2 would have not found an available member (all slots were allotted in the member ID table) with which to define your member.

You can also use the $D MEMBER command to verify the status of individual parameters. Two members cannot specify the same OWNMEMB= parameter; the

second member ends abnormally. When defining a member, IBM suggests that you explicitly code an OWNMEMB= parameter in the initialization stream of each multi-access spool member.

Following the IPL for the member defined in example **A**, only MEMBER(1) is active, MEMBER(2) through MEMBER(32) are defined but inactive.

Alternately, you can define a single-member MAS with the following initialization statements. (These same statements can be repeated for each member of the MAS, as you typically repeat all other initialization stream statements and modify them to meet the needs of the particular member. You must, of course, update the MEMBER number and the corresponding name as shown in example **C**.)

B Activate the first member of a JES2 MAS complex:

```
MEMBER(1) NAME=SY01          /*  Define SY01           */
MASDEF OWNMEMB=SY01,         /*  Define SMF ID for SY01*/
       .                     /*  Further define        */
       .                     /*     member SY01        */
       .                     /*     parameters         */
```

By adding the MASDEF initialization statement with the OWNMEMB= parameter, you have assigned a member name and made that one member active. Other parameter specifications should, of course, be added to this statement to further define the member. (See the MASDEF statement description in *z/OS JES2 Initialization and Tuning Reference* for a complete description of the other parameters.)

At this point, only MEMBER(1) is defined and active, all other members (MEMBER(2) - MEMBER(32)) remain undefined.

To add a second and third member to this single-member MAS add the following initialization statements to their **separate** initialization streams.

C Add a second and third member to this single-member MAS:

INITIALIZATION STREAM FOR MEMBER SY02:

```
         .
         .
         .
MEMBER(2)  NAME=SY02         /*  Define SY02           */
MASDEF  OWNMEMB=SY02,        /*  Define SMF ID FOR SY02*/
         .                   /*  Further define        */
         .                   /*     member SY02        */
         .                   /*     parameters         */
```

INITIALIZATION STREAM FOR MEMBER SY03:

```
         .
         .
         .
MEMBER(3)  NAME=SY03         /*  Define SY03           */
MASDEF  OWNMEMB=SY03,        /*  Define SMF ID FOR SY03*/
         .                   /*  Further define        */
         .                   /*     member SY03        */
         .                   /*     parameters         */
```

MEMBER(2) and MEMBER(3) are defined and added as the second and third members of the MAS by specification within their own initialization streams. All other members (MEMBER(4) - MEMBER(32)) remain undefined.

If you need to redefine a member's name this can be done just by adding the redefined statement, for example:

D Redefine a member's name:

```
MEMBER(3) NAME=MEM3           /*  Change MEMBER(3) from */
MASDEF OWNMEMB=MEM3,          /*   member SY03 to MEM3  */
         .
         .
```

You can only redefine a member when that member is inactive. If you attempt to change an active member, JES2 responds with $HASP876 MEMBER *memname* IS ACTIVE, CHANGE IGNORED.

Because JES2 keeps a copy of the multi-access spool complex configuration in the checkpoint, when you want to delete a member from your multi-access spool complex you must define a null member (see the following example); you cannot delete the member by just omitting the member on a new MEMBER statement.

E Deleting a multi-access spool member:

```
MEMBER(1) NAME=           /*  Delete Member SY01       */
MEMBER(2) NAME=SY02       /*  Member SY02 still active */
MEMBER(3) NAME=MEM3       /*  Member MEM3 still active */
           .
           .
           .
```

As a result of the initialization statement above, MEMBER(1) (defined as SY01) is deleted from the multi-access spool complex. Note that SY01 must be inactive. JES2 performs restart processing for SY01 to allow work on the deleted member to be eligible for restart on other members in the complex. MEMBER(2) - MEMBER(3) (SY02 and MEM3) remain defined and active. MEMBER(1) is now undefined, and MEMBER(4) - MEMBER(32) remain undefined.

Similarly, you cannot change the MEMBERs in a MAS by just changing the name on the MEMBER statement and performing a warm start. This results in JES2 issuing message $HASP435 MEMBER TABLE PARAMETER ERROR, because the member names you specified already exist in the checkpoint copy.

In order to reassign member names to different processors in a multi-access spool complex, you must:

1. Change the names to temporary identifiers
2. Stop JES2 on all members in the multi-access spool complex
3. Perform an all-member warm start
4. Change the temporary identifies to the names you want
5. Perform another all-member warm start.

For example, a 3-member multi-access spool complex has defined:

- MEMBER(1) NAME=A
- MEMBER(2) NAME=B
- MEMBER(3) NAME=C

The installation now wants to change the configuration to:

- MEMBER(1) NAME=B
- MEMBER(2) NAME=C
- MEMBER(3) NAME=A

Figure 13 shows an example of the steps above.

```
MEMBER(1) NAME=A        /*  ORIGINAL CONFIGURATION */
MEMBER(2) NAME=B
MEMBER(3) NAME=C
          .
          .
          .

Step 1:
MEMBER(1) NAME=X        /*  DEFINE TEMPORARY      */
MEMBER(2) NAME=Y        /*  MEMBER IDENTIFIERS    */
MEMBER(3) NAME=Z        /*  AND THEN WARM START   */
          .
          .
          .

Step 2:
$PJES2 on all members in MAS.

Step 3:
     Perform an all-member warm start with the above configuration.

Step 4:
     When JES2 starts redefine with the following statements:

MEMBER(1) NAME=B        /*  NEW CONFIGURATION     */
MEMBER(2) NAME=C
MEMBER(3) NAME=A
          .
          .
          .

Step 5:
     Perform an all-member warm start with the above configuration.
```

Figure 13. Reassigning Member Identifiers to Different Processors

Job submission and queuing

In a multi-access spool configuration, jobs enter the common queue from any input source (local or remote) attached to any member in the configuration. Normally, (unless special actions are taken), jobs are eligible to execute in any member in the configuration just as in a single-member operation. Started tasks and TSO/E users are an exception: they execute only in the member in which they are entered.

There are two different modes of initiators: those owned by JES2 and controlled by JES commands, and those owned and managed by workload management. Whether a job is selected by a JES initiator or by a WLM initiator is determined by the MODE= parameter on the JOBCLASS statement. Initiators from each JES2 member in the MAS select work from the common job queue independently of other initiators on other members of the JESplex. See “WLM control of batch job initiation” on page 93 for more details.

Job queue entries also contain a member affinity for up to 32 members on the maximum configuration and can contain an independent mode affinity.

Displaying or updating a job's affinity

Individual jobs can be given affinity to one or more members (less than the total configuration). Member affinity is required if a processor is operating in independent mode; that is, isolated from the complex and only running jobs specifically routed to it. Provide affinity for independent mode by using the SYSAFF= keyword on the /*JOBPARM statement.The $T command can be used to give member or independent-mode affinities to all jobs read from a specific input device (local or remote). The /*JOBPARM statement overrides the input device default.

Sysaff and independent mode are related to job submission, selection, and execution. They do not affect SYSOUT. Sysout can be selected regardless of the independent settings of the MAS members.

If a job's affinity is to specific members in the configuration or to independent mode, the job can be selected only by the members specified and only if the mode of the member (independent or not) matches that of the job. For a job transmitter, we check the independent mode matches, but not sysaff (member). This is because Offload and Network Job transmitters can, or will, send the job to another Node for selection and execution. Sysaff= and independent mode will be verified on the receiving side.

Member affinity is useful for special processing requirements (for example, emulation) not available on all members of the configuration. Independent mode is useful for testing new components with selected jobs while in a shared configuration. Independent mode must be checked (Member must match job characteristics) for any QGET type of operations. This includes Conversion, Execution, Purge and Job XMIT.

The display commands ($DA, $DN, $DQ, $DJ) can be used to determine (by SMF system ID) the member in which a job is active and the members eligible to process a queued job. The $T J command allows you to change affinities of individual jobs or of all jobs with given affinity. The $T MEMBER command allows you to place a member in independent mode. The $D MEMBER command displays the status of all members in the multi-access spool configuration.

Scheduling environment

Similar to member affinity, a job may be assigned a scheduling environment to ensure it executes on specific member(s) in the MAS with the SCHENV= keyword parameter on the JOB statement or using the $T Job JES2 command. Scheduling environments are installation-defined 16-character names which may be available on any of the MVS systems in the sysplex, or may be available on none of the systems.

JES2 can display the scheduling environment for a specified job, and list those jobs waiting for a specified scheduling environment.

SDSF can also be used to display all the scheduling environments defined to the WLM and on which members in the MAS each is available using the SE option. The scheduling environment assigned to a job can be update or deleted using the $T Job command.

For more information on scheduling environments, see “Job scheduling environment” on page 85.

Using RACF multi-level security

In support of multi-level security support through RACF, JES2 can now limit job selection based on "security label by system". Although the RACF data base is shared by all MAS members and most profiles apply to all systems, you can specify a subset of members to which SECLABELs apply. A RACF SETROPT command option (SECURITY LABEL BY SYSTEM) controls whether SECLABELs are active on all systems or only those you specify. SECLABELs control MAC (mandatory access control). JES2 maps the systems for which a SECLABEL is active against an affinity mask associated with each batch job. JES2 then uses that affinity mask to determine where a job can be selected for conversion and execution. This affinity can increase security within your system, but it can also prevent job selection.

If you activate security label by system (SECLABEL by system), you are probably limiting the MAS member(s) on which a job can run. Therefore, a job might not be able to run if there is no system that satisfies the requirements for system affinity (SYSAFF=), scheduling environment affinity (SCHENV_AFF=), and SECLABEL affinity (SECLABEL_AFF=). For a job to be selected, JES2 must find an active system in all three affinity lists. (Consider each list as a set within a Venn diagram - the intersection of the three sets defines the select group of systems on which the job can run. If the intersection is empty, the job cannot run). Use the JES2 $D Job,LONG command to display the list of system that are assigned SECLABEL affinity.

Conversion processing requires a system be on both the SYSAFF= and SECLABEL_AFF= lists.

If you use Exit 14 (Job queue work select - $QGET) to replace the job selection that JES2 normally performs in $QGET, you might need to update your exit to take SECLABEL by system (and the JQASCLAF mask) into consideration when selecting jobs for conversion and execution.

Restarting jobs

If a member fails, and jobs in execution are recovered and requeued for automatic restart (either by a warm start or the $E MEMBER command), those jobs that have journals are given affinity only to the failed member. If the failed member is unavailable, the operator can change affinity with the $T J command to attempt restart on another member.

You can also use automatic restart management to automatically restart batch jobs and started tasks (both are considered jobs). If the job is registered with automatic restart management, automatic restart management restarts jobs when either:

- An executing job unexpectedly ends
- The system on which a job is executing unexpectedly ends or leaves the sysplex.

Duplicate job names

Duplicate job name protection extends to all members; that is, if one job name matches another job name that is active in execution anywhere in the configuration, the job is temporarily delayed.

By default, JES2 enables duplicate batch job name protection for all members of the multi-access spool configuration. That is, if one batch job name matches another batch job name that is active in execution anywhere in the configuration, the

second job is temporarily delayed until the first job completes. You can disable duplicate batch job name protection by specifying DUPL_JOB=NODELAY on the JOBDEF initialization statement.

Note: DUPL_JOB= has no effect on TSO/E user logon and duplicate instances of a specific user ID.

See "TSO/E" for more information concerning related TSO/E processing and restrictions.

Priority aging

Priority aging is done only by the lowest-numbered active member in the configuration.

Output

Printed and punched output processing in a multi-access spool configuration differs very little from that of single-member operation. Member affinity does not apply to selection of work from the job output elements.

The selection of output for a device is governed by a list of work selection (WS) criteria. This list is specified through the WS= keyword on the output device initialization statement or through the WS operand on the JES2 $T command. Either you or the operator can choose from a list of characteristics (user-specified and JES2-specified criteria) to be considered when JES2 selects output for output devices.

RJE

JES2 ensures that the same remote terminal cannot sign on to more than one line anywhere in the configuration at any given time. For dedicated lines, the user must ensure this uniqueness by properly setting line and remote initialization parameters as previously described. JES2 sends a copy of the signon message back to the originating remote terminal as an acknowledgment of successful signon.

Remote operator messages can be sent across the entire configuration. That is, any remote operator can send messages to any other remote operator (even if attached to different members) and any central operator can send a message to any remote operator.

JES2 sends selected messages generated by remotely submitted jobs back to the remote terminal on which the job was submitted. A multileaving terminal with a console will receive messages on an output printer (not the console) if the terminal was inactive at the time the message was sent, or if the number of messages queued for the terminal exceeds the RMTMSG= parameter specification on the TPDEF statement.

Configuration considerations for RJE lines are discussed under "Remote line and device configuration" on page 72.

TSO/E

In z/OS V1R4, JES2 discontinued monitoring TSO/E logons. Because duplicate instances of a given user ID can be logged on at the same time, it is highly recommended that the GRS RNL list be updated to convert SYSIKJUA to a SYSTEMS ENQ in order to prevent such duplicates. If the RNL is not updated and duplicate instances of a user ID are logged on, then TSO GR and SEND messages sent to the duplicated logged on user IDs will not function.

To revert to pre-z/OS V1R4 processing of duplicate logon checking, use the supplied JES2 sample Exit 44 (HASX44A). see *z/OS JES2 Initialization and Tuning Reference* for further exit information. Using this exit might result in a delayed re-logon for a given user ID with its accompanying erroneous error message.

Jobs submitted by TSO/E can execute anywhere in the configuration, subject to their affinities (as previously discussed). However, held output data sets are accessible (by the TSO/E OUTPUT command) to the submitting user, regardless of where he is logged on or where the job was executed. If you are not using the shared broadcasting function (BRODCAST=NO is either specified on the OUTDEF initialization statement or the default), JES2 processes the NOTIFY messages as requested on the JCL JOB statement as follows:

- If the user is logged on, the NOTIFY message is returned to that user.
- If the user is not logged on, the NOTIFY message is issued on the member from which the job was submitted, and the message is added to the SYS1.BRODCAST data set associated with the originating member. Also, output generated from that job is not available until the originating member has issued the NOTIFY message.

If, however, you are using the shared broadcasting function (BRODCAST=YES is specified on the OUTDEF initialization statement), JES2 processes the NOTIFY message as follows:

- If the user is logged on, the NOTIFY message is returned to that user.
- If the user is not logged on, the NOTIFY could be issued on any member in the complex, and the message is added to the SYS1.BRODCAST data set.

Jobs submitted by TSO/E users can also execute on any configuration in the NJE network. When a job is successfully transmitted to another node for execution, it is purged at the entry node.

Functional subsystem support

The following section provides:

- An overview of what a functional subsystem (FSS) is
- How to use an FSS in a poly-JES environment
- The failure and recovery procedures unique to the FSS address space.

Functional subsystem and functional subsystem interface

A functional subsystem (FSS) is an extension of JES2 that runs in an address space separate from the JES2 address space. An FSS provides support for a function peripheral to JES2 processing, such as a peripheral device or other component. A functional subsystem application (FSA) executes within the functional subsystem address space and is defined to provide application-specific support to peripheral functions. An FSA allows devices to operate outside of direct JES2 control. The support contained in the FSS/FSA, then, is specific to the task. There is no limit to the number of FSSs that you can define, nor is there a limit to the number of FSAs that you can define to run under an individual FSS.

Note: All FSS-controlled printers, such as the 3820 or 3800-3 operating in full-function mode (no matter how distant the connection), are considered by JES2 to be “local”.

A functional subsystem can be defined to relieve JES2 of device-specific support associated with page-mode data. For example, because page-mode printers have processing requirements not required by line-mode printers (such as translating

complex data attributes), the Print Services Facility is an FSS/FSA defined to drive a 3800-3 or 3820 printer when operating in full-function mode.

Because an FSS is common to all FSAs connected to it, the FSS provides a common set of services to those applications. For example, HASPFSSM supports a number of functions to include acquiring data sets (GETDS) and releasing data sets (RELDS). These functions are invoked from the FSA and are used to communicate with JES2 for data set printing.

Functional subsystems communicate with both JES2 and the FSA that resides within the functional subsystem address space. JES2 communicates with the functional subsystem through the functional subsystem interface (FSI). JES2 supports AMODE(31) callers and data areas across the FSI. JES2 data areas reside in virtual storage, above 16 megabytes. The FSI is not a single definable piece of code; rather an FSI encompasses those individual pieces of code (resident within JES2 and the FSA) that provide the communications between JES2 and the FSA. Figure 14 depicts the relationship between JES2 and the functional subsystem components.

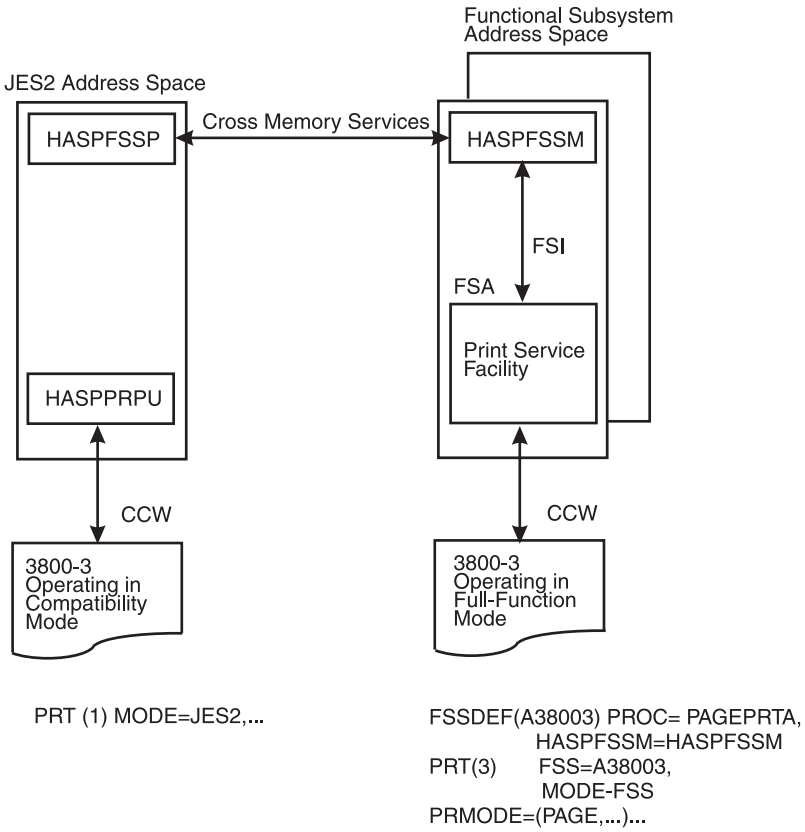


Figure 14. Representation of the relationship between functional subsystem components

Using functional subsystems in a poly-JES environment

In a poly-JES2 environment (that is, an MVS system in which more than one JES2 is running concurrently) either one or both JESs can start and pass data sets to a functional subsystem for processing (for example, page-mode printing). A functional subsystem is started by an internally produced MVS START command; JES2 processes the START command so that the functional subsystem started task

is associated with the appropriate JES2 (the JES2 that processed the START command). A functional subsystem **cannot** communicate with multiple subsystems simultaneously.

Functional subsystem recovery procedures

Processing of page-mode data sets occurs in a functional subsystem address space that is separate from the JES2 address space; therefore, the following two failure or recovery situations can arise:

- The functional subsystem address space abnormally ended.

  The functional subsystem attempts to halt the functional subsystem application (FSA) processing or itself through use of its own recovery routines. If the functional subsystem can recover in this manner, JES2 is not aware of the failure. If not, JES2 provides for cleanup of resources, such as the functional subsystem address space, and recovers any work assigned to the failing functional subsystem; all output devices associated with the failing functional subsystem are marked drained.
- JES2 abnormally ended.

  When JES2 ends, any started functional subsystem continues to print work assigned to it before the JES2 failure. After finishing the assigned work, the functional subsystem becomes idle. After JES2 is restarted (by a hot start), JES2 reestablishes communications with the active functional subsystem and normal processing; that is, assignment of work resumes.

Chapter 2. Controlling JES2 processes

This chapter describes the aspects of JES2 processing that you can affect through initialization statements (and their associated parameters) and operator commands. For specific information about coding initialization statements, see Chapter 1, "JES2 initialization," on page 1. For descriptions of initialization statements, see *z/OS JES2 Initialization and Tuning Reference*. For information about operator commands, see *z/OS JES2 Commands*.

Devices

With initialization statements and commands, you can specify the configuration of JES2 local devices, the JES2 internal reader facility, the JES2 remote lines and devices, and the JES2 spool volumes.

Local devices are attached to the MVS system and are used by JES2 for reading jobs and writing output. Local devices include card readers, card punches, internal readers, and printers.

Defining device configuration

You identify local devices JES2 uses with the RDR(nn), PRT(nn), PUN(nnnn), and LINE(nnnn) initialization statements. You can also associate JES2 processing parameters with each device, and indicate whether the device is available (started) on completion of JES2 initialization.

During JES2 initialization, JES2 determines the number of printers, punches, and readers by counting each unique PRT(nnnn), PUN(nn), and RDR(nn) initialization statement. JES2 handles any devices not defined during initialization through default parameter values for that type device. *z/OS JES2 Initialization and Tuning Reference* describes the default parameters used.

JES2 dynamically allocates any available local devices, and processing begins on that device when work exists for the device. An operator must start a drained or halted device with a JES2 start command ($S). During JES2 processing, you start devices with the JES2 start command ($S device) and deactivate devices with the JES2 stop command ($P device).

Assigning devices dynamically after initialization

After initialization, operators can use JES2 commands to add, assign, or reassign local printers, punches, readers, communication lines, and channel-to-channel (CTC) adapters. By specifying a device statement without a unit value, the device name and its attributes become known to JES2, but a physical device is not assigned until a unit value is added using a JES2 $T command. Defining devices in this manner allows installations to dynamically assign or reassign devices after initialization.

You can omit the UNIT= on the following JES2 device initialization statements:

- PRT(nnnn)
- PUN(nn)
- RDR(nn)
- LINE(nnnn)

If you do not specify UNIT= on the PRT(nnnn) statement, JES2 bases the defaults for COPYMARK, FLash, FCB, and UCS on the MODE= parameter. If MODE=FSS, these parameters default to specifications for a non-impact printer. If MODE=JES, these parameters default to specifications for an impact printer. To determine how JES2 parameters affecting FSS printer characteristics retain values across a hot start, see "Functional subsystem reconnection" on page 49.

During initialization, JES2 does not allocate or assign devices defined by these spare device statements. JES2 starts FSS printers with a valid FSS procedure whether or not UNIT= has been specified. After initialization completes, operators can dynamically assign or reassign a physical device by specifying the UNIT= parameter on any of the following commands:

- $T RDR(nn)
- $T PRT(nnnn)
- $T PUN(nn)
- $T LINEnnnn

Operators can dynamically assign as many local printers through the $ADD PRT(nnnn) command. However, dynamically assigned devices are saved across a JES2 hot start only.

Note:

1. To define a device to JES2 permanently, installations must specify a value on the UNIT= parameter in the initialization stream.
2. You must first drain a device before replacing it.
3. You can dynamically add a local printer (JES2- or FSS-managed) using an $ADD PRT(nnnn) command. You do not need to define spare initialization statements to allow the dynamic addition of printers when using the $ADD command.
4. If your installation is using Print Service Facility (PSF)* Version 1.4 or an earlier release of PSF, only those local FSS printers that have been defined to the base control program as installation-static can be dynamically assigned to JES2 using $T commands.

See *z/OS JES2 Commands* for additional information about dynamically assigning or reassigning local devices using $T commands.

Remote line and device configuration

JES2 supports both NJE and RJE. During JES2 initialization, you should:

- Identify and describe the NJE applications eligible to participate in an SNA NJE network with the APPL(avvvvvvv), NODE(nnnn), LINE(nnnn), and LOGON(nnnn) statements, or in a TCP/IP NJE network with the SOCKET(vvvvvvvv), NODE(nnnn), LINE(nnnn), and NETSRV(nnn) statements.
- Specify the maximum number of concurrent JES2/VTAM sessions allowed with the SESSION parameter on the TPDEF statement.
- If you want compaction, specify the compaction table or tables you want to use (with the COMPACT statement).
- Specify which RJE workstation(s) and devices will use compaction (with the RMT(nnnn), R(nnnnn).PR(m), and R(nnnnn).PU(m) statements).

  Note that you can also specify compaction on the $ADD RMT(nnnn) command and $T forms of each command listed above.

- Specify the maximum number of nodes (with NODENUM parameter on the NJEDEF statement) in the NJE network to which this JES2 member belongs.
- Identify and specify characteristics for each line, RJE workstation, NJE node and remote device. (By using the LINE(nnnn), RMT(nnnn), NODE(nnnn), R(nnnnn).RD(m), R(nnnnn).PR(m), and R(nnnnn).PU(m) statements.)

  Note: If you do not define lines during initialization, you cannot add RJE workstations through the $ADD RMT(nnnn) command. Also, JES2 sets the RMTNUM= parameter on the TPDEF initialization statement to zero and ignores any RMT(nnnn) initialization statements.
- Specify the number of network SYSOUT and job transmitters and receivers (with the SRNUM=, STNUM=, JRNUM=, and JTNUM= parameters on the NJEDEF or LINE statements) for each NJE line.

You can dedicate (permanently attach) physical and logical (that is, SNA) teleprocessing lines by specifying the line number in the RMT(nnnn) statement that describes the RJE workstation. Any RJE workstation dedicated on a line must use that line for the connection. Any RJE workstation or device that does not have a dedicated line specified can use any line not designated by a RMT(nnnn) statement. This type of line is a nondedicated line.

Network job entry (NJE)

JES2 supports communication with remote JES2 complexes through channel-to-channel adapters (CTCAs), binary synchronous communication (BSC), synchronous data link control (SDLC), system network architecture (SNA) communication facilities, and Transmission Control Protocol/Internet Protocol (TCP/IP). See Chapter 5, “Network job entry (NJE),” on page 251 for a complete description of NJE .

Remote job entry (RJE)

JES2 supports both BSC and SNA RJE terminals. Configuring these devices consists of defining RJE workstation facilities, teleprocessing lines, and logical lines for SNA terminals (for example, a 3770, 3790, or a System/32 workstation). The remote configuration can range from one remote terminal (for example, a 2770 or 3780) to an RJE workstation consisting of a system operating many devices. Support of multiple logical units permits the concurrent use of more than one device at an RJE workstation.

You define the devices for an RJE workstation through the RMT(nnnn) statement and the $ADD RMT(nnnn) command. Use the RMTNUM= parameter on the TPDEF initialization statement to specify the highest number you can define at your installation. If you do not specify this parameter at initialization, it defaults to the highest-numbered RJE workstation defined during initialization. If at initialization, you define an RMTNUM= value and inadvertently then specify an RJE workstation with a higher number, JES2 sets RMTNUM= to the higher value. On both the RMT(nnnn) statement and the $ADD RMT(nnnn) operator command, you can define up to a maximum of 15 devices as follows:

- **Up to 8 devices** that can consist of a combination of printers, punches, and an optional SNA console. You can define:
  - 1 SNA console
  - Up to 7 printers (or only 6 if you define an SNA console)
  - Up to 7 punches
- **Up to 7 devices** that consist of readers only.

Specifying SETUP=PDIR on either the RMT(nnnn) statement, the $ADD RMT(nnnn) command, or the $T RMT(nnnn) command allows you to spool data sets and create multiple copies of output (from a single, transmitted copy).

The RJE workstation operator controls the RJE workstation, the RJE workstation devices, and the jobs submitted through the remote console or the remote card reader. JES2 considers each RJE workstation an extension of the local JES2 facility. For more detailed information about remote devices controlled by JES2, see Chapter 6, "Remote job entry (RJE)," on page 325.

Directing responses to display commands

In some situations, you might want to send the output of a display command to a console other than the one on which the command was issued. For example, in the case of a display command that sends large amounts of output to the console, you might want to send the output to another console. Use the REDIRECT(vvvvvvvv) initialization statement to redirect the output of a display command from the in-line area of the console on which you entered the command to another console or to another area of the console. For example, the initialization statement,

```
REDIRECT(mastcons) DCONNECT=njeconsl-c
```

sends the output of commands that display network connections from the console named **mastcons** to the out-of-line area C of the console named **njeconsl.** See *z/OS MVS Planning: Operations* for information on planning and defining consoles.

z/OS JES2 Initialization and Tuning Reference contains a complete description of the REDIRECT initialization statement.

Job submission

JES2 accepts jobs through:

- Card readers allocated to JES2
- RJE devices allocated to JES2
- NJE job receivers from other nodes in an NJE network in which the local JES2 is a member
- Offload job receivers
- The JES2 internal reader facility, through which you submit jobs from TSO/E and started tasks (see *z/OS JES Application Programming* for details about the facility)

JES2 queues the jobs submitted to it in priority order. Use the various initialization statements and parameters to control input device characteristics such as input streams, job classes and their attributes, job priorities, whether to initially hold the job, or a job's default performance group. You can change the attributes of jobs coming into this system through the JES2 input service exits 2, 3, 4, 20, 50, 51, 52, 53 and 54. To enable these exits, see *z/OS JES2 Installation Exits*.

Card readers

JES2 automatically starts local card readers if you specify the START=YES parameter on the RDR(nn) JES2 initialization statement. The operator can then enter job streams into the system by placing the cards in the hopper and pressing the start button on the reader; the operator does not have to issue a JES2 start ($S) for the reader. The operator deallocates the card reader from JES2 by issuing the JES2 stop ($P) reader command.

Network jobs

In the NJE environment, you can pass jobs read into the system through the input service processor to other host nodes in the network for execution service. The input service processor treats jobs you want processed at other nodes in the network the same as jobs you want executed locally. This allows JES2 to determine the node of execution at any time during the scheduling process.

You can route job output from one node to another by:

1. Specifying a symbolic destination identifier on the following:
   - The /*ROUTE PRT JES2 control language statement
   - The /*ROUTE PUN JES2 control language statement
   - The DD JCL statement,
   - The OUTPUT JCL statement.
2. Using the same symbolic destination identifier as the subscript of the DESTID(jxxxxxxx) initialization statement.

For example, if Node 1 uses a symbolic, PRTBIG, to define a printer on Node 2, NODE 2 can change that printer in the future from remote 7 to remote 8 by specifying R8 on the DEST= parameter without having to advise Node 1 of the change.

At NODE 1, this is the DESTID(jxxxxxxx) initialization statement,

```
DESTID(PRTBIG) DEST=N2.PRTBIG
```

At NODE 2, this is the DESTID(jxxxxxxx) initialization statement,

```
DESTID(PRTBIG) DEST=N2.R7
```

Processing JES2 control statements

The input service processor also processes the /*XEQ control statement, /*ROUTE XEQ control statement, and /*XMIT control statement. JES2 converts the execution node specified in any of these control statements to a node number or specified userid, replacing the default (or previously specified) node number.

You can send jobs for execution on a guest virtual machine on a VM node by specifying a symbolic destination defined on a JES2 DESTID(jxxxxxxx) initialization statement. If you do not specify a symbolic destination, JES2 processing moves the second-level route code from the /*XEQ or /*XMIT JECL statement. VM uses the execution userid as the destination virtual machine for the jobs it receives.

If the execution node number does not match the input node, JES2 will not process any execution-related control statements. JES2 processes JES2 control statements on the input node and the execution node when they appear before the /*XEQ (or /*ROUTE XEQ) control statements. If you place the /*XEQ (or /*ROUTE XEQ) statement before other JES2 control statements, JES2 processes only the/*MESSAGE and /*SETUP control statements at the input node if they come before the /*XEQ statement. All other control statements are **not** processed until they reach the execution node.

Execution-related control statements include the following JES2 control statements:

- /*MESSAGE
- /*SETUP
- /*ROUTE (PRINT and PUNCH)
- /*JOBPARM

- /*OUTPUT

Use the /*XMIT control statement to submit data into the network. The transmitter considers all data following an /*XMIT statement, up to a delimiter statement as one network job. The delimiter statement defaults to /*, but can be changed through the DLM= parameter on the /*XMIT JES2 control language statement. For information about specifying this delimiter, see the *z/OS MVS JCL User's Guide*. The network job is transparent to all nodes but the receiving node.

Network job transmitter

A network job transmitter selects jobs from the job transmission queue. If there is a network job header and trailer, (the job has been received from a node other than the node where the job was created), the job transmitter updates them as appropriate. For example, if the job was destined for execution at this node but routed to another node (through either a $R operator command or a /*XMIT control statement), the job transmitter places the new job card information in the job header. If the job was received from the origin node, the job transmitter builds the NJE headers and trailers so that the job can be sent across the network.

The job transmitter sends the job exactly as the system read it in. The transmitter does not send any statements added by the local input service processor and recovers, for transmission, any statements which the input service processor deleted or modified. Because JES2 preserves the original job stream on the spool disk, the job transmitter can reconstruct the original job.

You can store job-related information with both job and SYSOUT data sets, and transmit installation-specific data from node to node using the $JCTX and $NHD services. See sample exit HASXJECL in the AHASSAMP JES2 distribution library. For more information about using these services, see *z/OS JES2 Macros*.

Network job receiver

The network job receiver component, which receives jobs over the NJE lines, is similar to the input service processor. The network job receiver reads in jobs from other host nodes in a manner similar to the method used to read in jobs from RJE workstations. The difference is the network job receiver expects NJE protocols from the other host nodes and there are some changes to job processing. When the receiver detects a job destined for another node, it reads the job in to the spool without examining the data. The node then uses the network job transmitter to continue the job's journey to the ultimate destination.

When a node receives a job from the network, JES2 uses the header and trailer it receives from the network job transmitter. If the received job header or trailer does not contain all the sections necessary for JES2 processing, JES2 adds them to the headers received. The JES2 input service processor treats multiple jobs received in a single file as a single job. Sending multiple jobs through the network in a single file could cause unpredictable results; a job might execute at a node other than the one you intended.

Identifying jobs received from the network: When a node receives a job from the network, the node attempts to assign the job identifier (JES2 job number) that appears in the jobheader, if the identifier is not in use. If unable to use the original job identifier, the input service processor assigns a new number as if the job entered the system on a device attached to the executing processor. However, the input processor does not replace the job identifier that appears in the job header. The original job identifier remains in the job header and remains with the job as it

progresses through the network. (You can exercise some control over the job identifiers by using the RANGE= parameter on the JOBDEF initialization statement.)

After the node receives the job, the input service processor queues the job for either JCL conversion or job transmission, depending on the value of the execution node in the job header. If the execution node in the job header matches the node identifier of the node that received the job, the input service processor queues the job for execution. If the execution node does not match, the job continues through NJE lines until the job reaches the execution node specified in the network job header.

Data previously spooled without examination (for example, data transmitted with a /*XMIT statement or data spooled for transmission at an intermediate node), and rerouted locally must again go through input service processing. A logical route transmitter and route receiver reroute network jobs to the local node.

Controlling network job headers and trailer areas

Installations can enhance the performance of their JES2 members by controlling the amount of storage allocated for NJE headers and trailers through the HDRBUF= parameter on the NJEDEF initialization statement.The HDRBUF= parameter allows you to specify the number of cells allocated at initialization through the LIMIT subparameter. Use this subparameter with great care; specifying too small a value could decrease member throughput by creating resource contention for the headers and trailers.

Consider the number of jobs actively transmitted or received by your installation at one time when setting this parameter. For most installations, the JES2 default provides enough cells for NJE headers and trailers.

However, if your installation transmits many small jobs (100 lines or less) across multiple active NJE lines, JES2 transmits headers more frequently and might need a larger number of buffers. One sign that indicates your installation requires a larger LIMIT value is a shortage of virtual storage. For the exact specifications of this parameter, see *z/OS JES2 Initialization and Tuning Reference*.

Controlling network job transmitters and receivers

Control the number of network job transmitters and receivers associated with each NJE line by specifying the JTNUM= and JRNUM= parameters on the NJEDEF statement. Use the STNUM= and SRNUM= parameters on the NJE statement to control the number of SYSOUT transmitters and receivers associated with an NJE line. The values specified for these parameters are related (see *z/OS JES2 Initialization and Tuning Reference*, for coding restrictions).

Job transmitter selection: The NJE job transmitters select files destined for nodes reachable through the NJE line with which they are associated. Job transmitters select files from the transmission ($XMIT) queue, which is ordered FIFO (first-in-first-out) within priority. You can specify multiple transmitters (and receivers) for a line as described in “Specifying transmitters and receivers” on page 272. You can specify work selection characteristics for job transmitters and for SYSOUT transmitters.

Conversion

The converter routine processes the JCL for a job, logon, or started task and converts the JCL into converter/interpreter text. The job is then available for execution, which occurs as soon as an initiator eligible to process the job is available.

JCL conversion

Unless a job is held, the job is eligible for JCL conversion as soon as JES2 places the job in the queue. The job converter, which is invoked for every job, receives converter parameters and a pointer to a cataloged procedure library from JES2.

There are two options for JES2 conversion processing: one calls the interpreter as part of the conversion process; the other calls the interpreter when the job is selected for execution. Calling the interpreter as part of the conversion process detects any errors earlier, and also processes OUTPUT JCL statements even if the job does not execute. However, this option requires using some additional spool space and has implications for a number of JES2 installation exits, particularly Exit 6.

The INTERPRET= parameter on the JOBDEF statement controls when the interpreter is called for a job. Specifying INTERPRET=INIT is the traditional method of calling the interpreter—when a job is selected for execution. Specifying INTERPRET=JES calls the interpreter during the conversion phase, which occurs before the job is selected for execution. When INTERPRET=JES is specified, the converter and the interpreter are both called in a persistent address space. The number of address spaces that are created depends on the CISUB_PER_AS setting on the JOBDEF statement. The number of conversion processes (PCEDEF CNVTNUM=), divided by the number of subtasks per address space (CISUB_PER_AS), can be used to calculate the number of address spaces that are created. The default number of created address spaces is 2, and the maximum number is 25.

Note: The INTERPRET parameter is only valid when JES2 is in z11 mode. The INTERPRET parameter is not supported in z2 mode.

The name of the address spaces are jes*x*CI*nn*, where jes*x* is the JES2 subsystem name and *nn* is a number (01-25) to create unique values. This address space accesses the PROCLIB data sets that are defined in the JES2 start PROC, which use the JES2 dynamic PROCLIB service.

You must ensure that a valid user ID is assigned to the address space (presumably the same user ID that is assigned to the JES2 address space), by using entries in the started procedures table (ICHRIN03) or by defining STARTED class profiles that match each new address space name. Optionally, you can make both the started procedures table and STARTED class profile available. Ensure that the user ID has read access to the data sets that are defined in the JES2 PROCLIB concatenations, either in the JES2 PROC or using the JES2 dynamic PROCLIB service.

You can enable Exit 6 (if the conversion phase occurs in the JES2 address space) or Exit 60 (if the conversion phase occurs in the JES2CI address space) to scan and change the Converter/Interpreter text: refer to *z/OS JES2 Installation Exits*.

Scanning the JOB statement accounting field

Use the ACCTFLD= parameter on the JOBDEF statement to control whether JES2 is to scan the accounting field of the JOB statement. The accounting field is valid if its format matches the format specified for JES2. (For a description of the JES2 format for the accounting field, see the accounting field parameter described in *z/OS MVS JCL Reference*.)

JOB statement accounting field scan exit

You can supply an Exit 3 or Exit 53 installation exit routine to scan and change the accounting field in the JOB statement. See *z/OS JES2 Installation Exits* for information about Exit 3 or Exit 53 and for the exit routine's relationship to the ACCTFLD= parameter on the JOBDEF initialization statement.

Network accounting

The JES2 NETACCT initialization statement determines the correspondence between the network account numbers and the local account numbers for a node. JES2 uses the parameters on the NETACCT statement to build the network account table used by input processing, the job and SYSOUT transmitters, and the job and SYSOUT receivers.

JES2 recognizes the following network accounting control statement:

/*NETACCT=vvvvvvvv
: where vvvvvvvv is a 1- to 8-byte alphanumeric character network account identifier. When the input service processor encounters this statement, it inserts the network account identifier in the job's job header and looks up the number in the network account table built at JES2 initialization, to find an associated local account number. If a local account number is found, JES2 will move the number into the job's job control table, overriding any account number present in the JOB statement. Processing always interprets the /*NETACCT statement, regardless of its position in relation to the /*ROUTE XEQ or /*XEQ statement.

When transmitting a job or its system output, JES2 sends the network account number as part of the job header. At each node, the job receiver and SYSOUT receiver perform the same table lookup as that of input processing did to obtain a local account number for use by that node. Thus, accounting is a local function using whatever set of account numbers the system programmer defined on a given node.

Converter parameters

The JOBCLASS(v) initialization statement contains parameters that define an installation's defaults for converter parameters. If you do not specify the converter parameters, JES2 assigns its own defaults during initialization processing. The converter parameters you can affect include; estimated execution time, JCL and allocation MSGLEVEL options, command disposition and authority, and bypass label processing options. See *z/OS JES2 Initialization and Tuning Reference* for descriptions of the specific parameters.

Defining a job's procedure library

SYS1.PROCLIB contains the JES2 cataloged procedure. This procedure defines the job-related procedure libraries. Figure 15 on page 80 is an example of defining procedure libraries. All the libraries reside in the system's master catalog (or in a user catalog, provided the data set has a high-level qualifier other than SYS1).

```
//PROC00    DD    DSN=SYS1.PROC1,DISP=SHR
//PROC01    DD    DSN=SYS1.PROC2,DISP=SHR
...
//PROCnn    DD    DSN=SYS1.LASTPROC,DISP=SHR
//anyname   DD
```

Figure 15. Specifying Procedure Libraries in the JES2 Procedure

The procedure library a job uses depends on the value of the PROCLIB= parameter on the JOBCLASS(v) initialization statement. Programmers can override the default specification by specifying a /*JOBPARM PROCLIB= statement in their JCL, and coding the name of the DD statement in the JES2 procedure that points to the library they want to use. For example, a programmer is running a job in class A. That class has a default PROCLIB of SYS1.PROC1. The programmer wants to use a procedure that resides in SYS1.LASTPROC. That programmer would include a /*JOBPARM PROCLIB=PROCnn in the JCL.

Another way to specify a procedure library is to use the JCLLIB JCL statement. See *z/OS MVS JCL Reference* for more information on the JCLLIB statement.

Any library that you will specify in initialization statements must have a name in the PROCnn form in the JES2 procedure or PROCLIB statements defining the logical PROCLIB. Other procedure libraries might have any DD name that follow the general rules for DD statements. If you require multiple data sets for one DD name, specify those data sets as concatenations in the JES2 cataloged procedure.

If the procedure library is not specified or specified and not found, PROC00 is used.

Job selection and execution

Jobs exist to be read in, converted, classified, and ultimately run. Many factors can affect how, where, when, and if they will run, such as:

- Job Class
- Control of Initiators by JES2 or WLM
- Job Selection Priority
- Job Hold Status
- Member Affinity
- Scheduling Environment
- Job Name
- Number of other Jobs queued for Execution
- Configuration of Active Initiators
- Workload Manager Policy

This section describes each of these factors and how they influence job flow during the execution phase.

The JES2 job queue

JES2 receives a job, places its JES2 control blocks on spool, and creates a job queue entry (JQE) on the JES2 job queue. JES2 uses a job queue index (JIX) based on the job number to more efficiently access the jobs on spool.

The queue entry for each job contains the job name, job priority, the IDs of the entry and execution nodes, a flag that indicates the held status of the job, pointers

to JES2 control blocks on the spool, and the next JES2 process for which the job is eligible (transmission to another node in an NJE network, JCL conversion, execution, output processing, purging).

Batch jobs are selected for execution by initiators, each one running in a separate address space. Initiators are controlled by JES2 or by MVS workload management (WLM). Who controls the initiators is determined by the job class and by the MODE= parameter (JES or WLM) on the JOBCLASS statement. JES2 maintains two different queue organizations for all jobs awaiting execution to service the two different selection mechanisms:

1. All jobs are queued by job class, priority, and the order in which they finished conversion. This is the queue from which JES2 managed initiators select jobs for execution. See "JES2 control of batch job initiation" on page 89 for more details.
2. Jobs awaiting execution in WLM managed job classes are also queued by their WLM assigned service class in the order they were made available for execution. This is the queue from which WLM managed initiators select jobs for execution. See "WLM control of batch job initiation" on page 93.

Started tasks and Time Sharing Users do not need initiators to begin execution. They are not affected by workload management nor other jobs in the queue but are initiated immediately by JES2.

Job class

JES2 provides 38 predefined job classes—the started task control (STC) and time-sharing logons (TSU) job classes, and 36 one character (A-Z and 0-9) execution job classes. Additional user-defined 2-8 character execution job classes are supported. Up to 512 total execution job classes can exist in a MAS.

For details on adding and deleting job classes, refer to the $ADD JOBCLASS and $DEL JOBCLASS commands in *JES2 Commands*.

Note:

1. The $ADD JOBCLASS command requires that all members of the new job class are running z/OS version 2.1, or later.
2. The $DEL JOBCLASS command can only be used for classes that were created by the $ADD JOBCLASS command.
3. None of the 38 predefined job classes can be deleted; however, they can be marked as inactive.

The CLASS= keyword on the JOB statement identifies the execution class for the job. If not coded on the JOB statement, the class associated with the particular device that the job entered the system on becomes the job's execution class. The INTRDR initialization statement defines a logical device for jobs entering the system through the internal reader facility.

There are no absolute rules for assigning job classes, and some experimentation is necessary.

Note: The CLASS= keyword on the JOB statement is ignored for started task jobs.

Job class characteristics

During JES2 initialization, you can assign job characteristics to jobs queued in each class to determine:

- A default performance group for each job.
- JCL conversion parameters.
- Whether to initiate jobs in this class by JES2 managed initiators or WLM managed initiators.
- Whether to impose limits on the maximum number of jobs that can be in execution at one time in the MAS.
- Whether to hold jobs in this class after conversion.
- Whether to copy jobs in this class directly to the message class output, without undergoing conversion or execution.
- Whether to convert the job and queue for output without execution.
- Whether to produce a JES2 job log for jobs in this class. (The JES2 job log is a list of all messages and replies issued by, or on behalf of, a job.)
- Whether to save a job journal for jobs in this class. If you use checkpoint restart or restart a job step, you need to save the journal or the system can not automatically restart the job if it fails or if there is a system restart.

  If you use automatic restart management to restart a job, you do not need to save the journal because automatic restart management of MVS does not use the job journal when restarting jobs.
- If execution batch monitoring facility jobs can run in this class.
- Whether to suppress output for jobs in this class (for example, started tasks).
- The procedure library (PROCnn) definition.
- SMF options.
- If the system can restart this job in the event of a JES2 warm start.
- JESLOG options.

When first creating your initialization data set(s), you can define a “universal” job class statement and specifications. Define JOBCLASS(?) as a single JOBCLASS statement; it provides a “universal” set of characteristics for **all** 36 job classes. (The '?' wild-card character refers to the one-character job classes, as opposed to '*' which would include the 'STC' and 'TSU' classes.) Then add specific JOBCLASS(v) statements after this statement for those classes that require individual specification.

JES2 overrides previously-defined initialization statements with duplicates of the statement it reads later in initialization processing. In this manner, only specific JOBCLASS(v) statements override their “universal” JOBCLASS(?) or JOBCLASS(*) specification and the “universal” characteristics remain valid for all other “non-specifically” defined classes.

For example, in the following initialization statements:

```
JOBCLASS(?) AUTH=IO,MSGLEVEL=(1,1),PERFORM=10
:
JOBCLASS(X) AUTH=INFO,IEFUSO=NO
```

Job class X overrides the previous value of AUTH=IO with AUTH=INFO, and the IEFUSO value (which defaulted to YES) is changed to NO. Job class X uses the values for MSGLEVEL= and PERFORM= from the JOBCLASS(?) statement.

A default SCHENV can be specified for each JOBCLASS (not for JOBCLASS(STC) or JOBCLASS(TSU)). This value will be provided each job at the end of JCL conversion provided that a SCHENV has not already been associated with the job.

A SCHENV can be associated by the user specifying SCHENV= on the // JOB statement or by JES2 installation exits which occur before the end of conversion.

SPINNING of JESLOG data sets is specified by the JESLOG operand.

Jobs are assigned to JES2 or WLM managed initiators based on the MODE parameter on the JOBCLASS statement. JES2 and WLM initiators select jobs using different criteria:

- JES2 uses job class and scheduling priority as selection criteria. See "JES2 control of batch job initiation" on page 89 for details.
- Workload management uses service class as selection criteria. See "WLM control of batch job initiation" on page 93 for details.

Note: All JOBCLASS parameter definitions are MAS-wide, and can only be changed with an operator command or a JES2 cold start.

Job class groups

To avoid a large number of 2-8 character job class names being associated with a single initiator or a device, you can create job class groups to manage these associations. In a manner similar to placing NJE nodes in SUBNETs, job classes can be defined to a job class group, as follows:

- A job class can only be in one job class group or in no job class group.
- A job class group is created when the first job class is added to the group.
- A job class group is deleted when the last job class is removed from the group.
- Deleting a job class also deletes the job class from its job class group.
- The maximum number of job class groups is 512 (in which case each group would contain one job class).
- Job class group names must be unique, from 2-8 alphanumeric characters and cannot match any existing job class name.

WLM classification

JES2 uses workload management classification services to assign a service class to each job after it is converted, regardless of its job class. The service class is assigned based on WLM rules which can include attributes such as Job Class, Job Name, Job Priority, Userid, Job Accounting Information, and PERFORM= specification on the JOB statement. See *z/OS MVS Planning: Workload Management* for details.

You should take care not to classify both JES2 managed classes and WLM managed classes into the same service class. Workload management may not be able to effectively meet its goals if some jobs are initiated through JES2 and others through the WLM.

Modifying jobs and classes

JES2 allows each installation to further control batch jobs by allowing you to:

- Switch job classes between JES mode and WLM mode with the $T JOBCLASS,MODE= command.
- Change the job class, or priority of a particular job with the $T Job command. (JES2 will call WLM to re-classify the job.)
- Modify the service class of a batch job with the JES2 $TJob,SRVCLASS= operator command.

Any of these actions may change the service class or scheduling characteristics of the job. For more information, see the JOBCLASS(v) statement in the *z/OS JES2 Initialization and Tuning Reference*, and the $T JOBCLASS and $T Job statements in the *z/OS JES2 Commands*.

Job scheduling priority

The priority on the /*PRIORITY statement determines the job's scheduling priority for JES2 managed initiators. If not specified, JES2 calculates this priority based on estimated execution time and the JOBPRTY(n) initialization parameters. You can increase the priority with the PRIOINC parameter up to the PRIOLIM limit on the RDR(nn), Rnnnnn.RD(m), or INTRDR initialization statements.

Jobs in JES2 controlled job classes are queued FIFO within priority within job class. Priority is not used directly to order jobs in WLM managed classes, but can be used as a criteria for determining service class.

Specifying priority

Specify priority on the /*PRIORITY statement if you have coded PRTYJECL=YES on the JOBDEF initialization statement (or allowed the parameter to default). If specified, the /*PRIORITY statement must immediately precede the JOB statement, or JES2 flushes the input stream until it finds another JOB or /*PRIORITY statement.You can also specify priority by using the PRTY= keyword on the JCL JOB statement provided PRTYJOB=YES appears on the JOBDEF statement of the initialization stream you used to initialize JES2.

Calculating priority

When a /*PRIORITY or PRTY= on the JOB statement do not specify the scheduling priority, JES2 calculates the priority from the estimatedexecution time. Supply these estimates through the: /*JOBPARM statement, the accounting information on the JOB statement, or the ESTIME initialization statement.

To calculate priority, JES2 uses the PRIORITY= and TIME= parameters on the JOBPRTY(n) initialization statement as a table of values. You can supply these values on the JOBPRTY(n) statement or allow them to default. The following example uses the default values to calculate priority:

Table 12. Priority Table Example

If TIME<=*	Then JOBPRTY(n)=	Then PRIORITY=**
2	(1)	9
5	(2)	8
15	(3)	7
279620	(4)	6
.	.	.
.	.	.
.	.	.
279620	(8)	2
279620	(9)	1

Table 12. Priority Table Example (continued)

If TIME<=*	Then JOBPRTY(n)=	Then PRIORITY=**

* The TIME values are estimates of the number of minutes a job requires to execute.

** These are the defaults for corresponding n values. For example, if you specify TIME= to estimate 4 minutes execution time,

```
4 minutes implies n=2 and n=2 implies PRIORITY=8
```

If the job is placed on the HARDCOPY queue, JES2 ignores the job priority (for example, 8, as calculated in the above example); the actual priority used (in this example) is the output priority.

By specifying other values for the tables (during JES2 initialization), you can more closely control your installation's priority assignments. If you build your own priority table based on estimated execution time, be certain the values that you supply appear in ascending order (as are the values in the default table in Table 12 on page 84). The order is important because JES2 uses the TIME= values in a "less than or equal to" order when calculating a priority that does not exactly match the values in the table.

For example, in Table 12 on page 84, if you submitted a job with an execution time of 10, JES2 calculates n as 3 because 10 is less than or equal to 15. If the estimated execution time is 16 to 279620, n is 4. You can use n values 4 through 9 to further control you installation's priority requirements.

Note: Values specified on the JCL /*JOBPARM statement override those in the accounting field of the JOB statement.

During JES2 initialization, you can specify the ACCTFLD parameter on the JOBDEF initialization statement and cause JES2 to ignore the accounting field on the JOB statement.

Member affinity

In addition to using job classes and initiator selection, batch jobs can specify which members in the MAS can process them by explicitly coding the member names on the SYSAFF parameter of the /*JOBPARM statement in their JCL. The member IDs are explicitly specified through this mechanism. You can modify the member affinity of a batch job with the $TJob,SYSAFF= command.

Job scheduling environment

Similar to but independent from member affinity, a job may also be assigned a "scheduling environment" to ensure it executes on the members in the MAS which have the required resources, or specific environmental states. The states may be associated with hardware resources such as a vector facility, an address space such as DB2®, or an abstract state such as time of day (shift) or day of the week. The environment can be specified with the SCHENV= keyword parameter on the JOB statement, or through an installation exit. They can also be assigned by a JOBCLASS default or using the $T Job JES2 command. Scheduling environments are installation-defined 16-character names which may be available on any of the MVS systems in the sysplex, or may be available on none of the systems.

Use workload management to define the scheduling environments and make them available or unavailable on each system, based on the "ON" or "OFF" state of resources.

The scheduling environment is validity-checked by the converter and must be defined to workload management, or else the jobs will fail with a JCL error. JES2 detects the availability of scheduling environments on each member, and allows an initiator to select jobs only if the specified scheduling environment is available. This is true for both JES2-managed initiators and WLM-managed initiators. In addition, any member affinity specified through the SYSAFF= parameter as well as the job class and initiator class must match.

For more information on the definition and control of scheduling environments, see *z/OS MVS Planning: Workload Management*.

Displaying and controlling scheduling environments

JES2 can display the scheduling environment for a specified job and list those jobs waiting for a specified scheduling environment. Use the `$DJnnnnn,LONG` command to display the environment specified for the job, or `$DJOBQ,SCHENV=environ_name` to display jobs waiting for a specific scheduling environment. In addition, you can display why a job will not run with the DELAY parameter, in which the reason might be because the scheduling environment (SCHENV) is not available.

When a scheduling environment is assigned to a job, you can change it or remove it using the $T Job JES2 command.

SDSF options: You can display the job scheduling details about specific jobs with the "Job Information" pop-up on the job input (I) panel. You can also display all the scheduling environments defined to the WLM and on which members in the MAS each is available using the SDSF "SE" option.

In addition, authorized SDSF users can display and set the state of various resources on each member using the "RES" option.

See *SDSF Guide and Reference* for specific SDSF information.

SMF recording: To help you monitor and manage the scheduling environment, SMF type 26 and 30 records have been expanded to include the scheduling environment name.

See *z/OS MVS System Management Facilities (SMF)* for details.

Duplicate job name control

By default, JES2 enables duplicate batch job name protection for all members of the multi-access spool configuration. That is, if one batch job name matches another batch job name that is active in execution anywhere in the configuration, the second job is temporarily delayed until the first job completes. You can disable duplicate batch job name protection by specifying DUPL_JOB=NODELAY on the JOBDEF initialization statement.

CAUTION:
If you specify DUPL_JOB=NODELAY, jobs with the same job name may run concurrently with adverse effects. Make certain that this serialization mechanism is not being used in your environment before changing this parameter.

Held jobs

JES2 does not remove a held job from the queue; instead JES2 marks the job ineligible for selection. You can hold a job at any time. Thus, when an operator holds a job in execution, the job is not eligible for output processing until the operator releases it. You can hold a particular job, a job class, or all jobs.

If you specify TYPRUN=HOLD on a JOB statement and then transmit that job through the network, the execution node holds the job after conversion of the JCL. If the job card specifies TYPRUN=JCLHOLD, the entry node and each of the destination node(s) holds the job.

Setting job class limits for execution

You can limit the number of jobs in each class that can execute simultaneously in the MAS by using the XEQCOUNT=MAXIMUM= parameter on the JOBCLASS statement. This applies to JES2 and WLM managed job classes, and can be altered with the $T JOBCLASS command. There is no facility to control the number of jobs in execution by class on an individual system basis.

Enabling job execution selection

You can stop or start the selection of batch jobs by JES2 or WLM managed initiators on an individual system with the $P XEQ or $S XEQ command.

The initial setting when JES2 first initializes is to allow selection. If you want to prevent this, place a $P XEQ command in your JES2 initialization deck.

Controlling job execution through exits

You can alter the characteristics of a job through the following exits:

- Exit 2 or 52 for JOB statement scan
- Exit 20 or 50 for end-of-job-input processing
- Exit 51 for end of phase processing
- Exit 6 or Exit 60 for converter/interpreter text

SMF exit IEFUJV can also change the job's scheduling attributes.

When jobs are queue for execution, you can also use these JES2 exits to control the job selection:

- Exit 14 for JES2 managed initiators only (do your own job selection)
- Exit 49 for both JES2 and WLM managed initiators (accept or reject the selected job)

See *z/OS JES2 Installation Exits* for information about these exits.

Controlling the sequence of job execution

Because multiple JES2 subsystems alter the execution queues in a shared-spool environment, jobs in the same class with the same priority can execute out of their reader (job number) sequence. You should be aware of the how various job stream environments are affected by this execution out of sequence and the steps you can take to control the execution sequence.

Serial job execution sequence

If you define in the PCEDEF initialization statement that your MAS is to have only one converter processor (CNVTNUM=1), then you force the JES2 system to convert

and execute jobs in the order of their reader (job number) sequence. Although this setup controls the order of job conversion and execution, it has the following implication and disadvantages:

- JES2 is limited to processing your workload in a serial manner without respect to the types of jobs that are being submitted. Large batch jobs and TSO logons are treated in a similar way.
- Each of the job's resources are required when the job executes. As such, any migrated libraries and data sets that the job needs, must be restored before the job can execute. The time spent waiting for these migrated resources can backup other jobs in the queues.
- System hangs and deadlocks can occur in the system. When jobs are waiting for resources that have been migrated, the migration program needs to run and restore these resources. The migration program is often a job that needs JES2 conversion before it running. Since only "1" JES2 converter processor is defined to do the conversion and it is waiting, the migration cannot occur and the system becomes deadlocked. As an example, this condition can occur for jobs that need JCLLIB data sets, which have been migrated by a product like DFHSM.
- If a job has an exclusive ENQ on a JCLLIB data set and another job needs that JCLLIB data set, the job must wait till the ENQ is released.

Concurrent job execution sequence

If you define in the PCEDEF initialization statement that your MAS is to have more than one converter processor (CNVTNUM=2 or CNVTNUM > 2), then concurrent JES2 converter job processing can take place. However, this concurrent processing does affect the sequence of job execution. Two or more jobs with the same class and the same priority can be in conversion processing at the same time. Whichever job finishes conversion first is then queued for execution first, regardless of the order in which the job was submitted. You need to consider the following implications when setting up your system with CNVTNUM=2 or CNVTNUM>2:

- JES2 conversion processing occurs in different ways depending on whether the conversion processor represents an "even" (2,4,6, and so on.) or "odd" conversion processor (3,5,7, and so on.). As an example, for CNVTNUM=5, you define "3" odd conversion processors (1,3,5) and "2" even conversion processors (2,4). An even-numbered processor continually processes jobs through JES2 conversion and does not wait. An odd-numbered processor waits for a job's resources to become available.

  Odd-Numbered Conversion Processing

  Odd-numbered conversion processing waits for the resources to be available. When the resources become available, conversion continues.

  There are two primary conditions that affect JES2 conversion processing:

  1. JCLLIB data sets need to be recalled for a job to proceed.

     Conversion processing waits for the recall of the JCLLIB data sets to occur then proceeds with the conversion.

  2. The JCLLIB data set is controlled (typically by an exclusive ENQ) by another job.

     Conversion processing waits for the exclusive control to be released.

  Even-Numbered Conversion Processing

  Even-numbered conversion processing does not wait; it requeues jobs that need resources so that they are processed by the odd-number conversion processor. Jobs that have all their resources are converted immediately.

Note: To have at least one converter PCE free to process jobs at all times, you need to define a minimum of "2" converter PCEs in the PCEDEF initialization statement.

- If you need one job selected before another and you accept the "default" number of conversion processors or define more, you can control the sequence of job execution through the following types of procedures:
  - Delay the selection of the second job by coding TYPRUN on the JOB statement for that job
  - Specify a job class for the second job that will force the job into hold status
  - In a shared spool environment, assign a specific system affinity to each reader in your system through the RDR(nn) initialization statement, $T RDRnn operator command, or on the /*JOBPARM statement for the job.
  - Add an additional step to the first job to submit the second job to the internal reader. As a result, the first job must complete before the second job can run.
  - You can affect the status of jobs entering the system, and the order in which JES2 selects jobs for processing by using the following JES2 installation exits:
    - Exit 2 or 52 for JOB statement scan
    - Exit 20 or 50 for end-of-job-input processing
    - Exit 51 for end of phase processing
    - Exit 6 or Exit 60 for converter/interpreter text
    - Exit 14 for job queue work selection (before a job is selected)
    - Exit 49 for job queue work selection (after a job is selected by QGET)
    - Exit 22 for processing of TSO/E STATUS and CANCEL commands
    - Exit 44 for determining whether a job waits for migrated resources

    z/OS JES2 Installation Exits gives details about using these exits.

JES2 control of batch job initiation

After conversion of a job's JCL, JES2 places the job into an appropriate class queue to await execution. If the job class is a JES2 managed class (JOBCLASS MODE=JES), then JES2 initiators select a job from the class queue (depending on the job's priority) and pass control to MVS to execute your program. If the job class is a WLM managed class (JOBCLASS MODE=WLM), see “WLM control of batch job initiation” on page 93.

During execution, the initiator selects non-held jobs in priority order within their class, and the non-held classes in the order specified for that initiator. That is, the JES2 initiator selects the lowest priority job in the first non-empty class ahead of the highest priority job of the next class. Exceptions to these selection criteria are explained above and include:

- Held Jobs (TYPRUN=HOLD, JCLHOLD, or operator hold)
- Held Classes
- Held Execution Environment ($P XEQ)
- Member Affinity (SYSAFF)
- Scheduling Environment
- Duplicate Job Name (optional in OS/390® Release 3)

With the introduction of 2-8 character job class names in z/OS 2.1, the syntax for specifying an initiator class list has changed. For MAS members running z/OS 1.x, the class definition and initiator syntax remains:

```
CLASS=ABCDE      - $D I(*), CLASS=A   WILL NOT MATCH
```

For MAS members running z/OS 2.1 or later, the class definition and initiator syntax is:

```
CLASS=(A,B,C,D,E) - $D I(*), CLASS=(A) WILL MATCH
```

In addition to listing job class names, you can initiate batch jobs using job class group names. When you initiate a job class group, there is no processing priority given to any of the job classes within the group; job classes within the group are processed in a round robin fashion.

Operator control of the batch job workload

The operator can ensure that the system resource manager has a sufficient number and variety of jobs to keep the system busy by varying the number of JES2 managed initiators and the classes from which they dequeue jobs. The mechanisms for such control are the JES2 commands to start, stop, set, and halt initiators ($S Ivvvv, $P Ivvvv, $T Ivvvv, and $Z Ivvvv). Each JES2 managed initiator controls the selection of one job at a time.

JES2 indirectly creates an address space for each active JES2 managed initiator. The operator can activate JES2 managed initiators and create additional batch job address spaces by means of the JES2 start initiator command ($S Ivvvv). In a similar manner, the operator can drain the JES2 managed initiators and cause termination of their address spaces by issuing the JES2 stop initiator command ($P Ivvvv).

The operator issues the halt initiator command ($Z Ivvvv) to deactivate a JES2 managed initiator, without terminating the address space (which is swapped out). This saves some time because JES2 need not recreate the address space if the operator starts the initiator later.

Table 13 summarizes the four useful commands for controlling batch workload (for syntax information, see *z/OS JES2 Commands*).

Table 13. JES2 Commands Useful in Controlling the Batch Job Workload

Desired Function	JES2 Command
Control the mix of batch jobs executable at the same time by assigning classes from which to select these jobs.	Set Initiator: $T INITvvvv
Activate JES2 managed initiators and cause creation of additional batch job address spaces.	Start Initiator: $S INITvvvv
Stop (“drain”) JES2 managed initiators and end batch job address spaces.	Stop Initiator: $P INITvvvv
Inactivate a JES2 managed initiator without terminating the address space	Halt Initiator: $Z INITvvvv

Note: The initiator identifier vvvv represents a 1- to 4-digit ID.

Use the DUPL_JOB= parameter on the JOBDEF statement or the $T JOBDEF,DUPL_JOB command to control whether JES2 is allowed to concurrently execute batch jobs of the same name in the same multi-access spool complex. Setting DUPL_JOB=DELAY instructs JES2 to wait for the first batch job to complete before starting the second with the same name.

Using job classes with JES2–managed initiators

Generally, you should assign the same job classes to jobs that have similar execution characteristics or jobs you are assigning special characteristics, such as:

- Rate of processor-to-I/O processing
- Use of special devices
- Number of devices used
- Use of central storage

For example, if several jobs are time-dependent and sort large amounts of data it might not be desirable to tie up all processor storage by having these jobs run concurrently. You could assign these jobs to class B (or C or D--class names have no inherent meaning); then if you define and start only one initiator to handle class B jobs, there will never be more than one of these jobs executing at once.

Assume you made the following assignments:

- Class B Jobs that are time-dependent
- Class C Jobs with high processor requirements
- Class D Jobs with high I/O requirements

You can specify initiator parameters such that:

- INIT(1) CLASS=BCD
- INIT(2) CLASS=CDB
- INIT(3) CLASS=DBC

If the three initiators (INIT(1), INIT(2), and INIT(3)) are processing jobs with the same priority and all necessary resources (for example, I/O devices and data sets) are available, then three jobs, one from each of the three different classes, run concurrently. If a job within one of the classes has higher priority than the others in the class, the system selects that job first.

For initiator processing, class groups exist within the priority-ordered class list. However, they are considered to be a conglomerated 'class'. Classes within a group are selected in a round robin scheme (across all job selection). This ensures an even distribution of work across all classes in the group. For example, if a group has classes A, B, and FRED, then those classes are chained in a ring off the job class group structure. One class is at the head of that loop and is the first searched when looking for work. If work is found in class B in this example, the head is updated to class FRED. The next selection will search class FRED first, and if work is found in that class, the head is updated to class A.

JES2 initiator attributes

Use the INIT(nnn) initialization statement to identify each of the JES2 managed initiators you need on your system. On this statement, you specify:

- the number for the initiator
- the job classes that the initiator can select
- whether the initiator will start automatically when JES2 starts
- the name the operator uses to see the initiator (optional).

Naming initiators: Installations can name ranges of initiators so that a single command controls all the initiators in that group. As an example, you can set up three initiators to process job classes A, B, and C:

```
INIT(101)  CLASS=ABC,NAME=FRED
INIT(102)  CLASS=CAB,NAME=FRED
INIT(103)  CLASS=BCA,NAME=FRED
```

Now you can issue commands such as $SI(FRED) or $PI(FRED) to start or stop all initiators in that group.

JES2-managed initiators can include jobs that are in WLM mode, but these initiators will not select jobs from classes in WLM mode. All initiators (JES2 and WLM ones) honor existing selection criteria. Those initiators in WLM mode only select jobs in a service class queue. The service class queue is passed to JES2 on "job select SSI." Exit 14 is not called for WLM initiators, but Exit 49 is invoked to allow installation modification of the job selection process.

Job priority aging

The system can increase the priority of a job based on the amount of time since the job entered the system if it is in a job class managed by JES2 initiators. You define this ability by specifying PRTYHIGH=, PRTYLOW=, and PRTYRATE= on the JOBDEF initialization statement.

PRTYHIGH= specifies the highest priority a job can achieve through automatic priority aging. PRTYLOW= specifies the lowest priority for a job to be eligible for automatic priority aging. PRTYRATE= specifies the number of times in a 24-hour period that the system will increase a job's priority (subject to the PRTYHIGH= limit) by one. PRTYRATE= has a default of 0, which means the system will not priority-age any jobs.

For example, if PRTYRATE=48, PRTYLOW=4, and PRTYHIGH=10, then all jobs with priorities of 4 to 9 will have their priority increased by 1 every 30 minutes (that is, 24 hours / PRTYRATE of 48 = 30 minutes) until the job priority reaches priority 10.

You set the lower priority limit, PRTYLOW=, to ensure that low priority work does not reach priority levels higher than new, more important work entering the system. In the previous example, all work with priorities 1, 2, and 3 will not priority age.

You set the upper priority limit, PRTYHIGH=, to ensure that high priority work maintains its importance, and doesn't compete with work that achieved its priority from length of time in the system. In the previous example, work with priority 11 through 15, will not priority-age but maintains higher priority than any jobs that can priority-age. See Figure 16 on page 93 for a graphic representation of the priority range setting.

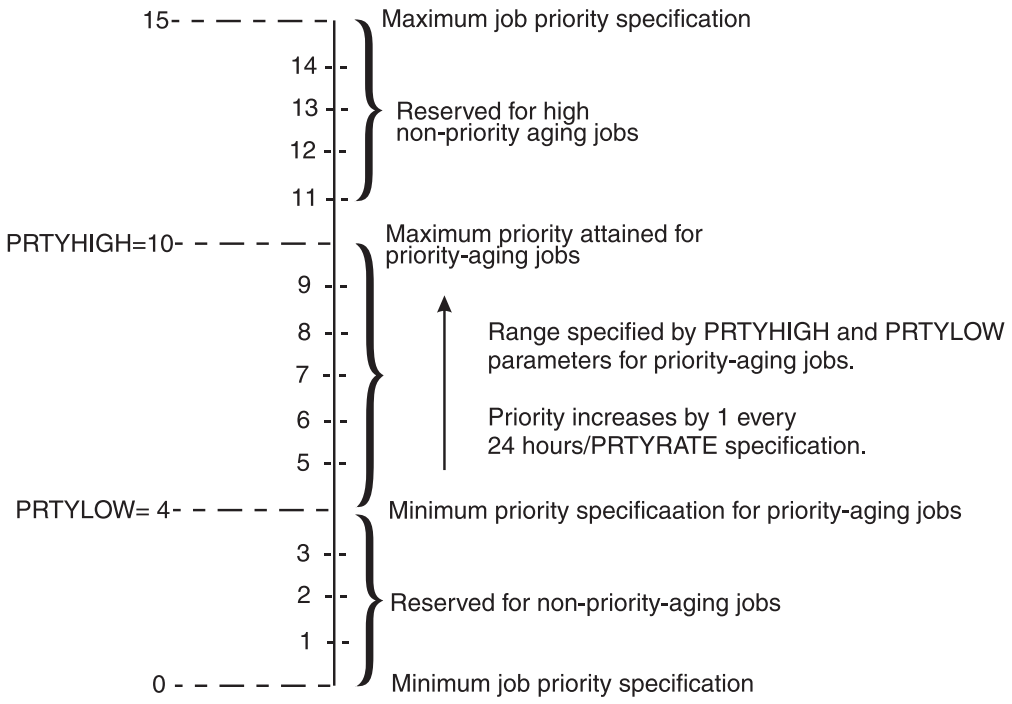


Figure 16. Example of the Interaction of Priority-Aging Parameters, PRTYHIGH=, PRTYLOW=, and PRTYRATE= on JOBDEF Statement

When the job is on the HARDCOPY queue, the only way to affect the priority is through the output priority.

For jobs that are run on a jobclass where MOD=WLM was specified on the JOBCLASS(n) initialization statement, job priority aging has no affect.

WLM control of batch job initiation

Job initiators for WLM-managed job classes are controlled dynamically by workload management. These initiators run under the Master Subsystem and are not assigned JES2 job numbers. WLM can adjust the number of initiators on each system based on:

- The queue of jobs awaiting execution in WLM managed classes.
- The performance goals and relative importance of this work.
- The success of meeting these goals.
- The capacity of each system to do more work.

Note: WLM initiators on a given member do not select work from service classes that have removed the member from its QAFF (Queue Affinity) mask. For example, if MEMA and MEMB are two members in an MAS, and there are jobs awaiting execution in service class DISCRETN, changing the SRVCLASS(DISCRETN) to have QAFF=MEMB means that MEMA will not select any jobs from the DISCRETN service class.

The following must also be true for WLM managed initiators to be active:

1. WLM must be operating in goal mode. (All members of the MAS must be in goal mode for consistent operation.)

See *z/OS MVS Planning: Workload Management* for more details.

Controlling batch job selection

Although workload management dynamically controls its initiators, there are several mechanisms by which you can manage them with JES2 commands:

- Limit the number of jobs in each class that can execute simultaneously in the MAS. See "Setting job class limits for execution" on page 87.
- Stop or start the selection of batch jobs by JES or WLM managed initiators on an individual system. See "Enabling job execution selection" on page 87.
- Control the system or member affinity with the $TJob command. See "Member affinity" on page 85.

  The scheduling environment for a job cannot be changed with operator commands. (See "Job scheduling environment" on page 85.)
- Immediately start the execution of a batch job in a WLM managed job class with the $SJ operator command. Workload management will select the job for immediate execution from the list of systems where the job is eligible to run based on SYSAFF and SCHENV parameters. This is processed even if the job is held.
- Change the classification of jobs entering the system by altering characteristics of the job through the JES2 exits. See "Controlling job execution through exits" on page 87.

SMF recording

To help you monitor and manage the batch job environment, SMF records include the following information:

SMF type 26 records: Workload Management section contains the service class name, scheduling environment, indicators that the job ran under a WLM or JES MODE initiator, or if the job ran because of $SJ command.

SMF type 30 records: Additional fields have been added to the Performance Section including queue times and indicators that the job was modified through operator commands, forced into execution with the $SJ command, or restarted.

SMF type 72 records: Subtypes 3 and 4 record the performance information for batch service classes including response times, velocity, transaction volumes, and JES delays.

See *z/OS MVS System Management Facilities (SMF)* for details.

Considerations for WLM goals applied to batch jobs

WLM goal types deserve special considerations for batch jobs. Remember that the queuing delays while waiting for initiation can affect response time calculations and performance indices. See *z/OS MVS Planning: Workload Management* for guidance on setting goals for batch jobs.

Batch jobs that are part of a critical path, such as the overnight "batch window" should remain in JES managed job classes.

Alignment of initiator mode and service classes: All jobs with the same service class should be managed by the same type of initiation. For example, if job classes A and B are both assigned to the HOTBATCH service class, and JOBCLASS(A) is MODE=WLM, while JOBCLASS(B) is MODE=JES, workload management will have a very difficult time managing the goals of the HOTBATCH service class without managing class B jobs.

Queue delay measurements: If you have large job execution queues, the queue delay can dominate the response time or velocity. This may cause the system to not address other delays because it would not significantly affect the performance index (PI).

Response time does not include TYPRUN=HOLD or JCLHOLD delays, but does include the following:

- Operational delays (jobs or job class held by operator command)
- System or resource affinity delay
- Scheduling delay because of class limits, duplicate jobnames
- Time waiting for an Initiator

Multiple JES2 MAS complexes in a sysplex: There is no requirement that all systems in the sysplex be in the same MAS complex. However, the WLM policy is sysplex-wide and independent of the JES2 member or MAS complex.

- Jobs with the same classification attributes (job class, and so on.) will be assigned the same service class, no matter which MAS they are in unless the subsystem collection name is included in the classification attributes.
- The service class determines the goal regardless of which MAS they are in.
- All jobs in the sysplex with the same service class should be managed by the same type of initiation. If you split a service class between WLM and JES managed initiators, workload management will have a very difficult time managing the goals. [JOBCLASS(c) MODE= setting should be the same in all MASes in the sysplex.]
- Limiting the number of jobs in execution by class through the JOBCLASS(c) XEQCOUNT parameter is only MAS-wide, not sysplex wide; however, limiting the number of jobs in execution by class through JOBCLASS(c) XEQMEMBER(mbr) is on a member-by-member basis.

Note: If you are using WLM-managed initiators, do not classify the batch work from more than one MAS into the same WLM service class.

WLM poly-JES considerations: If two copies of JES2 in the same MVS system (poly-JES) are not sharing the same spool, workload management initiators can select jobs from both the primary and secondary JES.

WLM does not support a secondary JES2 subsystem in the same MAS as the primary. Workload management initiators will only select jobs from the primary JES if the secondary JES is sharing spool with the primary.

The initiator cataloged procedure

SYS1.PROCLIB must contain an initiator cataloged procedure (named INIT) to create job address spaces for both JES2 and WLM managed initiators. JES2 and WLM use the MVS START command or the MVS ASCRE macro to create a system initiator. You can control the number of JES2 initiators by starting and stopping initiators. You cannot control the WLM initiators directly with operator commands.

Figure 17 illustrates the initiator cataloged procedure as supplied by IBM:

```
//IEFPROC EXEC PGM=IEFIIC,DPRTY=12
```

Figure 17. IBM-Supplied INIT Procedure

System resource manager control of the batch job workload

With JES2 managed initiators, you can have more batch jobs initiated and more time-sharing users logged on than the number of address spaces in central storage can accommodate. The System Resources Manager (SRM) can remedy this problem by swapping out heavy resource-using job(s).

Although you might expect over-initiation could compete with and slow the progress of the more important jobs, the SRM addresses this problem. The SRM processes individual jobs according to their service class goals and relative importance.

When the load on the resource eases, the SRM can choose from a varied mixture of swapped-out jobs to even the job mix.

Job monitoring

You can monitor a job through:

- Elapsed time (time since the job started)
- Processor time (actual time spent executing instructions)
- Output
  - Lines and cards produced
  - Pages produced
  - Number of bytes of output produced.

Use the ESTIME statement to cause JES2 to write a message to the operator when a job exceeds the elapsed time specified by the JOBPARM statement, and an additional message at each interval specified by the INT= parameter on this same statement. You can use the IEFUTL installation exit routine to enforce these values, if the IEFUJV installation exit routine places the time in the SMF user ID field. See *z/OS MVS Installation Exits* for information about these exits.

The JCL EXEC statement, the JOB statement, and the converter parameters on the JOBCLASS(v) statement allow you to specify the maximum execution time for a job. If a job exceeds this value, the system enters an IEFUTL installation exit routine, which can cancel the job or allow it to continue (after writing a warning to the job submitter). See *z/OS MVS Installation Exits* for information about this exit.

Use the ESTLNCT, ESTPAGE, ESTPUN, and ESTBYTE JES2 initialization statements to specify the total number of printed lines, printed pages, punched cards, and SYSOUT bytes that a job can produce before JES2 takes action. The OPT= parameter specifies the action JES2 should take. You can allow the job to continue after writing a message to the operator or cancel the job with or without a dump.

Even if a previous step ends with a completion code of 722 (output limit exceeded), the MVS scheduler selects (and begins executing) the next step if COND=EVEN or COND=ONLY appears on the EXEC statement. This step does not end immediately with ABEND722 for producing excessive line, page, or byte SYSOUT but ends when the job exceeds the next excess output interval (that is, NUM= specification plus INT= specification for the specific ESTLNCT, ESTPAGE, or ESTBYTE initialization statement).

Use JES2 Exit 9 to control job output overflow. See *z/OS JES2 Installation Exits* for information about this exit.

Your installation can specify SMF output limiting by class, with the JOBCLASS JES2 initialization statements. SMF can monitor the output (OUTLIM= on DD statement) for each data set.

Monitoring the successful completion of a job

JES2 displays the successful or unsuccessful completion of a batch job with the $HASP165 message. This message is displayed to an interactive user if the NOTIFY parameter is specified on the JCL JOB statement, or the JECL /*NOTIFY statement is specified. You can see the completion code or ABEND code as follows:

- Jobs ending normally have the maximum condition code displayed

  ```
  $HASP165 job_name ENDED AT node_name MAXCC=nnn
  ```

- Jobs ending abnormally have the ABEND codes displayed:

  ```
  $HASP165 job_name ENDED AT node_name ABENDED Sxxx Unnnn
  ```

Also, you can display the ABEND or MAXCC for a job still on the JES2 queues before they are purged, with the `$DJ( ),CC` command.

Entering commands from a job stream

Depending on your security policy, JES2 accepts JES2 commands and MVS commands at different points in a job stream, with different types of control.A job stream is the set of jobs submitted between the physical start of a reader and physical end-of-file, or between the opening and closing of an internal reader data set. See Figure 18 for a pictorial representation of the following description.

JES2 accepts JES2 commands in your jobstream in the form /*$command (for example, /*$S) in front of the JOB JCL statement. JES2 also accepts MVS commands in the /*$VS, 'systemcommand' statement if the command authority is similar to the authority for accepted JES2 commands. An example of submitting an MVS command this way is /*$VS, 'V 0CE,offline'.

JES2 ignores JES2 commands found in the job stream between the first JOB statement and EOF. Also, JES2 ignores JES2 commands entered in a job stream through the network job receiver.

The converter executes any MVS commands (//name COMMAND 'command' where the name parameter is optional) that appear in the job stream after the first JOB statement. Whether the converter actually issues the command depends on what you specified for the converter parameters. JES2 ignores all MVS commands appearing before the first JOB statement.

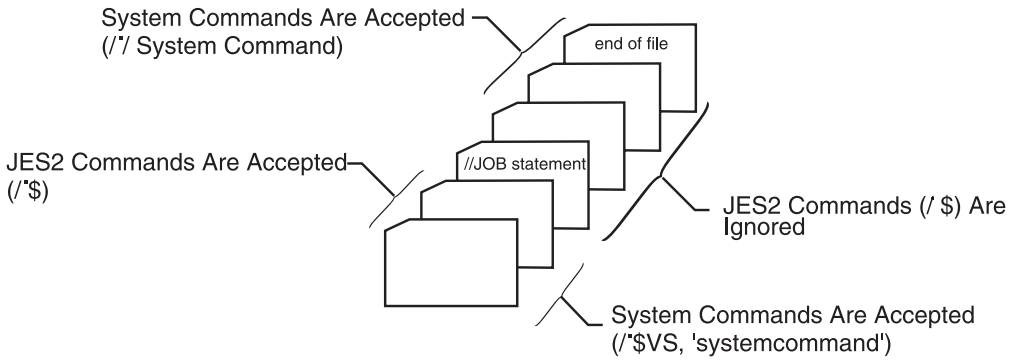


Figure 18. Entering Commands from a Job Stream

Execution batch monitoring (XBM) facility

The execution batch monitoring facility allows a user with minimal knowledge of JCL to process data. The user need only provide a valid MVS JOB statement and input data. This facility is useful for:

- Compile-and-go debugging compilers
- File-inquiry programs
- Hardware or software system emulators

XBM treats all statements including the JOB statement (up to the next JOB statement or end-of-file) except for the /*XMIT statement as input to a processing program. To use this facility:

- The XBM= parameter on a JOBCLASS initialization statement must name a cataloged procedure that resides in a procedure library named in the JES2 start procedure.
- The user must specify that job class on the JOB statement that is submitted.
- The EXEC statement of the cataloged procedure associated with the specified job class must name the program that processes the user's input data.

When a user submits a JOB statement specifying a job class that is associated with a cataloged procedure, JES2 generates an EXEC statement naming that procedure, and a SYSIN DD * statement. JES2 then makes a copy of the original JOB statement and places it after the generated statements. This JOB statement, and any other JCL statements that the user might have coded, except the /*XMIT JES2 control statement and the input data the user has provided, becomes input to the program named in the procedure. When you code the program to be invoked by the procedure, be sure to handle any JCL statements the program might encounter in the input stream (unless you restrict your users to only the JOB statement). Also, your program must define the DDNAME SYSIN as input.

Installations currently using the XBM facility before JES2 Version 3.1.3 can convert their existing applications with minimal impact to their customers. You must:

1. Ensure all of your customers have valid MVS JOB cards in their job streams.
2. Remove the XBATCH parameter from the JOBCLASS(v) and JOBDEF initialization statements.
3. Specify the name of the procedure you were using to process input for each individual job class on the XBM= parameter of the JOBCLASS(v) initialization statement. JES2 no longer generates the name of the procedure it is invoking, therefore, you need only provide one procedure name for each initiator class.
4. If your existing processing program reads past end-of-file in an attempt to obtain more data, JES2 issues a bad return code. This version of XBM processes each job individually, and does not support the additional read. You must change your processing program to accept the end-of-file as a valid end-of-input.
5. If you require the ability to accumulate statistics for a specific class of work, or data related to the number of jobs processed, you **must rewrite** your processing program to maintain that information between jobs because your program will end between jobs.
6. Ensure all user JCL contains valid JOB statements. Execution batch monitoring jobs must pass all converter validation before execution.

Note that the majority of the previous benefits of using the execution batch scheduling facility no longer exist. You might want to convert jobs that currently

use the facility to jobs with the appropriate execution JCL as soon as convenient. Also, all previous restrictions on dynamic allocation no longer exist.

Using the execution batch monitoring facility

Before using the execution batch monitoring facility you must:

- Write and test the program that will process the input. This program must handle any JCL it encounters in the input stream and use DDNAME SYSIN as input.
- Write and test a procedure to invoke your program.
- Place the procedure in a library defined in the JES2 start procedure. Figure 19 shows the contents of an example procedure named ORDPROC.

```
//PROCESS   EXEC  PGM=ORDERIT
//INPUT     DD    DDNAME=SYSIN
//OUTPUT    DD    DSN=NEEDED.BOOKS,DISP=OLD
//CONTROL   DD    DSN=BACKORD.INVENTOR,DISP=SHR
```

Figure 19. Contents of the Example ORDPROC Member

- Use XBM= on the JOBCLASS(v) initialization statement to specify the name of the data set member that contains the procedure. Figure 20 shows a JOBCLASS(v) statement coded to take advantage of the execution batch monitoring facility.

```
JOBCLASS(R)  ACCT=YES,COMMAND=IGNORE,LOG=NO,MSGLEVEL=(2,0),
             PERFORM=123,PGMRNAME=YES,PROCLIB=02,
             XBM=ORDPROC
```

Figure 20. Specifying the Procedure on the JOBCLASS(v) Statement.

- Issue $P JES2 to stop JES2 and close the procedure libraries.
- Issue S JES2 to restart JES2 and open the procedure libraries. JES2 can now access your procedure and you can submit jobs that use the procedure.

Figure 21 is an example of the JCL coded to submit input to the ORDERIT program. The /*ROUTE statement in the example routes the output to the destination, STR15.

```
//ORDER12   JOB    (STR12,15,5845),HUTCH,CLASS=R
/*ROUTE     PRINT  STR15
  JES2 Exits                J. Vazquez
  JES2 Commands             P. Kaskovich
  JES2 General Information  D. Wagenaar
  JES2 Introduction         D. Pizzuto
  JES2 Init Tuning Reference T. Koman
  JES2 Init Tuning Guide    J. Hutchinson
```

Figure 21. Example JCL to Invoke ORDERIT

If the procedure fails, you can temporarily use another procedure to process the input by:

1. Holding the job classes that use the failing procedure.
2. Issuing $T JOBCLASS(v),XBM=newproc to point the job class to a different procedure.
3. Releasing the job classes you held in the first step.

You can enable the execution batch monitoring facility for a previously defined job class, or switch procedures associated with a job class with the $T JOBCLASS(v) command. The job class can be managed by JES2 or WLM initiators.

Output

JES2 provides the following facilities for controlling system output by:

- Queuing of each output unit of work for a job according to a priority based on the volume of output represented by that work unit or thepriority specified on the JCL OUTPUT statement.
- Minimizing operator interaction because of forms, carriage tape, and print train loading for impact printers, and forms, overlay frames, and burster loading for non-impact printers.
- Supporting devices other than printers and punches by routing SYSOUT data to specific external writers through an external writer interface.
- Priority aging of job output elements based on priority parameter specifications, such that the longer the output remains on the output processing queues, the greater its priority becomes.
- Supporting a functional subsystem for non-impact printers (such as the 3800-3 functional subsystem printer) through a functional subsystem interface.
- Routing SYSOUT data sets among nodes in the network.
- Specifying the number of records processed, per job, on printers or punches.
- Holding the writing of an output data set and gathering data sets with the same set-up characteristics for printing as a single unit of output work.
- Supporting the purging of output from jobs, SYSOUT, started tasks (STCs), and time sharing users (TSUs) if the job ended normally (that is, as expected).
- Ensuring users can use specific output devices and output devices have a sufficient security to process specific output. See Chapter 7, “Providing security for JES2,” on page 349 for information on output security.

The system programmer controls system output by:

- Specifying print train and either carriage tape name or forms control buffers for SYSOUT directed to printers with these features, plus supporting the 3525 print and interpret features for SYSOUT data sets.
- Specifying all options and features of the AFP printer for SYSOUT data sets directed to the 3800.

Default data set characteristics

JES2 assumes defaults for any data set characteristics not specified on the SYSOUT DD statement, the JCL OUTPUT statement or the JES2 /*OUTPUT control statement. If a job does not specify an FCB, UCS, or FLASH parameter, JES2 uses the FCB, UCS, or FLASH default value specified by your installation. JES2 uses the name ‘****’ when requesting a default FCB or UCS image from the operator. The operator satisfies this request by mounting any image specified as a default. If the job does not specify one or more of the parameters (FCB, UCS, or FLASH), then any default will satisfy it.

Identify the form used for all data sets not specifying a form during JES2 initialization in the STDFORM parameter on the OUTDEF statement. This is the standard form for both printers and punches.

Output priority

JES2 queues a job for output processing when the job completes execution. You can change the default priority assigned to the output by coding the PRTY= keyword on the JCL OUTPUT statement.PRTY= assigns a single priority level between 0 and 255 to all work units within a particular output group. The keyword takes effect if you also specified PRTYOUT=YES on the OUTDEF initialization statement; PRTYOUT=NO is the default.

JES2 creates output groups during output processing or during execution in the case of spin (FREE=CLOSE or SPIN=UNALLOC on the DD statement) data sets. Each output group represents a unit of output work to JES2. JES2 places the output group in the job output element queue in order of output priority. If you did not specify a priority on the JCL OUTPUT statement, JES2 computes a priority for each output group from the tables specified by the PRIORITY= parameter on the OUTPRTY initialization statement.

Also, JES2 uses the RECORD= parameter (for line data), the PAGE=parameter (for page data) or both the RECORD= and PAGE= (for data containing line and page data) with the PRIORITY= parameter to determine the priority of the output. JES2 bases the priority on the total number of print lines, punch cards, or pages in the data set which the output group represents. If the output group represents a data set containing both line and page data, the output priority is the average of the priorities specified by the RECORD= and PAGE= parameters, respectively. The following tables show the default values for the priority calculation:

Table 14. Priority calculation default values: Record

For RECORD:	RECORD	n
RECORD values are estimates of the job output lines and cards	2000	1
	5000	2
	15000	3
	279620	4
	.	.
	.	.
	.	.
	16777215	9

Table 15. Priority calculation default values: Page

For PAGE:	PAGE	n
PAGE values are estimates of the number of job output pages	50	1
	100	2
	300	3
	16777215	4
	.	.
	.	.
	.	.
	16777215	9

Table 16. Priority calculation default values: Priority

For PRIORITY:	n	PRIORITY
The n value is determined from the PRIORITY table. The PRIORITY table shown contains the default for the corresponding n values	1	9
	2	8
	3	7
	.	.
	.	.
	.	.
	9	1

If the total number of output records (lines or cards) is 2000, then:

```
From RECORD, 2000 output records indicates n = 1
From PRIORITY, n = 1 indicates PRIORITY = 9
```

If the total number of output pages is 300, then:

```
From PAGE, 300 output records indicates n = 3
From PRIORITY, n = 3 indicates PRIORITY = 7
```

If the total number of output is 2000 lines **and** 300 pages, then the output priority is the average of both types of output based on the RECORD and PAGE values as follows:

```
From RECORD, 2000 output records indicates n = 1
From PAGE, 300 output records indicates n = 3
The average value of n = (1 + 3) / 2 = 2
From PRIORITY, n = 2 indicates PRIORITY = 8
```

By specifying other values for the tables (during JES2 initialization), you can more closely control your installation's priority assignments. If you do build your own priority table based on estimated record/page counts, be certain that you supply the values in ascending order as the RECORD, PAGE, and PRIORITY(n) values are in the default tables, above. This requirement exists because JES2 uses the RECORD= or PAGE= values in a "less than or equal to" order when calculating a priority that does not exactly match the values in the tables.

JES2 selects output groups for output processing based on the job's output priority. You can cause the priority to increase through the priority aging feature. You can set the original priority on the OUTPRTY(n) statement. However, if the priority is within the limits set by PRTYLOW= (the minimum) and PRTYHIGH= (the maximum), JES2 increases the priority of the job at a rate specified by PRYORATE=. See "Output priority aging" on page 103 for additional information about output priority aging.

When a job completes output processing (all of its output groups created and added to the job output element queue), JES2 queues the job for hard-copy.

When assigning such things as priority classes and forms requirements for data sets, you should balance the choices against the criteria JES2 uses to select the output data sets to process.

Output priority aging

JES2 can increase the priority of a job's output based on the time since the output was created. You define this ability by specifying PRTYHIGH=, PRTYLOW=, and PRYORATE= parameters on the OUTDEF initialization statement.

PRTYHIGH= specifies the highest priority which output can achieve through automatic priority aging. PRTYLOW= specifies the lowest priority for a job to be eligible for automatic priority aging. PRYORATE= specifies the number of times in a 24-hour period that the system will increase a job's priority (subject to the PRTYHIGH= limit) by one. PRYORATE= has a default of 0, which means the system will not priority age any output.

For example, if PRTYRATE=48, PRTYLOW=100, and PRTYHIGH=200, then all jobs with priorities of 100 to 199 will have their priority increased by 1 every 30 minutes (that is, 24 hours / PRTYRATE of 48 = 30 minutes) until the job priority reaches priority 200.

You set the lower priority limit, PRTYLOW=, to ensure that low priority output does not reach priority levels higher than new, more important output entering the system. In the previous example, all output with priorities between 1 and 99 (inclusive) will not priority age.

You set the upper priority limit, PRTYHIGH=, to ensure that high priority output maintains its importance, and doesn't compete with output that achieved its priority from length of time in the system. In the previous example, output with priority 200 or higher, will not priority age but maintains higher priority than any output that can priority age. See Figure 22 for a graphic representation of the priority range setting for output.

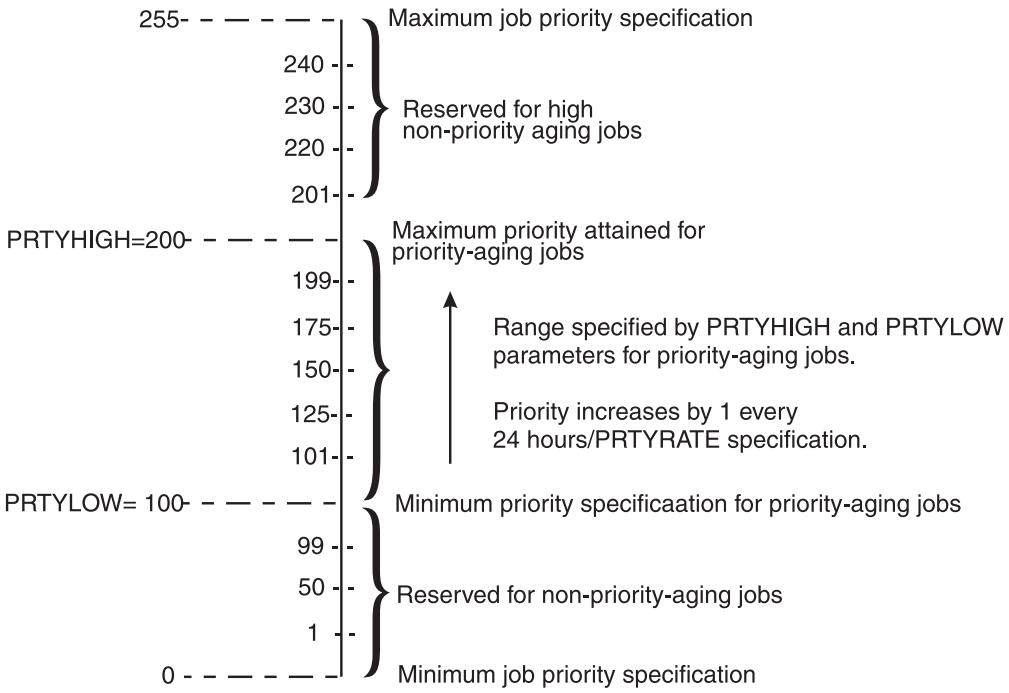


Figure 22. Example of the interaction of priority–aging parameters PRTYHIGH, PRTYLOW, and PRYORATE on OUTDEF statement

Output processing

The JES2 print/punch processor, the external writer, the network SYSOUT transmitter and spool offload SYSOUT transmitter can select only data sets which already have output groups constructed.Varying the number of output groups, (JOENUM parameter on the OUTDEF initialization statement) influences the processing of output.

Specifying many output groups makes more work available for selection within the configuration; however, the increased number of output groups also increases the size of the checkpoint data set thus adversely affecting performance.

An output group that does not yet describe a unit of work is a "free" JOE. When the building of output groups for a job depletes the number of available free JOEs, the job or spin data waits until more output groups become available.

The JES2 output processor builds output groups for a job without making a distinction as to whether the routing of the data sets is for the local node or for some other node. JES2 makes the distinction when it adds the output groups to the output queues; JES2 then places all the output groups for the local node on one of the class queues, and places all the others on the network queue.

Output data set groups

An output group can point to a data set group that defines data sets that have identical characteristics and contain the output of a job, a started task, or a time-sharing user. Each data set group is a processing entity with a set of processing characteristics.

JES2 groups output for jobs processed at the local node use the following processing characteristics:

- Burst specification (BURST)
- Output class (CLASS)
- Destination (DEST)
- Forms control buffer name (FCB)
- Forms overlay frame (FLASH)
- Forms (FORM)
- Group name (GROUPID)
- Output disposition (OUTDISP)
- Process mode (PRMODE)
- Print train (UCS)
- Installation writer name (WRITER)

You can use the OUTPUT JCL statement to describe these characteristics. Note that the SYSOUT DD statement and the /* OUTPUT JECL statement support all these characteristics except GROUPID and OUTDISP.

JES2 also groups output data sets on the following criteria which a user can specify on the OUTPUT JCL statement:

- Address
- Building
- Department (DEPT)
- Name
- Print complete notification message (NOTIFY)

- Room
- Title

These fields can also be modified through SDSF while output resides on the output or hold queue.

When JES2 groups on the information on these fields, it attempts to match the character string exactly. Therefore, it is important to ensure the user specifies the **exact** string in each of the above so that JES2 can group the output properly. For example, JES2 will not group NAME='JOHN' and NAME='JOHN ' together.

You can change the SYSOUT characteristics that affect grouping by using JES2 Exit 40. See *z/OS JES2 Installation Exits* for more information about Exit 40.

The following tables illustrate how one output group can represent one of several SYSOUT data sets:

Table 17. Relationship of SYSOUT specification to number of job output elements: Seven output groups built. The first row, which spans all three table columns, is the table header.

SEVEN OUTPUT GROUPS BUILT		
1	-	SYSOUT=(A,,4PLY), UCS=PN
2	-	SYSOUT=C,UCS=PN
3	-	SYSOUT=A
4	-	SYSOUT=B
5	-	SYSOUT=A,FLASH=FL01
5	-	SYSOUT=A,FLASH=FL01,MODIFY=CM03
6	-	SYSOUT=C,FCB=STD3,UCS=GF10
6	-	SYSOUT=C,CHARS=(GF10,FM10),FCB=STD3
7	-	SYSOUT=C,CHARS=(GF10,FM10),FCB=STD3,UCS=PN
3	-	SYSOUT=A

Table 18. Relationship of SYSOUT specification to number of job output elements: Three output groups built. The first row, which spans all three table columns, is the table header.

THREE OUTPUT GROUPS BUILT		
1	-	SYSOUT=A
1	-	SYSOUT=A
...	...	...
2	-	SYSOUT=B
1	-	SYSOUT=A
2	-	SYSOUT=B
.		
.		
.		
3	-	SYSOUT=(A,,2PLY),FCB=8LPI
.		
.		
.		
1	-	SYSOUT=A

JES2 duplicates output groups built for job-related output depending on the number of job copies requested on the /*JOBPARM statement (up to the maximum allowed by the COPIES= parameter on the OUTDEF statement). This allows the devices available for output to govern the number of copies processed for a

particular job. JES2 uses several factors to group SYSOUT data sets. The groups also conform to a set of rules, discussed below. Table 19 summarizes these rules.

Note: If the output groups are transmitted through an NJE network, do not use the /*JOBPARM JECL statement to produce multiple copies. Instead, use the COPIES= parameter at the DD level. If COPIES= is used at the DD level, JES2 does not duplicate output groups and the output groups are handled appropriately.

JES2 uses the following factors to group output:

- The data set output class matching the job's message class
- The USERSET specification on the OUTDEF statement.
- The user assigned an output group name using the GROUPID= keyword on the JCL output statement.
- The DMNDSET parameter specification on the OUTDEF statement.

JES2 forms output groups whose data sets always agree on the following characteristics:

- address
- building
- department
- destination
- external writer
- name
- output class
- output disposition
- output group name
- print complete notification message
- process mode
- room
- title

If DMNDSET=YES on the OUTDEF initialization statement and the data set matches the job's message class or if USERSET=YES on the OUTDEF initialization statement and the user assigned the output group a name, then JES2 scans the setup characteristics of the output group's data sets to determine which, if any, of the setup characteristics vary. Otherwise, the output groups will also agree on setup characteristics.

Table 19. Interrelationship Between the Four Factors Affecting Output Grouping

DMNDSET= *	**DOES MSGCLASS AND OUTPUT CLASS MATCH?**	**USERSET=YES**		**USERSET=NO**	
		OUTPUT GROUP NAME ASSIGNED		**OUTPUT GROUP NAME ASSIGNED**	
		YES	NO	YES	NO
YES	YES	1	1	1	1
	NO	1	2	2	2

Table 19. Interrelationship Between the Four Factors Affecting Output Grouping (continued)

DMNDSET= *	DOES MSGCLASS AND OUTPUT CLASS MATCH?	USERSET=YES		USERSET=NO	
NO	YES	1	2	2	2
	NO	1	2	2	2

* USERSET and DMNDSET are parameters on the OUTDEF initialization statement.
Note:

1. Data sets within groups formed agree on class, external writer, destination, process mode, output disposition, and output group name. However, setup characteristics (those characteristics that determine the data set groups eligible for processing a given device) can vary.
2. Data sets within groups formed agree on class, external writer, destination, process mode, output disposition, output group name and setup characteristics.

To delay printing data sets, an installation can specify:

- Reserved classes
- HOLD=YES on a JCL DD statement
- OUTDISP=HOLD on the JCL Output statement
- OUTDisp=HOLD or LEAVE on one of the following initialization statements:
  - JOBCLASS
  - OFF(n).SR
  - OFF(n).ST
  - OUTCLASS

"Output disposition for SYSOUT data sets" on page 121 discusses the OUTDISP parameter. *z/OS MVS JCL Reference* explains HOLD=YES on a JCL DD statement. Note that the use of the GROUPID= keyword on the JCL OUTPUT statement influences data set gathering. If your installation allows user control of output groups by specifying USERSET=YES on the OUTDEF statement, you can form output groups with dissimilar output characteristics.

JES2 does not group spin data sets; each data set constitutes a "group" of one data set.

SMF record summary: job-output related information

The following lists the SMF records that contain job-output related information. For a complete list of the record contents, see *z/OS MVS System Management Facilities (SMF)*.

Table 20. SMF Records Containing Job Output Information

Record Number	Action that Causes the Record to be Written
6	When JES2 finishes processing a job output element it will write this record if TYPE6=YES in the JOBCLASS statement.
26	After processing all SYSOUT data sets for a job, JES2 writes this record if TYPE26=YES in the JOBCLASS statement. Exit IEFUJP can also override the writing of this record.
57	After transmitting an output group to another NJE node, JES2 writes this record.

JES2 Exit 21 or SMF Exit IEFU83 can be used to override the writing of these records.

Routing output to other nodes

Output groups on the network queues are grouped by data set forms requirements, SYSOUT class, and other selection criteria. If you decide to route output on the network queue to the local node, remember that routing output does involve calls to an installation's security product, which can impact command performance. JES2 removes the output group from the network queue and places it on the appropriate local class queue. Similarly, JES2 routes output groups originally destined for local output to another node by removing them from their class queue, changing their route codes, and placing them on the network queue.

The JES2 commands that deal with output routing (for example, $R and $O) allow routing and rerouting of data sets. JES2 places the current routing of a data set in the data set header whenever the data set is transmitted.

The network SYSOUT transmitter selects output groups from the network transmission queue, which is ordered FIFO (first-in-first-out) within priority. For data sets created on the local node, the network SYSOUT transmitter creates network data set headers. The network SYSOUT transmitter transmits the data set headers, along with a job header, job trailer, and data set records over an NJE line to the destination node specified in the routing information for the data set. This transmission can be either store-and-forward through intermediate nodes, or direct transmissions to the destination node.

JES2 attempts to transmit as one job, any non-spin SYSOUT which are created under a single job and directed to a common destination (or along a common path). This processing is performed even if multiple SYSOUT data sets are grouped into different output groups with different output priorities.

At the other node, a network SYSOUT receiver receives the NJE headers and data sets. The network SYSOUT receiver uses the information in the NJE job header to create a new job on the receiving system. As a result of this processing, the network SYSOUT receiver places a job on the output queue.The output processor on the receiving node treats this job as a locally completed job. The output processor creates output groups describing the data set, thus making it ready for local print/punch processing or further network transmission, depending on the routing specified for the data sets.

NJE transmission selection: You can specify multiple transmitters (and receivers) for a line as described in “Specifying transmitters and receivers” on page 272. For individual SYSOUT transmitters, you can also specify work selection characteristics based on the size of SYSOUT files as described in “Specifying work selection values for NJE transmitters” on page 273.

NJE file sizes can range from very small (for example, one record, such as a Netmail acknowledgment) to very large (millions of records). After a transmitter has selected a file, it does not interleave it with other files or voluntarily interrupt the transmission until that file is completely transmitted. If the line drops or the transmission is otherwise interrupted, the entire file must be retransmitted from the beginning.

You can use the work selection (WS=LIM) parameter on the L(nnnn).ST(n) statement to cause a SYSOUT transmitter to select files based on their size, so as not to tie up a transmitter a long time with very large files. This allows a transmitter to be available for timely delivery of small files. See “Specifying work selection values for NJE transmitters” on page 273 for more information.

Considerations for output produced by APPC transaction programs

Advanced Program-to-Program Communication (APPC) is an implementation of the Systems Network Architecture (SNA) LU 6.2 protocol that permits interconnected systems to communicate and share the processing of programs. Installations can use APPC to allow MVS systems to participate in a cooperative processing environment.

Application programs that use APPC communication calls to perform cross-system requests are called **transaction programs**. Transaction programs can communicate with other transaction programs that reside on local or remote systems. For example, an installation might create a transaction program and use APPC to access MVS services on a host, such as updating a large data base; the MVS transaction program could subsequently return messages or data.

APPC transaction programs can exploit certain JES2 facilities, such as printing, network job entry (NJE), and job submission. JES2 treats all SYSOUT produced by APPC transaction programs as spin data sets. APPC transaction programs do not run as JES2 jobs; instead, transaction programs are associated with APPC transaction initiators, which are long-running jobs that can sponsor many transaction programs.

Output from APPC will have the 'PROTected' indicator, but this has nothing to do with RACF or other security servers. Also, the 'PROTected' indicator is not exclusive to APPC; it is found in OMVS and in other transactional environments where the transaction output is associated with the initiators such as ASCHINIT and BPXAS.

Because APPC work is treated differently from other JES2 work, you should be aware of the following processing considerations for SYSOUT data sets created by a transaction program:

System data sets

Because an APPC initiator can represent many transaction programs, the following system data sets are treated differently for APPC work than for other types of work in the system:

Data Set
APPC Considerations

$JOURNAL
Job journal data sets are not produced for APPC transaction programs. Therefore, transaction programs cannot be automatically restarted.

JESMSGLG
JES2 does not support JESMSGLG data set processing for APPC transaction provided programs. However, JES2 does write messages concerning APPC transaction programs into the TP message log which can be used for recovery and problem determination when an error occurs while an APPC transaction program is processing. See *z/OS MVS Planning: APPC/MVS Management* for additional information about the TP message log.

JESYSMSG
JESYSMSG data sets are not produced for APPC transaction programs.

JESJCL
Support for JESJCL is equivalent to that provided for other started tasks.

See *z/OS MVS Planning: APPC/MVS Management* for additional information about understanding and implementing APPC in an MVS environment.

Output class assignment

A problem program assigns its output an output class which JES2 processes. You can name one of a maximum of 36 SYSOUT classes by specifying the SYSOUT= keyword on the DD statement or by using the CLASS= keyword on the JCL OUTPUT statement.You can specify any single letter (A-Z), number (0-9), or asterisk as a class. An asterisk indicates the class assigned is the same as that assigned to the message class as specified by MSGCLASS= on the JCL JOB statement. The names have no inherent meaning; they just group similar output characteristics. The designation of a class as containing print or punch data, during JES2 initialization, is only for output limiting and job accounting purposes and has no bearing on the class assigned to the actual device.

JES2 assigns output class characteristics (such as data set and message hold status, track celling, and print-punch specification) on the OUTCLASS(v) initialization statement. See the OUTCLASS(v) statement in *z/OS JES2 Initialization and Tuning Reference* for a complete description of this statement.

Characteristics of output classes

Assign JES2 print/punch processors and external writers only to specific classes of output. Make the assignment:

- During JES2 initialization for JES2 devices.
- In the external writer cataloged procedure for external writers.
- At any time, for JES2 devices, by using JES2 modify ($T) commands.
- For external writers, by specifying classes on the MVS START or MVS MODIFY commands. See *z/OS MVS System Commands* for more information on these MVS commands.

If no device is active for an output class, output for that class remains on the queue until you start a device for that class.

Assign output classes in a manner that distinguishes types of output and results in the most efficient use of devices. However, keep the following in mind when assigning classes:

- Distinguish between data processed by standard JES2 writers and data processed by external writers.
- Allocate separate classes for data destined for different devices and data destined for similar devices but with different characteristics. It is not necessary, however, to use classes to separate data with different UCS and FCB requirements if a JES2 writer is to process the data because JES2 handles these parameters automatically. Use separate classes if an external writer will print this data.
- Assign classes to give different priority to different types of data, such as output printed on a different work shift.
- Do not use different classes to give priority to short data sets as the JES2 priority calculation is adequate for this purpose.
- Coordinate output classes in an NJE environment so print and punch classes are consistent among nodes in the network. The destination node places SYSOUT data sets in the class queues indicated in the NJE header. The class at the destination node determines the processing used for the data set.

If the classes are not consistent, the receiver might not get data in the expected form. For example if class G is print output on node A and punch output on node B, routing class G data from node A to node B causes print output to be punched.

By specifying a process mode (PRMODE) for the particular device on which you require the system to process the SYSOUT data set(s), you avoid the problem of multiple destinations assigning the same output class to different devices. For example, if you send a SYSOUT data set to another node (for printing on a AFP printer), specify PRMODE=U3800 on the PRT(nnnn) initialization statement for that device. If you then specify PRMODE=U3800 on the JCL output statement of the data set you want, JES2 allows the device with PRMODE=U3800 to select that data set for printing. Additionally, JES2 can select any other work which matches any other process modes specified for that device.

Blank truncation

Use the BLNKTRNC parameter on the OUTCLASS(v) statement to turn off blank truncation for any non-page-mode data (the default is BLNKTRNC=YES). This parameter specification has no effect on page-mode data, but generally saves significant spool space for line-mode data. In a typical page-mode SYSOUT data set that contains a mixture of page- and line-mode data, any spool savings applies only to the line-mode records. Do not set BLNKTRNC=NO except for those data sets that require right-hand padding with blanks to produce correct alignment. If required, set aside one or more specific classes for no blank truncation.

If you send SYSOUT data sets that have not been blank-truncated to a node initialized with blank truncation set on for that particular output class, the receiving node will truncate the data set. This same 'truncation rule' holds true for individual members of a multi-access spool complex. If a member initialized (or defaulted) with BLNKTRNC=YES executes a job it receives from a system where BLNKTRNC=NO, the data set is blank truncated regardless of the truncation characteristic of the sending member's SYSOUT class.

Setup characteristics

As established during JES2 initialization (and perhaps altered by the operator), each JES2-controlled printer and punch possesses setup characteristics. Setup characteristics for impact printers (for example, the 1403 and 3211) and punches are: process mode, forms, print train, and either carriage control tape or FCB. (For a 1403 printer, JES2 uses the FCB parameter as the carriage control tape name. JES2 requests that the operator mount carriage tapes by this name in a manner similar to mounting forms.) For non-impact printers (for example, the 3800 and 3820 printers), setup characteristics are: process mode, forms, flash, burst, UCS, and FCB specifications.

Setup characteristics determine the data set groups eligible for processing on a given device. Each locally attached printer and punch has either a route code of LOCAL or a specific device-name route code. If the device has a specific route code name or names, it is eligible to process only those data sets that JCL or the operator specifically routes to it. The route code name assigned to a remotely attached printer or punch is either the installation-assigned name of a remote pool of devices or the name of the workstation the device attaches to.

Demand setup and output data set grouping

This facility allows users to define the grouping of output data sets. For example, a user can group all of a job's output together, including the JES2 system data sets and assign a single output class to the group. Such grouping directs JES2 to print

all the data sets in the group on continuous paper.When grouping data sets, the data sets must have the same **grouping characteristics**; these include output class, destination, external writer name, security label, and process mode. See *z/OS MVS JCL User's Guide* for a description of how a user can control grouping characteristics through use of the JCL OUTPUT statement.

Demand setup is the setup of an output devices based upon user needs. **Demand setup groups** are groups of data sets with the same user-defined setup characteristics. The DMNDSET= and USERSET= parameters on the OUTDEF statement allows users to define demand setup groups. DMNDSET= allows users to define a single group to data sets with different device setup characteristics. USERSET= allows users to create JCL-defined output groups.

Spin data set processing

Coding FREE=CLOSE or SPIN=UNALLOC in the JCL of a SYSOUT DD statement or specifying the DALCLOSE text unit key on a dynamic allocation (SVC99) parameter list for a SYSOUT data set causes JES2 to spin the data set for output processing when it is closed. JES2 then has the capability of performing output processing on the data set while the job continues execution. JESLOG data sets can be spun through $T J/S/T,SPIN.

In most cases, JES2 frees spool tracks allocated to the data set after printing, canceling, or transmitting the last copy of the data set.

For applications that dynamically allocate a spin SYSOUT data set (unallocation at Close - Key X'001C') but fail to issue FREE=CLOSE or SPIN=UNALLOC, JES2 does not free spool tracks allocated to the spin data set until the job is purged. In this case, end-of-task processing deallocates the data set and job purge processing returns the allocated tracks.

Spin data sets in the NJE environment

JES2 transmits spin data sets separately from regular data sets. The SYSOUT receiver receives spin data sets as unique "jobs"; any later spin data sets or regular data sets cause the receiver to create a new job.

The SYSOUT receiver builds peripheral data definition blocks for the spin data sets in separate spin input/output tables, and the output processor builds separate spin output groups for them. If a regular (non-spin) data set has a copy count of 2 and the job has a copy count of 3, the data set should print 6 times. If a spin data set has a copy count of 2 and the job has a copy count of 3, the data set prints twice; job copy count does not apply to spin data sets.

System data set characteristics

JES2 must also route JES2 system data sets generated during the execution of a program to an output device. To ensure that the messages and data appear together, you must describe all data sets for the job in a single output group. There are a number of ways to assign an output class to JES2 system data sets:

- The MSGCLASS= parameter of the job statement
- The JESDS= parameter on the JCL output statement
- The CLASS= parameter the JCL OUTPUT statement

The default for MSGCLASS= is the class specified on the JOBCLASS statement, or the class assigned to the device on which the job entered the system.

The JESLOG specification will influence the generation or SPINNING of JOBLOG and SYSTEM messages data sets.

Note: Remember that all output grouping characteristics, not just class, must match to group system data sets with the output produced by the job.

Specifying JES2 output size limits

At initialization, you can set a range of values which limits the number of output records or pages JES2 can process (by output group) on each local and remote printer or punch.Set this range of values in the device's initialization statement. You can alter this output control at processing time by using the LIM= parameter with the $T command for line-mode data or the PLIM= parameter for page-mode data.

Variable work selection criteria

The work selection criteria keyword (WS=) on the L(nn).ST(n), OFF(n).XX, PRT(nnnn), PUN(nn), R(nn).PR(n), and R(nn).PU(n) statements defines:

- The precise specification of job and output characteristics that JES2 considers when selecting work for an output device
- The order of importance (priority) of the selection characteristics
- The characteristics that JES2 must match exactly.

JES2 initializes the setup characteristics of a device based on the specifications supplied on a device (printer/punch) statement. JES2 groups a job's output data sets into output groups (job output elements) based on the data sets' output requirements. The user defines these requirements in the job's JCL or by JES2-supplied defaults.

When JES2 selects work for a device to process, JES2 compares the device's characteristics to the output requirements associated with the output groups waiting for processing. If JES2 finds an output group that matches the device's characteristics, JES2 routes the data sets associated with that output group to that output device.

If JES2 can not match an output group with a device, selection of the output group does not occur. The output group remains on the spool until the operator uses a $T command to change an output device specification to match that output group. Alternatively, the operator can use the $T O command to change the data set's characteristics to allow the device to select the output group. If an output device selects an output group for processing, but certain setup characteristics do not match exactly, JES2 issues the $HASP190 message notifying the operator of the characteristics that require changing.

Specifying work selection criteria

The four criteria (Q,R,PRM, Volume), typically listed before the slash, also prioritize the lists they name. For example, on the PRT(nnnn) statement

```
PRMODE=(PAGE,UPLOT8,LINE)
```

specifies that the printer accepts PAGE-type output before UPLOT8-type output and UPLOT8-type output before LINE-type output. In fact, you must list PRM before the slash for printing to occur. If you specify Q, R, or V after the slash, JES2 will still match them exactly, but JES2 does not prioritize selection of their specification strings.

If the output group is a demand setup output group (see "Demand setup and output data set grouping" on page 111), JES2 will not match those output groups whose output setup characteristics match the specific device setup. The match does not occur because demand setup indicates that the setup characteristics of individual job output elements can vary throughout the output group.

Determining the "best" JOE

By using the JES2 work selection facility, you can add your own installation-defined work selection parameters. JES2 integrates these work selection parameters and those that JES2 provides [user specifications always override JES2 (HASP) table specifications] to build the work selection mask. The higher the mask value, the more likely a particular output group will receive **best JOE** status. The remainder of this section discusses the work selection processes and JES2 selection of the best JOE.

The following examples illustrate how JES2 allows the output device to select work. See Table 21 which presents a specific example of six output groups (job output elements) each of which relates to a particular job. In that figure:

- Output groups JOB5.1.1 and JOB53.6.2 are class A output on the class A output queue.
- Output groups JOB99.2.1 and JOB6.4.2 are class B output, on the class B output queue.
- Output groups JOB8.3.2 and JOB22.2.1 are class F output on the class F output queue.

Assume you have previously initialized three printers (using the PRT(nnnn) statements specified in Figure 23 on page 115 through Table 27 on page 116); note the only difference between each statement is the WS= keyword character string specification. (The values shown next to the output groups listed, in Table 21 for each printer are JES2-assigned values generated during the selection process.)

Table 21. Example one of output groups on output queues

CLASS A QUEUE	
JOB5.1.1	**JOB53.6.2**
ROUTECDE=LOCAL	ROUTECDE=R3
PRIORITY=15	PRIORITY=10
FORMS=FMS1	FORMS=FMS1
FLASH=NONE	FLASH=NONE
FCB=STD6	FCB=STD6
UCS=AN	UCS=A11
PRMODE=LINE	PRMODE=UPLOT8
PAGECNT=0	PAGECNT=0
RECORDCNT=50	RECORDCNT=60
BURST	BURST
JOBNAME=TEST	JOBNAME=TEST
W=ABC	W=ABC

Table 22. Example two of output groups on output queues

CLASS B QUEUE	
JOB99.2.1	**JOB6.4.2**
ROUTECDE=R105	ROUTECDE=R106
PRIORITY=01	PRIORITY=78
FORMS=FMS1	FORMS=FMS2
FLASH=TOPS	FLASH=CAPS

Table 22. Example two of output groups on output queues (continued)

FCB=8LP1	FCB=8LP2
UCS=GF15	UCS=GF15
PRMODE=PAGE	PRMODE=PAGE
PAGECNT=90	PAGECNT=40
RECORDCNT=0	RECORDCNT=0
NOBURST	BURST
JOBNAME=TEST3	JOBNAME=TEST7
W=NONE	W=ABC

Table 23. Example three of output groups on output queues

CLASS F QUEUE	
JOB8.3.2	**JOB22.2.1**
ROUTECDE=LOCAL	ROUTECDE=R3
PRIORITY=15	PRIORITY=78
FORMS=FMS2	FORMS=FMS3
FLASH=NONE	FLASH=NEWF
FCB=STD6	FCB=STD6
UCS=P11	UCS=GCF
PRMODE=U3800	PRMODE=UPLOT8
PAGECNT=0	PAGECNT=0
RECORDCNT=5000	RECORDCNT=1000
BURST	BURST
JOBNAME=TEST	JOBNAME=TEST7
W=ABC	W=NONE

A

```
PRT(1) BURST,CLASS=BAFGTD,FCB=8LP1,FLASH=TOPS,FORMS=FMS2,
       UCS=GF15,PRMODE=(PAGE,LINE,UPLOT8),W=ABC,
       LIMIT=(4000,8000),ROUTECDE=(R3,LOCAL,R106,R105),
       PLIM=(40-80),WS=(W,R,Q,PRM,LIM/F,T,C,P)
```

Figure 23. Example work selection criteria specifications and corresponding values assigned to particular JOEs on the output queue

Table 24. Example work selection criteria specifications and corresponding values assigned to particular JOEs on the output queue

WS=(	W	R	Q	PRM	LIM	/	F	T	C	P)
JOB99.2.1	00									
JOB6.4.2	FF	FD	FF	FF	FF		FF	FF	00	78
JOB5.1.1	FF	FE	FE	FE	FF		00	00	00	15
JOB53.6.2	FF	FF	FE	FE	FF		00	00	00	10
JOB8.3.2	00									
JOB22.2.1	00									

B

```
PRT(2) BURST,CLASS=BAFGTD,FCB=8LP1,FLASH=TOPS,FORMS=FMS2,
       UCS=GF15,PRMODE=(PAGE,LINE,UPLOT8),W=ABC,
       LIMIT=(4000,8000),ROUTECDE=(R3,LOCAL,R106,R105),
       PLIM=(50-90),WS=(Q,R,PRM,LIM/P)
```

Figure 24. Example work selection criteria specifications and corresponding values assigned to particular JOEs on the output queue

Table 25. Example work selection criteria specifications and corresponding values assigned to particular JOEs on the output queue

WS=(	Q	R	PRM	LIM	/	P)
JOB99.2.1	FF	FC	FF	FF		01
JOB6.4.2	FF	FD	FF	00		78

Table 26. Example work selection criteria specifications and corresponding values assigned to particular JOEs on the output queue

C

```
PRT(3)  BURST,CLASS=BAFGTD,FCB=8LP1,FLASH=TOPS,FORMS=FMS2,
        UCS=GF15,PRMODE=(PAGE,LINE,UPLOT8),W=ABC,
        LIMIT=(4000,8000),ROUTECDE=(R3,LOCAL,R106,R105),
        PLIM=(40-80),WS=(P,Q,R/PRM,LIM)
```

Table 27. Example work selection criteria specifications and corresponding values assigned to particular JOEs on the output queue

WS=(	P	Q	R	/	PRM	LIM)
JOB6.4.2	78	FF	FD		FF	FF
JOB22.2.1	78	FD	FF		FF	00

JES2 scans each output class queue and compares the output groups on the queues to the printer's characteristics (DCT) as initialized on the PRT(nnnn) statement. Output group scanning/scheduling for PRT1 proceeds as follows: (See Figure 23 on page 115 through Table 27 (Use section A to assist your understanding of the selection process.)

1. Because class B has the highest priority on the CLASS= parameter specification for PRT(1), JES2 examines JOB99.2.1 on the class B queue first. The checking process stops almost immediately because the writer name (W=NONE) specified for the output group does not match the writer name assigned to the printer (W=ABC). Writer name (W) appears before the slash, indicating an exact match must occur; because it does not match, JES2 assigns a value of 00 to this output group.
2. JES2 checks JOB6.4.2 next. All criteria before the slash matches; however JES2 assigns a value of FD to the destination (R) criterion. This value denotes a match two levels lower than the maximum priority (FF). Note that the ROUTECDE= keyword has set priority order as follows: R3, LOCAL,**R106**,R105. JES2 assigns a value of FFFDFFFFFFFFF0078 to this output group. This output group is now the current **best JOE** (that is, most likely to print).
3. Next, JES2 checks output group JOB5.1.1 in the class A queue because class A has priority after class B but before class F. Here again, all criteria specified before the slash matches; however, the destination (R), class (Q), and process mode (PRM) all receive a value of FE denoting a priority one level below the maximum (FF). The entire value obtained by this output group

(FFFEFEFEFF00000015) is greater than that of JOB6.4.2 (see step 2 above). This output group then replaces JOB6.4.2 as the current **best JOE**.

4. JES2 now checks JOB53.6.2 in the class A. Again, all criteria specified before the slash matches; however, the class (Q) and process mode (PRM) receive a value of FE denoting one level below maximum because of their respective priority positions. The entire value obtained by this output group (FFFFFEFEFF00000010) is greater than that of JOB5.1.1 (see step 3 above). This output group now replaces JOB5.1.1 as the current **best JOE**.
5. The fifth output group JES2 checks is JOB8.3.2 in the class F queue. The scanning process stops because the process mode (PRM) for the output group (U3800) does not match that of the printer (PAGE, LINE, or UPLOT8). JES2 assigns a value of 00 to this output group.
6. Lastly, JES2 checks JOB22.2.1 in the class F. Here again, the scanning process stops immediately because the writer name (W=NONE) for the output group does not match that of the printer (ABC). Again, JES2 assigns a value of 00.

JES2 selects JOB53.6.2 as the best JOE for hardcopy output. The scanning process begins again when the output device is ready to select another output group, and JES2 determines the next **best JOE**.

To further illustrate how the ordering of the selection criteria can affect output selection, see Table 21 on page 114 and Figure 23 on page 115 through Table 27 on page 116(Section B) and review the following two examples:

1. The PRT(nnnn) statement for PRT2 lists the WS= keyword as WS=(Q,R,PRM/P). By putting the class (Q) criteria first in the WS list, JES2 searches for work one queue at a time. Note that JES2 selected JOB99.2.1 in the class B queue after scanning only two output groups. The next output group selected is JOB6.4.2. JES2 scans no other queues until it processes all selectable work on this queue.
2. The PRT(nnnn) statement for PRT3 lists the WS= keyword as WS=(P,Q,R/PRM,LIM). By putting the priority (P) first in the WS list, JES2 selects those jobs with the highest priority. (In the example, JOEs JOB6.4.2 and JOB22.2.1 both have a priority level of 78. JOE JOB6.4.2 is in the class B queue and JOE JOB22.2.1 is in the class F queue; therefore, because all other criteria before the slash match for both output groups, JES2 selects JOE JOB6.4.2, with the greatest value, as **best JOE**.

Tuning work selection criteria

JES2 trace ID 20 tracks the number of output elements each device examines before finding selectable output. Use trace ID 20 records to determine the optimum settings for the WS= parameter. The $TRACE output shows the effect of reordering the work selection list in terms of the amount of time spent and output examined for every queue search.

Table 28 illustrates table eight output groups on the output queues where the only relevant criteria are class, routecde, and priority.

Table 28. Example of output groups on output queues

ROUTECDE	A	CLASS B	C
LOCAL	JOE1	JOE3	JOE5 JOE7
R1	JOE2		

Table 28. Example of output groups on output queues (continued)

ROUTECDE	A	CLASS B	C
R2		JOE4 JOE6	
R3		JOE8	

The following general guidelines should help you tune your work selection parameters for the best performance. Evaluate these guidelines in regards to your specific environment.

- Remove irrelevant criteria. Criteria after the slash need not match. If a criterion is not important to you, remove it from the list. In particular, this applies to RJE devices where the only pertinent characteristics might be ROUTECDE (R) and CLASS (Q).
- Place WRITER (W) first in the list if there is a lot of output for a particular writer. This causes print/punch devices to reject work for a particular writer before examining any other criteria.

  You can improve performance further by keeping electronic mail files in a separate class from other print/punch output. This prevents printers or punches from examining large amounts of mail when selecting work.
- Always place CLASS (Q) in the WS list. Leaving any criterion out of the list tells JES2 not to consider that criterion when examining output. If CLASS is not in the list, JES2 scans **every** class queue looking for output which matches all the other characteristics. To limit the number of queues scanned, include CLASS in the WS list, and limit the number of classes specified.

  Using Table 28 on page 117 as a reference for the output queues, Table 29 illustrates the search results when CLASS is specified and when CLASS is not specified. The column "Output groups scanned" indicates how many output groups are examined before the device selects one to process. The column "Element selected" indicates which output group of those examined was selected.

Table 29. Search results when specifying CLASS

WS Criteria	Order of Selection	Output groups scanned	Element selected
WS=(Q,R) Q=BC R=(R2,Local)	JOE4 JOE6 JOE3 JOE5 JOE7	2 2 2 3 2	2 2 1 2 2
WS=(R) R=(R2,Local)	JOE4 JOE6 JOE1 JOE3 JOE5 JOE7	4 4 6 5 4 3	4 4 1 2 3 3

In Table 29 on page 118, JOE1 is selected when WS=(R) is specified because all class queues are examined. JOE1 is not selected when WS=(Q,R) is specified because only queues B and C are examined.

Assume JOE4 and JOE6 have been selected first because their routecode was a best match. They are longer no longer on the queues.

When WS=(R) JES2 searches the queue for work as follows:

– JOE1 has a ROUTECDE=LOCAL which is not a best possible match
– Starting with CLASS=A
  - JOE2 has a ROUTECDE=R1 which does not match at all
  - Continue searching on next queue
– Go to CLASS B:
  - JOE3 has a ROUTECDE=LOCAL. It is not the best possible so continue looking
  - JOE8, the ROUTECDE does not match at all
– Go to CLASS C:
  - JOE5. It is not the best possible and not better than JOE3. Continue searching.
  - JOE7. It is not the best possible and not better than JOE3
– All JOEs have been examined
– JOE3 is selected because it is the best one found.

- Give CLASS (Q) a higher priority than other criteria, particularly ROUTECDE (R). This improves performance by reducing the amount of output examined. When CLASS has a higher priority than ROUTECDE (farther to the left in the list), JES2 prints all eligible work from the first class queue before moving on to the next class. If CLASS has a lower priority, JES2 might unsuccessfully search every class queue, and possibly every output element, looking for a perfect match.

  The following illustrates the search results when CLASS is before ROUTECDE and when CLASS is after ROUTECDE.

Table 30. Search results when specifying class before ROUTECDE

WS Criteria	Order of Selection	Output groups scanned	Element selected
WS=(Q,R) Q=ABC R=(Local,R1,R2)	JOE1 JOE2 JOE3 JOE4 JOE6 JOE5 JOE7	1 1 1 3 2 2 2	1 1 1 1 1 2 2
WS=(R,Q) Q=ABC R=(Local,R1,R2)	JOE1 JOE3 JOE5 JOE7 JOE2 JOE4 JOE6	1 2 5 5 4 3 2	1 2 5 5 1 1 1

- Give careful consideration to PRIORITY (P) in the work selection list. If CLASS (Q) is before the slash in the list, then placing PRIORITY after CLASS has no effect and adds unnecessary overhead to JES2 because the class queues are already in priority order.

  Placing PRIORITY (P) before CLASS (Q), and before the slash, causes JES2 to look for the highest priority piece of output in the system. This results in long searches while JES2 examines every piece of output looking for the highest priority work.

  The following illustrates the search results when specifying PRIORITY and when PRIORITY is not specified.

Table 31. Search results when specifying PRIORITY

WS Criteria	Order of Selection	Output groups scanned	Elements selected
WS=(Q,R)	JOE1	2	1
Q=ABC R=(R2,Local)	JOE4 JOE6 JOE3 JOE5 JOE7	3 3 3 4 3	3 3 2 3 3
WS=(P,Q,R)	JOE1	8	1
Q=ABC R=(R2,Local)	JOE3 JOE4 JOE5 JOE6 JOE7	7 6 5 4 3	2 2 4 2 3

- If you include JOBname in the WS list, it causes JES2 to gather all output for a job and attempt to process that output consecutively. The JOBNAME= parameter for a particular device determines what job name JES2 selects when JES2 has processed all output that matches the current job name.
  - If JOBname= is specified before the slash in the WS= list:

    JES2 selects only those jobs that match the value specified on JOBNAME= for a specific device. When JOBNAME= is specified as a generic name, JES2 only processes jobs that match that pattern and attempts to process all output for that job name consecutively.
  - If JOBname= is specified after the slash in the WS= list:

    JES2 uses the value specified on JOBNAME= to select new job names for the specific device. If there are no job names that match the JOBNAME= specification, JES2 selects jobs as if JOBname were not listed in the WS= list. A device gathers and processes all output for a particular job before again examining the value specified on JOBNAME= for more jobs.

    If you require JES2 to group jobs based on job name but do not want JES2 to prefer a particular JOBNAME= pattern, specify JOBNAME=* on the device statement and code JOBname after the slash in the WS= list.

    Note:

    1. Use a $T *devname*,JOBNAME=*jobname* command to stop processing jobs of the current JOBNAME= specification and begin searching for jobs of the new name.

2. Specifying JOBNAME in the WS list causes JES2 to examine all output while gathering output for a job and finding job name matches. This causes performance degradation.

– If your installation has many active remote devices, minimize the number of queue elements searched by spreading the output for different remotes over multiple classes.

Many RJE devices that share the same class queue unnecessarily search a large queue of output destined for other remotes.

Carefully consider all work selection settings. Specifying the WS list incorrectly or inappropriately causes exhaustive searches of the JES2 output queues and excessive processing.

Output disposition for SYSOUT data sets

JES2 lets you use several initialization statements to specify how JES2 should process a SYSOUT data set when the SYSOUT data set is eligible for selection. You can base the processing on the successful execution of a job or task. The user can alter these specifications through JCL.

Specifying output disposition

You can specify default output dispositions for groups of output by coding the OUTDISP=parameter on any of the following initialization statements.

- JOBCLASS(v) for system output associated with started tasks, time sharing users, and batch jobs in a specific JOB class. (For example: JES2 system data sets for batch jobs, the JES job log.)

  Note: Only code OUTDISP= on JOBCLASS when you require that system data sets have a specific output disposition regardless of their message class (MSGCLASS=) specification. If you have no such requirement, either code OUTDISP=() or leave it off the JOBCLASS statement completely.
- OUTCLASS(v) for all batch SYSOUT (other than JES2 system data sets)

The OUTDisp= parameter has the format:

```
OUTDisp=[()[normal disposition][,abnormal disposition][)]
```

Figure 25. The OUTDisp= Parameter

The *normal disposition* subparameter specifies the disposition for the output if the job completes normally. JES2 chooses the normal disposition when the job's JCL contains no errors, the job does not abend, and the COND= parameter on the JCL JOB statement does not cause the job to fail. The *abnormal disposition* subparameter specifies the disposition for the output if the job does not complete successfully.

Note: The disposition for spin output is determined at the time that the output is spun; it does not change if a job fails later. The OUTDisp= parameter allows you to specify the following dispositions:

HOLD
Hold the output. JES2 does not process the output until you either change the disposition to WRITE or KEEP, or release the output. When the output is released, the disposition changes to WRITE.

KEEP Process the output and then keep a copy of it on spool. After processing, the disposition of this output becomes LEAVE.

LEAVE
JES2 does not process the output until you change the disposition to either WRITE or KEEP, or release the output. When the output is released, the disposition changes to KEEP.

PURGE
Purge the output immediately.

WRITE
Process the output then purge it.

Table 32 shows the defaults for output dispositions based on what you code.

Table 32. JES2 defaults for the OUTDISP parameter

If you code:	JES2 assumes:
No OUTDisp	OUTDisp=(WRITE,WRITE)
Normal end disposition only For example: OUTDisp=HOLD	Both values will be the one you specified for normal end. As in the example: OUTDisp=(HOLD,HOLD)
Abnormal end disposition only For example: OUTDisp=(,KEEP)	The value of the normal end disposition will be WRITE, while the abnormal end will be the value you gave it. As in the example: OUTDisp=(WRITE,KEEP)
Both the normal and abnormal end dispositions For example: OUTDisp=(PURGE,HOLD)	They will both assume the values you coded. As in the example: OUTDisp=(PURGE,HOLD)

Overriding output disposition

For a particular SYSOUT data set, a user might need to change the default output disposition that was set by initialization statements. The user can override the defaults through one of the following:

- JCL statements
- TSO/E commands
- Operator commands

Changing output disposition through JCL: The user can override output disposition through JCL either by coding HOLD=YES on a DD statement that defines the SYSOUT data set or by coding an OUTDISP= parameter on an OUTPUT JCL statement associated with a SYSOUT data set. Specifying HOLD=YES on the DD statement forces JES2 to treat the SYSOUT as if OUTDISP=(HOLD,HOLD) appears on an OUTPUT JCL statement associated with the SYSOUT.

Changing output disposition through TSO/E: When a user views held output using the TSO/E OUTPUT command, the user can choose to release the data set. Releasing the output would change the disposition of the output from HOLD to WRITE or from LEAVE to KEEP. Only output whose disposition, when the job executed, was HOLD or LEAVE **and** whose disposition has not been modified is

eligible for TSO/E OUTPUT command processing. See *z/OS TSO/E Command Reference* for more information about releasing output when using the TSO/E OUTPUT command.

Changing output disposition through operator commands: Use the $O (output) operator command to release output if it is HOLD or LEAVE, or use $T O to change the disposition of output to another disposition. You can use these commands to change the disposition of all output based on the time elapsed (n days or n hours) since a job created the output. Changing held output's disposition based on its age releases spool space occupied by output that is no longer needed. For more information on the $O command, see *z/OS JES2 Commands*.

JES2 overrides output dispositions in the following order:

1. HOLD= on the DD statement
2. OUTDisp=(normal,abnormal) on the OUTPUT JCL statement
3. OUTDisp=(normal,abnormal) on the JOBCLASS initialization statement, for JES2 system data sets
4. OUTDisp=(normal,abnormal) on the OUTCLASS JES2 initialization statement.

You should use OUTDisp= on the initialization statements to set up default dispositions: typically WRITE for normal disposition and some form of hold for abnormal disposition. If you need to change a disposition for a specific job, use OUTDisp= on the OUTPUT JCL statement.

JES2 keeps a record of the time a job creates output. Use the OUTTIME= parameter on the OUTDEF statement to control whether JES2 changes the creation time of the output to the current time when you change any characteristics of the output. See *z/OS JES2 Initialization and Tuning Reference* for the syntax of the OUTDEF statement.

Purging JES2 system data sets

JES2 purges system data sets if CONDPURG=YES is specified on the JOBCLASS initialization statements and the job produced no output data sets (or the output disposition of all the system data sets was PURGE). If both conditions are not met, JES2 system data sets obtain their output disposition from the OUTDisp= parameters specified explicitly on the JOBCLASS initialization statements. If OUTDisp= is not specified on the JOBCLASS initialization statements, JES2 uses the disposition specified on the OUTCLASS(n) initialization statement for that message class.

Note: The CONDPURG= parameter **does not** apply for jobs that did not complete successfully.

Offloading and reloading SYSOUT based on output disposition

During a spool offload or reload operation, you can select SYSOUT with specific output dispositions for JES2 to offload or reload, by specifying OUTDisp on the WS= parameter and the OUTDisp= parameter on either the OFF(n).ST or OFF(n).SR initialization statement. For example,

```
OFF(1).ST    ...,
             OUTDisp=(KEEP),WS=(,OUTDisp)
```

Figure 26. Example of OUTDisp as a Work Selection Criteria

causes the offload SYSOUT transmitter for offload device 1 to offload only SYSOUT with an OUTDisp of KEEP. “Variable work selection criteria” on page 113 and “SPOOL offload facility” on page 176 describe the WS=parameter and the spool offload facility.

Changing output disposition during reload: Specifying MOD=(OUTDisp=disposition) on the OFF(n).SR statement causes JES2 to change the output disposition of SYSOUT selected during the reload process.

Transmitting output to another node

When JES2 transmits output to another node, JES2 indicates the output's disposition in the network header. If the receiving node has a level of JES2 SP Version 4 or Version 5 installed, the receiving node keeps the disposition set by the transmitting node. Otherwise, the receiving node will recognize the data as non-held if the disposition set by the transmitting node was WRITE. If the disposition set by the transmitting node was HOLD, KEEP, or LEAVE, then the receiving node will recognize the data as held.

When a node with JES2 SP Version 4 or Version 5 receives output from a node with a previous level of JES2 installed, the JES2 SP Version 4 or Version 5 member assigns an OUTDisp of HOLD to the output if the data was held. If the output was non-held, the JES2 SP Version 4 or Version 5 member assigns an OUTDisp of WRITE to the output.

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 runs 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 on setting up automatic restart management and see *z/OS MVS Programming: Sysplex Services Guide* for more detail on how MVS in a sysplex controls job's under automatic restart management.

The following JES2-specific items need consideration 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 want to make a SYSOUT data set available for JES2 output service processing when the data set is closed, you need to use the FREE=CLOSE JCL option for the corresponding dynamic allocation option. automatic restart management does not delete such data sets when the job is restarted.

  If you need to retain a SYSOUT data set that was still open and not yet closed at the time of failure, you need to use the SPIN=UNALLOC JCL option (or corresponding dynamic allocation option). Using this option causes the data set to be unallocated and made available to JES2 at job termination time (that is, when the job or system initially fails and the job is restarted).

  The following table summarizes the use of the FREE=CLOSE and SPIN=UNALLOC JCL options and the effect they have on SYSOUT data created by automatic restart management - registered jobs.

Table 33. Effects of FREE=CLOSE and SPIN=UNALLOC JCL on SYSOUT data

Option Used	SYSOUT Data Set Status at Time of Restart by automatic restart management	JES2 Processing
FREE=CLOSE	Closed	The data set is unallocated and spun-off when closed by the job.
FREE=CLOSE	Open	JES2 deletes the data set.
SPIN=UNALLOC	Closed	The data set is unallocated and spun-off at job termination.
SPIN=UNALLOC	Open	The data set is unallocated and spun-off at job termination.
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 termination.

Note: SYSOUT data sets created and maintained by JES2 (for example, JESMSGLG, JESYSMSG, JESJCL) are not deleted when a job is created by automatic restart management.

- You cannot use JES2 spool offload to offload jobs that are waiting for restart by automatic restart management.

Dividing output into smaller units

Applications can decrease the turn-around time of large amounts of printed outputs that jobs create by specifying the SEGMENT= parameter on the DD JCL statement that defines the output for line mode data. SEGMENT= forces JES2 to spin off output after producing a specified number of pages. There will be a segment identifier on the separator page for each segment if segmentation is in effect. See *z/OS MVS JCL Reference* for more information on specifying the SEGMENT= parameter.

Segmenting output into many small pieces, however, could cause performance problems as each segment requires an output group to represent it. Specifying SEGLIM= on the OUTDEF initialization statement minimizes the impact output segmentation has on your installation. When the number of segments a job creates for any piece of output exceeds the SEGLIM= value, JES2 stops segmenting the output and prints the remaining output as one segment. See *z/OS JES2 Initialization and Tuning Reference* for additional information about the OUTDEF statement.

If a SYSOUT data set is segmented, each segment is treated as a separate file for printing, punching, or NJE transmission. Output segmentation can be used to make portions of the SYSOUT data set available earlier for processing or transmission before the job completes or before the entire data set is available. For NJE transmission, this can also improve line utilization by reducing the time to retransmit files over NJE should there be a line failure or other unrecoverable error.

Defining output limits

There are several ways in which an installation can set job output limits and define the actions that JES2 takes when a job exceeds those limits. For example:

- You can specify defaults in the JES2 initialization stream that limit the number of bytes, cards, lines, or pages that batch jobs or APPC transaction programs can generate before JES2 initiates some predefined action, such as canceling the job, canceling the job with a dump, or issuing a warning message to the operator. You can use the following initialization statements to set installation-wide job output limits:

  Statement
  Purpose

  ESTBYTE
  Estimated spool utilization for output exceeded

  ESTLNCT
  Default estimated print output and options

  ESTPAGE
  Default estimated page output exceeded

  ESTPUN
  Default estimated punch output and options
- You can implement JES2 installation Exit 9 to make decisions when output exceeds output limits. For APPC transaction programs, JES2 Exit 43 can be used to override SYSOUT limits. See *z/OS JES2 Installation Exits* for information about these exits.
- For individual batch jobs, end users can define output limits by specifying output limits on the JCL JOB statement or the JECL /*JOBPARM statement.
- The system does not cancel recovery termination manager (RTM) jobs for output limit exceeded for ESTBYTE, ESTLNCT, ESTPAGE, and ESTPUN with ABEND code 722. JES2 overrides normal excession processing to allow RTM to honor, without limitation, the collection of diagnostic information defined by SYSUDUMP, SYSABEND, and SYSMDUMP DD statements. JES2 ignores these output limits whether set by JES2 initialization statement, the JECL /*JOBPARM statement, or the JCL JOB statement.

If more than one of the methods listed above are being used to set output limits, JES2 uses a predefined order of overrides to determine which output limits and resulting system actions to honor. The order of overrides differ depending on whether the job is a batch job or a transaction program.

For batch jobs, JES2 uses the following order of overrides:

- JES2 initialization stream
- JECL /*JOBPARM statement (overrides the initialization stream)
- JCL JOB statement (overrides the /*JOBPARM JECL statement)

For output produced by transaction programs, JES2 uses the following order of overrides:

- JES2 initialization stream (lowest level)
- JES2 Exit 43 (overrides the initialization stream)
- JCL JOB statement (overrides JES2 Exit 43)

See *z/OS JES2 Installation Exits* for information about Exit 43.

The following example demonstrates the order of overrides when using the JCL JOB and JECL /*JOBPARM statements:

```
ESTBYTE NUM=9999
ESTLNCT NUM=2
ESTPAGE NUM=40
ESTPUN  NUM=100
```

Figure 27. Sample Initialization Stream Definitions

```
//MRECKSA  JOB  MARTY,...,PAGES=200,BYTES=20000
/*JOBPARM       LINES=100,BYTES=10000
//STEP1    EXEC PGM=IEBDG
```

Figure 28. Sample JCL with Output Limits

In the example shown above, the following output limits would exist:

- The batch job is limited to 20,000,000 bytes (20000x1000 - because the byte definition is in thousands). The BYTES= keyword on the JOB card overrides the JECL /*JOBPARM and the initialization statement default (ESTBYTE statement).
- The batch job is limited to 100,000 lines (100x1000 - because the line definition is in thousands). The value from the /*JOBPARM statement is used because it overrides the initialization stream default (ESTLNCT statement) and the LINES keyword is omitted on the JOB card.
- The batch job is limited to 200 pages. The value specified on the PAGES= keyword on the job card overrides the initialization stream default (ESTPAGE statement).
- The batch job is limited to 100 cards. The limit is taken from the ESTPUN initialization statement because no override value exists on the JCL JOB or JECL /*JOBPARM statements.

If a batch job or APPC transaction program exceeds the output limits set by JES2, JES2 issues message $HASP375. However, you can use JES2 Exit 9 to specify what action(s) should be taken if a job exceeds output limits. See *z/OS JES2 Installation Exits* for information about Exit 9.

See *z/OS MVS JCL Reference* for additional information about specifying output limits on JCL JOB or JECL /*JOBPARM statements. See *z/OS JES2 Initialization and Tuning Reference* for additional information about the ESTBYTE, ESTLNCT, ESTPAGE, and ESTPUN initialization statements.

Considerations for started tasks and TSO LOGONs

Output limits for TSO/E transmits can be set by TSO/E using the TSO/E OUTLIM= parameter. JES2 also sets a limit internally. When SYSOUT is transmitted in the foreground for started tasks and TSO/E LOGONs, the member uses the lower of these two limits. JES2 sets the following output limits for started tasks and TSO LOGONs:

- 999,999 for lines, cards, and pages
- 2,147,483 (in 1000s of bytes) for spool utilization.

An installation can change the limits for started tasks or TSO LOGONs by using JES2 Exit 20 to change the limit for each particular started task or TSO LOGONs The limit for TSO/E transmits which are specified thorough the OUTLIM parameter, should not be greater than the limit JES2 sets for punches or a X'722' abend will occur. See *z/OS TSO/E Customization* for information about limiting the TSO/E TRANSMIT command.

SAPI POST and GET JOE work selection optimization

SYSOUT work selection is the process of selecting SYSOUT output groups (JOEs) from JES2. Increases in the numbers of JES2 JOEs and JES2 devices that are processing JOEs can result in a performance decline during SYSOUT work selection for SAPI devices, which include SAPI printers and any applications that select SYSOUT for printing. In this case, JES2 SAPI POST and GET JOE work selection optimization are available to help improve SYSOUT work selection performance. SYSOUT optimization reduces sequential searching, which in turn reduces the volume of JOE investigation that is required for POST and GET operations.

Note: To use SYSOUT optimization, no changes to SAPI applications are required. After enablement, work selection merely uses an alternate code path. The two optimization types are independent of one another and can be enabled in any combination. Both types of optimization are designed for SAPI applications that select on Class, Destination (route code), Disposition, or any combination thereof.

SAPI POST work selection optimization

SAPI applications identify output characteristic interest and JES2 POST notifies applications when work with required characteristics is received. SAPI POST optimization uses a cached results algorithm and a binary tree to reduce the CPU overhead that is incurred with notification.

SAPI GET JOE Index work selection optimization

JOE Index augments function provided by SYSOUT class queues. JOE Index optimization classifies work JOEs by Class, Destination (route code) and Disposition. JOEs with any combination of these attributes are chained off an index JOE which resides within the JES2 checkpoint. An index JOE is a unique type of JOE in addition to the work JOE, characteristic JOE and Free JOE. The totality of all index JOEs comprise JOE Index optimization.

Four balanced binary trees are contained within each index JOE. The following combinations of JOE attributes can be chosen as a search key:

1. Class/Destination (route code)/Dispostion
2. Class
3. Destination (route code)
4. Disposition

These binary trees can be used to efficiently implement JOE selection for a range of work selection requests that use these attributes. For SAPI GET operations, JOE Index optimization can reduce the number of JOEs that are investigated. To benefit from JOE Index optimization your SAPI applications must be selecting based on one or more of the JOE attributes. If your SAPI applications are not selecting based on the four attributes, then the cost to maintaining the binary tree structure might be greater than its benefit. If this is the case, you could consider restructuring your select output to take advantage of the JOE Index structure.

Note: If you are selecting held output, the JOE Index structure helps to organize the held output on your system. This can alter the order in which output is presented to the SAPI application, but also provides significant improvement over the alternative selection process for held output.

Index JOEs contain the JOE Index binary trees. Once created, index JOEs are not freed even if the number of work JOEs chained drops to zero. The number of unique JOE combinations of Class, Destination and Disposition dictates the number of index JOEs which is required. As new output is

created, existing index JOEs are used. Large JES2 environments can use 2,000-3,000 index JOEs. Use the following command to return the number of index JOEs that are currently used:

```
$doutdef,joeuse

$HASP836 OUTDEF
$HASP836 OUTDEF  CURRENT JOE UTILIZATION
      $HASP836         TYPE        COUNT
      $HASP836         -------- --------
      $HASP836         WORK        50000
      $HASP836         CHAR          440
      $HASP836         INDEX         300
      $HASP836         FREE        34000
```

If the number of index JOEs that are in use becomes excessive, you can disable and then re-enable JOE Index to reduce their number. Subsequently, any index JOE that had no chained work JOE is freed. There is no disruption of service when JOE Index is disabled; SYSOUT class queues revert to the z/OS version 1.13 level.

Enabling and disabling SAPI POST and GET JOE Index optimization

To enable SAPI POST and GET JOE Index optimization, use the applicable keywords on the $T OUTDEF command. Both types of optimization are disabled by default. SAPI POST optimization can also be enabled during initialization. JOE Index optimization can only be enabled by operator command.

SAPI POST optimization

To enable SAPI POST optimization, enter:

```
$TOUTDEF,SAPI_OPT=YES
```

The scope of this type of optimization is the MAS member and is only active until this member is restarted.

To disable SAPI POST optimization, enter:

```
$TOUTDEF,SAPI_OPT=NO
```

Disabling optimization reverts to the z/OS version 1.13 level of JES2 work selection processing.

To display whether SAPI POST optimization is active, enter:

```
$DOUTDEF,SAPI_OPT
```

SAPI GET JOE Index optimization

To enable SAPI GET JOE Index optimization, enter:

```
 $TOUTDEF,WS_OPT=YES
```

The scope of this type of optimization is the entire JES2 MAS and persists until it is explicitly disabled.

To disable SAPI GET JOE Index optimization, enter:

```
$TOUTDEF,WS_OPT=NO
```

Disabling optimization reverts to the z/OS version 1.13 level of JES2 work selection processing, which uses SYSOUT class queues.

To display whether SAPI GET JOE Index optimization is active, enter:

```
$DOUTDEF,WS_OPT
```

JES2-provided client print services

JES2 provides client print services by allowing applications such as the IBM z/OS Print Server to allocate space on JES2 spool for print jobs submitted from workstations in the TCP/IP network.

Print jobs that use print services can take advantage of traditional JES2 print services. These include work selection and work balancing across logical printer groups, use of JES2 default SYSOUT attributes, SYSOUT processing progress and monitoring, and SYSOUT canceling, holding, and releasing.

JES2 provides these client print services in a manner compatible with its standard operational characteristics. For example, client print services can:

- Query their jobs to determine status, as does the JES2 operator
- Set jobs priorities consistent with JES2 priority values, and validate that a job can print on JES2 queues
- Use the SYSOUT application programming interface (SAPI).

Printers and punches

An operator responds to a request for a setup change following a $HASP190 message by doing one of the following:

- Performing the setup requested, then issuing the $S command to the device.
- Allowing the use of the setup only for the data set group that requests it, by issuing the $P command, followed by the $S command. The $P command causes the device to stop after printing the current data set group.
- Forcing an alternate setup on this data set group by issuing the $T command, followed by the $S command. The operator must set up the device, however, and must repeat the $T and $S commands for each data set in the group. JES2 considers header and trailer pages data sets for this sequence.
- Causing the selection of an alternate data set group by holding ($H command) the job, and then issuing the $I or $E command, which requeues the data set group in a held state. The operator must later release the held group.
- Deleting the data set group by issuing the $C command to the device. The device then selects another data set group for the setup or requests another setup.

Print chain alias for 1403 and 3211 printers

The system assigns an alias for each installation's standard print chain not defined on a given 1403 or 3211 printer. This alias provides JES2 with flexibility in scheduling printers for SYSOUT data sets. For example, a request for the 1403 TN train would be assigned the T11 train, if the data set prints on a 3211. Table 34 lists the assigned aliases, which follow the naming conventions currently used in SYS1.IMAGELIB.

Table 34. Assigned alias names for the 1403 and 3211 printer chains

Image	Alias
UCS1AN	UCS1A11
UCS1HN	UCS1H11
UCS1PN	UCS1P11
UCS1TN	UCS1T11
UCS2A11	UCS2AN
UCS2H11	UCS2HN

Table 34. Assigned alias names for the 1403 and 3211 printer chains (continued)

Image	Alias
UCS2P11	UCS2PN, UCS2RN, UCS2QN
UCS2T11	UCS2TN

The image and aliases are included in SYS1.IMAGELIB at system installation.

Some trains, such as SN and G11, do not have aliases because they have no equivalent train on another printer. An installation can assign an alias if it so chooses. (See *MVS/DFP Linkage Editor and Loader* for details about the ALIAS statement.) JES2 uses any aliases supplied. If no alias is available, you should use an installation-defined SYSOUT class or a printer routing code (specified by the DEST parameter) to assign the data set to the correct printer. JES2 notifies the operator when you direct it to print a data set on a printer for which the proper image does not exist, if you have not used a SYSOUT class or printer routing code. The operator can then print the data set with a valid train or redirect the data set to the proper printer by using the JES2 $E command.

3211 indexing

JES2 supports 3211 indexing through the specification of the INDEX parameter on the /*OUTPUT control statement, JCL OUTPUT statement, or in the extended FCB image. With the extended FCB image, JES2 supplies two special FCBs: FCB26 for 6 lines per inch and FCB28 for 8 lines per inch (specified as FCB=6 and FCB=8, respectively). These FCBs contain a channel 1 indication in position 1, a special index flag in the third byte, and the number of lines per inch in the fourth byte of the image. The special index flag in the third byte of FCB26 and FCB28 contains X'80' plus a binary index value, in the range of 1 to 32 (if not specified, 1 is the default). The index value sets the left margin (1 indicates flush-left position; other values cause indentation of the print line by n-1 positions).

For JES2 to use any other FCB images, they must specify channel 1 in position 1; otherwise, JES2 incorrectly positions the forms in the printer. (STD1 and STD2 do not specify channel 1 in position 1, therefore, you should not specify these for JES2 unless you alter them.) If the third byte of any other FCB image contains a data character (specifying the number of lines per inch) other than X'80', JES2 uses that specification and supplies an index value of 1.

Defining the advanced function printer (using the PRT(nnnn) statement)

JES2 supports Advanced Function Printing (AFP) printers (such as the3800 model 3 executing in full function mode) as local printers only. A 3800 model 3 running in compatibility mode is functionally equivalent to a 3800 model 1.

Specify a printer to operate under the control of a single functional subsystem by using the FSS= keyword on the JES2 PRT(nnnn) initialization statement. You must define each functional subsystem (and the printer(s) under its control) individually so you can access the functional subsystem applications (FSA) separately and use separate PROCLIB procedures for each functional subsystem (FSS). (See "Defining a functional subsystem for advanced function Printing (using the FSS(acccccccc) statement)" on page 132.)

You can start a 3800-3 printer running under the control of either JES2 or a functional subsystem. Other AFP printers must be run under the control of an FSS. To do this, set the MODE= keyword on the PRT(nnnn) initialization statement to either JES or FSS as appropriate.

The following affects how JES2 sets the MODE= parameter. Note that all parameters referred to belong to the PRT(nnnn) initialization statement.

- If you specify MODE=JES, MODE is set to JES unless you also specify an FSS procedure name (FSS=fssname) and the unit address (UNIT=) of an AFP1 unit control block (UCB). Then MODE=FSS.
- If you specify MODE=FSS, MODE is set to FSS unless you fail to specify an FSS.
- If you do not specify the MODE= parameter:
  - MODE= defaults to JES if you do not define an FSS procedure name.
  - MODE= defaults to JES if you define an FSS procedure name and a unit address that does not point to an AFP1 UCB.
  - MODE= defaults to FSS if you define an FSS without specifying the a unit address.
  - MODE= defaults to FSS if you define an FSS and the unit address of an AFP1 UCB device.

Note: If you do not specify the UNIT= parameter for a MODE=JES printer, you must use the $T PRT(nnnn) command to specify a unit address (UNIT=) before you can start the printer through a $S PRT(nnnn) command.

Specify a procedure in one of the started task procedure libraries on SYS1.PROCLIB through the PROC= keyword on the FSS(accccccc) initialization statement. This procedure must contain the JCL needed to start the functional subsystem. JES2 starts the indicated functional subsystem with a name of procname.fssname (where the fssname is the subscript on the FSS(accccccc) statement).

As system requirements change, you can use the $T PRTnnnn command to alter the initial setting of the printer. Some cases require you to drain the functional subsystem before issuing the $T command; other cases do not. This requirement is printer-specific.

Defining a functional subsystem for advanced function Printing (using the FSS(accccccc) statement)

Associate an FSS with an AFP printer such that the device receives the necessary FSA support. Use the FSS(accccccc) initialization statement to define an FSS to JES2 and associate the JES2 device to the required FSS and corresponding FSA.

- Use the subscript (accccccc) on the statement FSS(accccccc) to define a functional subsystem token. Then specify this token on the PRT(nnnn) statement through the FSS= parameter to associate the device with the appropriate FSS.
- Specify the name of the JES2 functional subsystem support load module name if it is different from HASPFSSM (the default) through the HASPFSSM= parameter. Ensure that the functional subsystem that you define can access the proper JES2 functional subsystem support load module that you require. This process might require a STEPLIB in the functional subsystem procedure to override the default load module name, HASPFSSM, on the FSS(accccccc) initialization statement. The HASPFSSM load module resides in SYS1.SHASLNKE.

Using advanced function printing printers

You can operate Advanced Function Printing printers only under the control of a functional subsystem and managed by a print services facility (PSF). Except for a 3800-3, you **cannot** operate an AFP printer in compatibility mode; JES2 will not permit you to change the operation of an AFP printer to compatibility mode.

You can define a 3800-3 printer as running in either **full-function mode** and thereby capable of accepting "page" or "line" as the addressable unit of output or in **compatibility mode** with "line" as the addressable unit of output.

Compatibility-mode support

Although a page-mode printer is capable of printing either output type, a line-mode printer cannot process page-mode data.

A 3800-3 (which is an AFP printer) when running in **compatibility mode** is functionally equivalent to 3800-1 printers (which are line-mode printers). You need define no special address spaces or have any other special considerations when running a 3800-3 in compatibility mode. JES2 supports a 3800-3 running in compatibility mode as if it were a 3800-1. Figure 14 on page 69 provides an example of the JES2 initialization statements required to support a 3800-3 in either full-function or compatibility-mode.

Full-Function mode support

Full-function mode is using those functions of the printer that produce page-mode output. The concept of **page-mode** permits printed pages to contain both text data and graphical presentations. Page-mode printers allow for the specification of complex data attributes not available from printers without all-points-addressable capabilities.The user can define and request such attributes as:

- **Segments** - predefined portions of a page
- **Overlays** - predefined page templates
- **Images** - pictures and graphics
- **Type fonts** - collections of unique or stylized print characters.

The graphical material can include a wide range of print font types and sizes [1/18 inch (4 pts.) to 1/2 inch (36 pts.)] for use in text headings, logos, and imbedded artwork; shading of textural and user-produced graphics; and graph plotting. You can make the above print features available for page-mode printing by defining AFP printers as a page-mode printers.

The advanced function printing data stream (AFPDS) used in IBM page-mode printing does not allow the use of blank truncation within the data stream. Be certain that the output class that you define for the use of advance function printing (AFP) does not specify blank truncation. Specify BLNKTRNC=NO on the OUTCLASS statement; the JES2 default is BLNKTRNC=YES.

JES2 does not communicate directly with an AFP printer but rather through the functional subsystem interface (FSI) to an FSS. The FSI defined for an FSS specifies the manner in which the FSS/FSA and JES2 communicate. IBM designed a specific FSS/FSA, the Print Services Facility, to support the AFP printer. See "Functional subsystem support" on page 68 for a more detailed description and Figure 14 on page 69 for a pictorial description of these relationships.

Route data sets that include both page-mode and line-mode data to page-mode printers exclusively, and define the data sets as page-mode on the JCL OUTPUT statement. JES2 provides job output routing, through the printer work selection

criteria (WS= parameter the PRT(nnnn) statement). To define an AFP printer as page-mode, also specify PRMODE=(PAGE,...) on the PRT(nnnn) statement.

Functional subsystem support of a page-mode printer applies to all page-mode printers in a multi-access spool JES2 complex. Therefore, both you and the operator need to be aware of what printers in your complex can process of which type of work at any one time. Table 35 summarizes this output processing support by printer type.

Table 35. Summary of data type and printer mode compatibility. The first column describes data process mode. The second column describes printer process and page mode.

	PRINTER PROCESS MODE (PRMODE)
DATA PROCESS MODE (PRMODE)	**PAGE-MODE&tab;LINE-MODE***
LINE	YES**&tab;YES
PAGE	YES&tab;NO***

* A page-mode printer operating in 3800 compatibility mode is equivalent to a line-mode printer.

** This assumes the printer is defined to accept line mode data, that is, PRT(nnnn) PRMODE=(LINE,PAGE...)....

*** If the operator forces a line-mode printer to print a page-mode data set, the data set will print with undesirable results.

Creating output for a 3540 diskette writer

A SYSOUT data set to be written to a 3540 diskette *must* be written by a diskette writer, which is an external writer. You can write to a 3540 diskette through the following parameters on a JCL DD (data definition) statement:

- Specifying DSID= to indicate the data set identifier.
- Specifying SYSOUT= to indicate an output class or the diskette writer process.

See *z/OS MVS JCL Reference* for more information on both the the DSID= and the SYSOUT= parameter of the DD statement. See *OS/VS2 IBM 3540 Programmer's Reference* for details on the diskette writer.

The JES2 print separator

The JES2 print/punch processor writes separation records (pages or cards) before writing the output of each job. These separation records make it easy to identify and separate various job outputs written contiguously on the same printer or card punch device. The separator appears at the beginning and the end of a job's output or at the beginning of interrupted (continued) output.

The print separator consists of a detail box and, optionally, the jobname and jobid in block letters.Use the SEPPAGE= parameter of the PRINTDEF initialization statement to specify whether JES2 produces the print separator page and to specify the format of the print separator page for all local and remote printers. *z/OS JES2 Initialization and Tuning Reference* contains the syntax of the SEPPAGE= parameter. Figure 29 on page 135 shows an example of a complete START print separator page.

```
         JJJJJJJJJJ   000000000    BBBBBBBBB     000000000        1
         JJJJJJJJJJ  00000000000   BBBBBBBBB    00000000000      11
             JJ      00       00   BB      BB   00      000     111
             JJ      00       00   BB      BB   00     0000    1111
             JJ      00       00   BB     BB    00    00 00   11111
             JJ      00       00   BBBBBBBB     00   00  00      11
             JJ      00       00   BBBBBBBB     00  00   00      11
             JJ      00       00   BB     BB    00 00    00      11
         JJ  JJ      00       00   BB      BB   0000     00      11
         JJ  JJ      00       00   BB      BB   000      00      11
         JJJJJ       00000000000   BBBBBBBBB    00000000000      11
         JJJJJ        000000000    BBBBBBBBB     000000000       11


 JJJJJJJJJJ   000000000    BBBBBBBBB     000000000        1        1        1
 JJJJJJJJJJ  00000000000   BBBBBBBBB    00000000000      11       11       11
     JJ      00       00   BB      BB   00      000     111      111      111
     JJ      00       00   BB      BB   00     0000    1111     1111     1111
     JJ      00       00   BB     BB    00    00 00   11111    11111    11111
     JJ      00       00   BBBBBBBB     00   00  00      11       11       11
     JJ      00       00   BBBBBBBB     00  00   00      11       11       11
     JJ      00       00   BB     BB    00 00    00      11       11       11
 JJ  JJ      00       00   BB      BB   0000     00      11       11       11
 JJ  JJ      00       00   BB      BB   000      00      11       11       11
 JJJJJ       00000000000   BBBBBBBBB    00000000000      11       11       11
 JJJJJ        000000000    BBBBBBBBB     000000000       11       11       11


 ***START*****START*****START*****START*****START*****START*****START*****START***
 *                                                                               *
 * JOBID:          JOB0111                                                       *
 * JOB NAME:       JOB01                                                         *
 * USER ID:        ZOOT                                                          *
 * SYSOUT CLASS:   A                                                             *
 * OUTPUT GROUP:   GRP01.CLASSA.D3289                                            *
 * TITLE:          Corporate Checking Account History                            *
 *                                                                               *
 * DESTINATION:    NEW YORK                                                      *
 * NAME:           DB Pizzuto                                                    *
 * ROOM:           2G-54                                                         *
 * BUILDING:       Textile Building                                              *
 * DEPARTMENT:     Accounting                                                    *
 * ADDRESS:        999 W. 99th Street                                            *
 *                 New York, New York                                            *
 *                 10000                                                         *
 *                 212-555-3487                                                  *
 *                                                                               *
 * PRINT TIME:     13:47:13                                                      *
 * PRINT DATE:     1 APR 2003                                                    *
 * PRINTER NAME:   PRT2                                                          *
 * SYSTEM:         IBM1                                                          *
 *                                                                               *
 ***START*****START*****START*****START*****START*****START*****START*****START***
```

Figure 29. Sample JES2 print separator page

After defining the global separator page options through the PRINTDEF statement, you can suppress production of the separator page by specifying SEP=NO on the PRT(nnnn) or R(nnnnn).PR(m) statements, or through the use of an Exit 1 routine.

The user can supply the information JES2 uses to build the separator page in either the OUTPUT JCL statement or the JOB JCL statement. See *z/OS MVS JCL Reference* for the parameters of the OUTPUT and JOB statements. The following fields can also be modified through SDSF while output resides on the output or hold queue:

- ADDRESS

- BUILDING
- DEPARTMENT
- NAME
- ROOM
- TITLE

Modifying the JES2 print separator

The IBM-defined installation exit, Exit 1, allows you to provide an exit routine to control production of the standard separator page or to create your own separator page to replace or print with the standard separator page. Additionally, the IBM-defined installation exit, Exit 23, allows you to provide exit routines to control production of a job separator page for page-mode printers. This exit allows only separator page suppression or modification of the FSI job separator page area (JSPA) control block used to build the separator page. Print Services Facility creates the separator page.

JES2 writes the separator pages before and after processing the output that one output group represents.You can create separator pages for each data set, or a copy of each data set, by enabling Exit 15 (Output Data Set/Copy Separators). If you do not enable Exit 15, JES2 prints separator pages only for the start and end of an output group. See *z/OS JES2 Installation Exits* for information on installation exit routine coding and implementation.

Creating a JESNEWS data set

The JESNEWS data set is EBCDIC text (news) to be printed by the JES2 print processor immediately following the JES2 information at the beginning of the output separator page. Use the JESNEWS data set to distribute information of general interest to users of a computing installation. For example, you can inform your customers of updated libraries or scheduled outages. This news can appear on the separator page itself if the number of news lines plus the number of lines on the print separator fits on a single page.

The JESNEWS data set is represented in the system by a started task (STC) with the job name $JESNEWS. The news is shared among all processors in a multi-access spool configuration; when one processor updates the news, all processors automatically pick up the updated news.

The JESNEWS data set is created by a user-defined job through the use of a SYSOUT DD statement. The user-defined job's SYSOUT DD statement must specify an external writer name of JESNEWS. When RACF is active, the userid on the job creating the JESNEWS data set must have sufficient RACF authority to update the JESNEWS data set. If RACF is not active, JES2 issues message $HASP360 to ask the operator if the user-defined job has authority to create or change the JESNEWS data set. See Chapter 7, “Providing security for JES2,” on page 349 for additional information on protecting JESNEWS.

Any direct access file with a record length (LRECL) and block size (BLKSIZE) that is acceptable to IEBGENER as SYSUT1 input can be used to create the JESNEWS data set.

Note:

1. You must use ANSI-control characters for JESNEWS to print properly on a PSF-controlled printer.

2. If RACF and the SECLABEL class are active, RACF assigns the SECLABEL of the last job that updated JESNEWS to the JESNEWS profile. This could cause jobs with lower SECLABELs than the updating job to miss important information and RACF will record security violations for jobs accessing JESNEWS that did not previously occur.

For example, to create a JESNEWS data set, code:

```
//UP$NEWS    JOB     ...
//JESNEWS    EXEC    PGM=IEBGENER
//SYSIN      DD      DUMMY
//SYSPRINT   DD      DUMMY
//SYSUT2     DD      SYSOUT=(A,JESNEWS),DCB=(LRECL=80,
//           RECFM=FBA,BLKSIZE=80)
//SYSUT1     DD      *
                     NEWS
                     NEWS
                      .
                      .
                      .
                     NEWS
 JES2 is up and works fine.
 The computer center will be closed on Sunday, the 23rd.
/*
```

When defining the JESNEWS data set on the SYSOUT DD statement, all other DD parameters (except SYSOUT CLASS, DCB, HOLD, and FREE)are ignored. The class name subparameter of the SYSOUT parameter is used only to determine dummy and hold status.Any /*OUTPUT and JCL statements associated with the created JESNEWS data set are ignored. If JESNEWS is defined as a non-held data set, the news will be updated at unallocation time. If you define JESNEWS as a held data set, JES2 will update the news when the data set is released, either by the operator (through either a $O J or a $T OJ command) or by a time-sharing request. JES2 writes a message ($HASP316) to the console whenever a JESNEWS data set is replaced or deleted.

Both forms of carriage control, ANSI or machine code, are accepted (see the RECFM subparameter of DCB in *z/OS MVS JCL Reference*). If the JESNEWS data set exists, the page eject (skip-to-channel 1) operation following the printing of header pages is suppressed so that the installation news can be printed on the separator page. If you want the news to start on its own page, then the appropriate carriage control character should be used.

The JESNEWS is deleted by creating a null data set. This is accomplished by opening and then closing a JESNEWS data set that has no data, or by not opening the JESNEWS data set after it has been allocated. The JESNEWS data set is also deleted by a cold start. There are no restrictions as to when the JESNEWS data set can be changed.

To delete a JESNEWS data set by opening a null data set code:

```
//DEL$NEWS   JOB     ...
//           EXEC    PGM=IEBGENER
//SYSPRINT   DD      DUMMY
//SYSIN      DD      DUMMY
//SYSUT1     DD      DUMMY
//SYSUT2     DD      SYSOUT=(A,JESNEWS)
```

To delete a JESNEWS data set by not opening the data set code:

```
//DEL$NEWS   JOB    ...
//DELNEWS    EXEC   PGM=IEFBR14
//JESNEWS    DD     SYSOUT=(A,JESNEWS)
/*
```

Additionally, if you need to suppress the printing of the JESNEWS data set, use Exit 1 or Exit 23. See *z/OS JES2 Installation Exits* for information about these exits.

JES2 system data sets

The job log is a collection of data sets that include the user's console messages and replies to WTORs issued during job processing. The job log can appear in the job output on the page following the job's separator page. The job log is grouped into a single output group and only printed once, unless JESLOG=SPIN is specified. Installations can choose to print or suppress the job log by using the JES2 JOBCLASS initialization statement or by using JCL.

The system ID and system NODE found in the title line of the JES2 job log are the JES2 member name and NODE name of whichever phase of JES2 processing first puts messages into the job log.

To provide an indication when a particular message was logged for jobs that run multiple days, JES2 provides day/date messages in the JES2 JOBLOG. The time stamp on the first message matches the time that the first data record (PUT) was processed. The time stamp on all subsequent day/date messages is always midnight (00.00.00). See Figure 30 to illustrate the following example: If a job is submitted on day 1 (27 MAR 1995), issues no messages for the next three days (28, 29, and 30 March), and ends on day 5 (31 MAR 1995), the JOBLOG contains two dates only; the first at the top of the JOBLOG when the job began and the second before the end on day 5 at midnight. There is no entry for a day in which no messages were logged, and therefore, never two consecutive lines with the day/date message.

```
J E S 2 J O B  L O G - S Y S T E M  A Q T S - N O D E P O K

- MONDAY, 27 MARCH 2003 -
IRR010I  USERID PAULAC IS ASSIGNED TO THIS JOB.
ICH70001I PAULAC   LAST ACCESS AT 09:13:23 ON WEDNESDAY, MARCH 29, 2003
$HASP373 PAULACD  STARTED - INIT   12 - CLASS U - SYS AQTS
IEF403I PAULACD - STARTED - TIME=09.17.56
- ------------------------------------------------------------------------------
-                              REGION      - STEP TIMINGS -           -PAGING COUNTS-
- STEPNAME PROCSTEP PGMNAME  CC USED   CPU TIME ELAPSED TIME EXCP SERV PAGE SWAP VIO SWAPS
- TSO               IKJEFT01 00 452K 00:00:01.46 00:00:06.48  538 37817   0    0   0     0
- FRIDAY,   31 MARCH 2003 -
$HASP395 PAULACD   ENDED
IEF404I PAULACD  - ENDED - TIME=15.20.02
- ------------------------------------------------------------------------------
- NAME-PAULAK    TOTALS: CPU TIME=00:00:51.46 ELAPSED TIME=04:06:06.51 SERVICE UNITS=37817
- ------------------------------------------------------------------------------
- JES2 JOB STATISTICS -
  27 MAR 2003 JOB EXECUTION DATE
           72 CARDS READ
        7,287 SYSOUT PRINT RECORDS
```

Figure 30. Sample JES2 job log

JES2 produces JES2 system data sets for each job according to parameters defined on the JESDS JCL statement or the /*JOBPARM JES2 control statement. These data sets are collectively called the *job log*. These data sets are written by JES2 and

accompany user's output; they can also be accessed by the process SYSOUT (PSO) interface through the external writer or using the TSO/E OUTPUT command.

Table 36 shows the names and purposes of these JES2 system data sets.

Table 36. JES2 system data sets

Data Set Name	Purpose of data set
JESJCLIN	Indicates the actual JCL submitted.
JESMSGLG	Indicates any system messages for this job.
JESJCL	Indicates all job control statements in the input stream, that is, all JCL statements and JES2 control statements, plus all procedure statements from any in-stream or cataloged procedure a job step calls, plus all messages about job control statements.
JESYSMSG	Indicates the job's hardcopy log, which contains the JES2 and operator messages about the job's processing: allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.

JES2 does not produce JES2 system data sets for jobs that do not complete execution (that is, jobs that specify TYPRUN on JOB JCL statements, JCL errors, or failure because of an MVS system failure). See *z/OS MVS JCL Reference* for information about using the job log.

For more information on displaying or suppressing a job log, see *z/OS JES2 Initialization and Tuning Reference*.

The JES2 job statistics

The JES2 job statistics uses data accumulated while processing JCL statements and SYSIN data to determine the number of input records read, and, while creating SYSOUT, to determine the number of print and punch records. The number of records reflect the actual records written to the spool and might not necessarily indicate the number of cards, lines, or pages of output; this is also true for the number of SYSOUT spool bytes. For example, specifying multiple copies or using a punch with print commands can result in differences between the number of records in the JES2 job statistics and the number of lines or pages actually printed or cards actually punched.

The following is an example of the JES2 job statistics as they appear in the JES2 JOBLOG:

```
--------JES2 JOB STATISTICS--------

 14 OCT 2003 JOB EXECUTION DATE

     8,075 CARDS READ

       428 SYSOUT PRINT RECORDS

         0 SYSOUT PUNCH RECORDS

        82 SYSOUT PAGE RECORDS

        36 SYSOUT SPOOL KBYTES

      0.52 MINUTES EXECUTION TIME
```

The JES2 punch separator card

JES2 can optionally precede each job's punch output with an identification card.To make the card easy to identify, it has an 11-punch and a 12-punch in all 80 columns. To make the room number and job number easy to read, each digit extends over 6 columns. JES2 converts alphabetic characters into numerical digits as shown in Table 37.

Table 37. Alphabetic character conversion to digits for JES2 punch separator cards

Alphabetic Characters	Numeric Digits
A or J	1
B, K, or S	2
C, L, or T	3
D, M, or U	4
E, N, or V	5
F, O, or W	6
G, P, or X	7
H, Q, or Y	8
I, R, or Z	9

JES2 routes the punch separator card to the stacker designated as the default on each punch device as follows:

- The default for the 2540 punch device is stacker 2. JES2 punches a blank card immediately following the separator card for this device; then JES2 routes this card to stacker 1. JES2 also punches a blank card at the end of each data set and routes them to stacker 1. The blank cards in stacker 1 separate error cards from unselected stacker 2 data sets.
- The default for the 3525 punch device is stacker 1. JES2 punches a blank card at the end of each data set and routes the card to stacker 1. These blank cards separate data sets in stacker 1 from unselected cards.

On either device, JES2 routes any card that has an incorrect carriage control (command control) or incorrect print line to stacker 1.

Output routing

If allowed by the security product, you can route output to:

- A specific local or remote device
- A remote job entry (RJE) workstation
- A pool of remote job entry (RJE) workstations
- A JES2 node or nodes
- A non-JES2 node or nodes
- Remote terminals attached to a JES2 or non-JES2 node
- A specific destid at this or another node
- A specific userid at this or another node

Routing data sets

You can route a specific data set by using the DEST parameter of the DD statement, a specific data set or group of data sets by using the OUTPUT JES2 control statement or the OUTPUT JCL statement, or the entire print/punch output for the job by using the /*ROUTE statement.

JES2 uses the destination specified for a data set on the OUTPUT JES2 control statement, the OUTPUT JCL statement, or the DD JCL statements. However, output data sets routed to a remote workstation using the OUTPUT or DD JCL statements, or the DEST= DD parameter cannot be altered by a remote operator; the job-level default controls access to those data sets. To discover the job-level default, enter the $D J1-999 operator command. For data sets with no destination specified, JES2 uses the destination on the /*ROUTE statement or the default destination specified on the input devices from which the job was submitted.

The DEST=parameter on the DESTID(jxxxxxxx) initialization statement provides route codes for destinations on this node and other nodes.A complete destination specifies both a second-level destination and first-level destination for SYSOUT. Second-leveldestinations specify the final destination (LOCAL, Rmmmm, Unnnn, userid) at a node (the first-level destination) specified previously on either a /*ROUTE PRT or a /*ROUTE PUN JES2 control statement, or defaulted as specified below. For more information on how to specify and use destination identifiers (destids), see "Using destination identifiers to route output" on page 142.

Pooling remote devices

By assigning the same destid to a group of remote workstations or a group of devices, you can create a remote pool or device pool. Remote device pooling can identify a particular printer for more than one remote workstations. For more information on pooling remote devices, see Chapter 6: "Remote Job Entry".

Specifying default output routing on input devices

You can specify the default print and punch destination on the readers through which the system receives the job, using the following statements:

- RDR(nn) PRTDEST= ,PUNDEST= for local readers.
- R(nnnnn).RD(m) PRTDEST= ,PUNDEST= for remote readers.
- For INTRDR(nn) specifying a destination by way of JCL (DEST=).

(If you do not specify the default print and punch destination, the default output routing becomes the destination associated with the input device.)

In the case of an internal reader, the DEST= parameter for the internal reader allocation determines the default print and punch destination. If the DEST= parameter does not specify a destination for the internal reader, the destination for the output is the location that originally submitted the job. For example, you can submit a job on NODEA and route it for execution on NODEB. The output, however, returns to NODEA unless you specified the DEST= parameter as NODEB or some other location.

You can use JCL with either the /*ROUTE statement or the DEST= subparameter in the DD SYSOUT statement to specify a particular device.

Allowing output devices to select route codes: You can define output devices, such as printers and punches, so that they can selectroute codes for themselves and for remote workstations during JES2 initialization. Use the following initialization statements:

- OFF(n).ST for SYSOUT transmitters.
- PRT(nnnn) for local printers.
- PUN(nnnn) for local punches.
- R(nnnnn).PR(m) for remote printers.
- R(nnnnn).PU(m) for remote punches.

If you specify R on an output device's work selection parameter, the device can select only output with a route code that matches a route code specified on the output device's Routcde= parameter. To specify work selection values for an installation's readers, printers, and punches, see “Variable work selection criteria” on page 113. Most devices specify R in their WS= list. If they do not, JES2 assumes that the device can select output with any route code. JES2 does not allow printers and punches to select output destined to a different node.

Using destination identifiers to route output

JES2 allows users to specify where output should go by using destination identifiers. These destination identifiers (destids), together with JES2 route codes, allow installations a method of sending output to different nodes, remote workstations, and special local devices.

JES2 destinations are composed of two parts:

- A first-level destination indicating a particular node.
- A second-level destination indicating a remote workstation, special local route code, or userid at that node.

Each node in a JES2 network automatically receives a destination identifier through the NAME= parameter on the NODE initialization statement and the $T NODE command. If you do not specify a destination identifier for the NAME= parameter, JES2 provides a default value. For example, the NODE(1) NAME= parameter defaults to‘N1’where‘N’is a special prefix indicating this is a node.

Installations can specify a symbolic name as a destid through the DESTID initialization statement subscript (jxxxxxxx). The DESTID subscript (jxxxxxxx) specifies the 1- to 8-character name that installations can use to refer to these JES2 destinations.

The DEST= parameter on the DESTID(jxxxxxxx) initialization statement specifies a route code that JES2 uses to send both jobs and SYSOUT from one destination to another. These route codes can be defined as numbers preceded by a prefix that has special meaning. For example, the prefix ‘N’ can indicate that the destination identifier represents a particular node. Other prefixes represent a remote workstation (‘R’, ‘RM’, or ‘RMT’) or a special local route code (‘U’) defined across a number of installations.

Specifying SYSOUT destination through JES2 route codes: When SYSOUT data sets are created, they are assigned routing information. That routing information can cause the data set to be sent to another node or be processed locally. Local printers are also assigned routings that are used to select data sets to print. Routings that are valid for printer selection are:

- LOCAL or ANYLOCAL
- Remote routing (Rmmmm)
- Special local routing (Unnnn)
- User routing (8-character name with no special meaning).

Using remote routings or special local routings provides better performance because there tends to be less output queued to those routings. But these routings are difficult for end users to remember and cause problems if they need to change. By using the DESTID(jxxxxxxx) initialization statement, an installation can associate a meaningful name with a remote or special local routing that is easy to remember and insulates the end user from changes made to the system configuration.

Figure 31 illustrates how an installation can use the DESTID(jxxxxxxx) initialization statement to associate a meaningful name with a special local routing that insulates end users from changes made to the network configuration. Nodes 1 and 2 use a 3820 printer (PRT1) located at Node 1 to print all SYSOUT. The destid 'TOM' can be specified (DEST=TOM) on either a JCL statement or a JES2 control statement at other nodes to ensure that JES2 routes SYSOUT to the 3820 at NODE 1.

```
At Node1:  DESTID(TOM) DEST=N1.U5
           PRT(1) START=YES,SEP=NO,UNIT=0008,R=U5

At Node2:  DESTID(TOM) DEST=N1.U5
```

Figure 31. Using Destids to Route Output to a Printer at Another Node

If end users and application programs are using destids to route SYSOUT and jobs, they will be insulated from configuration changes. As configurations change, installations can change the routing associated with the DESTID(jxxxxxxx) initialization statements to match the new configurations.

For example, an installation wants to move printer 1 to Node 2 or switch the work load that had been sent to Node 1 to another printer at Node 2. The installation can define a printer at Node 2 that selects the same special local route code and change only the first-level destination identifier of DESTID(TOM) from 'N1' to 'N2'. SYSOUT with a destination of 'TOM' is now sent to Node 2, where it is selected by the printer with the same special local routing. Figure 32 displays the changed initialization statements.

```
At Node1:  DESTID(TOM) DEST=N2.U5

At Node2:  DESTID(TOM) DEST=N2.U5
           PRT(1) START=YES,SEP=NO,UNIT=0008,R=U5
```

Figure 32. Destids When Printer Moved to Node 2

Routing Output After a Configuration Change: To ensure that all output that had been created using the old destid of 'U5' does not remain on spool unprocessed, operators should route this output from special local routing 5 to destid 'TOM' through the $R command:

```
$R ALL,R=U5,D=TOM
```

Specifying SYSOUT destination through user routing: Insulation from configuration changes can also be achieved for different nodes in a network through user routing. Installations at nodes 1, 2, and 3 can logically move a printer from Node 2 to Node 3 without causing Node 1 to make any changes.

For example, consider the following figure. An installation has eliminated printer 1 from NODE2. Work that had printed on printer 1 should be routed to printer 3 at node 3. End users and application programs on nodes 1, 2, and 3 all route SYSOUT to the printer using a destid of 'TOM'. Figure 33 on page 144 shows the destids before and after the move.

```
BEFORE
At Node1:  DESTID(TOM) DEST=N2.TOM

At Node2:  DESTID(TOM) DEST=N2.U5
           PRT(1) START=YES,SEP=NO,UNIT=0008,R=U5

At Node3:  DESTID(TOM) DEST=N2.TOM

AFTER
At Node1:  DESTID(TOM) DEST=N2.TOM

At Node2:  DESTID(TOM) DEST=N3.TOM

At Node3:  DESTID(TOM) DEST=N3.U3
           PRT(3) START=YES,SEP=NO,UNIT=000A,R=U3
```

Figure 33. Moving Printers without Notifying Other Nodes

Notice in Figure 33, the DESTID(TOM) on node 1 was not changed. Node 1 defined the destination identifier 'TOM' as meaning: Route the SYSOUT to whatever 'TOM' is on node 2. Before the move, 'TOM' on node 2 was the special local routing 'U5'. After the move, 'TOM' on node 2 was updated to route the SYSOUT to whatever 'TOM' is on node 3. This technique of routing a destid to user routings at other nodes allows configuration changes at one site without the need to update all other nodes simultaneously.

The use of symbolic route codes allows an installation to associate a meaningful name(s) with a remote or special local routing. Care must be used when creating a DESTID with a symbolic DEST= parameter if that parameter is also used as a DESTID subscript or node name in the same initialization stream. JES2 enforces an order-dependency of destination definitions:

1. Destinations defined within the same initialization stream.

   JES2 will allow the use of a symbolic DEST= parameter if it is:

   - Never defined as a DESTID subscript or node name in the same initialization stream. In this case, the symbolic DEST= parameter is treated as if it were a userid.
   - Used as a DESTID subscript or node name in an initialization statement that occurs earlier in the initialization stream. In this case, the destination will resolve to that defined for the first initialization statement. This 'look-up' will continue until a non-user route code is encountered.

   If a DESTID is created with a symbolic DEST= parameter, and that parameter is used as a DESTID subscript or node name later in the same initialization stream, JES2 will treat it as an error condition. Message $HASP512 will be issued to indicate an error for the first DESTID statement.

2. Destinations defined using the JES2 $ADD command.

   JES2 will resolve destinations defined using a $ADD DESTID(jxxxxxxx), DEST=symbolic_parameter in a method similar to that described above for DESTID statements that appear in initialization streams. The following rules apply if the symbolic DEST= parameter is:

   - Used as a DESTID subscript or a node name in the initialization deck, the destination will resolve to that defined by the initialization statement.
   - Used as a DESTID subscript in a previous $ADD DESTID(jxxxxxxx) command, the destination will resolve to that defined by that (previous) command.

- Not already defined as a destination it will always resolve to the DEST= parameter value, regardless of whether it is defined in a subsequent $ADD command.

Table 38 shows how an installation can use symbolic route codes to allow users to route SYSOUT to one node using several different destination names.

SYSOUT can be routed to node N10 using the destination names defined in the initialization stream, 'NYC' and 'BIGAPPLE'. SYSOUT can also be routed to N10 using destination name NEWYORK after the first $ADD command is issued. Because NODE10 is defined as N10 in the third $ADD command, the fourth $ADD command can use route code NODE10 to define destination BIGCITY, which again resolves to node N10.

Table 38. How JES2 resolves destinations with symbolic route codes. Each data cell in the first column can have up to two rows associated with it

Initialization Statements/JES2 Commands	JCL Destination	Resolved Destination
DESTID(BIGAPPLE) DEST=N10	BIGAPPLE	N10
DESTID(NYC) DEST=BIGAPPLE N(10) NAME=RUDYJ	NYC	N10
$ADD DESTID(NEWYORK), DEST=BIGAPPLE	NEWYORK	N10
$ADD DESTID(NYCITY), DEST=N10	NYCITY	NODE10*
$ADD DESTID(NODE10), DEST=N10	NYCITY	NODE10*
$ADD DESTID(BIGCITY), DEST=NODE10	BIGCITY	N10
* Destination NYCITY will always resolve to NODE10, because NODE10 was not yet defined.		

Table 39 illustrates the error case described in the 'initialization statement' section above. Symbolic DEST= parameter BIGAPPLE is used in the first initialization statement. However, DESTID BIGAPPLE is created in a subsequent initialization statement. JES2 will issue message HASP512 to indicate that DESTID NYC is in error.

Table 39. How JES2 Resolves Destinations with Symbolic Route Codes. Each data cell in the first column can have up to two rows associated with it

Initialization Statements/JES2 Commands	JCL Destination	Resolved Destination
DESTID(NYC) DEST=BIGAPPLE	NYC	LOCAL
DESTID(BIGAPPLE) DEST=N10	BIGAPPLE	N10

How JES2 resolves destinations from node to node

A destination can be specified to a JES2 system through a variety of methods, including the following:

- JES2 commands
- The DEST= parameter on a JCL OUTPUT statement
- An SDSF over-type field

- The PRT= or PUN= parameter on a JES2 control ROUTE statement.

There are different types of JES2 processing for destinations received at the:

- Origin node
- Destination node

Origin-node processing

At the first node (the origin node) to receive a JES2 destination through one of the methods cited above, destinations can be translated to new values when the specified second-level destination equals the subscript on a DESTID(jxxxxxxx) initialization statement. Two types of change can occur:

Case 1: The specified first-level destination equals the first-level destination on the DESTID(jxxxxxxx) initialization statement.

Action: JES2 replaces the specified second-level destination with the second-level destination on the DEST= parameter on the DESTID(jxxxxxxx) initialization statement.

Case 2: The DEST= parameter does not have a first-level destination.

Action: JES2 replaces the specified second-level destination with the second-level destination on the DEST= parameter of the DESTID(jxxxxxxx) initialization statement.

Note: If the DEST= parameter specifies a DESTID previously created in the initialization stream, then the DEST parameters of the first DESTID(jxxxxxxx) initialization statement will apply.

Destination-node processing

At the node where a JES2 destination is received, that destination can be translated to new values when the second-level destination received from the network job header equals the subscript on a DESTID(jxxxxxxx) initialization statement.

Action: JES2 replaces the entire destination with the value of that DEST= parameter.

Note: The data could be routed to another node.

Examples of JES2 routing SYSOUT destinations: The following examples, read from left to right, display how JES2 receives data, interprets that data, and resolves the destination as it travels from node to node. Throughout these examples, the following parameters on the DESTDEF initialization statement use their default values.

- Ndest=
- R|RM|RMTdest=
- Udest=

Example 1:

Table 40. How JES2 resolves destinations -- example 1

1 **Destination Specified at NODE 1**	2 **DESTID Specified at NODE 1**	3 **Destination Sent (NJE Header)**	4 **DESTID Specified at NODE 2**	5 **Destination Resolved**
NODE2.TOM	DESTID(TOM) DEST=NODE2.FRED	NODE2 FRED	DESTID(FRED) DEST=BOB	NODE2.BOB

- 1 The destination specified on a //OUTPUT JCL statement, NODE2.TOM, means: Send this SYSOUT to destination TOM at Node 2.
- 2 JES2 determines whether Node 1 has a definition for destination TOM. It finds a DESTID(TOM) initialization statement. Comparing the first-level destinations and finding that they match (that is, NODE2 on the OUTPUT JCL statement and the DEST= parameter specifies NODE2), JES2 uses the second-level destination (FRED).
- 3 JES2 puts the destination of FRED at NODE2 into the NJE header and sends the SYSOUT to Node 2.
- 4 At Node 2, JES2 takes the incoming destination of FRED and checks to see if it is defined as a destination identifier. There is a DESTID(jxxxxxxx) initialization statement definition for FRED. Its DEST= parameter specifies only a second-level identifier of BOB. Therefore, the first-level identifier is assumed to be Node 2.
- 5 The resolved destination is BOB at Node 2.

Example 2:

Table 41. How JES2 resolves destinations -- example 2

1 **Destination Specified at NODE 1**	2 **DESTID Specified at NODE 1**	3 **Destination Sent (NJE Header)**	4 **DESTID Specified at NODE 2**	5 **Destination Resolved**
NODE2.JACK	DESTID(JACK) DEST=NODE3.R5	NODE2 JACK	DESTID(JACK) DEST=NODE3.R5	NODE3.R5

- 1 The destination specified on an SDSF over-type field, NODE.JACK, means: Send this SYSOUT data set to destination JACK at Node 2.
- 2 JES2 determines whether Node 1 has a definition for destination JACK. It finds a DESTID(JACK) initialization statement definition. Comparing the first-level identifiers, JES2 finds no match (that is, NODE2 specified on the SDSF over-type and the DEST= parameter specifies NODE3). JES2 ignores the DEST= specification.
- 3 JES2 puts the destination of JACK at NODE2 into the NJE header and sends the SYSOUT to Node 2.
- 4 At Node 2, JES2 takes the incoming destination of JACK and determines if it is defined as a destination identifier. There is a DESTID(JACK) initialization statement. Its DEST= parameter specifies NODE3.R5. Because this is the specification defined at Node 2 (the destination node), JES2 uses it.
- 5 The resolved destination is R5 (remote workstation 5) at Node 3.

Example 3:

Table 42. How JES2 resolves destinations -- example 3

1 **Destination Specified at NODE 1**	2 **DESTID Specified at NODE 1**	3 **Destination Sent (NJE Header)**	4 **DESTID Specified at NODE 2**	5 **Destination Resolved**
NODE2.BILL	DESTID(BILL) DEST=NODE2.JOHN	NODE2 JOHN	DESTID(JOHN) DEST=NODE3.U5	NODE3.U5

- **1** The destination specified on a /*ROUTE PRT JES2 control statement, NODE2.BILL, means: Send this SYSOUT data set to destination BILL at Node 2.
- **2** JES2 determines whether Node 1 has a definition for destination BILL. It finds a DESTID(BILL) initialization statement definition. Comparing the first-level identifiers and finding that they match (NODE2 specified on the /*ROUTE statement and the DEST= parameter specifies NODE2), JES2 uses the second-level destination (JOHN).
- **3** JES2 puts the destination of JOHN at NODE2 into the NJE header and sends the SYSOUT to Node 2.
- **4** At Node 2, JES2 takes the incoming destination of JOHN and determines if it is defined as a destination identifier. There is a DESTID(JOHN) initialization statement. Its DEST= parameter specifies NODE3.U5. Because this is the specification defined at Node 2 (the destination node), JES2 uses it.
- **5** The resolved destination is U5 (special local routing 5) at Node 3.

Example 4:

Table 43. How JES2 resolves destinations -- example 4

1 **Destination Specified at NODE 1**	2 **DESTID Specified at NODE 1**	3 **Destination Sent (NJE Header)**	4 **DESTID Specified at NODE 2**	5 **Destination Resolved**
NODE2.SCOTT	DESTID(SCOTT) DEST=R5	NODE2 R5	N/A	NODE2.R5

- **1** The destination specified on a $TO J5,ALL,D= JES2 command, NODE2.SCOTT, means: Send this SYSOUT data set to destination SCOTT at Node 2.
- **2** JES2 determines whether Node 1 has a definition for destination SCOTT. It finds a DESTID(SCOTT) initialization statement definition. JES2 cannot compare first-level destinations because the DEST= parameter provides only a symbolic second-level destination. Therefore, JES2 uses the first-level destination from the $T O command and combines it with the second-level destination on the DEST= parameter from the DESTID(jxxxxxxx) initialization statement.
- **3** JES2 puts the destination of R5 at NODE2 into the NJE header and sends the SYSOUT to Node 2.
- **4** At Node 2, because the incoming destination is an explicit destination (referring to a specific route code) as opposed to a symbolic destination, JES2 uses it.
- **5** The resolved destination is R5 (remote workstation 5) at Node 2.

Example 5:

Table 44. How JES2 resolves destinations -- example 5

1 **Destination Specified at NODE 1**	2 **DESTID Specified at NODE 1**	3 **Destination Sent (NJE Header)**	4 **DESTID Specified at NODE 2**	5 **Destination Resolved**
NODE2.ROY	DESTID(ROY) DEST=FRED	Node2 FRED	DESTID(FRED) DEST=U2	NODE2.U2

- 1 The destination specified on a //OUTPUT JCL statement, NODE2.ROY, means: Send this SYSOUT data set to destination ROY at Node 2.
- 2 JES2 determines whether Node 1 has a definition for destination ROY. It finds a DESTID(ROY) initialization statement definition. JES2 cannot compare first-level destinations because the DEST= parameter provides only a symbolic second-level destination. Therefore, JES2 uses the first-level destination from the OUTPUT JCL statement and combines it with the second-level destination on the DEST= parameter from the DESTID(jxxxxxxx) initialization statement.
- 3 JES2 puts the destination of FRED at NODE2 into the NJE header and sends the SYSOUT to Node 2.
- 4 At Node 2, JES2 takes the incoming destination of FRED and determines if it is defined as a destination identifier. There is a DESTID(FRED) initialization statement. Its DEST= parameter specifies the explicit destination U2. Because this is the specification defined at Node 2 (the destination node), JES2 uses it.
- 5 The resolved destination is U2 (special local routing 2) at Node 2.

Routing output to devices at another node

There are two ways to route output to devices at another node:

1. By defining both the target node and the specific destination on that node. ('N2.R5' signifies remote 5 at node 2.)

   Remember that if you define a destination node that is different from the node where the output was created, JES2 processing will interpret destinations (for both jobs and SYSOUT) at both the creation and destination nodes.
2. By using a DESTID defined on that node.

   'BOST1.R2' signifies a remote workstation 'R2' at node BOST1 Node 2 for the example in Figure 34 on page 150.

   An installation could change BOST1.R2's routing on the DEST= parameter to a different route code without affecting other nodes and user applications in the network.

To illustrate this process, see the initialization parameters defined in Figure 34 on page 150.

```
   WASH1                                BOST1

 NODE(1) NAME=WASH1                   NODE(1) NAME=WASH1
 NODE(2) NAME=BOST1                   NODE(2) NAME=BOST1
 DESTID(FIRST) DEST=BOST1.R1
 DESTID(SECOND) DEST=BOST1.SECOND     DESTID(SECOND) DEST=BOST1.R2
 DESTID(THIRD)  DEST=BOST1.THIRD      DESTID(THIRD)  DEST=DET1.THIRD

   DET1

 NODE(1) NAME=WASH1
 NODE(2) NAME=BOST1
 NODE(3) NAME=DET1
 DESTID(SECOND) DEST=BOST1.R2
 DESTID(THIRD)  DEST=DET1.R1
```

Figure 34. Sample JES2 DESTID(jxxxxxxx) Initialization Values on Three Nodes

Output at WASH1 routed to destination FIRST is sent to Remote 1 at BOST1. The destination is resolved at BOST1. Because a BOST1 remote number is defined in the initialization statement shown for WASH1, the destid FIRST need not be defined at BOST1.

Output at WASH1 routed to destination SECOND is sent to BOST1 and the output sent to remote 2. The destination SECOND must be defined on both nodes. WASH1 does not need to know which remote on BOST1 represents SECOND. If SECOND was not defined at BOST1, JES2 would send output to userid SECOND. If there were no such user on the system, the output would remain on the userid queue at BOST1 until an operator changed its destination or the data set was purged.

Output at WASH1 routed to destination THIRD is sent to BOST1, where it is rerouted to THIRD at DET1. Finally, DET1 receives the output and routes the job to remote workstation 1 at DET1. If no destid had been defined as DESTID(THIRD), JES2 would have routed the data to userid 'THIRD' at DET1.

Avoiding endless looping in a network

When planning a network, ensure that all NODE(nnnn) and DESTID(jxxxxxxx) initialization statements have been defined to prevent endless looping throughout the network. For example, in Figure 35, the output destined for node 2 is routed back and forth in a loop because node 1 assumes that DEST is userid NY6 at node 2 and node 2 assumes that DEST is userid NY5 at node 1.

All nodes in a network need to define JES2 destids using the same conventions so that each system programmer knows what devices, remote workstations, and userids exist at each node.

```
   NODE 1                          NODE 2
   ------                          ------
DESTID(NY5)  DEST=N2.USER       DESTID(NY6) DEST=N1.USER
```

Figure 35. JES2 DESTID(jxxxxxxx) Initialization Values that Loop

Specifying TSO/E userIDs on a JES2 system

In order for a userid to receive mail or notification messages from a JES2 subsystem, **never** define userids of the following forms as TSO/E userIDs:

- ANYLOCAL|LOCAL
- Any destid name defined on a DESTID(jxxxxxxx) initialization statement
- Any destid name added through a $ADD DESTID(jxxxxxxx) operator command.

- Any name specified on the NAME= parameter of a NODE(nnnn) initialization statement.
- Any name specified on the NAME= parameter of a $T NODE(nnnn) operator command.

IBM suggests that installations avoid confusion when routing SYSOUT from one destination to another by not specifying the following forms as userids on a JES2 system:

- Nnnnn
- Rmmmm
- RMmmmm
- RMTmmmm
- NnnnnRmmmm
- Unnnn

An installation using these userids might experience unexpected results. For example, output intended for userid R001 is instead routed to remote 1, even if that remote has not been defined on the system.

Installations that find it necessary to define TSO/E userids beginning with the prefixes 'N', 'R', 'RM', 'RMT' and 'U' can do so by specifying USER on the following parameters of the DESTDEF initialization statement:

- Ndest=
- Rdest=
- RMdest=
- RMTdest=
- Udest=

Installations can also control the number (1-32767) of nodes in a network through the NODENUM= parameter on the NJEDEF statement. For example, if NODENUM=7, JES2 would resolve the route code N17 as a userid at its destination.

Additionally, the LOCALNUM= parameter of the DESTDEF initialization statement definesthe maximum number that JES2 identifies as a special local route code. In an installation that defines this parameter, JES2 presumes that any number exceeding the LOCALNUM= value is a userid. For example if LOCALNUM=29, JES2 would resolve the route code U321 as a userid at its destination.

The prefixes R, RM, and RMT can precede a number in the range from 0 to 32767 JES2 provides no means of controlling this range.

Altering destination processing through DESTDEF

JES2 destination processing recognizes certain destination prefixes as having a special meaning. This can cause problems if an installation is using those special prefixes as TSO/E userids: notify messages and netmail files might not reach the intended users. Installations can use options on the DESTDEF initialization statement to prevent these problems. These options alter which prefixes JES2 destination processing recognizes and which it does not.

These are the options on the DESTDEF statement that an installation uses to control which prefixes JES2 recognizes:

Ndest Nodal routing Nnnnn and NnnnnRmmmm

Rdest Remote routing Rmmmm and NnnnnRmmmm

RMdest
Remote routing RMmmmm

RMTdest
Remote routing RMTmmmm

Udest Special local routing Unnnn

When any of these options are set to 'USER', JES2 no longer recognizes any special meaning for the corresponding destinations. This applies to JCL, JES2 control statements (JECL), commands, and dynamic allocation processing. For initialization statement processing, JES2 ignores the meaning of these prefixes for all destination identifiers that appear **after** the DESTDEF statement in the initialization stream unless performing a hot start. If this is a hot start, the values from when JES2 was last active apply to the entire initialization deck.

To avoid different interpretations of destinations on a hot start, the DESTDEF statement should appear early in the initialization stream and before any references to destinations.

The DEST= parameter on the DESTID(jxxxxxxx) initialization statement and the $T DESTID and $ADD DESTID operator commands are the only places where DESTDEF options never apply. This allows for the creation of destids that route output to a remote even when DESTDEF RDEST=USER.

To illustrate how the DESTDEF initialization statements can be used to enforce particular policies at an installation, consider the following two examples, in which both RDEST=User and SHOWUSER=WITHlocal.

The $T O command routes all the SYSOUT from job 4 to userid 'R0007'. This installation can require all JCL and JES2 control statements submitting jobs to specify all userids with the single-letter prefix 'R', while still using 'RM' (RMDEST=User) and 'RMT' (RMTDEST=User) to specify remote workstations.

In response to the operator command:

```
$TOJ4,ALL,D=R0007
```

JES2 displays:

```
$HASP686 OUTPUT(IRRDPTAB)  OUTGRP=2.1.1,BURST=NO,FCB=****,
$HASP686                   FLASH=****,FORMS=STD,HOLD=(NONE),
$HASP686                   HOLDRC=,OUTDISP=HOLD,PAGES=,
$HASP686                   PRIORITY=144,PRMODE=LINE,QUEUE=A,
$HASP686                   RECORDS=(4 OF 4),ROUTECDE=LOCAL.R0007,
$HASP686                   SECLABEL=,TSOAVAIL=YES,UCS=****,
$HASP686                   USERID=++++++++,WRITER=
```

In this example, the route code 'LOCAL.R0007' indicates a userid of R0007 at the local node.

This installation can alter the meaning of 'R0007' by dynamically adding a destid through the following command from the operator console:

```
$ADD DESTID(R0007),DEST=R7
```

JES2 displays:

```
$HASP822 DESTID(R0007)    DEST=R7,STATUS=DESTID,PRIMARY=NO
```

Now, if an operator enters the command:

```
$TOJ5,ALL,D=R0007
```

JES2 displays:

```
$HASP686 OUTPUT(IRRDPTAB)  OUTGRP=2.1.1,BURST=NO,FCB=****,
$HASP686                   FLASH=****,FORMS=STD,HOLD=(NONE),
$HASP686                   HOLDRC=,OUTDISP=HOLD,PAGES=,
$HASP686                   PRIORITY=144,PRMODE=LINE,QUEUE=A,
$HASP686                   RECORDS=(4 OF 4),ROUTECDE=R0007,
$HASP686                   SECLABEL=,TSOAVAIL=YES,UCS=****,
$HASP686                   USERID=++++++++,WRITER=
```

In this example, the route code 'R0007' indicates the destid R0007 rather than a user destination.

Preparing installations to alter destination processing

Before changing how JES2 interprets destinations, installations should review their JCL and automated commands to ensure that they are notusing the prefixes being affected. For example, before changing Ndest to 'USER', ensure that there are no JCL references to N123, N0123, N000123.

Destids can be used to support cases where it is not practical to change JCL statements. However, the subscript on the DESTID(jxxxxxxx) initialization statement must match the DEST= parameter specification on the JCL statement exactly. For example, an installation with the following initialization statement can route data to node 1 using OUTPUT JCL statements that specify DEST=N1.

```
DESTID(N0000001)
```

To route the data to Node 2, you must specify DESTDEF Ndest=User on a warm start, add the following DESTID,

```
DESTID(N1) DEST=N2
```

and add DEST=N2 to the DESTID(N0000001) initialization statement. If you change any DESTDEF initialization statement parameters to specify User, the processing of explicit destids can be altered. Use the DESTDEF initialization statement to alter destination processing only after carefully preparing the initialization stream definitions and JCL statements carefully. For more information on how to prepare an installation for such a change, see both "Specifying TSO/E userIDs on a JES2 system" on page 150 and "Altering destination processing through DESTDEF" on page 151.

An installation can specify whether JES2 displays a destid or an explicit route code for jobs and SYSOUT through the PRIMARY= parameter on both the DESTID(jxxxxxxx) initialization statement and the $ADD DESTID operator command. For example, by specifying 'DESTID(NYWK),DEST=U2,PRIMARY=YES', an installation ensures that the destid 'NYWK' is displayed instead of the explicit special local route code of 'U2'.

Use of PRIMARY=YES can aid both operators and system programmers in tracking jobs and SYSOUT across a network, as illustrated in Figure 36 on page 154. An installation specifies that destids beginning with the 'RMT' prefix should be processed as userids (RMTdest=USER), then creates the following destids in its initialization stream.

```
DESTDEF      RMTDEST=User,SHOWUSER=WITHlocal
DESTID(RMT6) DEST=R6,PRIMARY=YES
DESTID(DR1)  DEST=R1,PRIMARY=NO
DESTID(DR2A) DEST=R2,PRIMARY=NO
DESTID(DR2B) DEST=R2,PRIMARY=NO
DESTID(DR3A) DEST=R3,PRIMARY=YES
DESTID(DR3B) DEST=R3,PRIMARY=NO
DESTID(DR4A) DEST=R4,PRIMARY=YES
DESTID(DR4B) DEST=R4,PRIMARY=YES
DESTID(DR5)  DEST=N1.R2,PRIMARY=NO
        :
        :
NJEDEF       OWNNODE=1
```

Figure 36. DESTID(jxxxxxxx) and DESTDEF Initialization Stream Definitions

In Table 45, you can see how these destinations are resolved.

Table 45. How JES2 resolves destinations from DESTDEF initialization statements

Destid Routing	Console Display	Reason
DR1	DR1	Displays the destid because no other destids specify this routing.
DR2A	R2	Displays explicit destination because neither destid with this routing specified PRIMARY=YES.
DR3A	DR3A	Displays the destid because PRIMARY=YES.
DR3B	DR3A	Displays alternate destid because DR3A shows PRIMARY=YES.
DR4A	DR4A	Displays the destid because PRIMARY=YES.
DR4B	DR4A	Displays alternate destid because both show PRIMARY=YES. and DR4A is first alphabetically. in the initialization stream.
DR5	R2	Displays explicit destination because PRIMARY=NO.
R5	R5	Displays explicit remote workstation routing.
RM5	R5	Displays explicit remote workstation routing.
RMT5	LOCAL.RMT5	Displays user destination because RMTdest=User.
RMT6	RMT6	Displays destid because PRIMARY=YES. and RMTdest=User.

The PRIMARY= parameter on the DESTID(jxxxxxxx) initialization statement allows an installation to specify how JES2 resolves multiple destid displays. As Table 45 shows, specifying PRIMARY=YES allows an installation to specify which destid appears in displays when two statements specify the same DEST= routing. For example, destid 'DR3B' will never display on the console because destid 'DR3A' is routed to remote workstation 3 too, and it specifies PRIMARY=YES.

Because the PRIMARY= parameter defaults to NO, you must remember to specify which destid takes precedence when defining two or more with the same DEST= route code.

By specifying SHOWUSER=WITHlocal on the DESTDEF initialization statement, this installation provides a level of clarification in the RMT5 destid routing that displays the first-level identifier 'LOCAL' in front of the userid 'RMT5.'

Routing to multiple destids

If there are multiple destids that can be used for a particular routing, one of the destids should be defined as PRIMARY=YES. Failure to do so causes output destined to that routing to display in explicit format, such as Nn, Rm, NnRm, or Un. SDSF displays the routing in this explicit format and attempts to use this format in commands. This explicit routing does not work as expected if one of the DESTDEF initialization statement options is set to 'User'.

For example, in Figure 37, all three destids would match a piece of output routed to special local 5 at the local node (node 1). If you specify the $DO J command to list the job output for these destids, the console displays a routing of D=U5. However, with Udest=User (and no other destid), using a destination of 'U5' would not result in the same routing as when using a destination of 'DAVE'. If installations define either DESTID(DAVE) or DESTID(MIKE) as PRIMARY=YES, output routed to special local 5 displays with a destination that can be used to route other output to the same place.

```
DESTID(DAVE) DEST=N1.U5
DESTID(MIKE) DEST=N1.U5
DESTID(TOM)  DEST=U5
```

Figure 37. Multiple Destids that Match Output Routed to Special Local 5

For an example of how installations can route jobs and SYSOUT data sets through remote workstation devices, see "Routing to RJE devices when Rdest=User."

Routing to RJE devices when Rdest=User

An installation with RJE devices that needs to set Rdest=User must ensure that SYSOUT can still be routed to the RJE device. By default, RJE devices are set up so that SYSOUT from jobs submitted from the RJE device is routed back to the RJE workstation. But if a job submitted on remote 5 is routed to another node to execute, when the SYSOUT returns, it will be routed to destination 'R5'. With Rdest=User, 'R5' will not be interpreted as remote 5 but rather as user routing 'R5'. The remote printer will be unable to print the SYSOUT. To prevent this problem, create a DESTID(Rm) DEST=NxRm where 'x' is the local node number (NJEDEF OWNNODE=) and 'm' is the remote number.

When an RJE device signs on, JES2 attempts to interpret the workstation name sent from the remote while honoring the value specified on the DESTDEF initialization statement. First, JES2 searches the destids for a match to determine the actual remote number. If there is no match, JES2 ignores the values set on DESTDEF (assuming that Ndest=, R|RM|RMTdest=, and Udest= have all been set to USER) while interpreting the workstation name. This would allow a remote to sign on using RMT5, even if RMTdest=USER and there is no DESTID(jxxxxxxx) initialization statement with a subscript of RMT5. If a remote tries to sign on as RMT5 and there is a DESTID(jxxxxxxx) initialization statement with a subscript of 'RMT5' with a DEST= that does not specify a remote on the local node, then the sign on attempt will fail.

Reducing printing costs through remote pooling: Remote pooling allows the printers on multiple remote workstations to print the output of any remote in the pool. For more information about pooling remote workstations, see "Pooling RJE workstations" on page 345. JES2 changes any destination that specifies a specific remote workstation in the pool to the routing of the pool. This change occurs when JES2 interprets a destination starting with 'R', 'RM', or 'RMT'. If Rdest, RMdest, or RMTdest is set to 'USER', then remote pooling does not occur (even if there is a DESTID(R5)).

Even so, much of the function of remote pooling can be achieved by using destids. For example, in Figure 38, remotes 5, 6, and 9 are pooled. The destids will ensure that SYSOUT routed to R5, R6, or R9 will receive a routing of remote 5.

```
DESTDEF RUSER=USER

RMT(5) ROUTECDE=5                /* Define the remotes as pooled   */
RMT(6) ROUTECDE=5
RMT(9) ROUTECDE=5

DESTID(R5) DEST=N1R5,PRIMARY=YES  /* Define the DESTIDs as routing  */
DESTID(R6) DEST=N1R5              /* to the pooled remote           */
DESTID(R9) DEST=N1R5
```

Figure 38. Remote Pooling with RDEST=USER

Note: In this example, if a remote tries to sign on with a workstation name of 'R9', that workstation will actually sign on to remote 5. To avoid this problem, sign on with a workstation name of 'REMOTE9' or RMT9 (if RMTdest=REMOTE or there are no destids for RMT9).

External writers

An external writer can select output from JES2 after JES2 builds the output group from the data sets. Your installation can supply its own writer routines or use the external writer processor supplied by IBM by naming the writer as a subparameter of the SYSOUT= DD keyword. The operator then starts an external writer in a private address space, and the external writer writes the data using QSAM.

For details on the external writer, see The External Writer.

Held job and data set considerations

You can put a data set into a **hold** condition by explicitly defining it as held or assigning the output to a held SYSOUT class. The operator releases the job using the output ($O) command or the $T OJ,NDISP= command; a TSO/E user uses the TSO/E OUTPUT command on a time-sharing terminal.

You can transmit held data sets to an offload data set for future retrieval. If you transmit jobs to other nodes for execution, you must consider when and how the hold status takes effect. JES2 treats jobs placed in hold by the operator differently than jobs placed in hold through JCL or TSO/E commands. The following sections discuss, in further detail, how to define, modify, dispose, and offload held data sets.

Defining held data sets

- JES2 **holds** SYSOUT data sets if:
  - The JCL SYSOUT DD statement contains HOLD=YES regardless of the OUTDisp specification on the OUTCLASS(v) initialization statement.
  - The JCL OUTPUT statement contains OUTDISP=(HOLD,HOLD) or OUTDISP=(KEEP,KEEP), regardless of the OUTDisp specification on the OUTCLASS(v) initialization statement.
  - The SYSOUT parameter on the JCL DD statement specifies a class defined as held on the OUTCLASS(v) initialization statement with the OUTDisp parameter.
- JES2 **does not hold** SYSOUT data sets if:
  - The JCL SYSOUT DD statement contains HOLD=NO, regardless of the OUTDisp specification on the OUTCLASS(v) initialization statement.

- The JCL OUTPUT statement contains OUTDISP=(WRITE,WRITE), regardless of the OUTDisp specification on the OUTCLASS(v) initialization statement.
- The SYSOUT parameter on the JCL DD statement specifies a class defined as non-held on the OUTCLASS(v) initialization statement with the OUTDisp parameter.

Held output processing

When output is held, JES2 builds a JOE for each job as it does for non-held output. JES2 then retrieves such held output in LIFO order, **not** FIFO order as it does for non-held output.

Modifying and disposing held data sets

A data set remains in held status until either the TSO/E user releases it on a time-sharing terminal with the TSO/E OUTPUT command or from the SDSF panel, or until the operator modifies the disposition with the $O (output) command. The operator can either release or cancel held data sets based on time (that is, n days or n hours old) with the $O command or the $T OJ command. In this manner, you can easily dispose of held output unnecessarily using spool space and job queue elements (JQEs) for too long a time. If you free up these JQEs regularly, you reduce your installation's JQE needs. As your TSO environment grows in size, it is important to prevent this “wasting” of spool space with unneeded or forgotten held output data sets.

The operator can determine spool usage by issuing the $D SPL command. This command can also display job utilization of specific spool volumes. (see *z/OS JES2 Commands* for more information on this command.) If the $D SPL response indicates a particularly high spool usage by a particular job, the operator can take action (cancel, offload, or print) to eliminate that job. Here again, the age parameters on the $O command are useful.

Type 26 SMF records provide further job spool utilization information. These records include the number of track groups and spool buffers used. If you and the operator monitor spool usage and eliminate unnecessary held data set spool usage, you should be able to more effectively tune the track group size parameter (TGSIZE= on the SPOOLDEF statement), thereby using available spool space more efficiently.

Offloading held jobs and data sets

An installation can offload held jobs. This is a useful tool you can use to free up spool space. Offload jobs by specifying HOLD=YES and WS=(HOLD/) on the job transmitter (OFF(n).JT). Note that you can also transmit held jobs to the offload data set if you do not add HOLD to the work selection list. JES2 does not change the hold status when you receive the offloaded job. Of course, the operator can purge (or otherwise alter its disposition) it if held for longer than an installation determines to be an appropriate length of time. Or, the operator can release the job with the $A J operator command.

An installation can also offload held data sets. As with held jobs, JES2 remembers the held attribute across the offload operation. However, for data sets held at the local node, only the job-level work selection criteria on for the OFF(n).ST statement apply. You must explicitly request transmission of held data sets. If you add OUTDISP=(HOLD,LEAVE) to the OFF(n).ST statement, the offload device can then select held data sets. A request of this type means JES2 will transmit all held data sets for a particular job.

JES2 treats data sets held at the destination node (the target system of the offload operation) as normal SYSOUT subject to the work selection criteria for a SYSOUT transmitter.

Held data sets in the NJE environment

If a job generates a SYSOUT data set and specifies "HOLD=YES" on the DD statement for the data set, the "hold" takes effect on the system which receives the data set. If the job's execution node and the data set's destination node are the same, JES2 performs the hold as soon as the job deallocates the data set. If the job's execution node differs from the job's destination node, JES2 removes the local hold on the data set. This permits JES2 to group the data set into an output group; JES2 then places the output group on the network queue for output groups. The SYSOUT transmitter then selects the data set for transmission.

When JES2 transmits a held data set (possibly at the same time it transmits the job's other data sets), the data set header will reflect the hold status of the data set. When a node receives a SYSOUT data set marked as held, the SYSOUT receiver checks the data set's destination against its own node name. If the data set destination is the receiver's node, the receiver performs normal held data set processing. A TSO/E user can then inspect the data set. When an operator or TSO/E user releases the held data set, the system no longer holds the data set even though the operator or TSO/E user changed its destination before printing the data set. At each point along the way, you can use the JES2 $R command to change the data set's destination.

Improving JES2 processing

If JES2 is slow to process work or respond to operator commands, see the description of the following initialization statements (in *z/OS JES2 Initialization and Tuning Reference*) and the other noted sections of this manual for tuning recommendations:

- BERTNUM= parameter on the CKPTSPACE statement
- DEBUG statement
- JOBNUM= parameter on the JOBDEF statement
- JOENUM= parameter on the OUTDEF statement
- MEMBER(n) initialization statement
- "The JES2 health monitor" on page 395
- "Accessing the CKPTn data set in a MAS" on page 216

In a multi-access spool environment, also refer to the descriptions of the queue control parameters, HOLD= and DORMANCY= on the MASDEF statement and "MASDEF queuing parameters" on page 322 for recommended initial parameter value and tuning information.

Chapter 3. SPOOL volume configuration, control, and performance

The spool is the repository for all input jobs and all system output (SYSOUT) that JES2 manages. SYS1.HASPACE (the data set used for storing JES2–maintained data and control blocks) is a JES2-required data set. **Note that the name, SYS1.HASPACE, is the JES2 default name for the required spool data set. If you prefer, change this name by specifying the DSNAME= parameter on the SPOOLDEF initialization statement. However, for convenience of this discussion, we will use the default name here.** The specification, management, and placement of this data set affects JES2 performance. This chapter deals with the management of spool data sets, including their specification and important performance considerations.

SPOOL configuration

JES2 requires that the SYS1.HASPACE data set be present on direct access volumes. In a multi-access spool configuration, these volumes must be accessible from all member systems.

The following section discusses this data set, the dynamic addition and deletion of JES2 spool data sets, spool space for individual jobs, and the allocation of tracks and track cells on a volume.

Chapter 7, “Providing security for JES2,” on page 349 discusses how to protect spool data sets with RACF.

The SYS1.HASPACE data set

Before JES2 can be initialized, space on a direct access device must be allocated for SYS1.HASPACE. SYS1.HASPACE is the name of one or more data sets used for storing JES2-maintained data and control blocks. These volumes store all job input, job output, JES2 control blocks, and messages queued between shared spool members. The requirements for specifying these data sets follow.

Naming conventions

Up to 253 volumes can be designated as spool volumes. Spool volumes are identified to JES2 by a volume serial number, the first four or five characters of which are specified by the VOLUME= parameter on the SPOOLDEF statement during JES2 initialization. The first four or five characters specifying the volume serial number of each volume must be identical to the characters specified by the VOLUME= parameter. The fifth and sixth characters can be assigned to designate individual spool volumes and can be any characters that are valid in a volume serial number.

Any volume for which the first four to five characters of the volume serial are a value matching the VOLUME= parameter value on the SPOOLDEF statement is assumed to be a spool volume and should have an allocated SYS1.HASPACE data set. If however, a SYS1.HASPACE data set is not on the volume, JES2 issues message $HASP414 and the volume is not used as a spool volume. The data set name must be SYS1.HASPACE (the default) unless the DSNAME= parameter on the SPOOLDEF statement specifies a different data set name. (see the description

of SPOOLDEF in *z/OS JES2 Initialization and Tuning Reference* for further information on this statement, if required.)

Placement

Spool volumes can reside on any combination of direct-access devices which are supported by JES2 and your operating system. That is, you can have a mix of DASD types and models such as a 3390-2, a 3390-3, and several 3380s of various models. See Appendix A, "IBM devices supported by JES2 and how to use them," on page 387 for these device types. Be certain to consider efficient values for the SPOOLDEF initialization statement parameters BUFSIZE=, TGSIZE=, and TGSPACE=(MAX=) for each specific device type. Optimal values for DASD types and models are presented in Table 80 on page 388 and Table 82 on page 389. JES2 utilizes space from each spool volume, ensuring balanced use of all allocated space. All members of a multi-access spool configuration must have at least one channel path to the devices containing all spool and checkpoint volumes.

IBM suggests that you define the UCBs for spool volumes less than 16 megabytes (Mb) in common virtual storage. If you define the UCBs for spool volumes greater than 16 Mb in common virtual storage, JES2 makes a copy of the UCB less than 16 Mb in common virtual storage. Because there is no savings to storage, there is no advantage to defining UCBs for spool volumes greater than 16 Mb in common virtual storage. For more information on defining UCBs, see *z/OS HCD Planning*.

Through spool-partitioning installation exit routines (which you provide), you can control the specific volumes to which a job is allocated, thus limiting the number of jobs affected by the failure of a single spool volume. Also, use the FENCE parameter on SPOOLDEF initialization statement to limit each job to an installation–defined number of SPOOL volumes. See "SPOOL partitioning" on page 165 for a discussion of how the SPOOLDEF parameters and Exit 11 and Exit 12 interact to provide your installation with its required spool partitioning needs.

IBM suggests that each spool volume be entirely devoted to JES2. To allocate other frequently-referenced data sets on a spool volume would degrade the efficiency of JES2's direct-access allocation algorithm.

Note: All JES2 spool data sets are position-dependent when they are allocated. To prevent a spool data set from being moved by a utility program, specify each spool data set as unmovable. Do this by specifying DSORG=PSU on the DCB parameter of each DD statement used to define a spool data set.

Space allocation

Allocate spool data sets as single-extent data sets; if you allocate additional extents, JES2 uses only the first extent for spool space. Each spool volume must contain a data set named SYS1.HASPACE (unless the DSNAME= parameter on the SPOOLDEF statement specifies another name) in order to be used as a spool volume.

Note: JES2 supports using a data set more than 64K tracks in size based on the SPOOLDEF LARGEDS= parameter. If LARGEDS=FAIL is specified, the SYS1.HASPACE data set has to be less than 64K tracks in size. Specify LARGEDS=FAIL to allow pre-z/OS 1.7 levels of JES2 to co-exist in an MAS. If you no longer have any pre-z/OS 1.7 members in your MAS, you can specify LARGEDS=ALLOWED to create SYS1.HASPACE data set up to 1,048,575 tracks.

The unit of direct-access space allocation for JES2 is the track group. Obtain the number of tracks in a track group is obtained by the following formula:

$$\text{Number of track / track group} = \frac{\text{TGSIZE on SPOOLDEF stmt}}{\text{buffers per track}}$$

where: *TGSIZE=number* of buffers in a track group, and buffers per track is detailed in Table 80 on page 388based on BUFSIZE and device type.

You can allocate spool space by using any valid space specification, but keep the following considerations in mind:

- To minimize unused DASD space, you should allocate spool space contiguously, because JES2 only uses the first extent of the spool data set.
- The spool allocation *must* be equal to or greater than the number of tracks in a track group.
- TRK (track) allocations waste less DASD space than CYL (cylinder) allocations, because you can specify the allocation as an integral multiple of a track group.

For further information concerning the JCL SPACE parameter and its syntax, see *z/OS MVS JCL Reference*.

An example of defining spool data sets appears in Figure 39.

For multi-access spool configurations, all spool and checkpoint volumes must reside on devices that have at least one channel path to each JES2 system in the multi-access spool environment. If the multi-access spool configuration is a member of an NJE network, NJE commands and messages sent among JES2 systems attached to the shared spool are queued on the spool volume. To calculate the number of track groups normally needed in SYS1.HASPACE for NJE messages, use the following formula:

$$\text{Number of track groups} = \frac{n^2 + n^3}{2}$$

where n is the number of JES2 subsystems attached to the shared spool.

```
//ALLOCATE JOB  (...),'PREPARE FOR JES2',MSGLEVEL=1
//ALLOCAT  EXEC PGM=IEFBR14
//*
//SPOOL1   DD   DSN=SYS1.HASPACE,UNIT=3390,
//              VOLUME=SER=SPOOL1,DISP=(NEW,KEEP),
//              SPACE=(CYL,884,,CONTIG),DCB=(DSORG=PSU)
//*
//SPOOL2   DD   DSN=SYS1.HASPACE,UNIT=3390,
//              VOLUME=SER=SPOOL2,DISP=(NEW,KEEP),
//              SPACE=(TRK,13256,,CONTIG),DCB=(DSORG=PSU)
/*
```

Figure 39. Example of defining and allocating the JES2 SYS1.HASPACE data set

This job assumes a record size (SPOOLDEF BUFSIZE=) of 3992. For other buffer sizes, replace the two occurrence of 3992 with your buffer size value.

```
//ALLOCSPL JOB  (...),'SPOOL FORMAT',MSGLEVEL=1
//FORMAT   EXEC PGM=IEBDG
//*
//SPOOL    DD   DSN=SYS1.HASPACE,UNIT=3390,
//              VOL=SER=SPOOL2,DISP=(NEW,KEEP),
//              SPACE=(CYL,884,CONTIG),
//              DCB=(DSORG=PSU,RECFM=U,BLKSIZE=3992)
//*
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
               DSD OUTPUT=(SPOOL)
               FD NAME=SPOOL,FILL=X'FF',LENGTH=3992
               CREATE NAME=(SPOOL),QUANTITY=99999999
               END
/*
```

Figure 40. Example of allocating and formatting a SPOOL volume before starting it

Defining SPOOL space

Effective use of the JES2-managed spool data sets and subsequent performance of JES2 is based on correct control of the JES2 spool volumes. Spool partitioning, use of track celling, and use of marking bad tracks unavailable are effective spool management techniques.

SPOOL allocation

During any of the stages of processing, a job can allocate spool space. The logical record length (LRECL) of the spool allocation is dependent on the device that performs the allocation. Table 46shows the maximum LRECL for various devices.

Table 46. Maximum spool LRECLs

Source	Type	Length
BSC/RJE	Input	80
	Punch*	80
	Print**	32K
SNA/RJE	Input	254
	Output***	32K
Local	Input****	80
	Output	32K
INTRDR	Input	32K
NJE	Input	32K
	Output	32K
External Writer	Output	32K
TSO/E submits	Input	80

* LRECL is 81 with ASA or machine control.

** Truncated to PRWIDTH.

*** This is LRECL written to spool. Maximum transmitted LRECL is 255.

**** Truncated based on device type.

Controlling SPOOL space

Use the SPOOLDEF initialization statement to define your system-wide spooling environment. All parameters on this statement can be displayed by using the $D SPOOLDEF operator command, and several can also be modified by the $T SPOOLDEF command. However, if modified, JES2 does not honor the changed parameter values across a JES2 restart, with the exception of the FENCE= parameter specification. (See *z/OS JES2 Commands* for a description of these commands.)

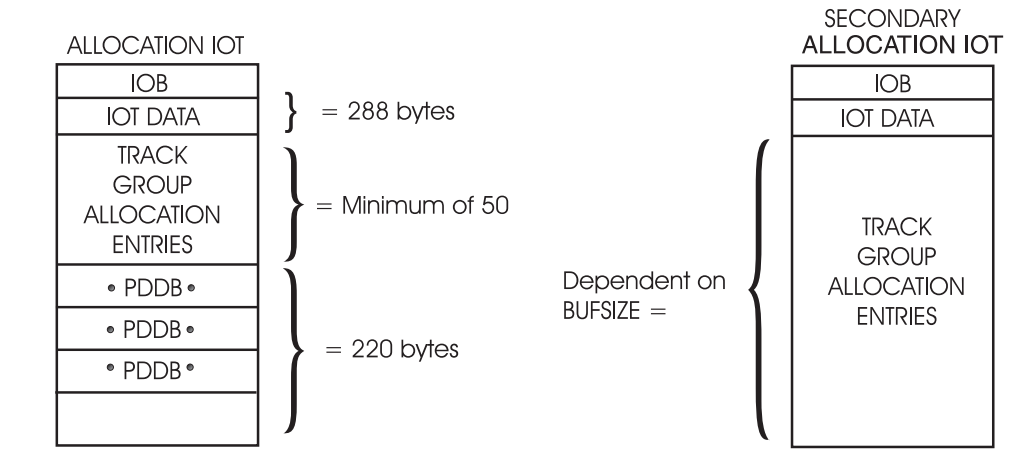
The SPOOLNUM= parameter on the SPOOLDEF statement specifies the number (1 - 253) of JES2 spool volumes available to your complex. Be aware that as you do increase this value above 32, each additional 32 volumes uses 8 times the JOBNUM= specification additional bytes in the JES2 address space. (JOBNUM= is a parameter on the JOBDEF statement.)

Spool volumes are allocated in multiples of 32 only. If you specify other than a multiple of 32, the value is rounded up to the next multiple of 32 not to exceed the maximum number of spool volumes (253). JES2 will automatically decrease SPOOLNUM to 253 if the value specified is 254, 255, or 256. All specified values greater than 256 use the default for SPOOLNUM, which is currently 32. Also, if you specify other than a multiple of 32, JES2 returns the rounded spool number value in response to the $DSPOOLDEF operator command. For example, if you previously specified SPOOLNUM=37, JES2 allocates 64 spool volumes and displays 64 in response to the display command.

To specify the size of the track group map (TGM) use the TGSPACE=(MAX=) parameter on the SPOOLDEF statement. JES2 automatically compresses the TGM whenever a spool volume is deleted. This prevents various-length entries from fragmenting TGM space and preventing the use of all available track groups. To keep the operator informed of the current status of track groups on a particular volume, JES2 issues messages $HASP850 and $HASP851 when spool volumes are either added or deleted. Use this information to monitor spool volume usage.

Defining the allocation IOT

The track group allocation entries (TGAEs) and peripheral data definition blocks (PDDBs) are contained in the **allocation IOT** (input/output table). The allocation IOT must fit on a single (4K) page of storage. See Figure 41 on page 164 for a representation of the allocation IOT while reading the following discussion.



For an Allocation IOT:
  Minimum Fixed IOT Size = 288 bytes
  Space for PDDB's = BUFSIZE - 288 bytes - (50 X TGAE size)
  Number of TGAE's = (reminder from previous divide/TGAE size) + 50

For a secondary Allocation IOT
  No PDDB's are allocated
  Fixed IOT Size = 288 bytes
  Number of Trace Group Allocation Entries = (BUFSIZE - 288) /3

Figure 41. Allocation IOT and Minimum Storage Requirements

Track group allocation entries (TGAE) identify the spool data set on which the track groups reside and the specific track groups within the spool data set. Each TGAE represents a single MTT (the spool allocation unit) or MQT. The allocation IOT contains a list of all TGAEs allocated to this job and data set. JES2 determines and reserves the amount of space within the allocation IOT for the TGAEs. Space for 50 TGAEs is the minimum reserved. Each TGAE is three or five bytes of information: the first byte contains the extent upon which the track group resides, and the other two or four bytes identify the relative track number within the specified extent.

Peripheral data definition blocks (PDDBs) also reside in the allocation IOT. These are required to describe the characteristics of the output data within your system. The number of PDDBs that the allocation IOT can hold is determined by the buffer size you have previously specified with the BUFSIZE parameter on the SPOOLDEF statement. See Appendix A, “IBM devices supported by JES2 and how to use them,” on page 387 for information on and recommended values for specific device types to maximize storage and device utilization.

If a job uses all the TGAEs available within the allocation IOT, JES2 obtains a second spool record, the **secondary allocation IOT**. Except for the IOT data area, the entire secondary allocation IOT contains TGAEs; additional tables are constructed if required. As many secondary allocation IOTs are built as required. These are chained in a push down stack which allows the most recently built IOT to be the most readily accessible (it likely resides in central storage whereas any older IOTs have likely been written to spool). In this manner, JES2 minimizes spool I/O and maximizes performance.

For spin data sets, JES2 reserves the maximum space possible for TGAEs. Only one PDDB is required for spin data sets; this minimizes the possibility of needing a secondary allocation IOT to contain the TGAEs.

Defining the track group maps

Two track group maps are defined to JES2: the master track group map and the "badtrack" map defining those tracks specified as bad by JES2. Use the TGSPACE=(MAX=) parameter on the SPOOLDEF initialization statement to define the number of track groups in your JES2 complex. The maximum value is 16581184; however, the realistic limit is determined by your virtual storage limitations. The rounding factor is 16288.

Note: If the number of track groups you have allocated on each spool volume is not divisible by 8, there will be a discrepancy between the maximum allocatable track groups and the sum of the allocated track groups plus the number of track groups available.

Use the $TSPOOLDEF command to modify the TGSPACE=(MAX=) parameter; therefore, you can increase the number of track groups as current system usage requires. In this manner, the operator can dynamically add spool volumes with the $S SPOOL command to a maximum of 253 spool volumes. Decreasing this value requires a JES2 cold-start. (See "Dynamic addition and deletion of SPOOL volumes" on page 167 and *z/OS JES2 Commands* for further information on this procedure.)

SPOOL partitioning

Spool partitioning is a facility provided within JES2 that permits the explicit identification of spool volumes from which a particular job or job class can allocate track groups. This facility is also referred to as spool fencing.

Spool partitioning can be a useful tool to your installation, especially if you anticipate future changes to your spool configuration. Standard JES2 processing allows all jobs to allocate track groups on all available spool volumes; therefore, the deletion of a single volume can involve many jobs. Isolation of long-running jobs, frequently-referenced system jobs, or jobs frequently put into HOLD status for long periods of time to one spool volume (or a small number of spool volumes as specified on SPOOLDEF FENCE=) can greatly benefit future spool volume reconfiguration. Also, if only particular volumes are allowed to allocate track groups to these types of jobs, the status of other volumes can be changed more easily, as required.

The use of spool partitioning is also a tool that is useful for increasing JES2 performance. Isolating frequently-run system jobs on separate volumes increases JES2 performance by permitting access to other volumes containing user-defined work. The manner in which you define spool partitioning to JES2 follows. The SPOOLDEF initialization statements and two installation exits, Exit 11 and Exit 12, provide methods for accessing and setting the **spool partitioning mask work area**. This allows the installation to assign masks based on, for example, JOBCLASSes or JOBNAMEs. You need to determine installation policy to compromise between performance and limiting the impact of a failure. For more information on installation exits, see *z/OS JES2 Installation Exits*.

JES2 processes your fencing requirements based on the number of volumes you specify and the system affinity to those volumes.

- **Volume Limits:** JES2 will fence a job to an installation–defined number of volumes. In this form of fencing, the job starts with a zero spool partitioning mask work area. As the job allocates spool space, each volume used will correspond to a bit set in the mask. The job will be forced to use only volumes listed in the mask. Minimum fencing is defined as setting the volume limit to "1".
- **MAS member association:** Each spool volume has masks of systems that can allocate space on that volume. Jobs will be limited to the spool volumes associated with a system. You assign spool volumes to particular systems by using the $T SPOOL command. (There are no initialization options to do this.)

You must assess the benefits and drawbacks of spool partitioning and determine whether it would be to your advantage to use it in your spooling environment. Not using spool partitioning at all can provide increased performance because all jobs can allocate space on all volumes. However, not using spool partitioning can also cause severe performance degradation in some cases. For example, if frequently-referenced and long-running jobs are spread across all spool volumes your system might experience spool contention.

In most cases, using spool partitioning increases hardware processing reliability and is more likely to increase performance if you correctly isolate frequently-referenced jobs. Review the following sections, the descriptions of the initialization parameters controlling spool partitioning in *z/OS JES2 Initialization and Tuning Reference*, and the material concerning Exits 11 and 12 in *z/OS JES2 Installation Exits* before using these facilities to modify your spooling environment.

The following section provides a discussion of the initialization statements and the installation exits affecting spool partitioning, their interaction, and some examples.

Using the FENCE parameter to partition SPOOL volumes

The FENCE parameter on the SPOOLDEF statement further defines the spooling environment by controlling the partitioning of the spool volumes. Use the FENCE parameter to tell JES2 how to partition spool volumes when performing I/O operations. If fencing is turned off, then jobs that have already obtained some spool space when fencing was active will not be fenced. Use FENCE=(ACTIVE=YES,VOLUMES=nnn) to indicate that all space that a job requires will be allocated from "nnn" volumes. (FENCE=(ACTIVE=YES,VOLUMES=1) is equivalent to FENCE=YES in prior releases.) These volumes are determined by JES2 as the first available volume from which this job can allocate space; this is determined by the setting of the spool partitioning mask work area. The spool partitioning mask is set with bits on to indicate the spool volumes from which the job can allocate space. If you specify FENCE=(ACTIVE=NO,VOLUMES=nnn), the spool partition mask will not indicate any available volumes (no bits are set on). JES2 sets all the bits on (indicating that all volumes are usable) and no spool partitioning is permitted; the space these jobs require is allocated across all available spool volumes.

Although, FENCE=(ACTIVE=YES,VOLUMES=nnn) indicates that JES2 will fence to nnn volumes in most cases more than one nnn volume can be used in certain situations such as when spool volumes fill or if the job obtains spool space on different systems as it goes through different phases. JES2 will allocate space from a subsequent volume(s) when, and if, the "nnn" volumes become full. At that time, JES2 will allocate track groups from the next available volume based on any other criteria that you have defined.

Associating SPOOL volumes with MAS members: JES2 uses a control block, the BLOB, to provide rapid access to spool space for running jobs. By associating spool volumes to systems in a mask, there will be a limit to the entries the BLOB can hold.

For example, you have a MAS where you specify:

```
SPOOLDEF FENCE=(ACTIVE=YES,VOLUMES=2)
```

Member A has spool affinity for volumes 1,2, and 3. Member B has spool affinity for volumes 1,2, and 4. A job converts on Member A and uses volumes 1 and 3. The job then executes on Member B. The job will potentially use space from three volumes before the fencing limit of 2 is reached. Because Member B does not have affinity for volume 3, the job will be allowed to get spool space from Volume 1 and either Volume 2 or 4 before the fencing limit is reached. A similar situation to this could occur if all spool volumes with affinity for the member temporarily fill and a job begins using space from a volume without affinity for the member.

See the FENCE= parameter on the SPOOLDEF statement in *z/OS JES2 Initialization and Tuning Reference* for how to use this parameter.

Refining spool volume selection: You can further refine spool volume selection by using JES2 installation Exits 11 and 12. See *z/OS JES2 Installation Exits* for information about these exits.

Allocating tracks and track cells

Specify the manner in which the tracks of the volumes are allocated and subdivided into physical records by specifying the BUFSIZE=, TGSPACE=(MAX=), and TGSIZE= parameters on the SPOOLDEF initialization statement.

The tracks are also subdivided into track cells, which are sets of JES2 buffers (physical records) grouped together on a spool volume in a logical order. Indicate the number of records in each track cell by specifying the TRKCELL= parameter on the SPOOLDEF statement during initialization. If the track-cell method is used for despooling data, each track cell that is to be sent to a printer, rather than each spool buffer, can be taken from the spool volume in one operation; thus, it is not necessary to have a separate I/O operation for each spool buffer.

Handling defective track groups

JES2 isolates bad track groups encountered on JES2 spool volumes. When an unrecoverable I/O error occurs, the operator receives message $HASP094 that identifies the address of the track where the error occurred. During purge processing, track groups that are allocated to the job or data set are normally returned to the pool of available track groups; the defective track groups are not returned. To return a defective track group to service, JES2 attempts to recover that track group. If the recovery is successful for all the tracks on a track group, that track group will be returned to service. The verification takes place when the job that encountered the I/O error is purged and during a warm start. To cause JES2 to remember bad tracks across a cold start, identify them using the BADTRACK initialization statement(s).

Dynamic addition and deletion of SPOOL volumes

The spool is the repository for all input jobs and most system output (SYSOUT) that JES2 manages. Because of fluctuating workload requirements, such as month-end processing, your installation might need to add spool space, or for maintenance reasons such as I/O errors, hardware failure, or utilization needs, you might need to delete idle spool packs. In order to optimize system availability, JES2

provides a facility to dynamically add and delete spool data sets. Through the use of operator commands ($S SPL, $P SPL, and $Z SPL) spool data sets can be added, drained, or halted without interrupting JES2 processing.

Spool data sets can be managed by the operator without affecting the normal processing of work, or any accompanying online system or processor availability. Therefore, the operator has the flexibility of adding or deleting spool data sets to meet your installation's varying workload needs. This facility permits multiple addition and deletion of spool data sets simultaneously, provided the maximum of 253 spool data sets is not exceeded at any one time. (Your maximum number of data sets can be less than 253. This number is dependent upon your JES2 track group size, JES2 buffer size, and the amount of space allocated on each spool data set.)

On a cold start, all mounted volumes whose first four or five characters match the VOLUME= parameter on the SPOOLDEF statement and contain the JES2 data set SYS1.HASPACE are automatically started. Spool volumes that do not contain the SYS1.HASPACE data set that are required for processing can be started dynamically in the future by use of the operator commands after the SYS1.HASPACE data set has been added to that volume.

Following a warm start, JES2 restores the status of the spool configuration. A spool data set can be deleted at an all-member warm start if that data set is not available (for example, the volume is not mounted). However, if you delete a spool data set during an all-member warm start, you will lose all work which resides on those spool data sets (which can include: JESNEWS and spooled messages). If the data set is put into inactive status by the operator response (GO) to the $HASP853 message, all work on that data set can later be recovered by the $S SPL command. However, if the data set is drained by the operator response (PURGE) to the $HASP853 message, all work on that data set is lost.

The operator can use the $D SPL command to display the status of the data set and the jobs currently using a specific data set. The operator should be aware of the contents of a spool data set before deleting it. The deletion of a spool volume will not complete while jobs allocated to that volume are active in one of the phases of job processing. For example, if jobs allocated to the volume are in execution, the deletion command will not complete until the job finishes execution. Never- or seldom-ending jobs such as SYSLOG, IMS™, CICS* and VTAM® generally must be given special consideration. You must establish procedures to prevent such situations from occurring, and, if appropriate, establish procedures to handle redistribution of long-running jobs and system data sets. Refer to “Performance considerations and recommendations” on page 171 for the description of device partitioning procedures. Additionally, the $D J,SPOOL command will display those data sets which are currently allocated to specific jobs.

Status of SPOOL volumes

A spool volume is classified as having one of six status types, dependent upon whether it can be used to perform work and its ability to allocate or deallocate spool space. These status types and their definitions are presented in Table 47.

Table 47. Spool volume status types and definitions

Status	Explanation
ACTIVE	The spool volume is in use. Allocation and deallocation of spool space is permitted. The volume is allocated to all systems in the multi-access spool complex.

Table 47. Spool volume status types and definitions (continued)

Status	Explanation
STARTING	The spool volume is not yet ACTIVE. It is in the process of dynamic allocation or formatting. During this time, the volume is being dynamically allocated to each system in the multi-access spool complex. Allocation or deallocation of spool space is not permitted.
DRAINING	Jobs already residing on the volume are selectable for processing; also deallocation of spool space is permitted. Allocation of new spool space is not permitted. The volume must be completely empty before the drain process can complete. When the volume is completely empty, the volume will be dynamically unallocated to all systems in the multi-access spool complex. The spool volume then becomes DRAINED.
DRAINED	The spool volume is completely empty and unallocated, that is, no allocated spool space remains on the volume, and the volume is unallocated to all systems in the multi-access spool complex. The volume is no longer known to JES2.
HALTING	The spool volume stops allocation of space and does not select new work. Currently active work (that is, executing, printing, punching, and so on.) is allowed to complete that phase. When all currently active work on the volume completes, the volume will be dynamically unallocated to all systems in the multi-access spool complex. The spool volume then becomes INACTIVE.
INACTIVE	The spool volume can contain allocated tracks, but does not permit further allocation or the processing of currently allocated tracks. The volume has been unallocated on all systems in the multi-access spool complex and can be removed by the operator.

Table 48 summarizes the characteristics of the six status classifications of spool volumes:

Table 48. Summary of spool volume status characteristics. The table footer describes the meaning of each column heading.

STATUS*	IN USE*	ALLOCATABLE*	SELECTABLE*
STARTING	No	No	No
ACTIVE	Yes	Yes	Yes
DRAINING	Yes	No	Yes
DRAINED	No	No	No
HALTING	Yes	No	No
INACTIVE	No	No	No

*

STATUS
The spool volume status

IN USE
Active jobs are using space on the volume

ALLOCATABLE
New spool space can be allocated on a spool volume

SELECTABLE
Jobs that have space on a spool volume can be selected for JES2 processing

Addition of SPOOL volumes

Spool volumes are added either at a cold start or dynamically through use of operator commands (refer to *z/OS JES2 Commands* for specific command usage). You can define a maximum of 253 spool volumes to JES2 at any one time. If the dynamic addition of another spool volume is attempted (thereby exceeding the maximum of 253 spool volumes) the command will be ignored, the volume will not be added, and JES2 issues message $HASP411 to the operator.

If however, the maximum of 253 spool volumes is exceeded during a cold start, JES2 issues a message informing the operator that more spool volumes were found than expected from the SPOOLNUM= specification on the SPOOLDEF initialization statement. Then, the operator can determine whether to end initialization. To display the contents and count the number of defined spool volumes the operator can issue the $D SPL, ALL command. All volumes listed (to include those in **draining** status) should be included in your count. Only after such volumes are completely **drained** are they considered undefined to JES2.

The operator can issue the $S SPL command to add (and format) a new spool volume or return an inactive volume to the configuration. The SPOOL data set on the new SPOOL volume can be pre-allocated or created by JES2 by using the SPACE= parameter on the $S SPL command. During the addition phase of a spool volume, other processing within the system continues normally.

Deletion of SPOOL volumes

Spool volumes can be deleted dynamically by the operator as required to meet your installation's requirements. The operator can either halt ($Z SPL) or drain ($P SPL) entire spool volumes.

Some considerations must be recognized and proper precautions must be taken to prevent improper usage of the spool volume deletion commands. If the volume being deleted (halted or drained) contains spooled or remote messages or the JESNEWS data set, they are automatically moved to another volume; however, this process might not be immediate. Spooled remote messages and SYSOUT are moved from the draining volume when they become the lowest-numbered job on the volume. Also, if there are no active volumes available for these data sets, the deletion process will not complete until an active volume is available to accept them.

The $Z SPL command can be used to allow only currently active work to complete. No new work is selected, and no new tracks can be allocated. A volume is classified as **halting** until the currently active workload on the volume has completed the current phase of job processing and the volume is unallocated to all systems in the multi-access spool complex. When complete, the volume is classified as **inactive**. However, the operator can display information about the volume or individual jobs on that volume. You must issue a $S SPL command to bring the volume back into the JES2 environment.

The $P SPL command prevents any available tracks from being selected for allocation. Until all jobs currently allocated to the volume have completed all phases of job processing the volume is considered to have a status of **draining**. The spool volume is **drained** when no allocated spool space remains on that volume, and the volume is unallocated to all systems in the multi-access spool complex. When the spool volume is drained, JES2 does not retain any information concerning that volume. Therefore, the operator cannot display information about the volume.

The CANCEL operand can be added to the drain command ($P SPL) to cancel and deallocate all cancellable jobs on either an **active** or **inactive** spool volume. All jobs and SYSOUT will be deleted from the volume; any spooled remote messages and the JESNEWS data set (if present) are automatically moved to another volume. As with the $P SPL and $Z SPL commands, the removal of these items might not occur immediately. Also, if any data sets residing on another volume belong to a job being canceled, they are also purged. JES2 issues message $HASP890 to inform the operator of all such jobs. All other resources are recovered.

You can use the $P SPL,CANCEL command to drain an inactive volume and purge all jobs allocated to it. **This can result in lost track groups on other volumes.** Any lost space can be recovered at an all-member warm start or the JES2 automatic spool reclamation function will recover them within 1 week, whichever occurs first. If the spool volume can be remounted, the $S SPL,P,CANCEL command will prevent the loss of spool space.

Before reusing a spool volume, you should erase (overwrite with binary zeros) the volume. This avoids any inadvertent disclosure of classified or sensitive information.

Attention: If ***all*** spool volumes are deleted, JES2 will not be capable of performing any work. For information concerning the use and syntax of the various operands that can be used with spool volume configuration commands, see *z/OS JES2 Commands*.

Performance considerations and recommendations

You need to be aware of several performance considerations when taking advantage of the dynamic addition and deletion feature of spool volumes. You must review the following recommendations to avoid serious performance degradation.

Spool partitioning (that is, explicitly limiting or identifying the spool volumes from which a particular job is permitted to allocate track groups) can be a useful tool when future alteration of the spool configuration is anticipated. Because standard JES2 processing allows all jobs to allocate track groups on all available spool volumes, the deletion of a single volume can involve many jobs. Isolation to an installation–defined number of volumes of long-running jobs (or jobs frequently put into hold status for long periods of time) to one spool volume (or as few as possible) can greatly facilitate future spool volume reconfiguration. If however, only particular volumes are allowed to allocate track groups to long-running jobs or jobs often put in hold status, for example, the status of other volumes can be changed as required more easily.

Use the IBM-supplied Exit 11 and Exit 12 to take full advantage of the dynamic addition and deletion of spool volumes without degrading performance. See *z/OS JES2 Installation Exits* for information about these exits.

The spool offload facility can also be used to facilitate the deletion of a volume. Jobs and SYSOUT residing on the volume to be drained can be offloaded to an offload data set, and reloaded at any future time. If needed, these data sets can be reloaded immediately; they will reside on other spool volumes because a halting or draining volume cannot allocate space. (See “SPOOL offload facility” on page 176 and *z/OS JES2 Commands* for further information on the spool off-load facility.)

Copying a SPOOL volume

Because of the manner in which SPOOL volumes and the checkpoint data set interact and because of their constant state of flux as JES2 updates them with job and output status, **you cannot copy a SPOOL volume on a live system to another location and expect to preserve system integrity**. Any attempt to copy and use a SPOOL volume can lead to JES2 problems and will render the copied data out of date and useless. The checkpoint data set is the map into the SPOOL volumes, contains all the job queue elements (JQEs), and job output elements (JOEs) which then point to the related job and output data on SPOOL. The relationships between the checkpoint and the SPOOL volumes must remain unaltered; any such changes will introduce errors and JES2 will eventually lose track of current jobs and output.

If you need to drain a SPOOL volume on one system and reload on another for disaster recovery, migration, or other needs, use the JES2 SPOOL offload facility. See "SPOOL offload facility" on page 176. This is the only supported means to perform a function similar to a copy, and it exists to lessen release-to-release or cold start impact.

Note: IBM suggests that you do not place your checkpoint data set(s) on a SPOOL volume for performance reasons. If you need to move your checkpoint data sets, be aware that you must use the checkpoint reconfiguration dialog. See "Checkpoint reconfiguration: An overview" on page 222 for a complete description of your use of the checkpoint dialog.

Recovering from SPOOL volume failure

If a spool failure occurs, JES2 halts all paths to each spool volume in error the next time data is requested from that system. If an I/O request to a spool volume fails for some other reason, you can enter the $Z SPL command to halt the volume. JES2 allocates spool space from the active volumes only. When you performs an all-member warm start on all the active spool volumes, JES2 issues the $HASP424 message to indicate that the halted spool volume is not mounted. Then JES2 issues the $HASP853 message, to which the operator must reply GO, PURGE, or QUIT. Reply:

- GO if the volume can be recovered. JES2 mounts the volume in an inactive state: jobs previously allocated from this volume are not available for processing until the spool volume is started when an operator enters the $S SPL command.

  Note: If JES2 is interrupted during spool validation (through either an ABEND or a re-IPL), JES2 does not provide the GO option to the operator.
- PURGE only if the spool volume is permanently damaged. JES2 removes all jobs with any space allocated from that volume.

  Use this response only if the volume cannot be recovered. JES2 does not issue any messages to indicate that all these jobs have been purged from the system.

  Note: If JES2 issues message $HASP424 when recovering from a spool failure, generally, you will respond PURGE to $HASP853. However, if you are uncertain if the volume is permanently damaged and want to keep the system running until you have verified its status, reply GO. If you later determine that the spool volume is lost, you can re-IPL and then reply PURGE to $HASP853.
- QUIT ends JES2.

The **$D JOBQ,SPL=(V=volser)** command displays all jobs currently allocated on the spool volume. If you enter GO in response to the $HASP853 message, you can enter this command to examine the spool's contents when you have started the volume.

The **$D A,ALL,X,DEV,V=volser** command displays all jobs that are active and have space on the spool volume.

Use the FENCE=(ACTIVE=YES,VOLUMES=nnn) option on both the SPOOLDEF initialization statement and the $T SPOOLDEF command to minimize the effect of a permanently damaged spool on jobs in your installation. When FENCE=(ACTIVE=YES,VOLUMES=nnn) is specified, JES2 allocates all track groups for a job from the installation specified number of volumes. If the FENCE= parameter is not used, JES2 allocates track groups for each job from all volumes available in the installation. If a volume fails on an installation that does **not** use fencing, that installation is likely to lose the majority of jobs on spool. For more information, see "Using the FENCE parameter to partition SPOOL volumes" on page 166.

In addition, JES2 installation exits 11 and 12 can limit the spool volumes from which a job can allocate space. For more information, see *z/OS JES2 Installation Exits*.

Replacing a damaged spool volume

If a permanently damaged spool volume has been replaced by a new volume, trying to start the new volume using the $S SPOOL command can fail and return message $HASP401. This error occurs if the new spool dataset is not at precisely the same location as the original dataset.

To avoid this error, you must ensure that the new spool dataset has the same attribute values as the original dataset, as follows:

1. List the original attribute values, which are stored on the JES2 checkpoint:

```
$DSPL(SY60JA),UNITDATA
$HASP893 VOLUME(nnnnnn)  UNITDATA=(EXTENT=00,TRKRANGE=(002D,
$HASP893                 826D),RECMAX=12,TRKPERCYL=15)
```

2. Use the attribute values to locate the new spool dataset at the same location as the original:

```
Start track:  X'002D' or 45
Last track:   x'826D' or 33 389
Extent size:  33 389 - 45 + 1 = 33 345
```

3. Use the values to reallocate the new spool dataset on the spool volume:

```
//STEP1  EXEC PGM=IEFBR14
//DD1    DD   DSN=SYS1.HASPACE,DISP=(,KEEP),UNIT=3390,
//            VOL=SER=SY60JA,SPACE=(ABSTR,(33345,45))
```

4. Rerun the $S SPOOL command.

SPOOL performance considerations

You can improve the performance of spool data sets through:

- "Selecting SPOOL devices" on page 174
- "Allocation of SPOOL space" on page 174
- "Track celling" on page 175.

Selecting SPOOL devices

When selecting devices for spool volumes, consider that the volume(s) which contain JES2 control blocks, job input and output data, and spool message queue records (for remote terminals) should go on a device that has reasonable speed and good capacity, such as a 3390. Do not use DASD intended to transmit bulk data. SYS1.HASPACE is the default name of this data set; you can rename the data set through the DSNAME= parameter on the SPOOLDEF initialization statement. You can define from 1 to 253 spool volumes with the name SYS1.HASPACE to use as JES2 spool. The BUFSIZE= parameter on the SPOOLDEF initialization statement defines the length of each record.

For best performance, it is desirable to dedicate spool volumes (that is, do not share a volume with paging data sets or other data sets). If there is more than one spool device, they should be put on separate channels. The channel need not be dedicated, however, because JES2 channel utilization is low. The $S SPL,FORMAT command, however, does result in extremely high channel utilization, but this is only a temporary condition, lasting for the duration of spool formatting.

You can enhance the performance of spool volumes through the use of cache controllers that provide high-speed buffered data. In particular, the 3990-3 cache controller provides JES2 spool volumes with:

- Larger cache memory sizes
- Non-volatile storage (allowing the high-speed buffering of DASD writes and DASD reads)
- Image copies of DASD devices.

The cache is a large electronic buffer that provides quick channel access for data that is currently in use and already stored on DASD. DASD Fast Write is a method of reading from and writing to the data stored in these caches. If you define spool volumes to enhance performance by spreading I/O across different volumes, DASD Fast Write allows you to define fewer large spool data sets without having high IOS queue times (the span of time between when a JES2 member requests IO and when that IO is processed). By reducing the amount of DASD supporting JES2, you can reduce costs at your installation.

Note: IBM suggests running RMF DASD reports to indicate JES2 spool volume performance.

Allocation of SPOOL space

JES2 allocates spool space for a job by dividing each spool volume into track groups. A track group consists of one or more DASD tracks. JES2 allocates spool space to a job one track group at a time. When a given track group is exhausted, JES2 allocates the next track group as close to the perimeter of the device as possible on the spool volume. Track groups are obtained from eligible spool volumes on a rotational basis to balance spool I/O. Seek time is therefore minimized.

The TGSIZE= parameter on the SPOOLDEF statement controls the number of JES2 buffers in a track group. JES2 uses this value to calculate the size (in tracks) of a track group for each spool device. Track group size is rounded to the minimum integral number of tracks that can contain TGSIZE buffers (each set to the BUFSIZE= specification on the SPOOLDEF statement). In an environment with mixed spool devices, it is best to specify a value for TGSIZE= that results in consistent spool space allocation for each job regardless of the device type.

Small values for TGSIZE= might cause an excessive number of track group allocations per job. Large values for TGSIZE= might waste spool space and increase seek time. See the descriptions of BUFSIZE=, TGSIZE=, and TGSPACE=(MAX=) parameters on the SPOOLDEF statement in *z/OS JES2 Initialization and Tuning Reference* for more information on the specification of these parameters, allocation of spool space, and their relation to maximizing device utilization and overall system performance.

Track celling

JES2 uses track celling to maximize its efficiency of performing I/O operations on the spool.

Track cells are sets of JES2 buffers, or track records, grouped together in a logical order on a track. The TRKCELL= parameter on the SPOOLDEF initialization statement indicates the number of records in each track cell. When track celling is used, each track cell that is to be sent to a printer, rather than each record, can be taken (despooled) from the spool volume in one operation. Track celling can be used to reduce spool contention. Note that track-cell despooling is used for local and functional subsystem (FSS) printers only; it is not used for remote job entry (RJE), network job entry (NJE), or the process SYSOUT (PSO) interface.

There are three ways JES2 performs despooling

- Single record reads (no track celling)
- Standard track celling
- Advanced track celling

Standard trackcelling is recommended particularly for high-speed printers operating under the control of a functional subsystem. This is particularly recommended for page-mode data sets that contain graphic images. However, track-cell despooling is not required or appropriate for low-speed printers such as the 3820. By not using track celling for these devices, considerable storage is saved in the FSS address space and in central storage. Therefore, JES2 sets the default for printers to not use track-cell despooling. If an installation requires the option, it must overtly request it by initialization statement, that is, you must add TRKCELL=YES to the PRT(nnnn) statement. To prevent performance degradation, TRKCELL=YES must be specified for all high-speed printers running under the control of an FSS.

Standard track celling can be used if the following two qualifications are met:

- The SYSOUT class must be one for which track celling was specified through the OUTCLASS(v) initialization statement.
- TRKCELL=YES must be specified for the printer on the PRT(nnnn) initialization parameter.

Both nonimpact and impact printers can have the track-cell despooling capability.

The standard track-cell method of despooling can affect the efficiency of both the system and the printers. A printer employing track-cell despooling fixes its buffers and the channel program used to send the data from the buffers to the printer. This storage is not used for anything but sending output to the printer, thereby reducing the number of pages fixed and freed by the MVS system. (The size of this fixed storage area depends upon the SPOOLDEF statement parameters, BUFSIZE= and TRKCELL=.) If too high a value of TRKCELL= is specified, too much central storage capacity is lost because of the increase in the amount of fixed-buffer storage. This can result in decreased system performance.

Specification of TRKCELL= can potentially leave short track cells at the end of each track. These track cells will be allocated to data sets of a SYSOUT class that does not have the track-cell characteristic (TRKCELL=NO) specified on the OUTCLASS(v) statement.

If, after allocating as many track cells as possible per track, at least TRKCELL/2 records remain, then these remaining records are also considered to be a track cell. These **short track cells** detract from the full performance benefits of track celling, but they do allow greater utilization of DASD space. However, if after allocating as many track cells as possible per track, less than TRKCELL/2 records remain, then these remaining records can be allocated to non-track cell data sets within the job. But because these records have a rather high (job dependent) probability of not being assigned to non-track cell data sets, the under–utilization of DASD space may result.

Ideally, specification of TRKCELL= should divide evenly into the number of records in a track group (TGSIZE=) for all spool volumes to provide the greatest DASD utilization. See Appendix A for recommended TRKCELL= values based on device type.

When choosing a value for TRKCELL=, note that during print processing the value specified for TRKCELL= is also the number of JES2 buffers that will be fixed into central storage (twice as many are fixed when RDBLBUFR=YES parameter on the PRINTDEF statement is specified); these pages will not be available to the rest of the system for the duration of the output. These buffers would normally be fixed by the MVS I/O supervisor during print processing; however, to reduce the overhead of constantly fixing and freeing these pages, JES2 leaves them fixed.

Advanced track celling is an internal JES2 process of grouping track group buffer sets. Each set holds an entire track group worth of data, and JES2 reads the entire track group into a buffer. Also, SPIN data sets, SYSIN data sets, and SYSOUT data sets from NJE are candidates for both standard and advanced track celling.

SPOOL offload facility

Spool is the primary medium for all JES2 data. All input jobs and system output (SYSOUT) are stored on the spool. JES2 gives your installation the capability to offload (transmit) data from the spool to a data set and later reload (receive) from the data set to the spool. The spool offload facility can use either tape or DASD as the offload medium.

Spool offload eases the migration from release to release by lessening the impact of a cold start. The facility provides a release-independent means of moving the data on the spool between releases.

Note that spool transmissions can contain clear text versions of all JOB statements. *If they are not properly secured, unauthorized discovery of RACF userid and passwords could occur.*

Overview

The spool offload facility is useful when performing the following tasks:

- Preserving pre-execution jobs and SYSOUT across a cold start by migrating your installation to another release of JES2.
- Relieving a full-spool condition during high-use periods; you can reload at a later time.

- Converting to another DASD type for spool.
- Backing-up spool data sets.
- Backing-up NJE connections.

The following topics explain how to use the spool offload facility, providing details on:

- "Defining the offload facility."
- "SMF record summary: transmitting and receiving jobs and SYSOUT" on page 180.
- "Offloading all pre-execution jobs and SYSOUT" on page 180.
- "Reloading all pre-execution jobs and SYSOUT" on page 184.
- "Offloading selected jobs and SYSOUT" on page 186.
- "Reloading selected jobs and SYSOUT" on page 188.
- "Using the SPOOL offload facility for networking" on page 189.

Defining the offload facility

Spool offload uses the order and format of NJE records and BSAM (Basic Sequential Access Method) processing to perform I/O to either tape or DASD data sets. Transmitter and receiver logical devices act as a connection between the spool and the offload data set, passing noncompressed data back and forth. JES2 allows more than one transmit or receive operation active at a time. Be certain to protect offloaded data by defining the offload data set to RACF. During the offload process RACF ensures that the owner of the SYSOUT data has access to the offload device and on reload ensures that the proper data is reloaded before writing it to spool. You must ensure that your local node is defined to RACF, so that your SYSOUT is reloaded with the proper userid. For more information, see "Protecting SPOOL offload data sets" on page 369.

Use the following five statements to define the logical offload devices:

- OFFLOAD(n)
- OFF(n).JT
- OFF(n).ST
- OFF(n).JR
- OFF(n).SR

The OFFLOAD(n) statement defines the logical offload device that is associated with the offload data set(s). Jobs and SYSOUT are transmitted to and from a data set specified by the DSN= parameter of the OFFLOAD(n) statement. You must specify this initialization statement to use the spool offload facility. This statement is numbered to correspond to the offload transmitter statements. You can define up to eight offload devices per system by the OFFLOAD(n) statement. However, only eight offload devices can be active within a multi-access spool complex at a time. An offload device started on one member of a multi-access spool complex cannot also be started at a second member. If you attempt to do so, JES2 issues the $HASP593 message, indicating that the start has been rejected because the device is already in use.

The statements listed previously collectively control the transmitting and receiving of offload data. If you do not include transmitter and receiver statements following the OFFLOAD(n) statement, JES2 creates them for you. That is, JES2 provides four

default statements: a job transmitter, a job receiver, a SYSOUT transmitter, and a SYSOUT receiver. JES2 provides default parameters values, but does not set any parameters that do not have defaults.

To control the transmission of jobs and SYSOUT that you intend to offload, use the OFF(n).JT and OFF(n).ST initialization statements respectively. For example, the statement parameters can control the transmitting of only specific classes, range of jobs, processing mode, or routecode. (See *z/OS JES2 Initialization and Tuning Reference* for a more complete description of each statement.)

To control the receiving of offloaded jobs and SYSOUT, use the OFF(n).JR and OFF(n).SR initialization statements respectively. As with job and SYSOUT transmitters, see *z/OS JES2 Initialization and Tuning Reference* for a complete description of what work selection criteria you can control.

Selecting jobs and SYSOUT through work selection

JES2 provides the WS= parameter on many device initialization statements to allow you to control output selection. Before using the following work selection subparameters, refer to *z/OS JES2 Initialization and Tuning Reference*.

If printers can select the same work as offload devices, the printers and offload devices compete for the work. If a printer gains access first, it purges the data set or output group if its output disposition (OUTDisp=) is either PURGE or WRITE, making its SYSOUT unavailable for transmitting to an offload data set.

JES2 provides two methods for avoiding such device competition:

1. Creating multiple print instances for each piece of output
2. Printing the output, then creating another copy.

For both methods, establish an installation policy to ensure that the JCL used to submit jobs specify particular work selection characteristics. Such a policy ensures that installation devices select the proper output.

You can create multiple print instances for each piece of output by specifying a different route code (Routecde=), writer name (Writer=), or any other work selection characteristic for the printer and the offload devices. Then JES2 is aware that, until both devices have processed this SYSOUT, the data set cannot be purged.

The following JCL sample exploits this alternative:

```
//sampjcl JOB ....
   :
//PRTOUT OUTPUT DEST=LOCAL
//OFFLOD OUTPUT DEST=TAPE01
   :
//DD1    DD SYSOUT=A,OUTPUT=(*.PRTOUT,*.OFFLOD)
```

At a minimum, IBM suggests defining one offload device for selective offload and another offload device for total spool offload.

An installation can also ensure that all output has printed, then create another copy of the output by specifying OUTDISP=KEEP on all output. When a printer has selected an output group, that output group's disposition changes to LEAVE. Then you can specify OUTDISP=LEAVE on the offload device and change the output's disposition to purge by specifying DISP=DELETE on the OFF(1).xT initialization statement, where *x* specifies either a job or SYSOUT transmitter.

Note: Regardless of how you avoid device contention, remember that all output must print to ensure that it is not lost in the event of a spool volume failure.

Using uncataloged data sets

If you are offloading to DASD, you should pre-allocate the data set because you cannot specify the SPACE= parameter on the OFFLOAD initialization statement. If the output data set has not been pre-allocated and pre-cataloged, you can use the OFFLOAD(n) initialization statement to define the tape unit on which the data set(s) will be defined and specify the tape label and security information about the data sets. The following OFFLOAD(n) parameters specify:

- LABEL= the type of label processing for the specified tape.
- PROTECT= whether the data set receives System Authorization Facility (SAF) protection.
- RETPD= the number of days JES2 retains the tape.
- UNIT= provides either a unit address, device type, or group name (to specify a general group of devices) on which JES2 writes the data set. The second value on this parameter defines the number of devices JES2 allocates to the data set.

Using these OFFLOAD initialization statement parameters frees the operator from additional involvement whenever a new offload data set is created.

Using pre-cataloged data sets

If you use a tape management system, you should not pre-catalog tapes for the spool offload facility. If you pre-catalog a tape with a data set name and a tape catalog that have no match in the z/OS system catalog, JES2 ends the program abnormally with an 8FB reason code and issues message $HASP078.

If you use pre-cataloged data sets, JES2 ignores any parameters you might have specified on the OFFLOAD(n) initialization statement. If the UNIT= parameter is ignored, you could overwrite another data set of the same name. Make sure the spool offload data set defined on the OFFLOAD(n) DSN= statement matches the data set you have cataloged.

JES2 limits pre-cataloged data sets to a 20-volume multi-volume expansion. When preparing to offload jobs and SYSOUT, make sure the data being offloaded does not require more than this limit. If you exceed the number of volumes, JES2 responds with an allocation abend. For example, if you pre-allocate and pre-catalog the offload data set and you specify 6 volumes in the allocation, you can expand up to 20 volumes. JES2 would abnormally end after the twentieth volume.

If you pre-allocate a spool offload data set, JES2 always uses the following data set attributes:

- Undefined record format (RECFM=U)
- 4042-byte block size (BLKSIZE=4042).

If you specify different values for these attributes, JES2 ignores them.

JES2 dynamically allocates the offload data set when the operator specifies that transmitting or subsequent receiving is to begin. JES2 supports simultaneous transmitting to and receiving from different offload data sets.

SMF record summary: transmitting and receiving jobs and SYSOUT

The following lists the SMF record written when transmitting or receiving jobs and SYSOUT. For a complete list of the record contents, see *z/OS MVS System Management Facilities (SMF)*.

Table 49. SMF Records Used by JES2 Spool Offload Facility

Record Number	Action that Causes the Record to be Written
24	• JES2 writes a pre-execution job to an offload data set successfully. • JES2 writes a SYSOUT data set to an offload data set successfully. • JES2 reads a pre-execution job from an offload data set successfully. • JES2 reads a SYSOUT data set header from an offload data set successfully.

Offloading all pre-execution jobs and SYSOUT

Offloading jobs and SYSOUT from the JES2 spool to a sequential data set requires the operator to complete the following 5-step procedure:

1. Display the job output forms queue, information about active jobs, and the number of jobs queued on the current spool.
2. Display the offload device's characteristics and the status of its transmitters and receivers.
3. Set and verify the logical offload data set.
4. Start offloading both jobs and SYSOUT.
5. Stop the offload device when all transmitters are inactive and the transmission is complete.

The following explains each step in this procedure and the JES2 processing considerations that apply.

1. Display the job output forms queue, information about active jobs, and the number of jobs queued on the current spool.

   You can use SPOOL Display and Search Facility (SDSF) or the following JES2 commands to display the current status of jobs on spool. Performing this task before offloading ensures that you have data to compare when you display the contents of the new spool after reloading.

   Command
   Console Display

   $D N Displays job queue information about jobs processed on the spool offload device.

   $D F Displays the job output forms queue.

   $D OJ Displays the output processing characteristics of individual job output elements.

   $D Q Displays the number of jobs on a queue and the percentage of spool use.

   Note: Because the system is not quiesced, the status of jobs on spool can change before you begin offloading jobs.

2. To display the offload device's characteristics and the status of its SYSOUT transmitters (ST), SYSOUT receivers (SR), job transmitters (JT), and job receivers (JR), enter the command:

   ```
   $D U,OFFLOAD1
   ```

   Through this command, you must ensure that the devices display **STATUS=STARTABLE**; which indicates that the devices are drained and ready to begin offloading jobs and SYSOUT. The offload transmitter you intend to use must be drained. Note that Figure 42 displays devices that are drained and available to begin offloading jobs and SYSOUT.

```
$DU,OFFLOAD1
$HASP882 OFFLOAD1 740
$HASP882 OFFLOAD1  DSN=BECKER.DUMPER01,STATUS=DRAINED,LABEL=SL,
$HASP882           PROTECT=NO,RETPD=0,UNIT=(,1),VOLS=255
$HASP884 OFF1.JT 741
$HASP884 OFF1.JT   STATUS=STARTABLE,CLASS=B,CREATOR=,
$HASP884           DISP=DELETE,HOLD=,JOBNAME=,NOTIFY=NO,RANGE=(J
$HASP884           1,65534),ROUTECDE=(),START=YES,SYSAFF=(),
$HASP884           VOLUME=(,,,),WS=(CL/)
$HASP886 OFF1.ST 742
$HASP886 OFF1.ST   STATUS=STARTABLE,CREATOR=,DISP=DELETE,
$HASP886           OUTDISP=(WRITE,KEEP),HOLD=,JOBNAME=,
$HASP886           NOTIFY=NO,RANGE=(J1,65534),ROUTECDE=(),
$HASP886           START=YES,VOLUME=(,,,),WS=(Q/),BURST=,FCB=,
$HASP886           FLASH=,FORMS=,LIMIT=(0,*),PLIM=(0,*),
$HASP886           PRMODE=(),QUEUE=X,UCS=,WRITER=
$HASP883 OFF1.JR 743
$HASP883 OFF1.JR   STATUS=STARTABLE,CLASS=,CREATOR=,HOLD=,
$HASP883           JOBNAME=,MOD=(CLASS=,HOLD=,ROUTECDE=,
$HASP883           SYSAFF=),NOTIFY=NO,RANGE=(J1,2147483647),
$HASP883           ROUTECDE=(),START=YES,SYSAFF=(),WS=(/)
$HASP885 OFF1.SR 744
$HASP885 OFF1.SR   STATUS=STARTABLE,
$HASP885           OUTDISP=(WRITE,HOLD,KEEP,LEAVE),CREATOR=,
$HASP885           HOLD=,JOBNAME=,MOD=(BURST=,OUTDISP=,FCB=,
$HASP885           FLASH=,FORMS=,HOLD=,PRMODE=,QUEUE=,ROUTECDE=,
$HASP885           UCS=,WRITER=),NOTIFY=NO,RANGE=(J1,2147483647),
$HASP885           ROUTECDE=(),START=YES,WS=(Q/),BURST=,FCB=,
$HASP885           FLASH=,FORMS=,PRMODE=(),QUEUE=X,UCS=,WRITER=
```

Figure 42. Display of offload devices

If no offload devices are available, you can wait for the device currently being used, or enter the following command to close, dynamically deallocate, and stop the selection process of the offload data set:

```
$P OFFLOAD1
```

Note that if your installation uses one offload data set to back up the primary offload data set, closing the data set would eliminate the back-up offload device. This command should be used with caution.

If you define at least two offload devices and reserve one for offloading the entire spool, an installation can:

- Avoid having to change the work selection criteria through operator command
- Ensure that no jobs or SYSOUT are lost in the transition from the spool volume to another offload data set.

Reserve one offload transmitter to transmit all pre-execution jobs and SYSOUT to the offload data set by specifying

```
WS=(/)
```

in the initialization stream for the OFF(n).JT and OFF(n).ST initialization statements. Note that this form of offloading does not avoid the device competition problem mentioned in "Selecting jobs and SYSOUT through work selection" on page 178. To avoid device competition with this method, you must ensure that no other devices can select data while the transmitter is offloading.

If the transmitters do not display WS=(/), you must remove the specified work selection criteria through the modify offload command.

Enter the following command:

```
$T OFF(1).xT,WS=(-c/-c)
```

where **x** indicates a Job (J) or SYSOUT (S) transmitter and **c** indicates the specific criteria being removed. JES2 transmits both held and non-held jobs, and SYSOUT.

3. Set and verify the logical offload data set. If you are using a pre-cataloged data set, ensure that the displayed data set is the appropriate data set for the offload task. If not already specified, set the data set name by entering:

   ```
   $T OFFLOAD1,DSN=dsname
   ```

   where **dsname** is the name of the sequential data set to receive the transmitted spool data.

   This command allows an operator to change the name of an offload data set **before** it is allocated to an offload device.

   If you do not use pre-allocated, pre-cataloged data sets, JES2 dynamically allocates them. You must tell JES2 where to place the dynamically allocated data set(s) through the UNIT= parameter of the OFFLOAD statement. **If you do not provide a UNIT= specification, JES2 is unable to proceed with dynamic allocation**. This dynamic allocation facility is only valid for $S OFFLOADn,TYPE=TRANSMIT commands.

4. To begin offloading jobs and SYSOUT, enter the command:

   ```
   $S OFFLOAD1,TYPE=TRANSMIT
   ```

   Note that the offload data set is overwritten each time you enter this command. Therefore, nothing is saved on the offload data set.

   You must start the offload device before its associated transmitters and receivers can be activated. Figure 43 on page 183 provides an example of the interaction of the offload transmit function and the required initialization statements. Note that OFF(n).JT and OFF(n).ST, device statements that are offloading from the JES2 spool (SYS1.HASPACE) to the sequential data set (specified by the OFFLOAD1 statement) are numbered "1" in this example.

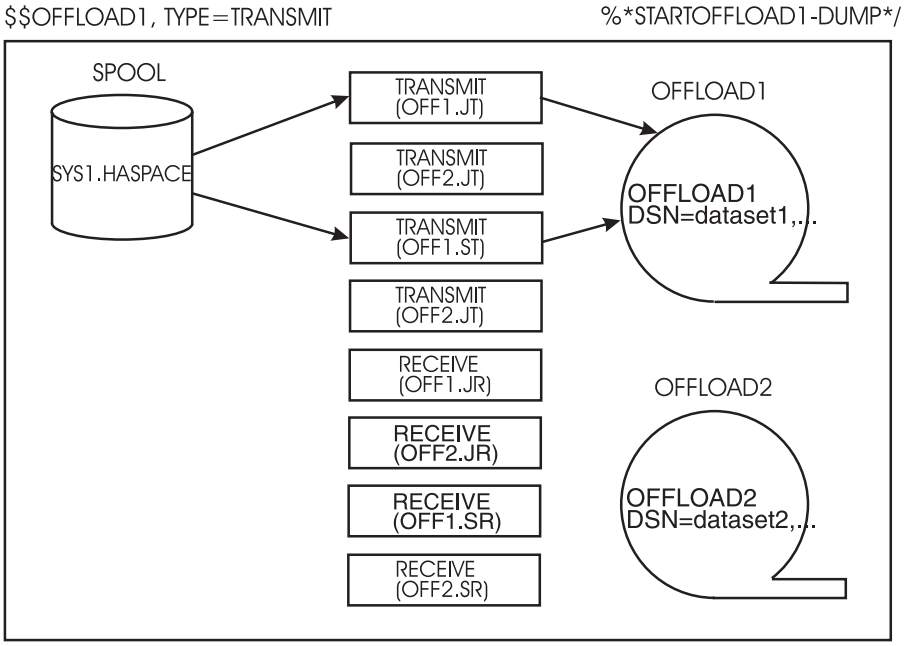


Figure 43. Offload Initialization Statements and $S OFFLOADn,TYPE=TRANSMIT Relation

When transmissions are complete and devices have become inactive, JES2 issues the $HASP524 message for job transmitters and the $HASP534 message for SYSOUT transmitters.

5. To stop the offload device when the job and SYSOUT transmitters are inactive and the transmission is complete, enter the command:

   ```
   $P OFFLOAD1
   ```

   This command causes the offload data set to be closed and dynamically deallocated. Then, it stops the selection process of its associated transmitters and receivers.

   If you fail to enter this command, the transmitters remain active and continue to select jobs and SYSOUT as they become available on the system.

Job disposition

Installations can specify the disposition of both jobs and SYSOUT processed by the offload facility. Specify this disposition on the DISP= KEEP | HOLD | DELETE parameter of the OFF(n).JT and the OFF(n).ST initialization statements. If you allow the DISP= parameter on these statements to default, JES2 deletes the selected output groups after they have been offloaded to the offload data set. Modify the disposition through the $T OFF(n).JT,DISP= and the $T OFF(n).ST,DISP= operator commands. You can specify the following for the DISP= parameter:

- KEEP specifies that the job or SYSOUT remains on the current queue following the offload transmit operation. The same offload device (for example, OFF.xT1) can never select this piece of work; however, another offload device (for example, OFF.xT2) can select it.
- HOLD specifies that the job or SYSOUT remains on the current queue after the offload transmit operation, however, the job is marked held at that time.

  Note: An operator can change this disposition through the $A J command and the $T O,J command.

- DELETE specifies that the job or SYSOUT is purged after the offload transmit operation.

By using KEEP or HOLD, the spool offload facility is useful as an archival system: both the job and SYSOUT remain on spool and a copy is written to the offload data set.

Reloading all pre-execution jobs and SYSOUT

The following procedure describes how to reload (receive) data onto a spool volume. Reloading all jobs and SYSOUT from the sequential data set onto the JES2 spool requires the operator to complete the following 4-step procedure:

1. Display the offload receiver's characteristics.
2. Set and verify the logical offload data set.
3. Receive pre-execution jobs and SYSOUT onto the new spool data set.
4. Display the job output forms queue, information about active jobs, and the number of jobs queued on the new spool data set.

Before beginning this task, consider JES2 naming conventions and spool allocation practices. The node name of the system that performed the offload (transmit) procedure should be known to the system performing the reload (receive) procedure. If not, routing of jobs and SYSOUT can be unpredictable on the reloading system. Any unresolved routing results in jobs and SYSOUT executing and printing locally. Because JES2 does not allocate spool space for jobs that have already been processed, the percentage of spool utilization on a reloaded spool volume is often less than it had been on the old spool volume.

The following explains each step in this procedure and the JES2 processing considerations that apply.

1. To ensure that the work selection parameter (WS=) is empty so that all jobs and SYSOUT can be read onto the spool volume. Display the offload receiver by entering:

   ```
   $D OFF1.xR
   ```

   where **x** indicates either the job (J) or the SYSOUT (S) receiver. If the work selection parameter is empty, proceed to the next step.

   If the work selection parameter has values, you must use the $T OFFn.xR to remove these values.

   Enter the following command:

   ```
   $T OFF(n).xT,WS=(-c/-c)
   ```

   where **x** indicates a Job (J) or SYSOUT (S) transmitter and **c** indicates the specific criteria being removed. JES2 transmits both held and non-held jobs, and SYSOUT.
2. Set and verify the logical offload data set. Ensure that the correct data set (DSN=) is used for the receive process. If it is not, set the data set name by entering

   ```
   $T OFFLOAD(n),DSN=dsname
   ```

   where **dsname** is the name of the sequential data set that contains the spool data for reloading.
3. To receive jobs and SYSOUT that have been offloaded to the offload data set onto the new spool volume, enter

   ```
   $S OFFLOAD1,TYPE=RECEIVE
   ```

This command directs JES2 to reload the data set based on the OFF(n).JR and OFF(n).SR initialization statement specifications. Just as the transmitter statements control what jobs and SYSOUT are offloaded, these two statements control what jobs and SYSOUT are received. Number these job/SYSOUT receivers to correspond to the OFFLOAD(n) statement.

Figure 44 provides an example of the receive function, showing how the required initialization statements interact. Here, the offload receiver statements, OFF(n).JR and OFF(n).SR, specify the data sets and jobs to be received. You must also number this set of receiver statements alike. As many as eight sets of receiver statements can be specified to correspond to the offload data sets specified by the OFFLOAD(n) statement. Note that in this example, OFF(n).JR, OFF(n).SR, and OFFLOAD(n) are numbered "1" to reflect a specific logical device and its corresponding transmitters and receivers.

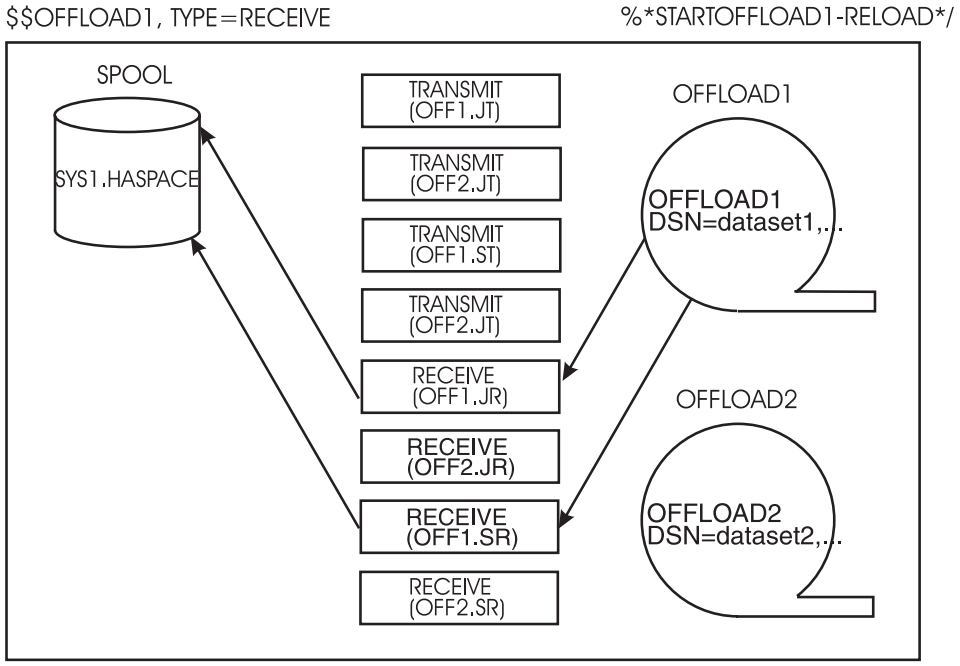


Figure 44. Offload Initialization Statements and $S OFFLOADn,TYPE=RECEIVE

JES2 calls the security authorization facility (SAF) to verify that a userid is valid and has access to data with specific security labels.

The receiving system assigns a new date and time to the SYSOUT receiver upon reload unless CRTIME=RESTORE is specified. If CRTIME=RESTORE is specified, the time to assign is the original creation time (before the data was offloaded).

Unlike the offload (transmit) operation, when both SYSOUT and job receivers are inactive, JES2 drains these devices automatically. The $HASP097 message appears when these operations complete.

4. Ensure that all jobs queued on the spool that held the data before it was transmitted to the offload data set have transferred cleanly to the new spool. See 180 for the list of commands or see *z/OS JES2 Commands*.

   When ensuring that all data has been correctly reloaded to spool, you should note that job IDs are preserved from one spool to another, provided the job number is not already assigned, and either the job number is inside the

JOBDEF RANGE, or RASSIGN=YES. (The default is RASSIGN=YES.) Otherwise JES2 assigns a new job ID from available job numbers within the JOBDEF RANGE= specification.

Offloading selected jobs and SYSOUT

Offloading selected jobs and SYSOUT from the JES2 spool to a sequential data set requires the operator to complete the following 5-step procedure:

1. Display the job output forms queue, information about active jobs, and the number of jobs queued on the current spool.
2. Display the offload device's characteristics and the status of its transmitters and receivers.
3. Set and verify the offload characteristics.
4. Start the offload device to begin offloading selected data.
5. Stop the offload device when all transmitters are inactive and the transmission is complete.

JES2 offloads selectable SYSOUT (ready data sets) by default if output disposition is specified in the SYSOUT transmitter work selection criteria (WS=OUTDisp). However, you can offload held SYSOUT data sets by specifying OFF(n).ST OUTDisp=HOLD|LEAVE.

The following explains each step in this procedure and the JES2 processing considerations that apply to offloading data selectively.

The spool offload facility supports selective offloading of pre-execution jobs and SYSOUT from the spool to offload data sets through the work selection (WS=) parameter. As devices select work selection criteria, so do spool offload devices. These work selection criteria are the same as those used by local and remote printers and punches to select work.

JES2 offloads all selectable work (ready data sets) by default, but can offload held data by specifying OUTDisp=(HOLD,LEAVE) on the OFF(n).ST initialization statement. Specify these selection criteria on the job transmitter and receiver statements during JES2 initialization. The 4 offload transmitter and receiver statements can be modified through $T operator commands.

1. Display the job output forms queue, information about active jobs, and the number of jobs queued on the current spool.

   You can use SPOOL Display and Search Facility (SDSF) or the following JES2 commands to display the current status of jobs on spool. Performing this task before offloading ensures that you have data to compare when you display the contents of the new spool after reloading.

Command	**Console Display**
$D N	Displays job queue information about jobs processed on the spool offload device.
$D F	Displays the job output forms queue.
$D OJ	Displays the output processing characteristics of individual job output elements.
$D Q	Displays the number of jobs on a queue and the percentage of spool use.

Note: Because the system is not quiesced, the status of jobs on spool can change before you begin offloading jobs.

2. To display the offload device's characteristics and the status of its transmitters and receivers, enter the command:

   ```
   $D U,OFFLOAD1
   ```

3. To change work selection characteristics before transmitting jobs and SYSOUT to spool, enter the command:

   ```
   $T OFF1.ST WS=(n)
   ```

   where **n** is a work selection criterion.

   The criteria that you can specify in a specific work selection (WS) list is individualized for the offload job and SYSOUT receivers and the offload job and SYSOUT transmitters. See Table 50 for the complete list.

   You can tailor your offload procedure to filter based upon the work selection of your choice. If you want to offload SYSOUT based on queue selection, enter:

   ```
   $T OFF1.ST WS=(Q/)
   ```

   To offload SYSOUT based on job selection, enter:

   ```
   $T OFF1.ST WS=(JOB/)
   ```

   To offload jobs based on job name, enter:

   ```
   $T OFF1.JT WS=(JOBNAME/)
   ```

   If you find these selection criteria inadequate, you can create installation-defined work selection criteria through the $WSTAB macro. (See *z/OS JES2 Macros* for details on using this macro and extending this function.) You are limited, however, to a maximum of 19 WS criteria (USER and JES2 combined) for any one device.

Table 50. WS Criteria for Spool Offload Devices

Job Receivers	Job Transmitters	SYSOUT Receivers	SYSOUT Transmitters
• CLass • CReator • Hold • JOBname • RANGE • Routecde • SYSaff	• CLass • CReator • Hold • JOBname • LIMit • Priority • RANGE • Routecde • SYSaff • Volume	• Burst • CReator • FCB|C • FLash|O • Forms • Hold • JOBname • OUTDisp • PRMode • Queue • RANGE • Routecde • UCS|T • Writer	• Burst • CReator • FCB|C • FLash|O • Forms • Hold • JOBname • LIMit • OUTDisp • Priority • PRMode • Queue • RANGE • Routecde • UCS|T • Writer • Volume

4. To begin offloading jobs and SYSOUT, enter the command:

   ```
   $S OFFLOAD1,TYPE=TRANSMIT
   ```

Note: To change work selection characteristics after you have begun transmitting jobs and SYSOUT to spool, enter the command:

```
$Z OFFLOAD(1)
```

It halts an offload device temporarily, without deallocating the offload data set.

5. To stop the offload device when the job and SYSOUT transmitters are inactive and the transmission is complete, enter the command:

   ```
   $P OFFLOAD1
   ```

Reloading selected jobs and SYSOUT

Reloading selected jobs and SYSOUT from a sequential data set to JES2 spool requires the operator to complete the following 4-step process:

1. Display the offload receiver's characteristics.
2. Set and verify the logical offload data set.
3. Receive selected jobs and SYSOUT onto the new spool data set.
4. Display information about active jobs, the job output forms queue, and the number of jobs queued on the new spool data set.

The following explains each step in this procedure and the JES2 processing considerations that only apply to offloading data selectively.

1. Display the offload receivers by entering:

   ```
   $D OFF1.JR
   ```

   and

   ```
   $D OFF1.SR
   ```

2. Set and verify the logical offload data set. Ensure that the correct data set is used for the receive process. If it is not, set the correct data set name by entering:

   ```
   $T OFFLOAD(n),DSN=dsname
   ```

   where **dsname** is the name of the sequential data set that contains the spool data for reloading.

   To further control the reload (receive) process, you can allow the job and SYSOUT receivers to modify some of the data set and job characteristics when the data sets and jobs are reloaded by entering the command

   ```
   $T OFF1.xR,MOD=(...)
   ```

   For available values, see the following figure.

Table 51. MOD Criteria for Spool Offload Receiving Devices

Job Receivers	SYSOUT Receivers
• CLass • Hold • Routecde • SYSaff	• Burst • FCB • FLash|O • Forms • Hold • OUTDisp • PRMode • Queue • Routecde • UCS • Writer

3. To receive jobs and SYSOUT that have been offloaded to the offload data set onto the new spool volume, enter:

   ```
   $S OFFLOAD1,TYPE=RECEIVE
   ```

   When reloaded, these data sets become automatically selectable for output processing to devices that were not originally able to select them. For example, you can set up the WS criteria for all printers and punches to select data sets with these "modified" characteristics. The MOD= parameter allows you to reload data sets originally assigned to a held class, and have JES2 change them automatically to nonheld status. Then change the class to one only selectable by a remote printer. As appropriate, the previously held SYSOUT are available to be selected by remote printers.

   The range of selectable characteristics allows you to avoid reloading jobs or SYSOUT that waste spool space.

   For more information on how to use the work selection parameters, see the following:

   - "Variable work selection criteria" on page 113
   - Spool Offload initialization statements in *z/OS JES2 Initialization and Tuning Reference*

   For more information about how to create installation-specific work selection criteria, see the following two sections in *z/OS JES2 Installation Exits*:

   - "Defining JES2 Tables"
   - "JES2 $SCAN Facility"
4. Ensure that all jobs queued on the spool that held the data before it was transmitted to the offload data set have transferred cleanly to the new spool. See 186 for the list of commands or see *z/OS JES2 Commands*.

Using the SPOOL offload facility for networking

The spool offload facility provides a primitive form of job and SYSOUT transmission. By offloading a subset of jobs at one location and reloading them at another, the jobs will be effectively transmitted from location to location. Although this form of job networking is of limited value with widely separated machines, it is a possible alternative for NJE lines; you can offload spool files on one node, switch or physically transfer the offloaded data to the target JES2 system and use a spool offload receiver to reload the data.

In the following example, the local node offloads all jobs and SYSOUT with the route code SANJOSE. When completed, this tape can be physically sent to the SANJOSE node. At SANJOSE operators can reload the tape, successfully transmitting the jobs.

```
$T OFF1.JT,R=SANJOSE,WS=(R)
$T OFF1.ST,R=SANJOSE.*,WS=(R)
$S OFFLOAD1,TYPE=TRANSMIT
$P OFFLOAD1
```

Figure 45. Offloading Data to Send to Another Node

Note: You cannot specify a second-level destination on a route code associated with pre-execution jobs.

Chapter 4. Checkpoint data set definition and configuration

The following section contains checkpoint data set definition and configuration information.

JES2 checkpoint function

The JES2 checkpoint function performs two separate functions:

1. Job and output queue backup to ensure ease of JES2 restart
2. Multi-access spool (MAS) member-to-member workload communication to ensure efficient independent JES2 operation.

The function(s) the checkpoint data set will perform in your configuration depends on whether you have a JES2 single-member MAS, or a multi-member MAS with as many as 32 members.

Checkpointing is the periodic copying of a member's in-storage job and output queues to the **checkpoint** data set, which can reside on either a DASD volume or a coupling facility structure. Checkpointing ensures that information about a member's in-storage job and output queues is not lost if the member loses access to these queues as the result of either hardware or software errors. Because all members in a JES2 MAS configuration operate in a loosely-coupled manner, each capable of selecting work from the same job and output queues, checkpointing allows all members to be aware of current queue status.

Within the single-member environment, the checkpoint function operates solely as a backup to the "in-storage" job and output (SYSOUT) queues maintained by JES2.

In a MAS environment the checkpoint data set not only backs up the job and output queues, it also links all members. It is the commonly accessible repository of member activity that allows each member to communicate and be aware of the current work load. The checkpoint data set contains a record of member values that describe the overall configuration of the MAS environment and specific characteristics and information that describes the current status of each member. The checkpoint allows all members to access and update (write to) the checkpoint data set and also allows all members to refresh their in-storage queues by reading from the checkpoint data set.

The following discussion describes:

- "Placement of the checkpoint data set" on page 192

  This topic explains how workload performance can be enhanced by placing a checkpoint data set on a coupling facility structure rather than on DASD.
- "Checkpoint data set specifications" on page 193

  This topic explains how to specify your primary and secondary checkpoint data sets during initialization, using the CKPTDEF initialization statement for checkpoints that reside on both a coupling facility structure and on DASD.
- "Determining the size of your checkpoint data set" on page 197

  This topic explains how to determine the number of 4K records required for a member's checkpoint data set(s).
- "Placing a checkpoint on a coupling facility for the first time" on page 201

This topic describes how to start JES2 with a checkpoint data set residing on a coupling facility for the first time. Also, it contains guidance on how to correct errors in specifying the size of the coupling facility structure.

- "The checkpoint cycle" on page 203

  This topic describes the stages through which JES2 reads and writes checkpoint records in a MAS environment. This data is especially helpful when using ID TRACE 17.

- "Checkpoint configuration modes" on page 207

  This topic explains how you decide what mode to specify for your checkpoint data set(s) (DUPLEX with backup, DUPLEX without backup, or DUAL), using the MODE= and DUPLEX= subparameters on the CKPTDEF initialization statement.

- "Accessing the CKPTn data set in a MAS" on page 216

  This topic describes how you can use the HOLD= and DORMANCY= parameters on the MASDEF initialization statement to control how access to the checkpoint is shared in a MAS environment.

- "Replacement data set (NEWCKPTn) definition" on page 218

  This topic describes how you can specify replacement checkpoint data set definitions to ease recovery when JES2 loses access to the CKPTn data set(s).

- "Checkpoint reconfiguration: An overview" on page 222

  This topic describes how you can redefine the JES2 checkpoint data set configuration dynamically to replace, discontinue use, or resume using a checkpoint data set.

- "Recovering from member failures on other JES2 members" on page 241

  This topic explains how you can use JES2 operator commands or the AUTOEMEM option to make jobs from a failed member eligible for restart on an active MAS member.

- "Providing copies of the JES2 checkpoint to application programs" on page 248

  This topic explains how authorized application programs can obtain a copy of the checkpoint data set.

Placement of the checkpoint data set

Your installation can place checkpoint data set(s) (primary or secondary) on a coupling facility structure, a DASD volume, or a combination of these locations. DASD used to hold checkpoint data sets can be all the same device type and model or any combination of JES2-supported (see Appendix A, "IBM devices supported by JES2 and how to use them," on page 387) device types and models.

Within a MAS, an installation can specify:

1. "Primary checkpoint on a coupling facility structure; secondary checkpoint on DASD"
2. "Both checkpoints on DASD" on page 193

Primary checkpoint on a coupling facility structure; secondary checkpoint on DASD

Placing the primary checkpoint on a coupling facility structure provides more equitable access for all members of a MAS than that available through DASD because access occurs using coupling facility lock serialization. To ensure the security of that data through the DUPLEX copy retained on DASD, IBM suggests this placement for MAS configurations with four or more members.

Both checkpoints on DASD

Placing both checkpoints on DASD volumes reduces throughput of data and can lock out smaller MAS members from access to the checkpoint. For information on tuning checkpoints residing on DASD, see "Controlling the size of the change log" on page 196 and "Controlled checkpoint data set access" on page 216.

For further information concerning specification of checkpoint data sets, see the CKPT1= and CKPT2= initialization parameter descriptions on the CKPTDEF statement in *z/OS JES2 Initialization and Tuning Reference*.

Attention: IBM does not recommend placing both checkpoints on coupling facility structures or placing the primary checkpoint on DASD and the secondary checkpoint on a coupling facility structure. If both checkpoints reside on Coupling facilities that become volatile (a condition where, if power to the coupling facility device is lost, the data is lost), your data is less secure than when a checkpoint data set resides on DASD. If no other active MAS member exists, you can lose all checkpoint data and require a JES2 cold start. Placing the primary checkpoint on DASD while the secondary checkpoint resides on a coupling facility provides no benefit to an installation.

Note: All combinations of checkpoint data sets on DASD and coupling facility structures are supported as potential error recovery scenarios through the checkpoint reconfiguration dialog.

Benefits of using a coupling facility

Placing a checkpoint on a coupling facility structure ensures equal access among MAS members and provides a better use of system resources than that available through DASD. Data can be transferred to and from a coupling facility much faster than to and from DASD.

JES2 uses the coupling facility lock to serialize access to the checkpoint data set. This coupling facility lock is better than the hardware RESERVE and RELEASE macros required for DASD because it ensures all members fair access to the checkpoint through first-in, first-out (FIFO) queuing. The coupling facility lock affects only the checkpoint data set, while the hardware RESERVE macro locks the entire volume upon which the DASD checkpoint resides.

Checkpoint data set specifications

The following describes how to specify checkpoint data sets through the CKPTDEF initialization statement.

The CKPT1= specification is used for the primary data set copy of the JES2 checkpoint information, and the CKPT2= specification is used for either the DUPLEX data set copy (if operating in DUPLEX mode) or the second primary data set copy (if operating in DUAL mode). Before setting the DUPLEX=ON|OFF parameter for a particular member of your configuration, you should consider whether it is practical for your MAS to use an entire volume for the duplex data set.

If the checkpoint resides on DASD, prefix the checkpoint data set names with SYS1. to provide a complete data set name (for example, SYS1.JESCKPT1 and SYS1.JESCKPT2). Define the placement of these data sets using the VOLser subparameter of the CKPT1= and CKPT2= parameters on the CKPTDEF initialization statement.

If the checkpoint resides on a coupling facility structure, you do not specify a volume. In this manual, checkpoints that reside on coupling facility structures are represented as SYS1_CKPT1 and SYS1_CKPT2. Either the CKPT1= or the CKPT2= parameter on the CKPTDEF initialization statement must be specified; there is no default for the data set name, the volume serial number, or the structure name. Proper placement is vital to JES2 performance, particularly in a multi-access spool configuration environment.

Chapter 7, "Providing security for JES2," on page 349 discusses using RACF to protect the checkpoint data sets that reside on DASD. Checkpoint data sets that reside on a coupling facility structure are RACF-protected by the coupling facility service.

CKPTn data set definition and placement on a coupling facility

Checkpoint data sets on a coupling facility structure are suitable for MAS configurations of all sizes (2- to 32-member). The coupling facility lock removes the need for both the software lock and the hardware RESERVE and RELEASE logic. JES2 use of the coupling facility lock provides for more equitable sharing of the checkpoint among all members. On DASD, a faster processor can monopolize a processor's shorter channel path to the checkpoint. When the checkpoint resides on a coupling facility, the first-in, first-out (FIFO) method of queuing ensures that all MAS members can have equal access to the data.

To place a JES2 checkpoint data set on a coupling facility structure, you must first define that structure in your installation's administrative coupling facility resource management (CFRM) policy. When an installation activates an administrative policy, it becomes the active policy. For more information about installing a coupling facility, see *z/OS MVS Setting Up a Sysplex*. Specify the coupling facility structure's name through the STRNAME= subparameter of the CKPT1= and CKPT2= parameters on the CKPTDEF initialization statement.

Recommendation for initializing a checkpoint on a coupling facility structure

IBM suggests initializing JES2 with the checkpoint data sets defined on DASD, and then forwarding them to a previously-defined coupling facility through the checkpoint reconfiguration dialog. For more information about how to perform this task, see both "Replacement data set (NEWCKPTn) definition" on page 218 and "Checkpoint reconfiguration: An overview" on page 222.

Note: When JES2 connects to the specified coupling facility structure, RACF protection is provided.

Ensuring the integrity of checkpoint data

Through the VOLATILE= parameter, JES2 allows you to specify how an installation responds when a coupling facility with a checkpoint data set structure becomes volatile. Note that a coupling facility is always considered volatile unless you have taken measures to protect against the loss of power to that coupling facility. See *ES/9000 and ES/3090 PR/SM Planning Guide* for a description of how to monitor the volatility of coupling facilities.

When a coupling facility becomes volatile, if power to the coupling facility is lost, all data on the coupling facility structure is lost; JES2 cannot guarantee the integrity of the data. If you do not specify this parameter, it defaults to WTOR for both the ALLCKPT= and the ONECKPT= subparameters.

Note: When a volatile coupling facility becomes non-volatile, JES2 does not inform you of the change. Use the $D CKPTDEF command to monitor the volatile status of your checkpoints.

There are two values for VOLATILE=

- **ONECKPT** - Used by JES2 when another available (INUSE=YES) checkpoint is still non-volatile.
- **ALLCKPT** - Used by JES2 when all available (INUSE=YES) checkpoints are volatile.

Ignoring the volatile status: If you have a primary checkpoint data set residing on a coupling facility and a secondary checkpoint data set residing on DASD, you can specify:

```
CKPTDEF .... VOLATILE=(ONECKPT=IGNORE)
```

If the one structure becomes volatile, JES2 ignores the status change; the member issues no message and does not force an operator into the checkpoint reconfiguration dialog. In this instance, you might want to specify IGNORE for ONECKPT= because the secondary checkpoint data set ensures the integrity of your installation's data. Because DASD data sets are always non-volatile, the ALLCKPT setting has no effect in this configuration.

Requesting operator intervention for volatile status: If both checkpoints reside on different Coupling facilities, you can be notified when both checkpoints become unstable. You can specify:

```
CKPTDEF .... VOLATILE=(ONECKPT=IGNORE,ALLCKPT=WTOR)
```

WTOR ensures that JES2 sends a message to the operator console. You can choose to run with an unstable checkpoint or enter the checkpoint reconfiguration dialog to change the primary checkpoint data set. Because ONECKPT=IGNORE, JES2 issues this message only when both Coupling facilities become volatile (unstable). This is the state where, if power to the coupling facility is lost, all data on the coupling facility is lost. However, if both checkpoints have been specified as coupling facility structures, but the secondary checkpoint specifies INUSE=NO, JES2 uses the ALLCKPT specification, because the primary checkpoint data set is the only available checkpoint.

For JES2 to issue a message when only one of the checkpoint data sets becomes volatile, specify

```
CKPTDEF .... VOLATILE=(ONECKPT=WTOR,ALLCKPT=WTOR)
```

If either or both checkpoints become volatile, JES2 issues the message that allows you to decide whether to enter the checkpoint reconfiguration dialog or continue processing without making a change. Because it is important to monitor the status of the checkpoint, the VOLATILE= parameter defaults to WTOR for both the ONECKPT and ALLCKPT subparameters.

Requesting immediate invocation of dialog: If you would rather enter the checkpoint reconfiguration dialog as soon as a coupling facility becomes volatile, specify:

```
CKPTDEF .... VOLATILE=(ONECKPT=DIALOG,ALLCKPT=DIALOG)
```

For more information about using the checkpoint dialog to **forward** checkpoint specifications, see “Checkpoint reconfiguration: An overview” on page 222.

CKPTn data set definition and placement on DASD

Checkpoint data sets should reside on a high-speed, low-usage device, and they should be allocated on a cylinder boundary. If available, the best location for each data set is on a dedicated 3390 device. To obtain optimum performance, the CKPT1 and CKPT2 data sets should be the only data sets on their respective DASD volumes if your configuration is operating in DUAL checkpointing mode. If your configuration is operating in DUPLEX mode, the data set defined by the CKPT1= parameter should reside on a separate volume. (See "Checkpoint configuration modes" on page 207 for an explanation of DUAL and DUPLEX modes.)

IBM suggests placing the CKPTn data sets on separate volumes to ease recovery. If the CKPTn data sets are physically removed from one another, they have separate power sources, separate control units, and no single common point of failure.

Note: All members of a MAS configuration must have at least one channel path to the device containing the primary checkpoint data set(s).

Figure 46 provides an example for defining checkpoint data sets.

```
//ALLOCATE JOB  (...),'PREPARE FOR JES2',MSGLEVEL=1
//ALLOCAT  EXEC PGM=IEFBR14
//*
:
//*
//CHECK1   DD   DSN=SYS1.JESCKPT1,UNIT=3390,
//              VOLUME=SER=CHECK1,DISP=(NEW,KEEP),
//              SPACE=(ABSTR,(45,15)),DCB=(DSORG=PSU)
//*
//CHECK2   DD   DSN=SYS1.JESCKPT2,UNIT=3390,
//              VOLUME=SER=CHECK2,DISP=(NEW,KEEP),
//              SPACE=(TRK,13256),DCB=(DSORG=PSU)
/*
```

Figure 46. Example of Defining Checkpoint Data Set

Performance can be severely degraded if the checkpoint data set(s) resides on a SPOOL volume; avoid placing a checkpoint data set on a SYS1.HASPACE data set.

If you use global resource serialization, it is strongly recommended that you add the checkpoint data set(s) to the member exclusion resource name list (RNL). If you do, global resource serialization does not delay a JES2 member's access to the data sets checkpoint data sets. See *z/OS MVS Planning: Global Resource Serialization* for further information on this procedure.

Attention: Do not use a hardware RESERVE to serialize access to any other data set on the CKPT1 or CKPT2 volumes. If you do, serious degradation of JES2 throughput or a deadlock situation can result.

Controlling the size of the change log

When the checkpoint data set resides on DASD, the change log communicates checkpoint updates to the member that is reading the checkpoint. Because the change log only communicates the changes to the 4K records (not each entire 4K record that has a change recorded on it), checkpoint performance is enhanced. If the change log is small, it will reside only on the first track of the checkpoint data set. This is desirable because it can then be read at the same time as the Lock, Check, and Master Records during the initial (READ1) portion of the checkpoint

cycle thereby decreasing the amount of I/O to the checkpoint. If the used portion of the change log is large and spills onto the second track, a second read is required.

You can use the LOGSIZE= parameter on the CKPTDEF initialization statement to control the size of the change log. If you do not assign a value to that parameter, JES2 calculates a value based on the checkpoint mode (DUPLEX versus DUAL; for more information on the modes, see "Checkpoint configuration modes" on page 207) and the number of change log records that fit on track 1 of the checkpoint data set defined by CKPT1= and CKPT2= on the CKPTDEF statement. (See the LOGSIZE= parameter on the CKPTDEF statement for the specific algorithm used to calculate the default change log size.) The LOGSIZE= value can be changed either on a cold start or an all-member warm start.

Storage management considerations on DASD

Be careful when using an automated storage management tool on a volume that contains the JES2 checkpoint. Because MVS does not record that JES2 allocated the checkpoint data set, the tool could attempt to move the data set erroneously.

Do not allow any automated storage management tool to move the JES2 checkpoint data set while the JES2 that uses it is active. You **must** inform the tool not to move the checkpoint data set by specifying DCB=(DSORG=PSU) on the DD JCL statement that allocates the checkpoint data set. However, this method might not work with all products.

Determining the size of your checkpoint data set

Adequately determining the size of your checkpoint data set is critical to JES2's ability to initialize, and it should be large enough for future expansion. This section presents it as a 2-step procedure:

- Step 1: Determine the number of 4K records needed to contain the checkpoint.
- Step 2: Use the 4K value to determine the number of cylinders for DASD or kilobytes for a structure size.

If you don't want to calculate the size, you can use a trial-and-error approach by over-allocating the space (two or three times your current allocation) and let JES2 tell you with the $HASP296 or $HASP542 message if you need more. The largest checkpoint that JES2 will currently use is approximately 5,850 tracks on a 3390 device (Approximately 70,000 4K records). Allocating a new data set of this size will ensure that it can accommodate increasing any section to the maximum size.

Determining the current checkpoint size

The $HASP537 message at JES2 initialization displays the number of 4K records in your checkpoint. However, this number will not allow for any increase to these parameters.

Use the $D CKPTSPACE command to determine the size of the checkpoint and amount of free space in the current checkpoints.

Determining the new checkpoint size

The $HASP537 message at JES2 initialization displays the number of 4K records needed for the checkpoint data set. This number does not allow for any increase to these parameters nor for expansion by the $ACTIVATE command which is necessary to enable new functions at the current JES2 level.

To determine how much checkpoint space to allocate to allow the $ACTIVATE command you can issue $DACTIVATE and the response will display the number of 4K pages needed.

To determine how much checkpoint space to allocate for a new release or for expansion:

1. Cold start a secondary JES2 specifying the maximum values to be supported for the specifications identified in Table 52. You should use values for the specifications that reflect what you might need in the future. If you want values for z2 mode, be sure to specify PARM=UNACT since otherwise COLD start assumes you will be in z11 mode.
2. The $HASP537 message will display the number of 4K records required for the checkpoint data set.

Table 52. JES2 parameters affecting checkpoint size

Description	Initialization Parameter	Default
Number of Spool Volumes	SPOOLDEF SPOOLNUM=	32
Number of Track Groups on Spool	SPOOLDEF TGSPACE=(MAX=)	16288
Size of Job Queue	JOBDEF JOBNUM=	1000
Size of Job Output Queue	OUTDEF JOENUM=	2.5 x JOBNUM
Size of Change Log on Checkpoint	CKPTDEF LOGSIZE=	1 (if MODE=DUPLEX) 1 to 9 (if MODE=DUAL)
Size of Block Extension Reuse Table	CKPTSPACE BERTNUM=	Greater of the two values: (JOBNUM + JOENUM/4 + 100) or 399
Number of NJE nodes	NJEDEF NODENUM=	1

If you want to over-allocate checkpoint for expansion but do not want to determine specific values, consider allocating 2 to 3 times your current checkpoint space.

If you need more space, you can use the checkpoint reconfiguration dialog to move the checkpoints to larger data sets or structures.

Use the $D CKPTSPACE command to determine the size of the checkpoint and amount of free space in the current checkpoints. Use the $D ACTIVATE command to see if $ACTIVATE has been issued.

Calculating the checkpoint size

The 4k (4096 byte) record number that you obtained from various sources does not include the space that is needed for control information. There is a fixed amount of control information that must be added including five 4K records and a variable amount based on the number of 4K records. In general, you should add one 4K (4096 byte) record of control information for each 1000 records of checkpoint data (rounding up any partial amount to the next 4K record). Then use this number as described in Table 53 on page 199 (Number of 4K Records per Track and Cylinder for Various DASD Devices) to determine the number of cylinders on DASD. To determine the size of the structure required for checkpoint on a coupling facility use Figure 48 on page 201 or the coupling facility wizard at http://www.ibm.com/systems/support/z/cfsizer/. The number of cylinders on DASD or size of the structure in the coupling facility must be sufficient or JES2 will not initialize.

Remember you should define two checkpoints (CKPT1 and CKPT2) and back-up checkpoints (NEWCKPT1 and NEWCKPT2). The previous calculations provided the size of the checkpoint, and each data set or structure is allocated individually.

Don't forget to allow for extra space for dynamic checkpoint expansion.

Specifying the storage of checkpoints on DASD

When you have calculated the number of 4K records for the checkpoint, use Table 53 to determine the number of tracks or cylinders on DASD, and round up to the nearest whole number.

Table 53. Number of 4K records per track and cylinder for various DASD devices

DASD Device Type	**3380**	**3390**	**9340**
# of 4K Records per Track	10	12	10
# of 4K Records per Cylinder	150	180	150

Note:

1. Track 1 contains only the Check, Lock, and Master records, plus as many change log records (4K each) that will fit on the remainder of the track. Track 1 correction is needed because track 1 is typically not filled to capacity.
2. The change log records can overflow onto subsequent tracks.
3. If the data set is not large enough, JES2 issues the $HASP296 message to indicate the number of additional tracks required. (There is no penalty for over-estimating the number.)

For example, 4K records require the following amount of space on DASD Device Type 3380 or 3390, if the number of 4K records for type 3380 is 376 and the number of 4K records for type 3390 is 378:

DASD Device Type 3380:
38 tracks (376 / 10 = 37.6), or 3 cylinders (376 / 150 = 2.5)

DASD Device Type 3390:
32 tracks (378 / 12 = 31.5), or 3 cylinders (378 / 180 = 2.1).

Syntax for checkpoint space allocation on DASD

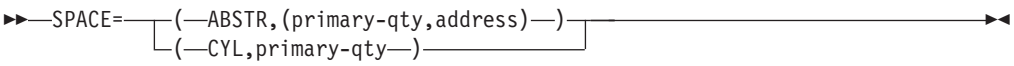
When you have the number of tracks or cylinders required for the checkpoint, you can use the JCL SPACE= parameter to define the size of the checkpoint on a particular volume. Figure 47 on page 200 presents an example of DASD checkpoint data set definition and allocation. Following the example is the syntax for the JCL SPACE= parameter.

```
//ALLOCATE JOB  (...),'PREPARE FOR JES2',MSGLEVEL=1
//ALLOCAT  EXEC PGM=IEFBR14
//*
//CHECK1   DD   DSN=SYS1.JESCKPT1,UNIT=3390,
//              VOLUME=SER=CHECK1,DISP=(NEW,KEEP),
//              SPACE=(ABSTR,(45,15))
//*
//CHECK2   DD   DSN=SYS1.JESCKPT2,UNIT=3390,
//              VOLUME=SER=CHECK2,DISP=(NEW,KEEP),
//              SPACE=(ABSTR,(45,15))
/*
```

Note: These data sets are defined on the CKPTDEF initialization statement with the DSName=, and VOLser= subparameters on the CKPT1= and CKPT2= parameters, respectively.

Figure 47. Example of defining and allocating DASD JES2 checkpoint data sets

The checkpoint data set should be allocated as a single extent starting on a cylinder boundary. JES2 will use only the first extent. Specify space allocation for the checkpoint data set either in absolute tracks (ABSTR) or cylinders (CYL) as follows:



SPACE
specifies the amount of space needed for a new data set on a direct access volume.

ABSTR
specifies that the data set is to be placed at a specific location on the volume. Be certain this location is at the start of a cylinder boundary. ABSTR is recommended if you do not intend to dedicate an entire volume to the checkpoint data set.

CYL
specifies that the data set is to be allocated in cylinders. The CYL specification is available as an alternative method of allocating space for those installations that dedicate an entire volume to the checkpoint data set.

primary-qty

- For ABSTR - specifies the number of full tracks to be allocated
- For CYL - specifies the number of full cylinders to be allocated

address
specifies the track number of the first track to be allocated

Specifying the storage of checkpoints on coupling facility structures

When the checkpoint resides on a coupling facility structure, the checkpoint data set requires additional storage for that structure. JES2 uses a list structure for its checkpoint data set structure. Approximately 85% of the structure can reside in non-control storage (expanded storage).

Approximate storage requirements on coupling facility structures

When you have calculated the number of 4K records, you can calculate the structure requirement by the formula:

```
(# of 4K records) x 4.4 (round up to 256K increment)  +  256K
```

Figure 48. Estimated Structure Sizes for JES2 Checkpoint on a Coupling Facility

Specify the size of the JES2 structure in your installation's active Coupling Facility Resource Management (CFRM) policy as described in *z/OS MVS Setting Up a Sysplex*.

Note:

1. Allocation is in 256K increments.
2. The Check record requires 4K bytes.
3. The Lock record is not written.
4. The Master record is broken into 4K segments.
5. There is no Change Log on a coupling facility.
6. If the structure size is not large enough, JES2 issues the $HASP542 error message to indicate the number of additional 4K records required. (There is no penalty for over-estimating the number.)

426 4K records require ***2304***-kilobytes for a coupling facility structure. Use the number 2304 as the maximum list-set-entry count.

Detailed storage requirements on coupling facility structures

If you want more precise calculations for the size of the coupling facility structure, you can use the formula for determining the size of a serialized list structure in *ES/9000 and ES/3090 PR/SM Planning Guide*.

A structure intended for use as a JES2 checkpoint data set requires the following attributes:

- List-structure-type with names but neither keys nor adjuncts.
- Target entry-to-element ratio value of 1
- List-element-characteristic value of 4
- Maximum data-list-entry size value of 1
- Lock-table-entry characteristic value of 0
- Lock-table-entry count value of 2
- List count value of 2
- Maximum list-set-entry count derived from the number of 4K records determined through either the $HASP537 message or the JES2 checkpoint data set formula.

For further information, visit the CFSIZER Web site at the following URL: http://www.ibm.com/systems/support/z/cfsizer/.

Placing a checkpoint on a coupling facility for the first time

When placing a checkpoint data set on a coupling facility structure for the first time, IBM suggests defining the JES2 checkpoint on a secondary JES2 (preferably a test system). Starting the checkpoint on a secondary JES2 ensures connectivity to the coupling facility and allows you to verify the accuracy of the coupling facility resource management (CFRM) policy definition. Then you can use the checkpoint reconfiguration dialog to forward the primary JES2 member's checkpoint data set onto the coupling facility structure. To start a JES2 member with a checkpoint data set residing on a coupling facility structure for the first time, use the following procedures:

1. Ensure that you have followed the instructions for predefining a coupling facility resource management (CFRM) policy as described in "Defining JES2 structures in a CFRM policy" on page 10.
2. Start a primary JES2 member as described in "Starting JES2 for the first time (a cold start)" on page 39.

   Note the number of 4K records specified on the $HASP537.
3. Cold start a secondary JES2 member with the checkpoint defined on a coupling facility structure, using the STRNAME= subparameter instead of the DSNAME= and VOLSER= subparameters on the CKPTDEF initialization statement. Follow the directions provided in "Defining a secondary JES2 member to MVS" on page 54.

   Ensure that the secondary member's JES2 definitions match those of the primary JES2 member for the following initialization parameters that define the checkpoint data set:
   - SPOOLNUM= and TGSPACE=(MAX=) on the SPOOLDEF initialization statement
   - JOBNUM= on the JOBDEF initialization statement
   - JOENUM= on the OUTDEF initialization statement
   - BERTNUM= on the CKPTSPACE initialization statement.
   - NODENUM= on the NJEDEF initialization statement.
4. If problems occur when initializing the JES2 secondary, see both "Correcting a structure error of size" and "Correcting a structure error of incorrect attributes" on page 203.
5. When JES2 is fully operational (after you receive the $HASP400 message), verify that the coupling facility structure has been allocated by using the $D CKPTDEF command to display the status of all defined JES2 checkpoint data sets.
6. After you have completed testing the checkpoint on a coupling facility, bring down the secondary JES2 member and invoke the JES2 reconfiguration dialog through operator command to forward the checkpoint data set to the coupling facility.

   For complete instructions on how to use the checkpoint reconfiguration dialog, see "Operator-initiated entrance into a checkpoint reconfiguration dialog" on page 233.

Correcting a structure error of size

During initialization, JES2 issues the following messages if the size of the coupling facility structure is too small to contain the checkpoint data set:

- The $HASP536 message with the following error message:

  08, STRUCTURE IS NOT LARGE ENOUGH
- The $HASP542 message with the following text:

  $HASP542 MEMBER *member_name*--STRUCTURE*strname*
  IS NOT LARGE ENOUGH TO CONTAIN ALL CHECKPOINT DATA.
  THE CURRENT SIZE IS *ccccc*K. THE SIZE MUST BE INCREASED
  TO HOLD AN ADDITIONAL *nnnnn* 4K ELEMENTS. THIS WOULD
  REQUIRE A STRUCTURE SIZE OF AT LEAST *mmmmmm*K.

When you receive this message sequence, you must do one of the following tasks (the first being the preferable task):

1. If you have a predefined structure of adequate size to contain the JES2 checkpoint data set in a CFRM policy, you can activate that structure through the STRNAME= value on the CKPTDEF initialization statement.
2. If you do not have a predefined structure of adequate size in any CFRM policy, run the MVS utility cited in "Defining JES2 structures in a CFRM policy" on page 10 and define a structure in a CFRM policy before continuing JES2 initialization.

   Then, use the SETXCF commands to specify a new active CFRM policy that contains a predefined structure with enough 4K records to contain the checkpoint.
3. If you cannot run the MVS utility to create a CFRM policy, invoke the initialization version of the checkpoint reconfiguration dialog and specify the checkpoint data sets on DASD.

   Note: For more information about how to use the SETXCF commands in specific scenarios, see *z/OS MVS System Commands*.

Correcting a structure error of incorrect attributes

During initialization, JES2 issues the $HASP536 message with return codes 1 through 7 if a structure intended to hold a JES2 checkpoint contains the wrong attributes. In this case, ensure that you have specified the correct STRNAME on the CKPTDEF initialization statement. If you have, use the SETXCF commands to disconnect from the structure and then RESPECIFY the same structure during initialization. Respecifying the same STRNAME allows you to change the attributes of the original structure. See *z/OS MVS Programming: Sysplex Services Guide* for more information on coupling facility structure attributes.

If this does not work, you might have to specify DASD values for the checkpoint data set through the initialization version of the checkpoint reconfiguration dialog.

The checkpoint cycle

Although the actual checkpoint cycle consists of a number of reads and writes, it can be considered to include only four basic phases. Depending on whether your complex is processing in DUPLEX mode or DUAL mode (see "Checkpoint configuration modes" on page 207), the actual specifics of the checkpoint process will vary somewhat; however, the basic processing is as follows:

1. Gaining checkpoint control (and RESERVE for DASD)
2. Owning the checkpoint to read and make updates
3. Final writing (and RELEASE for DASD)
4. Allowing other members access opportunity

As you read the following description of the checkpoint cycle, see Figure 49 on page 204, which represents a 3-member multi-access spool configuration.

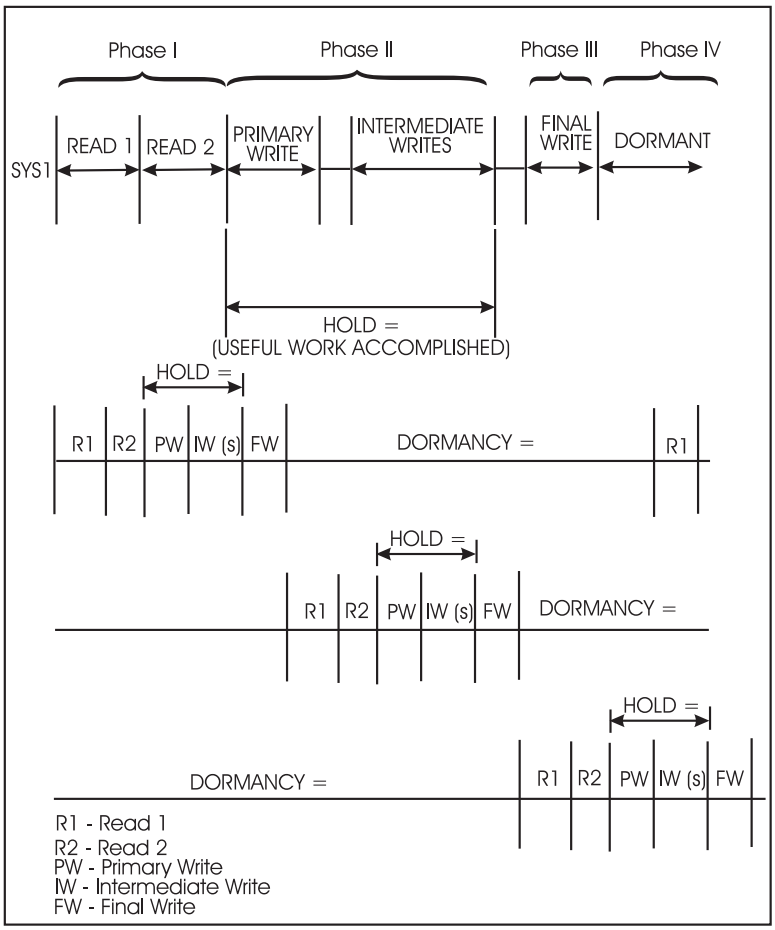


Figure 49. The checkpoint cycle (in a 3-member multi-access spool environment)

Checkpoint cycle overview

If a checkpoint data set resides on a coupling facility structure, serializing access to the checkpoint is greatly simplified because the coupling facility uses its own lock. As a result, checkpoint data sets residing on a coupling facility do not use RESERVE/RELEASE logic or the JES2 software checkpoint lock. Placing the checkpoint data sets on a coupling facility structure reduces the amount of dormant (non-useful) time in the checkpoint cycle.

When a checkpoint data set resides on DASD, JES2 obtains exclusive ownership of the checkpoint data set with a RESERVE (using the MVS RESERVE macro). The RESERVE is always issued against the data set defined by CKPT1=, unless that data set is inactive. If CKPT1 is inactive, the RESERVE is issued against the data set defined by CKPT2=. No other member is allowed access to the checkpoint data set until the member performing the update issues a RELEASE (using the MVS DEQ macro) for the volume on which the data set resides.

To detect and avoid problems because of the early release of the hardware lock initiated by the RESERVE, JES2 uses a backup software checkpoint lock for the checkpoint data set. The checkpoint lock is the first record on the first track of each data set. When a member has control of the primary checkpoint data set, it writes a member-dependent value into the key and data portions of the checkpoint lock

record. Therefore, even if the RESERVE fails, other members in the multi-access spool configuration that try to access the checkpoint data set are prevented from doing so by the checkpoint lock.

If the member holding the checkpoint lock fails before releasing the lock, it prevents other members from checkpoint data set access. Then JES2 issues the following message:

```
$HASP264 WAITING FOR RELEASE OF CHECKPOINT LOCK [BY membername]
```

If the checkpoint resides on DASD and the member holding the checkpoint lock fails before releasing the lock, use the $E CKPTLOCK,HELDBY= command to reset the lock and allow access to the checkpoint data set by the next member requiring checkpoint data set information. Before you reset the checkpoint lock, ensure that the member currently holding the checkpoint lock has failed. If you have enabled the AUTOEMEM option, JES2 requeues failed jobs for processing within the sysplex.

If the checkpoint resides on a coupling facility and the member holding the coupling facility lock fails before releasing the lock, the $HASP263 message is informational; wait for the MVS system to release the coupling facility lock. Then you can issue the $E MEMBER operator command or use the AUTOEMEM option to requeue the failed member's jobs on an active member. To use the AUTOEMEM option, see “Recovering from member failures on other JES2 members” on page 241.

Phase I - Checkpoint RESERVE, lock, and read

When a particular member obtains the checkpoint lock (whether the JES2 software lock or the coupling facility lock), that member can access the checkpoint data.

The JES2 member:

1. Reads the first track of the checkpoint data set (**READ1**).
2. Reads the updated 4K records (**READ2**).

   If the data set resides on DASD and MODE=DUAL, JES2 could have to read in the remainder of the change log if it has spilled onto track 2. The size of the change log that must be read is, of course, dependent upon the number of changed control blocks that reside there.

   Also, in DUAL mode, if more than one checkpoint cycle has taken place since the previous access (by this member), additional 4K records might have to be read.
3. Merges changes into the in-storage copy of the checkpoint before setting the HOLD time.

   This task must be completed before useful work requiring the checkpoint can begin.

Phase II - Updating the checkpoint

When a member has read the checkpoint data set, it can update its in-storage copy of its job and output queues. The changes are recorded on either DASD or the coupling facility by a series of writes: the primary write, intermediate write(s), and the final write. These writes allow the member holding the lock to update checkpoint data to reflect its “view” of member activity.

1. The **primary write** updates the 4K records of the checkpoint data set that is not the most current. These 4K records are located after the first track of the checkpoint data set.

If the checkpoint data set resides on DASD, these 4K records follow the change log if the change log ends on track 1. If, however, the change log is contained on a subsequent track(s), the 4K records are located on that track immediately following the change log.

If the checkpoint data set resides on a coupling facility, there is no change log.

2. The **intermediate writes** record all changes a member makes to the checkpoint data.

If the checkpoint data set runs in DUAL mode, JES2 records these changes in the change log.

If the checkpoint data set is in DUPLEX mode, JES2 records these changes in the 4K records of the data set.

In either case, these changes are read during the next member's phase I; that member's in-storage copy is updated to reflect the current status. During the ownership period, the owning member can make updates to the checkpoint data. Intermediate versions of the checkpoint information are written without losing checkpoint control.

In DUAL mode, the MAS configuration operates with two primary checkpoint data sets:

- The most recently updated data set
- The "down-level" data set.

The down-level data set holds checkpoint information that is one checkpoint cycle behind. This down-level data set was read from, but not written to (updated), during the last checkpoint cycle. In both modes of operation (DUAL and DUPLEX), the first write (that is, the primary write) is always to the down-level data set in order to bring it up to date.

If one member holds the checkpoint a sufficient length of time to have done 10 intermediate writes, the down-level data set is updated following the tenth intermediate write.

Phase III - Final checkpoint write and data set release

The member does a **final write** just before releasing the checkpoint lock. If the checkpoint resides on DASD, when this final write is completed the member holding the RESERVE issues a DEQ, and other members are able to access the newly-updated information. If the checkpoint resides on a coupling facility, when this final write is completed, the coupling facility lock releases the checkpoint.

Phase IV - Other member access

Based on the specifications you assigned to the DORMANCY= parameter on the MASDEF statement, the member that just had checkpoint access will not be allowed access for a period of time. This dormant period allows other members a time of "non-contended" access. Minimally, to assure access of the checkpoint by other members, DORMANCY= should be set to a value not less than the sum of the HOLD= specifications for all other members. See "Member-specific parameter recommendations" on page 217 for performance recommendations for setting this parameter.

Checkpoint configuration modes

There are three ways of specifying the checkpoint configuration mode for your JES2 checkpoint configuration:

- "DUPLEX-mode processing (with backup)"
- "DUPLEX-mode processing (without backup)" on page 211
- "DUAL-mode processing" on page 212.

The type of mode you select depends upon your JES2 configuration. For example, if a checkpoint data set resides on a Coupling facility structure, the mode must be DUPLEX. If an installation specifies MODE=DUAL on the CKPTDEF initialization statement, JES2 processing forces the mode to DUPLEX and notifies the operator through an informational message.

Checkpoint mode can be set through $T CKPTDEF, MODE=.

The following examines performance considerations and explains:

- How to initialize your member to process in a selected mode
- Recommendations on what mode is most appropriate for your member configuration
- Recovery procedures available in the event of an ABEND, I/O error, or hardware problem

DUPLEX-mode processing (with backup)

JES2 can operate its checkpoint configuration in DUPLEX mode if you have defined two checkpoint data sets, a primary and a duplex.

All members of your JES2 configuration (whether a single member operating in a single-member environment or multiple members operating as part of a multi-access spool configuration), should, if possible, operate with DUPLEX=ON on the CKPTDEF statement. This provides the greatest protection from checkpoint error.

DUPLEX (with backup) on DASD

The primary data set is specified by the CKPT1= parameter on the CKPTDEF initialization statement and the duplex (secondary) data set is specified by the CKPT2= parameter on that same statement:

```
CKPTDEF CKPT1=(DSN=dsname,VOL=volser,INUSE=YES),
        CKPT2=(DSN=dsname,VOL=volser,INUSE=YES),
        MODE=DUPLEX,DUPLEX=ON
```

There is no requirement that these two data sets be on unique volumes; however, for performance reasons, and increased recovery potential, it is strongly recommended that you place them on separate volumes.

DUPLEX (with backup) on Coupling facility

If you define both checkpoint data sets on different structures within different Coupling facilities, performance should be increased significantly. The primary data set is specified by the CKPT1= parameter on the CKPTDEF initialization statement and the duplex data set is specified by the CKPT2= parameter on that same statement:

```
CKPTDEF CKPT1=(STRNAME=strname,INUSE=YES),
        CKPT2=(STRNAME=strname,INUSE=YES),
        MODE=DUPLEX,DUPLEX=ON
```

The CKPT1 data set is the primary data set, that is, it is read from and written to at every intermediate read and final write of the checkpoint by all members in your multi-access spool configuration. The duplex data set, CKPT2, is updated less frequently (that is, once for every 10 writes to the primary). If the primary data set suffers an error, the duplex can provide either an exact duplicate of the primary or a very similar, almost current (depending on when it was last updated) copy, of the checkpoint data.

Figure 50 on page 209 and Figure 51 on page 210 provide diagrammatic views of this processing. See these figures as you read the following example of duplex processing. Assume a 2-member multi-access spool configuration: members JESA and JESB, the primary checkpoint named CKPT1 and duplex named CKPT2.

- **A** If JESA gains control of the checkpoint (Figure 50 on page 209), it reads from CKPT1 and
- **B** Does the **primary write** to CKPT2.
- **C** **Intermediate writes** are to CKPT1 with every tenth write (if there are 10 or more) also written to CKPT2.
- **D** The **final checkpoint write** (before JESA releasing its RESERVE) is to CKPT1.

  Note that if the checkpoint data set resides on a Coupling facility structure rather than on a DASD volume, JESA access to the checkpoint is serialized through the Coupling facility lock; the hardware RESERVE is not obtained or released.

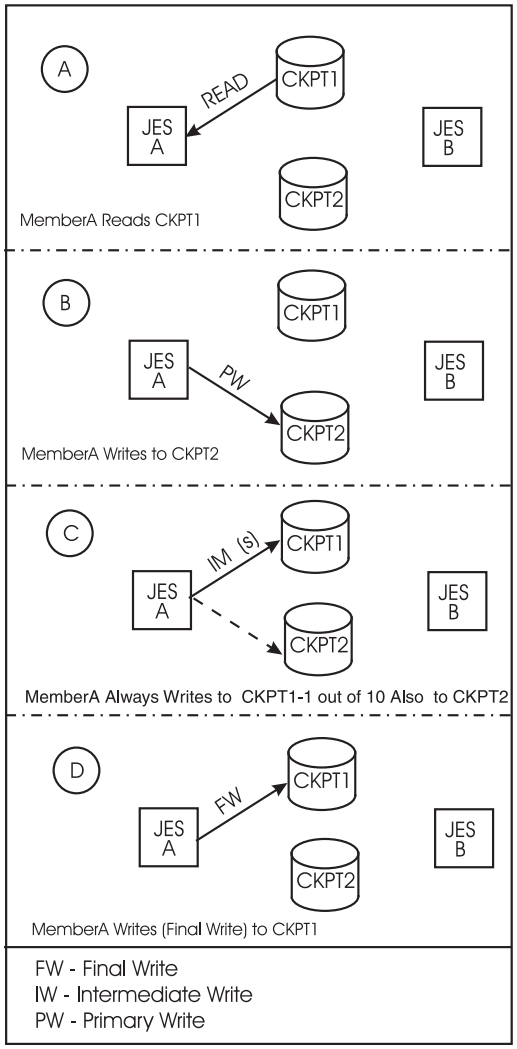


Figure 50. Checkpoint data set processing, DUPLEX mode, two-member multi-access spool configuration - JESA cycle

JESB now gains checkpoint control (Figure 51 on page 210);

- **E** It, too, reads from CKPT1, and
- **F** Performs the primary write to CKPT2.
- **G** Intermediate writes are to CKPT1 with every tenth write to CKPT2, and
- **H** SYSB's final write is to CKPT1.

This process continues, unless an error occurs, at which time you can dynamically reconfigure (forward, suspend, or resume use of the checkpoint data sets) your checkpoint data set specification by responding appropriately to checkpoint reconfiguration dialog messages. (See “Checkpoint reconfiguration: An overview” on page 222 for an explanation of these procedures.)

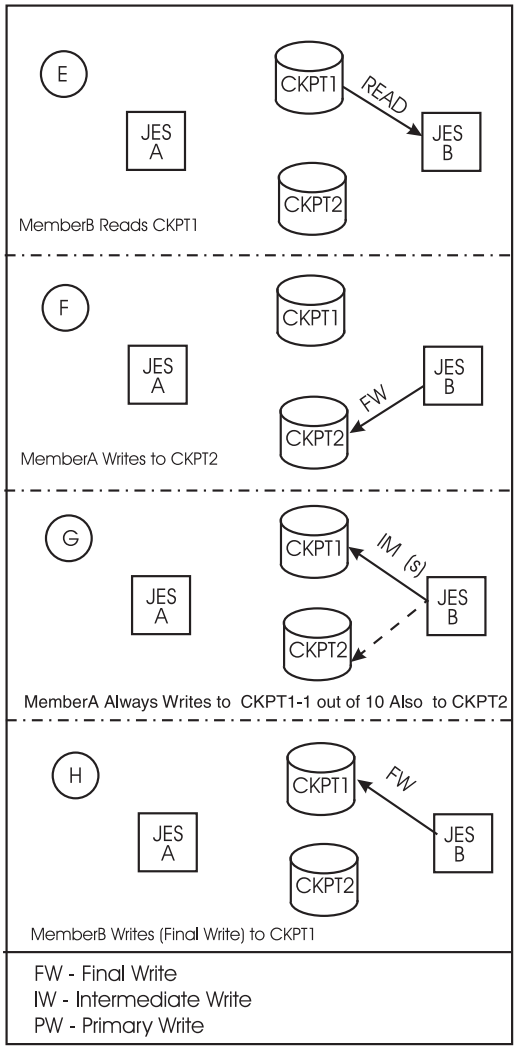


Figure 51. Checkpoint data set processing, DUPLEX mode, two-member multi-access spool configuration - JESB cycle

Even if the primary checkpoint data set, CKPT1, is updated every checkpoint cycle, and the duplex, CKPT2, is updated on a regular basis, when an error occurs on the primary checkpoint data set, a more current copy of the checkpoint might be available to the member that experienced the error.

Each member holds an "in-storage" copy of the checkpoint, which it uses to update the checkpoint. This in-storage data set can be copied into the "replacement checkpoint", NEWCKPTn, data set if one has been predefined (by initialization statement), or dynamically added (by operator command), or in response to checkpoint reconfiguration dialog messages. However, if the member holding this up-to-date in-storage copy has also suffered a failure, the most up-to-date copy of the checkpoint available would be defined by the duplex checkpoint data set.

Attention: DUPLEX mode checkpoint processing is recommended for single member configurations and required if any checkpoint data set resides on a Coupling facility structure.

DUPLEX-mode processing (without backup)

You can specify that your member operate with a single checkpoint data set. In most cases, this mode is not recommended. Effectively, you are processing in an environment in which there will be no backup copy of the checkpoint data set. As a result, JES2 will not need to read/write to two data sets. If that data set becomes damaged, lost, or suffers an I/O error, and your member experiences a concurrent failure, you will have no checkpoint data set available to restart JES2.

However, depending on your specific volume or structure use and configuration, it might not be practical for particular members of your JES2 configuration to maintain a backup checkpoint. For example, a 32-way MAS could be composed of two 3090 processors that manage a heavy batch workload and 30 smaller processors that manage on-line transaction programs. To reduce the cost of storage overhead, this installation might run without a backup checkpoint on the 30 smaller processors, while ensuring that the 3090s operate with DUPLEX=YES.

Another scenario could involve a MAS composed of different level processors. An installation can determine that it is not efficient to run with a backup checkpoint on the slower processors.

DUPLEX (without backup) on DASD

Define a single checkpoint data set on DASD by specifying the following initialization statement:

```
CKPTDEF CKPT1=(DSN=dsname,VOL=volser,INUSE=YES),
        MODE=DUPLEX,DUPLEX=OFF
```

DUPLEX (without backup) on Coupling facility

Define a single checkpoint data set on the Coupling facility by specifying the following initialization statement:

```
CKPTDEF CKPT1=(STRNAME=strname,INUSE=YES),
        MODE=DUPLEX,DUPLEX=OFF
```

Effects of error situations when DUPLEX=OFF

The effects of an I/O error to your checkpoint data set can be disastrous:

- If an I/O error occurs and the checkpoint volume becomes unavailable or damaged, you might be able to enter the checkpoint reconfiguration dialog from an active member of your MAS configuration and forward the data set to a new location.
- If your member fails while losing the checkpoint volume and there are no other active members from your MAS configuration, you will lose all checkpointed data.

You can lessen the possibility of causing such a disruption to JES2 processing by defining a DUPLEX checkpoint data set and by adding the CKPT2= and DUPLEX=ON parameter specification to the CKPTDEF initialization statement. For an example of DUPLEX-mode processing with a backup, see “DUPLEX-mode processing (with backup)” on page 207.

The complexity of DUAL processing is not required in a single-member environment.

DUAL-mode processing

Whereas DUPLEX-mode checkpoint processing uses two checkpoint data sets in a primary-duplex scheme (as described previously), DUAL-mode checkpoint processing uses the same two data sets in a "flip-flop" scheme. DUAL mode is recommended if your installation is suffering from degraded system performance because of high checkpoint data set I/O. The use of the change log provides reduced I/O to the checkpoint data set during most, if not all, update accesses by a member of your multi-access spool configuration.

Note that if any checkpoint data set resides on a Coupling facility structure, you cannot use DUAL mode.

Example of DUAL mode in a 2-member MAS configuration

Consider the basic example illustrated in Figure 52 on page 213. The definition of this checkpoint configuration example is:

```
CKPTDEF CKPT1=(DSN=dsname,VOL=volser,INUSE=YES),
        CKPT2=(DSN=dsname,VOL=volser,INUSE=YES),
        MODE=DUAL
```

Assume member JESA acquires the checkpoint RESERVE.

- **A** It must first read from the most current checkpoint data set to obtain current JES2 queue and system data. For purposes of this discussion, assume CKPT1.
- **B** JESA will then write to the CKPT2 data set (and every tenth write will also be to the CKPT1 data set).
- **C** If JESB now gains control of the checkpoint, it will read from CKPT2 and
- **D** Write to CKPT1 (and every tenth write will also be to the CKPT2 data set). Each successive cycle will then continue that alternating (flip-flopping) procedure.

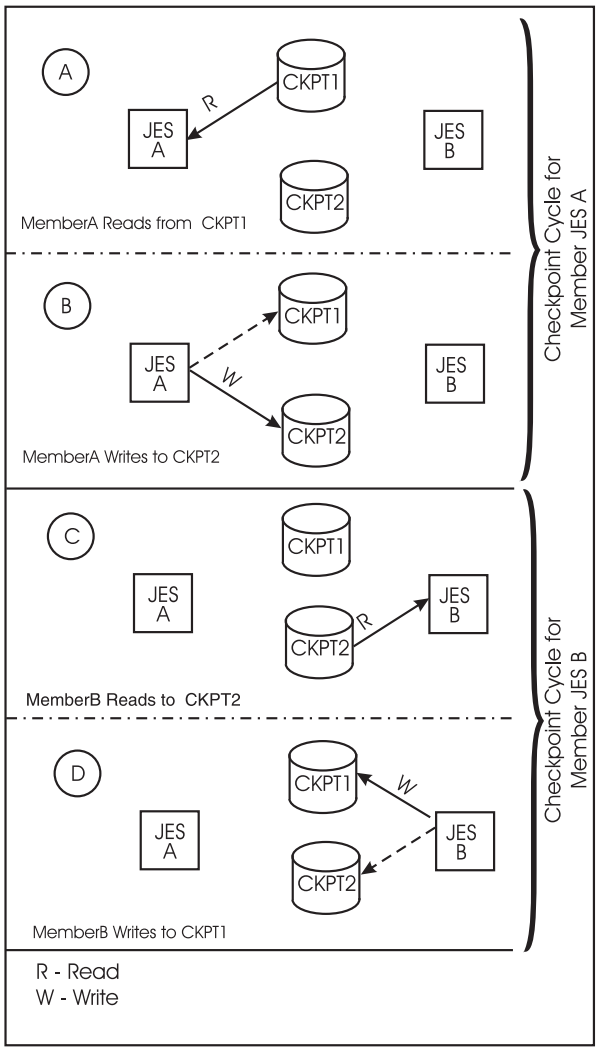


Figure 52. Checkpoint data set processing, DUAL mode, two-member multi-access spool configuration

Example of DUAL mode in a 3-member MAS configuration

Individual members do not always read and write to the same CKPTn data set. However, the member performing a read always uses the checkpoint data set last updated. The sequence of reads depends upon how many members you have in your MAS configuration and the specifications you have assigned to the queue control parameters (DORMANCY= and HOLD= on the MASDEF initialization statement). To illustrate this condition, a second, more complex example follows; see Figure 53 on page 215.

Assume a 3-member multi-access spool configuration; members named JESA, JESB, and JESC and the DUAL checkpoint data sets named CKPT1 and CKPT2. As in the previous example, assume JESA obtains checkpoint data set control.

- **A** JESA initially reads from CKPT1 and writes to CKPT2.
- **B** JESB now gains control, and reads from CKPT2 and writes to CKPT1.

To this point, all I/O is the same as the first example presented in Figure 52 on page 213. For simplicity, let's assume JESC now gains control.

- **C** JESC reads from CKPT1 and
- **D** Writes to CKPT2.
- **E** When JESA again gains control, during its next checkpoint cycle, it reads from CKPT2 and
- **F** Writes to CKPT1.

As noted previously, this is a simplistic view without regard for your specific queue control parameter specification. The ordering of reads and writes can be rather complicated, because any one member does not always read or write to a particular data set in DUAL mode. For example, should an error occur, you cannot assume that if the error occurred on JESB, CKPT2 is at fault. With this background of processing modes, you will be better able to understand the checkpoint reconfiguration process should you need to recover from an error or decide to alter your checkpoint data set definition.

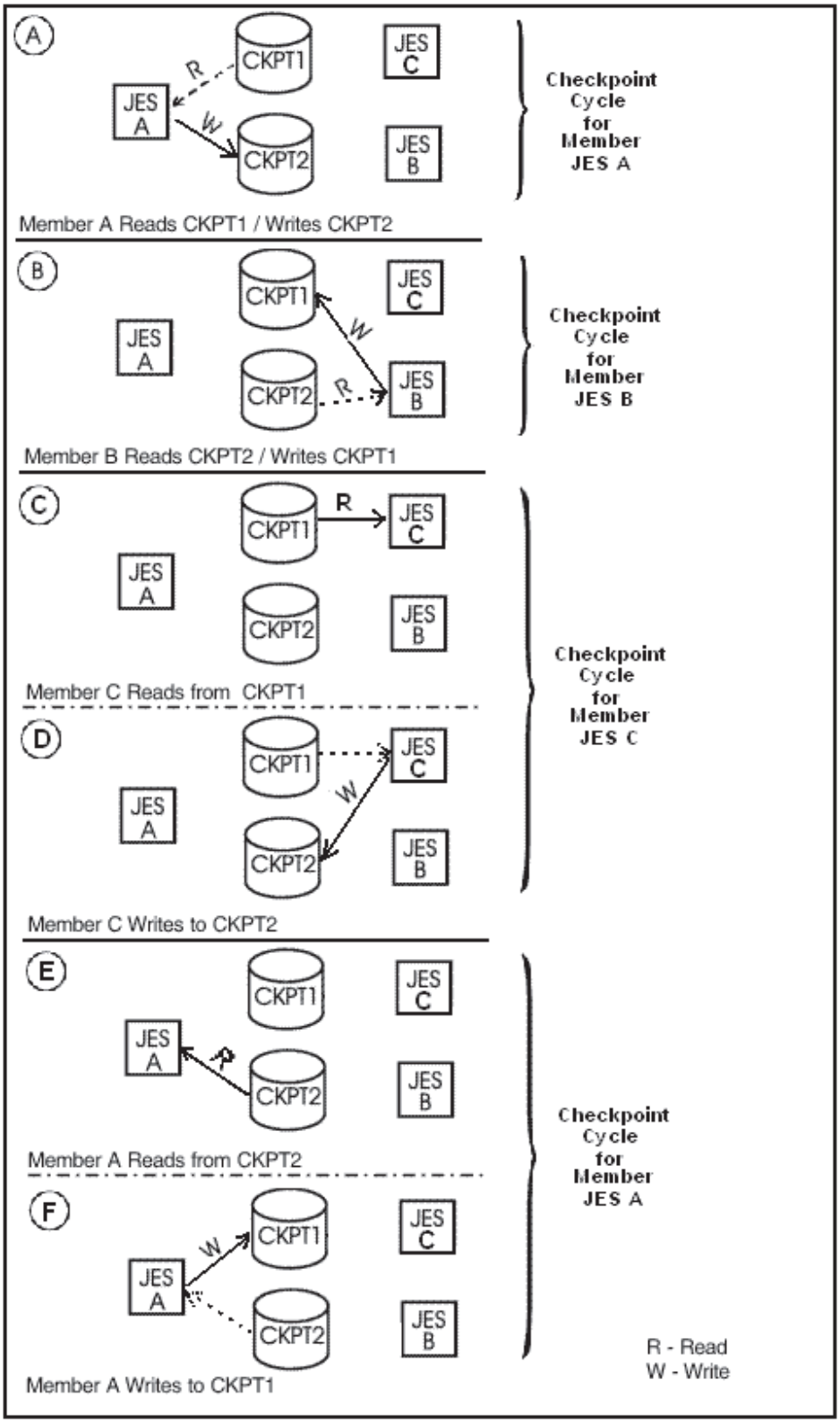


Figure 53. Checkpoint data set processing–DUAL mode–three-member multi-access spool configuration

Accessing the CKPTn data set in a MAS

There are two basic methods of operating a multi-access spool (MAS) configuration to provide individual members access to the checkpoint data set: a controlled environment and a contention-driven environment. This section discusses how you can control your multi-access spool configuration to prevent performance problems caused by the need to access the checkpoint.

Controlled checkpoint data set access

The recommended method of checkpoint access in a multi-access spool environment is a "controlled" environment where each member is allowed limited and regulated data set access. Use the queue control initialization parameters, HOLD= and DORMANCY=, on the MASDEF initialization statement to control how individual members contend for checkpoint data set control and the length of time each phase takes to complete.

The objective of setting the queue control parameters, HOLD= and DORMANCY= on the MASDEF initialization statement, is to balance access to the checkpoint data set based on the workload of each member in a multi-access spool (MAS) configuration. These parameters allow each member adequate access to the shared job and output queues (that is, the checkpoint data set).

HOLD= parameter on the MASDEF statement

The HOLD= parameter specifies the minimum time interval that a member will hold the checkpoint data set after acquiring it. If set properly, this parameter can reduce the thrashing back and forth between members that could occur in a contention-driven environment.

The actual time that the checkpoint data set is reserved for a member (and unavailable to others) is the sum of:

1. If the checkpoint data set resides on DASD, the length of time it takes to set the RESERVE.
2. Regardless of where the checkpoint data set resides, the length of time it takes to read the MASTER RECORD, read the change log (if operating in DUAL mode) and changed 4K records.
3. The value specified on HOLD=.
4. The length of time it takes to write the changed records to the checkpoint data set.

Setting HOLD= too high needlessly locks out other members in the configuration. Setting it too low causes more delays in local JES2 processing and requires more frequent access (and overhead). A good initial value is 50 (0.5 seconds).

This parameter defaults to 99999999 (11+ days), which should be used only in a single-member environment. If you take this default in a multi-access spool configuration consisting of two or more JES2 members, you will effectively prevent other MAS members from accessing the checkpoint.

DORMANCY=(mmmm,nnnn) parameter on the MASDEF statement

The DORMANCY= parameter specifies both the minimum time interval (mmmm) that a member must wait, after releasing the checkpoint data set, before attempting to regain control of it and the maximum time interval (nnnn) that this member is allowed to wait before attempting to acquire the checkpoint data set. This

parameter and the HOLD= parameter can prevent one member of the configuration from monopolizing the checkpoint data set by acquiring it too frequently.

DORMANCY(mmmm) specification: Setting (mmmm) too high will increase delays in JES2 processing on this member. Setting it too low will create excessive contention with other members in the configuration.

A good initial value for a particular member is the sum of:

1. All HOLD= values for all other members in the configuration.
2. The length of time it takes to read and write the checkpoint. If the checkpoint data set resides on DASD, a good initial estimate for calculating this length of time on DASD is 15 (that is 0.15 seconds per member to both read and write the checkpoint).

   If the checkpoint data set resides on a Coupling facility structure, checkpoint I/O is even quicker, because it does not have to set the hardware RESERVE or read the change log.

The default for the DORMANCY(mmmm) specification is 100 (1 second).

DORMANCY(,nnnn) specification: This parameter allows an inactive member to periodically scan the work queues to determine if any work for which it is eligible has entered the member from other members in the configuration.

Setting (nnnn) too high will cause this member to bypass opportunities for JES2 to process work. Setting it too low will create more contention with other members in the configuration.

The default, 500 (5 seconds), is a good initial value for the DORMANCY(,nnnn) specification.

Member-specific parameter recommendations

When assigning the queue control parameters (HOLD= and DORMANCY= on the MASDEF statement), be aware of the type and amount of workload on the individual members. For example, a heavy batch, TSO/E, print, or RJE processor workload requires more frequent access to the checkpoint data set than a processor in the configuration devoted mainly to non-TSO online applications.

In configurations that consist of members that are unequal in speed, it might be necessary to set parameters to favor slower members. This will prevent those slower members from becoming locked out from access to the checkpoint data set. Such parameter manipulation is less crucial when the checkpoint data set resides on a Coupling facility structure. The Coupling facility lock ensures equity by not locking out slower members, therefore, the lower (mmmm) DORMANCY= variable provides little benefit. However, if you set the maximum DORMANCY= variable too high, the Coupling facility lock that JES2 uses might not be able to detect the lockout time. The following are recommended queue control parameter settings for specific multi-access spool configurations and processing requirements.

- If running a single-processor, the following parameter specifications might be good initial values because there are no other processors waiting to gain access to the queues.

```
MASDEF HOLD=99999999,DORMANCY=(10,500)
```

- If running a member under VM, that is sharing queues with production members, the following parameter specifications might be good initial values in order to minimize the delays because of VM members holding the checkpoint.

```
MASDEF HOLD=0,DORMANCY=(200,1000)
```

- If running a member with heavy-batch, heavy-TSO, or printing (including RJE), the following parameter specifications might be good initial values:

```
MASDEF HOLD=50,DORMANCY=(100,400)
```

- If running a member with an online system that does not need JES2 services very often, the following parameter specifications might be good initial values:

```
MASDEF HOLD=40,DORMANCY=(200,1000)
```

Contention-driven checkpoint data set access

The other type of checkpoint data set access is a contention environment, in which the members compete for checkpoint data set control. If DORMANCY=(0,100) is specified, a member attempts to regain access as soon as it needs access (when a processor control element (PCE) requests access).

Some forms of contention-driven access can provide benefits. Setting a low HOLD= value (HOLD=10) and a low minimum dormancy value can benefit a response-oriented configuration where members are of equal size and strength. This type of checkpoint control (or lack of control) has the advantage of ensuring that a member is always requesting the checkpoint. As a result of the low HOLD= value, no member retains the data set for too long.

However, contention-driven access has the disadvantage of causing potential member lockout and severe member degradation in a MAS environment.

Replacement data set (NEWCKPTn) definition

You can specify replacement checkpoint data set definitions to ease recovery from checkpoint I/O errors and the subsequent loss of checkpoint data set access. All 4 checkpoint data sets (NEWCKPTn and CKPTn) must have unique names. For checkpoint data sets residing on DASD, you can have the same DSNAME= value if the VOLser= specification is different. For checkpoint data sets residing on a Coupling facility structure, the STRNAME= parameters must be different. Specify replacement checkpoint data sets on either DASD or on Coupling facility, regardless of what has been specified on the CKPT*n*= parameters. Generally, replacement data sets are prefixed with SYSn to provide a complete data set name. For consistency in the following discussion, SYS1.REPLACE1 and SYS1.REPLACE2 are used for the NEWCKPT1= and NEWCKPT2= parameter specifications.

These replacement data set definitions can also be used in non-error situations, such as scheduled hardware maintenance. In both DUPLEX and DUAL mode, NEWCKPT1 provides the replacement definition for CKPT1 and NEWCKPT2 provides the replacement definition for CKPT2.

Space allocation for NEWCKPTn data sets

The space you allocate for the NEWCKPTn data sets (both on DASD and on a Coupling facility) should be sufficient to hold all the information in its primary,

CKPTn, counterpart. Therefore, allocate this space as at least equal to that allocated for the CKPTn data sets. (For further information, see the "Syntax for checkpoint space allocation on DASD" on page 199.)

If these data sets are not defined in your initialization data, JES2 will not provide default data sets, but you can define these data sets during a checkpoint reconfiguration dialog. If defined at initialization, these data sets are **only defined** and **not allocated**.

To respecify the CKPTn data set, you must define the respective NEWCKPTn data set information using one of four methods. These methods are presented in order of recommendation.

Specify the NEWCKPTn data set during initialization

Define the replacement checkpoint data sets by adding the NEWCKPT1= and NEWCKPT2= parameters to the same CKPTDEF initialization statement that you use to define your primary (CKPT1 and CKPT2) data sets. These data set are only defined to JES2; they are not allocated until they are put into use during the checkpoint reconfiguration dialog.

For checkpoints on DASD:

```
CKPTDEF NEWCKPT1=(DSNAME=dsname,VOLSER=volser),
        NEWCKPT2=(DSNAME=dsname,VOLSER=volser),
```

For checkpoints on Coupling facility:

```
CKPTDEF NEWCKPT1=(STRNAME=strname),
        NEWCKPT2=(STRNAME=strname),
```

Specify the NEWCKPTn data set through the $T CKPTDEF command

These data sets are only defined to JES2 at some time after initialization; they are not allocated until they are put into use during the checkpoint reconfiguration dialog.

If you have not specified a value for a NEWCKPTn parameter in the initialization data set, a $D CKPTDEF,NEWCKPT1 operator command displays the following:

```
$HASP829 CKPTDEF  NEWCKPT1=(DSNAME=,VOLSER=)
```

To remove a NEWCKPTn value without specifying an alternative value, specify:

```
$T CKPTDEF,NEWCKPT1=(DSNAME=,VOLSER=) | (STRNAME=)
```

To specify DASD values for NEWCKPT*n*:

```
$T CKPTDEF,NEWCKPT1=(DSN=dsname,VOL=volser),
           NEWCKPT2=(DSN=dsname,VOL=volser)
```

To specify Coupling facility values for NEWCKPT*n*:

```
$T CKPTDEF,NEWCKPT1=(STRNAME=strname),
           NEWCKPT2=(STRNAME=strname)
```

Specify the NEWCKPTn data set during reconfiguration ($HASP271)

If these data sets have not been pre-defined (by initialization statement or by operator command), and you want to define the NEWCKPTn specifications during the dialog, the operator can specify the DSName and VOLser or the STRname= attributes by responding:

```
CKPTDEF NEWCKPTn={(DSName=,VOLser=)|(STRNAME=)}
```

in response to the

```
$HASP271 CHECKPOINT RECONFIGURATION OPTIONS
$HASP272 ENTER RESPONSE (ISSUE D R,MSG=$HASP271 FOR RELATED MSG)
```

and message pair. (The reconfiguration dialog is more fully discussed in "Checkpoint reconfiguration: An overview" on page 222.)

JES2 will dynamically allocate these data sets at this time.

Specify the NEWCKPTn data set during reconfiguration ($HASP282)

If you attempt to replace the NEWCKPTn specifications with those of the NEWCKPTn data set, and it has not been previously defined or it has become unavailable because of an I/O error, JES2 will issue $HASP282 NEWCKPTn DSN NULL and $HASP272 ENTER RESPONSE or the $HASP282 NEWCKPTn DSNAME, VOLUME, AND STRNAME ARE NULL and $HASP272 ENTER RESPONSE message pair.

If you want to define the NEWCKPTn data set at this time, you must respond:

```
CKPTDEF NEWCKPTn={(DSName=,VOLser=)|(STRNAME=)}
```

See *z/OS JES2 Messages* for a complete description of all the valid responses to the $HASP282 message.

Placement of the NEWCKPTn data sets

The volumes you assign to the two replacement NEWCKPTn data sets need not be unique, although IBM suggests that these two data sets reside on a separate volume from their primary data set counterparts. Both backup data sets can be specified on a single volume. This will generally provide sufficient protection from checkpoint data set loss; however, this configuration can cause some degree of performance degradation until the unavailable data set or volume is again available.

If the checkpoint data set resides on a Coupling facility structure, you can exercise some control over its placement through the use of preference and exclusion lists. For more information about how to specify these lists in your Coupling facility resource management (CFRM) policy, see *z/OS MVS Setting Up a Sysplex* and *z/OS Communications Server: New Function Summary*.

Using the NEWCKPTn data sets

In both error and non-error situations, the NEWCKPTn data set specifications can be used during the checkpoint reconfiguration dialog to replace the CKPTn specifications. For example, if an error occurs on CKPT1, JES2 **forwards** the CKPT1 specification to the NEWCKPT1 specification to replace the unavailable CKPT1 data set. (If the NEWCKPT1 data set has not been pre-defined, or has already been used, the operator can dynamically define it during the dialog. See "Replacement data set (NEWCKPTn) definition" on page 218 for further information.)

Forwarding is the process of replacing the CKPTn data set name and volume serial or structure name (if defined on a Coupling facility) with the specifications of its NEWCKPTn counterpart (NEWCKPT1 for CKPT1 and NEWCKPT2 for CKPT2), and the copying of the in-storage version of the checkpoint data to this newly defined CKPTn data set. The NEWCKPTn data set specification then becomes null.

The operator forwards the CKPTn specifications by selecting the appropriate option to the $HASP271 CHECKPOINT RECONFIGURATION OPTIONS and $HASP272 ENTER RESPONSE message pair. (See $HASP271 in *z/OS JES2 Messages* for these options and an explanation of each.) When the forwarding process is complete, the other members will automatically begin using this newly defined CKPTn data set.

For an example of the forwarding process, see Figure 54 on page 222 which uses the following checkpoint data set definition:

Note that this example applies equally to checkpoints residing on a Coupling facility structure, except that the STRname= subparameter replaces the DSN= and the VOL= specifications in the following examples.

A Your configuration is running in DUAL mode (that is MODE=DUAL) and your initialization data set has defined a backup data set for both primary data sets. These are specified as SYS1.REPLACE1 and SYS1.REPLACE2 both on volume CHECK3.

```
CKPTDEF    CKPT1=(DSN=SYS1.JESCKPT1,VOL=CHECK1),
           CKPT2=(DSN=SYS1.JESCKPT2,VOL=CHECK2),
         NEWCKPT1=(DSN=SYS1.REPLACE1,VOL=CHECK3),
         NEWCKPT2=(DSN=SYS1.REPLACE2,VOL=CHECK3)
```

B Assume an I/O error occurs to the CHECK2 volume while reading the CKPT2 data set. Data set SYS1.JESCKPT2 on volume CHECK2 becomes unavailable. Effectively, it no longer exists.

C You can **forward** CKPT2 from the unavailable data set whose name is SYS1.JESCKPT2 on volume CHECK2 to SYS1.REPLACE2 on volume CHECK3 because you have previously defined NEWCKPT2 in your initialization data set. Remember, although you have not altered your initialization data set (it remains as shown in the example) you have altered the data set specifications to the following:

```
CKPT1=   (DSN=SYS1.JESCKPT1,VOL=CHECK1)
CKPT2=   (DSN=SYS1.REPLACE2,VOL=CHECK3)
NEWCKPT1=(DSN=SYS1.REPLACE1,VOL=CHECK3)
NEWCKPT2=(DSN=,VOL=)
```

Note: Following a JES2 restart, the checkpoint data set definitions, as defined in your initialization data set, are read. If the checkpoint data sets have been forwarded, and you have not updated your initialization data set to reflect the new data set definitions, JES2 will use the definitions to provide a pointer to the new data set definitions. However, If you have updated your initialization data set to reflect the new data set definitions, JES2 will use these definitions without doing any forwarding.

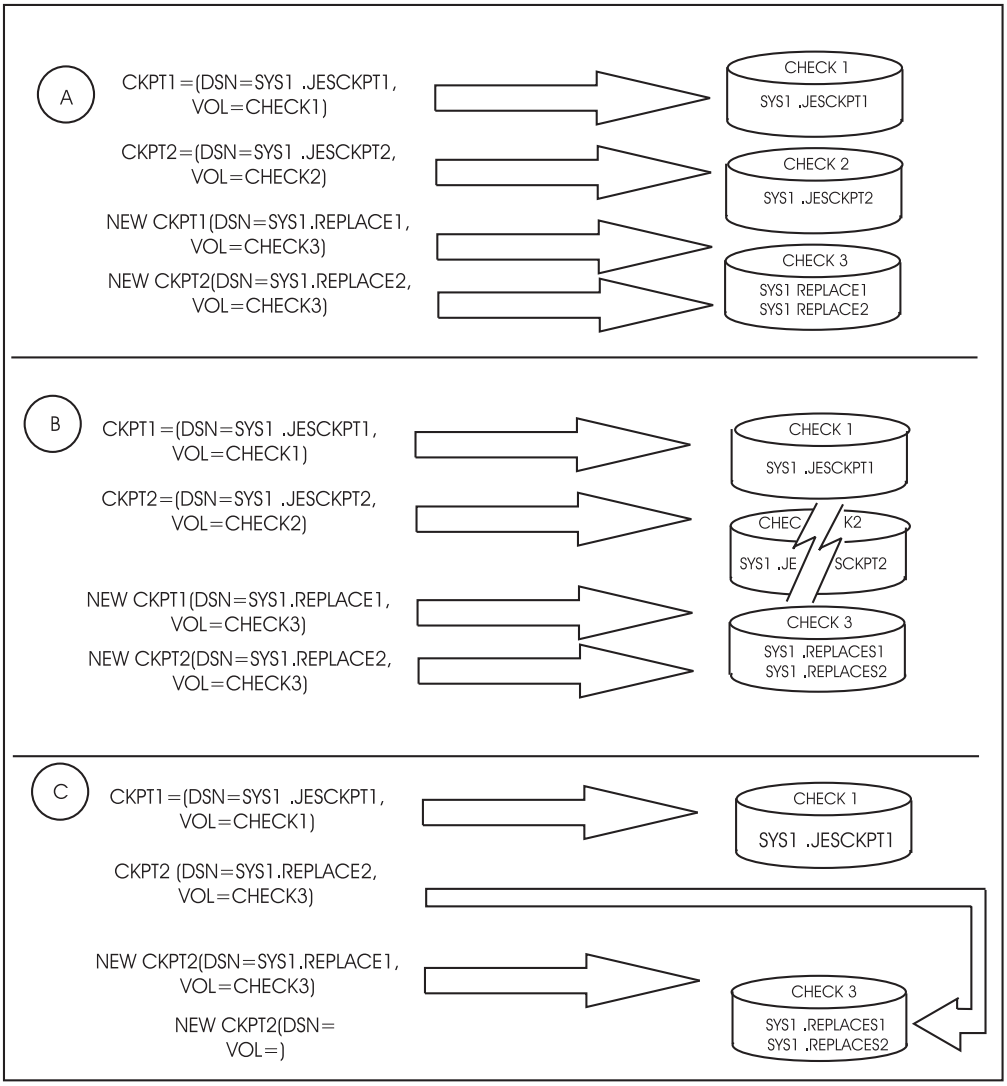


Figure 54. Example of checkpoint data set forwarding

Note that CKPT2 is now provided with a new data set name and volume specification (the specification provided by forwarding it to the NEWCKPT2 specification). The NEWCKPT2 specification is now null.

Checkpoint reconfiguration: An overview

The checkpoint reconfiguration facility allows you to dynamically redefine the checkpoint data set definitions for your JES2 multi-access spool (MAS) configuration. Whether the data sets reside on a DASD volume or on a coupling facility structure, the reconfiguration, through its operator dialog allows you to:

- Replace a checkpoint data set.
- Discontinue the use of a checkpoint data set.
- Resume using a previously suspended checkpoint data set.
- Terminate JES2 during the dialog.
- Start using a new checkpoint data set.

CAUTION:
JES2 does not support moving the checkpoint data set from one volume serial to another unless the checkpoint dialog is used.

Additionally, you can either enter a checkpoint reconfiguration dialog to change the DASD to any supported type or model or change the DASD while involved in a dialog initiated for other reasons. Device types used to hold the checkpoint data set can be a mix of any JES2-supported DASD (see Appendix A, "IBM devices supported by JES2 and how to use them," on page 387 for a list of these devices) or coupling facility structures.

If your checkpoint is on a coupling facility and all you want to do is move it to a different coupling facility, then you can use the SETXCF command to start a rebuild instead of using checkpoint reconfiguration. However, only checkpoint reconfiguration can move the checkpoint from a coupling facility to DASD. See "Moving a JES2 checkpoint to a coupling facility using system-managed rebuild" on page 240 for more information.

Depending on the situation, either JES2 or the operator can initiate the checkpoint reconfiguration process. The following situations initiate a checkpoint reconfiguration dialog:

- JES2-initiated:
  - During initialization processing:
    - Either JES2 could not determine the checkpoint data set specifications, or JES2 requires operator verification of the data set specifications.
    - You specified PARM=RECONFIG as a JES2 start option.
    - You specified RECONFIG=YES on the CKPTDEF initialization statement.

    For a discussion of the JES2-initiated checkpoint reconfiguration at initialization, see "Initialization" on page 228.
  - Because of I/O error:

    JES2 automatically initiates a reconfiguration when an I/O error occurs during a read or write of the checkpoint data set. For a discussion of the JES2-initiated checkpoint reconfiguration see "I/O error" on page 229.
  - Because of coupling facility structure becomes volatile:

    As result of processing based on your CKPTDEF VOLATILE= specification when a coupling facility on which a checkpoint resides becomes volatile. A coupling facility structure is considered volatile if when power to the coupling facility is lost, all data on the facility is lost. A volatile structure might not be an immediate problem, but is serious enough that JES2 prompts you to take precautions to prevent potential data loss.

    Note that a coupling facility is always considered volatile unless you have taken measures to protect against the loss of power to that coupling facility. See *ES/9000 and ES/3090 PR/SM Planning Guide* for a description of how to monitor the volatility of coupling facilities.

    For a discussion of the JES2-initiated checkpoint reconfiguration because of the coupling facility structure becoming volatile, see "Volatile coupling facility" on page 232.
- Operator-initiated:

  Depending on your installation's use of DASD or coupling facility structures or both to maintain your checkpoint data, you might need to occasionally change your checkpoint definitions. For example, you might choose to initiate a checkpoint reconfiguration because the device on which the checkpoint data

set(s) resides is scheduled for service. However if your checkpoint is on a coupling facility, it may be simpler to use system managed rebuild to move your checkpoint to another coupling facility. See "Moving a JES2 checkpoint to a coupling facility using system-managed rebuild" on page 240for more information

For a discussion of the operator-initiated checkpoint reconfiguration see "Operator-initiated entrance into a checkpoint reconfiguration dialog" on page 233.

The reconfiguration allows minimal disruption across all checkpoint data set failures, thus preventing the need for a JES2 restart.

If your checkpoint is on a coupling facility and all you want to do is move it to a different coupling facility, then you can use the SETXCF command to start a rebuild instead of using checkpoint reconfiguration only if the structure attributes remains the same. In the event that the structure size needs to be enlarged, then the system managed rebuild (SETXCF command) will be unsuccessful.

Additionally, system managed rebuild (SETXCF command) is only supported when the structure is to be moved to a different coupling facility, it cannot be used to move the checkpoint structure to another structure which resides within the same coupling facility.

The reason why JES2 does not support system managed rebuild to move the structure within the same coupling facility, or does not support resize of structure to another coupling facility is because JES2 connects (IXLCONN) to the structure with the ALLOWALTER option set to NO.

If the structure size needs to be changed, then this can only be done using the dynamic reconfiguration dialog process.

Checkpoint reconfiguration concepts

Whenever JES2 detects a situation that requires that it initiate a checkpoint reconfiguration, JES XCF, the JES common coupling services, provide the communication mechanism between the MAS members. By default, JES2 MAS members join a JES XCF group and are mapped in a one-to-one relationship with the JES XCF group members. As such, all members are linked through the JES XCF interface and JES2 is able to determine which MAS member will be considered the **driving member** for the reconfiguration. The driving member is the member that 1) has the most up-to-date version of the checkpoint data in-storage and 2) will be the member that directs (or drives) the other members through the reconfiguration process. All other members are referred to as **non-driving members**. When the reconfiguration process begins, if the driving member subsequently fails, JES2 selects the next best candidate to replace it as the new driving member. (If you require further background information on the JES common coupling services, see *z/OS MVS Programming: JES Common Coupling Services* or for recommendations on configuring your MAS in a sysplex environment see *z/OS MVS Setting Up a Sysplex*.)

Preparing for checkpoint reconfiguration

Depending on whether you maintain your checkpoint on a coupling facility structure or DASD, there are a few steps you should take to ease an unplanned checkpoint reconfiguration.

Becoming involved in a checkpoint reconfiguration can have serious consequences if you do not understand the implications of the choices JES2 presents though the dialog messages. Be certain you and your staff understand the definitions and the concepts explained in this section. (The checkpoint dialog messages and the options they present are fully explained in *z/OS JES2 Messages* as well.)

Attention: Use a non-production JES2 MAS to test your reconfiguration dialog and establish your recovery procedures. JES2 automatically enters a checkpoint reconfiguration if a checkpoint I/O error occurs.

JES2-initiated because of I/O error

You can minimize or eliminate operator involvement during the checkpoint reconfiguration process prompted by JES2 because of an I/O error by defining replacement checkpoint data set(s) in advance. Further, specify OPVERIFY=NO on the CKPTDEF initialization statement or use the $T CKPTDEF OPVERIFY=NO command to eliminate operator involvement.

During a checkpoint reconfiguration because of an I/O error, JES2 attempts to allocate and use a replacement checkpoint data set. If you do not predefine a replacement checkpoint data set, JES2 prompts the operator for the DASD definition (name and volume serial number) or coupling facility structure name of a suitable replacement. Several prompt messages (for example, $HASP273 and $HASP282) allow you to tell JES2 to suspend using a particular data set, modify CKPTDEF initialization statement options, cancel the reconfiguration, or terminate JES2. Use the NEWCKPT1= and NEWCKPT2= options on the $T CKPTDEF command (outside a checkpoint reconfiguration) to predefine replacement checkpoint data sets.

Predefining replacement checkpoint data sets

You can minimize the chance of errors and operator involvement during a checkpoint reconfiguration by defining and allocating replacement checkpoint data sets or structure definitions. Predefining replacement checkpoint data sets, however, does not guarantee that JES2 will be able to successfully allocate them during the checkpoint reconfiguration process.

JES2 can specify a replacement checkpoint data set on DASD. This procedure requires that you minimally perform step 1; step 2 is optional.

1. You can specify your backup checkpoint data set (NEWCKPTn) either on the CKPTDEF initialization statement or by operator command `($T CKPTDEF,NEWCKPTn=(DSN=...,VOL=...))` .

   Using this definition, JES2 can dynamically allocate the space for this data set if this data set did not previously exist on the volume.
2. To ensure that the space for the data set is available if needed, pre-allocate the space on the volumes for your replacement checkpoint data sets. See "Replacement data set (NEWCKPTn) definition" on page 218 for information about defining replacement checkpoint data sets. See "Syntax for checkpoint space allocation on DASD" on page 199 for information about how to determine the space requirements for checkpoint data sets.

If you maintain your checkpoint on a coupling facility structure, be certain to have previously defined the structure for NEWCKPTn=(STRNAME=) in a coupling facility resource management (CFRM) policy. JES2 cannot dynamically add this name to the CFRM policy. Therefore, if you expect JES2 to forward a checkpoint to a new structure, and you have failed to define the structure in the CFRM policy,

JES2 will prompt the operator for a valid checkpoint specification. See *z/OS MVS Setting Up a Sysplex* for information on defining coupling facility structure and a CFRM policy.

Using the checkpoint reconfiguration dialog

As stated earlier, either JES2 or the operator can initiate the reconfiguration process. This section provides some important concepts you should understand before entering a checkpoint reconfiguration dialog. It presents some considerations for checkpoint message routing, how you can determine why JES2 entered a particular type checkpoint reconfiguration, and some very important differences for the dialog responses of *CANCEL* and *TERM*.

Message routing during a checkpoint reconfiguration

The reason JES2 entered a checkpoint reconfiguration, the message route code, and your installation's specification of the MVS parameter MSCOPE= on the MVS CONSOLxx member of SYS1.PARMLIB all determine how the checkpoint reconfiguration messages are routed.

Based on reconfiguration entrance type, messages are directed as follows:

- Operator initiated - the messages are directed to the console that requested the dialog

  Therefore, if the MVS system to which this console is attached subsequently fails, the checkpoint reconfiguration messages might not be displayed on any console. To prevent this situation, when planning your sysplex console configuration consider the systems to which your consoles are attached. See *z/OS MVS Initialization and Tuning Reference* for a description of the GROUP and NAMES parameters on CNGRPxx and the ALTGRP, MSCOPE, and UD parameters on CONSOLxx. Also see *z/OS MVS Planning: Operations* for recommendations on how to define your consoles to be certain messages are routed as expected in a sysplex environment.
- JES2 initiated - the messages are directed to consoles based on their route code(s)

  The console on which reconfiguration messages appear is influenced by which member drives the checkpoint reconfiguration and your MSCOPE= specification on the CONSOLxx parmlib member. Also, if during a checkpoint reconfiguration, the driving member fails, the messages issued by the new driving member might appear on a different set of consoles based on MSCOPE= specifications.

  Depending on your MSCOPE= specifications for your consoles, exception messages issued by non-driving members, such as $HASP424, might not appear on the same console on which $HASP235 and $HASP272 messages are displayed.

Messages are directed to console by route code or console ID. Additionally, some messages are also issued only to the HDRCPY log. Those issued to the HDRCPY log are useful for diagnostic purposes and allow you to follow the progress of the reconfiguration on each member during a checkpoint reconfiguration. For example, all JES2 members within the MAS issue $HASP285, $HASP233, and $HASP255 to HDRCPY.

Determining the reason for a checkpoint reconfiguration

Multiple members can enter a checkpoint reconfiguration concurrently for any number of reasons. For example, you might request an operator-initiated

reconfiguration at the same time an I/O occurs. If so, JES2 determines which reason takes precedence in the following order:

1. I/O error on CKPT1
2. I/O error on CKPT2
3. Coupling facility on which the checkpoint resides becomes volatile
4. Operator request.

Message `$HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS` *`reconfiguration-reason`* informs you for which problem JES2 entered the reconfiguration.

If an I/O occurs on CKPT1 and CKPT2 on separate members at the same time, JES2 enters the reconfiguration for the I/O error on CKPT1. When that reconfiguration is complete, JES2 then enters a subsequent reconfiguration if the problem with CKPT2 still exists. JES2, however, does not enter subsequent reconfigurations if a concurrent coupling facility became volatile or an operator requested a dialog.

Dialog options: CANCEL and TERM

Several WTOR messages throughout the dialog give an option of 'CANCEL' or 'TERM'. The two options, CANCEL and TERM are quite different in their meaning, and the two are never presented as options at the same time.

- TERM, an option within I/O error checkpoint dialogs

  If selected, TERM indicates to JES2 to terminate (that is, force abnormal termination) of **all** members that experienced an I/O error on the CKPTn being processed by this checkpoint reconfiguration as indicated by $HASP233. The remaining members continue processing, but terminated members require a restart.

  $HASP233 provides which checkpoint (CKPT1 or CKPT2) experienced the I/O error; it also provides the number of members affected. See "Determining the reason for a checkpoint reconfiguration" on page 226 for a discussion of the precedence of reconfiguration types if multiple checkpoint problems occur. $HASP275 provides an individual message for each affected member; it also provides a $Knn reason code specific to the member that issued the message. If you reply TERM in response to $HASP272, this reason code becomes the $Knn error code reflected in the $HASP095 message when the member abnormally terminates. To verify which members received the $HASP275 message and determine the members that will be affected (terminated) by a reply of TERM, issue `D R,L,CN=(ALL),MSG=$HASP275` (This MVS command assumes that CONDEF CONCHAR=$ on all MAS members.)

  If you need to terminate a single JES2 member while within a reconfiguration, you need to complete the following 2-step process:

  1. Enter the $P JES2,ABEND command

     JES2 responds with:

     ```
     $HASP697 COMMAND PURGED--RECONFIGURATION DIALOG IN PROGRESS
     ```

  2. Enter the `$P JES2,ABEND,FORCE` command.

  TERM is more serious and always causes all member(s) experiencing the I/O on CKPTn to abend. IBM does not recommend its use because the remaining member(s) might subsequently reenter a checkpoint reconfiguration because of the same error situation. If the error situation is temporary, you should forward the checkpoint (options '1' or '2') or suspend using the checkpoint (options '5' or '6') and immediately reenter a new reconfiguration dialog to resume (options '7' or '8') using the suspended checkpoint.

- TERM, an option within JES2-initiated dialogs for initialization

  A response of TERM to the $HASP289 and $HASP272 message pair while within a JES-initiated dialog for initialization affects (terminates) the initializing member only. (A JES2-initiated initialization dialog has a scope of the single member, it never affects other MAS members.)
- CANCEL, an option for other reconfiguration dialog types.

  CANCEL provides the operator a means to easily end the dialog and allow processing to continue as before. If you replied CKPTDEF to change CKPTn or NEWCKPTn and then CANCEL a reconfiguration, those changes are lost.

Attention: Although TERM and CANCEL are never presented simultaneously as options on the same message, if you inadvertently specify either TERM or CANCEL when that option is **not** presented, JES2 will treat that response as a synonym for the presented option. For example, if $HASP273 offers an option of TERM and you reply CANCEL in response to $HASP272, JES2 **terminates** all JES2 members that experienced an I/O error; JES2 does not end the dialog directly.

JES2-initiated entrance into the dialog

JES2 will enter a checkpoint reconfiguration for any of these reasons:

- To complete initialization processing
- To correct an I/O error to the checkpoint data set
- To move the checkpoint off a volatile coupling facility structure.

Initialization

JES2 can enter the checkpoint reconfiguration dialog during initialization processing for the following reasons:

- JES2 could not determine, or required verification of, the checkpoint data set specifications
- You specified PARM=RECONFIG as a JES2 start option
- You specified RECONFIG=YES on the CKPTDEF initialization statement.

JES2 automatically enters a dialog either because of an error condition (as noted in the first instance) or because you have requested it to do so (as noted in the second and third instances).

A checkpoint reconfiguration entered at initialization is limited to the initializing member only; it is not a multi-system reconfiguration as are all other reconfiguration types. As such, the reasons for entrance can be unique. For example, consider the following scenario:

1. An I/O error occurred and CKPTn was forwarded at that time
2. The I/O error prevented JES2 from writing the forwarding information into the data set in error or possibly from reading it during this JES2 initialization. See “Using the NEWCKPTn data sets” on page 220 for a discussion of JES2 checkpoint data set forwarding.
3. You did not update the initialization data set to reflect the new checkpoint data set definition(s)
4. When initializing, JES2 will prompt the operator for the CKPTn definition(s) currently being used by the other MAS members.

Note: After completing all interactions with the operator, JES2 will verify the last written time stamp of the data set it will read the checkpoint from. The checkpoint is read from either the CKPTn specified on the PARM= on JES2 start or as determined by JES2 processing. This verification is an attempt to verify that JES2 is

reading from the correct checkpoint data set. Ultimately it is the installations' responsibility to ensure that the correct data sets are used to start JES2. However, some common problems are detected by JES2. Normally, the $HASP478 message is not highlighted and is issued for informational purposes only. However, if JES2 detects a potential problem with the last written times of the checkpoint data set(s), then the message is highlighted and followed by the $HASP441 WTOR. When this happens, the installation should verify that the time stamps associated with the data set being matched with the time the MAS was last active and would have last written the checkpoint. If not, then reply 'N' to the $HASP441 message to terminate in response to $HASP478.

Example reconfiguration dialog at initialization: The following example illustrates a typical console sequence when the operator initiates a checkpoint reconfiguration dialog by specifying RECONFIG on the start JES2 command. The system configuration is based on the following checkpoint definition.

```
CKPTDEF CKPT1=(DSN=SYS1.JES2CKPT1,VOL=CHECK1,INUSE=YES),
        CKPT2=(DSN=SYS1.JES2CKPT2,VOL=CHECK2,INUSE=YES),
        NEWCKPT1=(DSN=SYS1.JES2REP1,VOL=CHECK3),
        NEWCKPT2=(DSN=SYS1.JES2REP2,VOL=CHECK4),
        MODE=DUAL,OPVERIFY=YES
```

Assume you need to restart JES2 to change the MODE= specification from DUAL to DUPLEX. This change requires an all-member warm start. Restart JES2 by entering: *S JES2,PARM='WARM,RECONFIG'...* in order that JES2 will enter a reconfiguration dialog to allow you to make the change. You then receive the following series of messages:

1. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - JES2 INITIALIZATION
2. $HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED... VALID RESPONSES ARE...
3. $HASP272 ENTER RESPONSE
4. Operator response: R nn,CKPTDEF MODE=DUPLEX
5. $HASP829 CKPTDEF ...,MODE=DUPLEX,...
6. $HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED... VALID RESPONSES ARE...
7. $HASP272 ENTER RESPONSE
8. Operator response: R nn,END

Attention: If, on a JES2 restart, you specify RECONFIG as an option in response to the $HASP426 SPECIFY OPTIONS message or you have previously added RECONFIG to the EXEC statement of the JES2 PROC, JES2 will ignore all prior checkpoint data set forwarding performed by previous checkpoint reconfigurations. JES2 will override all forwarding and use the checkpoint data set definitions as defined in your initialization data set. (See "Using the NEWCKPTn data sets" on page 220 for a discussion of JES2 checkpoint data set forwarding.) If this is inappropriate to your restart situation, be certain to have either redefined your checkpoint data sets in your initialization data set or **do not** use the RECONFIG option. See "Specifying the Start Options" on page 28 for information on specifying JES2 start options and *z/OS MVS JCL Reference* for specifying the JES2 PROC statement.

I/O error

If an I/O error on a checkpoint data set occurs, JES2 enters checkpoint reconfiguration processing automatically. An I/O error can occur during either a

read or write of the checkpoint data set on any member. Unless a concurrent member failure occurs, JES2 automatically initiates the checkpoint reconfiguration dialog on all members in your MAS.

When JES2 initiates a checkpoint reconfiguration, the degree of operator involvement depends on whether:

- Operator verification is not required (OPVERIFY=NO on CKPTDEF)
- You have predefined a replacement checkpoint data set (NEWCKPTn= on CKPTDEF)
- An error occurs during the reconfiguration. (JES2 normally performs reconfiguration processing without operator involvement if the preceding two conditions are met. However, if JES2 encounters an error when attempting to complete an automatic reconfiguration, JES2 initiates an operator dialog to complete reconfiguration processing. Therefore, be certain to have a knowledge of the dialog procedure and have a data center procedure in place to handle such possibilities.)

Minimizing operator involvement: If you suppress operator verification **and** you have predefined a replacement checkpoint data set, JES2 automatically forwards the checkpoint data set that experienced the I/O error. To suppress operator verification, specify OPVERIFY=NO on the CKPTDEF initialization statement or on a $T CKPTDEF command. Automatic forwarding allows you to minimize disruption to your production systems. JES2 follows your predefined instructions and forwards the checkpoint data set as specified on the CKPTDEF statement. You are not prompted to change those instructions at reconfiguration time, thereby reducing the time required to complete checkpoint reconfiguration.

If you suppress operator verification (that is, if you specify OPVERIFY=NO) and NEWCKPTn= has been specified, JES2 takes the following default reconfiguration actions:

- Rather than issuing $HASP273, JES2 replaces the checkpoint data set with its replacement data set defined on the NEWCKPTn parameter of the CKPTDEF statement.
- Rather than issuing message $HASP278 (with the CREATE option), when JES2 cannot locate the specified replacement checkpoint data set, JES2 automatically creates one on the previously specified DASD volume, or JES2 can connect to a new coupling facility structure, but **only** if your coupling facility resource management (CFRM) policy has defined the structure.

  Note: If however, OPVERIFY=YES and a CFRM policy does exist, you will be prompted to define a replacement checkpoint at this time.

Requesting operator involvement: You can request that JES2 prompt the operator during the checkpoint reconfiguration process by specifying OPVERIFY=YES on the CKPTDEF initialization statement or on a $T CKPTDEF command. With operator verification enabled, the operator can:

- Accept the new checkpoint configuration. - Reply CONT
- Terminate all JES2 members that encountered I/O errors on the CKPTn being processed. See “Dialog options: CANCEL and TERM” on page 227 for a description of this procedure. - reply TERM
- Discontinue using the specified data set. - Reply DELETE

  This option (DELETE) is only available if the reconfiguration is due to an I/O error and both CKPT1 and CKPT2 are being used.

- Specify a different replacement checkpoint data set. - Reply CKPTDEF (with operands)

If you enable operator verification, JES2 prompts the operator during the checkpoint reconfiguration dialog with several messages, the most likely message pairs to be issued and requiring a response are:

```
$HASP273 JES2 CKPTn DATA SET WILL BE ASSIGNED TO NEWCKPTn
         STRNAME strname | dsname ON VOLUME volser
         VALID RESPONSES ARE:
⋮
$HASP272 ENTER RESPONSE

                                  -or-

$HASP278 UNABLE TO LOCATE OR UNABLE TO USE CKPTn|NEWCKPTn
         STRNAME strname | dsname ON VOLUME volser
         DOES NOT EXIST OR IS NOT USABLE - REFER TO PREVIOUS MESSAGE(S)
         VALID RESPONSES ARE:
⋮
$HASP272 ENTER RESPONSE
```

Message $HASP273 is used to request operator validation of the reconfiguration action JES2 is about to take. It is only issued if OPVERIFY=YES or JES2 requires operator involvement because of an unexpected processing condition. Message $HASP278 indicates that you have not preallocated a backup checkpoint data set or, if allocated, it was specified in error. Message $HASP273 can be eliminated by being certain you have specified valid NEWCKPTn data set(s) or structures.

Example reconfiguration dialog for I/O error: The following example illustrates a typical console sequence when JES2 initiates a checkpoint reconfiguration dialog with operator verification enabled. The system configuration is based on the following checkpoint definition:

```
CKPTDEF CKPT1=(DSN=SYS1.JES2CKPT1,VOL=CHECK1,INUSE=YES),
        CKPT2=(DSN=SYS1.JES2CKPT2,VOL=CHECK2,INUSE=YES),
        NEWCKPT1=(DSN=SYS1.REPLACE1,VOL=CHECK3),
        NEWCKPT2=(DSN=SYS1.REPLACE2,VOL=CHECK3),
        MODE=DUPLEX,OPVERIFY=YES
```

Assume an I/O error occurs on CKPT2 on member SY1

1. $HASP275 CKPT2 I/O ERROR - Knn
2. $HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS CKPT2 I/O ERROR(S) ON 1 MEMBER(S)
3. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY MEMBER SY1
4. $HASP273 JES2 CKPT2 DATA SET WILL BE ASSIGNED TO ...
5. $HASP272 ENTER RESPONSE
6. Operator response: R nn,CONT
7. $HASP280 JES2 CKPT*n* DATA SET...
8. $HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE
9. $HASP256 FUTURE AUTOMATIC FORWARDING OF CKPT2 IS SUSPENDED UNTIL NEWCKPT2 IS RESPECIFIED. ISSUE $T CKPTDEF,NEWCKPT2=(...) TO RESPECIFY

   In this example, $HASP256 is a highlighted message and remains on the screen until you issue $T CKPTDEF,NEWCKPT2=(...).

If you alternatively specified OPVERIFY=NO on the CKPTDEF statement, message pair $HASP273 and $HASP272 is not presented to the operator. All messages issued in a JES2-initiated automated checkpoint reconfiguration are informational and none require response.

If an automatic checkpoint reconfiguration takes place while the operator is away from the console, on return, the only message displayed will be $HASP256. Therefore, you can automate a reconfiguration by automating the response to this message only. To do so, establish a pool of replacement checkpoint data sets from which an automated $T CKPTDEF,NEWCKPTn=(...) reply in response to $HASP256 can choose a replacement.

Operator intervention when errors occur during a reconfiguration: JES2 requires operator intervention during a JES2-initiated dialog, if any of the following conditions are true:

- You have not predefined a replacement checkpoint data set (NEWCKPTn=)

  -- or -- You have not predefined a replacement structure name defined to a CFRM policy and specified a structure name on NEWCKPTn.
- JES2 cannot allocate the specified replacement checkpoint data set.
- The operator mistakenly specifies an unusable replacement data set.

Volatile coupling facility

JES2 enters a checkpoint reconfiguration if the coupling facility structure on which the checkpoint data set resides becomes volatile and subsequent CKPTDEF VOLATILE= processing determines a checkpoint reconfiguration is required. Note that a coupling facility is always considered volatile unless you have taken measures to protect against the loss of power to that coupling facility. See *ES/9000 and ES/3090 PR/SM Planning Guide* for a description of how to monitor the volatility of coupling facilities.

When forwarding a checkpoint data set to move it from a volatile coupling facility, you could inadvertently forward it to another volatile coupling facility. JES2 issues $HASP237 to inform you this is the case, and you are prompted to CONT (continue), CANCEL, or define an alternate checkpoint (structure or DASD data set). If you decide to forward your checkpoint to a volatile coupling facility (that is, you respond CONT to message pair $HASP237 / $HASP272), JES2 accepts that response and does not provide any further CKPTDEF VOLATILE= processing for this structure while the structure remains volatile.

Example reconfiguration dialog for volatile coupling facility: The following example illustrates a typical console sequence when JES2 initiates the checkpoint reconfiguration dialog because the coupling facility on which the checkpoint resides has become volatile. The system configuration is based on the following checkpoint definition:

```
CKPTDEF CKPT1=(STRNAME=SYS1_CKPT1,INUSE=YES),
        CKPT2=(STRNAME=SYS1_CKPT2,INUSE=YES),
        NEWCKPT1=(STRNAME=SYS1_NEWCKPT1),
        NEWCKPT2=(STRNAME=SYS1_NEWCKPT2),
        MODE=DUPLEX,
        DUPLEX=YES,
        OPVERIFY=NO,
        VOLATILE=(ALLCKPT=DIALOG,ONECKPT=DIALOG)
```

Assume the coupling facility on which the structure SYS1_CKPT1 resides becomes volatile. Messages are directed by route code. See “Message routing during a checkpoint reconfiguration” on page 226 for a more complete discussion.

1. $HASP233 REASON FOR JES2 RECONFIGURATION CHECKPOINT IS COUPLING FACILITY HAS BECOME VOLATILE
2. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY MEMBER SY8
3. $HASP271 CHECKPOINT RECONFIGURATIONS OPTIONS...VALID RESPONSES ARE...
4. $HASP272 ENTER RESPONSE

   If the operator did not notice the message that indicated which checkpoint(s) is volatile, the $D CKPTDEF command is useful to display that information. (A reply of CKPTDEF provides a display of modifiable parameters only and therefore cannot provide this information.)
5. Operator response: R nn,2
6. $HASP273 JES2 CKPT2 DATA SET WILL BE ASSIGNED TO NEWCKPT2...VALID RESPONSES ARE ...
7. $HASP272 ENTER RESPONSE
8. Operator response: R nn,CONT
9. $HASP256 FUTURE AUTOMATIC FORWARDING OF CKPT2 IS SUSPENDED UNTIL NEWCKPT2 IS RESPECIFIED. ISSUE $T CKPTDEF,NEWCKPT2=(...) TO RESPECIFY

   In this example, $HASP256 is a highlighted message and remains on the screen until you issue $T CKPTDEF,NEWCKPT2=(...) in step 11
10. $HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE
11. enter: $T CKPTDEF,NEWCKPT2=(...)

Operator-initiated entrance into a checkpoint reconfiguration dialog

The operator can initiate a checkpoint reconfiguration dialog; this is generally done to:

- Replace a checkpoint data set
- Update and start (resume) using a checkpoint data set
- Discontinue using a checkpoint data set.

You can enter a reconfiguration dialog at any point in JES2 processing by entering:

```
$T CKPTDEF,RECONFIG=YES
```

Note: Several messages throughout the dialog give the option of 'CANCEL'. If the operator decides to end the reconfiguration process on a member, a response of 'CANCEL' will cause JES2 to exit from the dialog. If you replied CKPTDEF to change CKPTn or NEWCKPTn during the reconfiguration, those changes are lost.

Initiating a reconfiguration dialog on any member causes JES2 to automatically initiate the reconfiguration process on all other active members of the MAS configuration. The checkpoint reconfiguration dialog messages, which are used to control JES2-operator interactions, are sent to the console on which the operator initiates the dialog. Starting in z/OS V1R7, these messages are also sent to the consoles with routing code (ROUTCODE) 42.

Moving a checkpoint data set

If the checkpoint needs to be moved to another location, it can be done in two ways:

- System managed rebuild (SETXCF command) if the structure attributes are the same and it is moved to a different coupling facility. Process should be as follows:

  ```
  SETXCF START,REBUILD,STRNAME=xxxxx
  or
  SETXCF START,REBUILD,CFNAME=xxxxx
  ```

  This process can be very useful when maintenance needs to be done on the coupling facility, or it needs to be powered down.
- Use the JES2 dynamic reconfiguration dialog and move the structure to DASD or a different structure.

If the checkpoint size needs to be increased, it can only be done using the dynamic reconfiguration dialog. Again, two possibilities:

- Move the checkpoint using the dialog to another larger structure or data set.
- If the checkpoint needs to return to the original structure: invoke the dialog to move the checkpoint to another structure or DASD.

When complete, a D XCF of the original structure will show it to be allocated with no connection.

Activate the new policy which will be in a policy change pending status.

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

Issue the rebuild to enlarge the original structure:

```
SETXCF START,REBUILD,STRNAME=xxxxx
```

When complete, the reconfiguration dialog must be invoked to move the checkpoint back into the original structure.

Replacing a checkpoint data set

Replacing a checkpoint data set is a relatively simple procedure. Whether you have previously defined the corresponding replacement data set will determine how this can be done during the reconfiguration dialog.

If you have previously specified the corresponding replacement data set, NEWCKPTn={(DSName=,VOLser=)|(STRNAME=)}, either on the:

- CKPTDEF initialization statement (only during a cold start) or
- $T CKPTDEF command.

and it has not already been used, you can copy the appropriate in-storage checkpoint (CKPT1 or CKPT2) to the corresponding NEWCKPTn data set. Respond to the $HASP271 CHECKPOINT RECONFIGURATION OPTIONS and $HASP272 ENTER RESPONSE message pair with the appropriate “FORWARD” option (that is, a reply of ‘1’ or ‘2’).

The time you and JES2 are involved in a checkpoint reconfiguration dialog is time that the checkpoint is unavailable to the JES2 members. It is important to minimize this length of time if possible and understand how to best consolidate $T commands. Use a single *$T CKPTDEF,NEWCKPTn=(...),RECONFIG=YES* outside a reconfiguration to speed reconfiguration processing. Specifying NEWCKPTn and RECONFIG=YES on the same command eliminates any ambiguity to JES2 whenever multiple NEWCKPTn specifications are entered on different JES2 members within the same checkpoint cycle.

If you have not previously defined the corresponding replacement data set, or it has already been used, you can define it by responding CKPTDEF NEWCKPTn={(DSN=dsname,VOL=volser)|(STRNAME=strname)} to the $HASP271 CHECKPOINT RECONFIGURATION OPTIONS and $HASP272 ENTER RESPONSE message pair. JES2 will then reissue messages $HASP271 and $HASP272 so that you can copy the in-storage checkpoint as described previously.

Discontinue using a checkpoint data set

If you want to discontinue using a checkpoint data set, respond to the $HASP271 CHECKPOINT RECONFIGURATION OPTIONS and $HASP272 ENTER RESPONSE message pair with the appropriate "SUSPEND" option (that is, a reply of '5' or '6').

Note: If your MAS is running in DUPLEX mode and one of more members specified DUPLEX=OFF, you can still specify discontinue using CKPT1. JES2 will attempt to make CKPT2 your primary data set. After JES2 completes SUSPEND processing, DUPLEX=ON is set for every MAS member.

Example operator-initiated reconfiguration dialog

The following example illustrates a typical console sequence when an operator initiates the checkpoint reconfiguration dialog. The original checkpoint configuration as defined in the JES2 initialization data set is:

```
CKPTDEF CKPT1=(DSN=SYS1.JESCKPT1,VOL=CHECK1,INUSE=YES),
        CKPT2=(DSN=SYS1.JESCKPT2,VOL=CHECK2,INUSE=YES),
        NEWCKPT1=(DSN=SYS1.JES2REP1,VOL=CHECK3),
        NEWCKPT2=(DSN=SYS1.JES2REP2,VOL=CHECK4),
        MODE=DUAL
```

Assume that during a previous checkpoint reconfiguration, CKPT1 was forwarded to NEWCKPT1 on volume CHECK3. If you issue a $D CKPTDEF command, JES2 displays the following current checkpoint definition:

```
CKPTDEF CKPT1=(DSN=SYS1.JES2REP1,VOL=CHECK3,INUSE=YES),
        CKPT2=(DSN=SYS1.JESCKPT2,VOL=CHECK2,INUSE=YES),
        NEWCKPT1=(DSN=,VOL=),
        NEWCKPT2=(DSN=SYS1.JES2REP2,VOL=CHECK4),
        MODE=DUAL
```

To begin a checkpoint reconfiguration dialog and return CKPT1 to its original definition:

1. Enter

   ```
   $T CKPTDEF,NEWCKPT1=(DSN=SYS1.JES2CKPT1,VOL=CHECK1),RECONFIG=YES
   ```

2. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
3. $HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS OPERATOR REQUEST
4. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY MEMBER SY4
5. $HASP271 CHECKPOINT RECONFIGURATION OPTIONS...VALID RESPONSES ARE...
6. $HASP272 ENTER RESPONSE
7. Operator response: R nn,1
8. $HASP273 JES2 CKPT1 DATA SET WILL BE ASSIGNED TO NEWCKPT1 ...
9. $HASP272 ENTER RESPONSE
10. Operator response: R nn,CONT

11. $HASP256 FUTURE AUTOMATIC FORWARDING OF CKPT1 IS SUSPENDED UNTIL NEWCKPT1 IS RESPECIFIED. ISSUE $T CKPTDEF,NEWCKPT1=(...) TO RESPECIFY

    In this example, $HASP256 is a highlighted message and remains on the screen until you issue $T CKPTDEF,NEWCKPT1=(...) in step 13.
12. $HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE
13. Enter

    ```
    $T CKPTDEF,NEWCKPT1=(...)
    ```

Checkpoint reconfiguration recovery

The JES2 checkpoint reconfiguration provides its own recovery for the following:

- Driving member failure - JES2 selects a new driving member and re-synchronizes reconfiguration processing
- JES2 member failure - an unrecoverable error occurs during reconfiguration processing - JES2 prompts the operator to create an emergency copy of the checkpoint before terminating
- Initiating member(s) failed in start-up processing - JES2 cancels the reconfiguration
- Reconfiguration delayed - a member fails to respond as requested by the driving member - JES2 issues $HASP257

Driving member failure

The driving member fails and JES2 selects a new driving member. If successive members fail during a checkpoint reconfiguration, be aware that each new driver contains a less current in-storage copy of the checkpoint than the previous driver(s). Depending on how many driving members failed and how tolerant your operations are to down-level data, you might consider terminating all JES2 members and using the SYS1.EMERGNCY.HASPCKPT you created in response to $HASP279 on the original driver.

It is possible that when JES2 selects a member to be the driving member for a checkpoint reconfiguration, that driving member then fails. JES2 then selects another driving member to take its place. JES2 issues $HASP285 JES2 CHECKPOINT RECONFIGURATION RE-SYNCHRONIZING - DRIVEN BY MEMBER *member-name* to indicate that a new member has been selected as the new driving member. No operator involvement is required for the selection of the new driver. However, whenever a driving member fails, JES2 involves the operator for the remainder of the reconfiguration regardless of your OPVERIFY= specification.

Recovery from JES2 member failures

If a JES2 member experiences an unrecoverable error during the reconfiguration dialog, or if you reply 'TERM' to a dialog message while processing an I/O error, JES2 attempts to create an emergency copy of the checkpoint data set. JES2 then issues message $HASP279 SPECIFY EMERGNCY CKPT VOLUME SERIAL OR '*NONE*' to which you can specify a volume on which to place this data set. (JES2 can only create an emergency data set on DASD.) If JES2 is successful, JES2 names this data set SYS1.EMERGNCY.HASPCKPT. You then receive message $HASP274 SYS1.EMERGNCY.HASPCKPT CREATED ON VOLUME *volser* before JES2 termination. This data set can then be used when restarting JES2 on an all-member warm start.

When you respond to $HASP279 requesting that JES2 create an emergency checkpoint, IBM suggests that you minimally create an emergency checkpoint from the driving member(s).

Initiating member(s) failure in start-up processing

If JES2 cannot determine the reason for the reconfiguration, such as if a member requests a reconfiguration and then fails during reconfiguration start-up processing, JES2 will not be able to determine the reason for the request. JES2 then cancels the reconfiguration and issues $HASP255 JES2 CHECKPOINT RECONFIGURATION CANCELLED BY JES2.

If this type error occurs, just restart the failing member. Unless there was a need to reconfigure the checkpoint as indicated by further messages from other members, the problem is cleared up by the restart alone.

Member delayed

JES2 issues message

```
$HASP257 MEMBER delayed-member-name -- JES2 CHECKPOINT
RECONFIGURATION DELAYED BY MEMBER causing-member-name.
delayed-member-name IS WAITING FOR ACTION=action-code,
action-text
```

to provide an indication of which system member is experiencing problems. If this message is issued because of a temporary delay, JES2 deletes the message, and no operator action is required. However, the message might indicate a more serious problem indicating your need to fail the indicated system. Based on the set of $HASP257 messages that you receive and their explanations, you might be able to diagnose the checkpoint reconfiguration problem(s). See *z/OS JES2 Messages* for the message text that this message can issue and a detailed explanation for each.

Clean-up after a checkpoint reconfiguration

When you have completed a reconfiguration dialog or JES2 has completed a checkpoint reconfiguration, you may want to:

- Return your checkpoint definitions to those typically used within your installation.
- Restart JES2 from an emergency checkpoint.
- Make the new checkpoint data set definitions permanent.

Returning to your typical checkpoint data set definitions

If you need to return your checkpoint data set definitions to the specifications as defined in your initialization data set, enter a reconfiguration dialog on any member by using the $T CKPTDEF,NEWCKPTn=(...),RECONFIG=YES command. You can then reassign the replacement checkpoint (NEWCKPTn) data set to your original checkpoint definition and forward the primary (CKPTn) data set to the NEWCKPTn data set.

Although this reconfiguration process can be accomplished a number of different ways, IBM suggests the method used in the “Example operator-initiated reconfiguration dialog” on page 235 because it minimizes the time that the checkpoint data set is unavailable compared to other methods of redefining NEWCKPTn.

If the new configuration is sufficient and causing no system performance or availability problems, you can perform a reconfiguration at your convenience or not at all.

Restarting JES2 from an emergency checkpoint

If your checkpoint reconfiguration did not complete successfully, and you are unable to restart JES2 from your checkpoint, perform an all-member warm start using an emergency checkpoint data set. If in response to $HASP279 issued by several members, you have created several emergency checkpoints, attempt to restart JES2 with an available emergency checkpoint that you select in the following order:

1. Original driving member
2. Subsequent driving member (if a previous driving member failed)
3. Other available emergency checkpoint.

Follow this example reconfiguration dialog when warm starting JES2 at initialization. This example assumes you are capable of returning to the original checkpoint DASD specifications that follow:

```
CKPTDEF CKPT1=(DSN=SYS1.JES2CKPT1,VOL=CHECK1,INUSE=YES),
        CKPT2=(DSN=SYS1.JES2CKPT2,VOL=CHECK2,INUSE=YES),
        NEWCKPT1=(DSN=SYS1.JES2REP1,VOL=CHECK3),
        NEWCKPT2=(DSN=SYS1.JES2REP2,VOL=CHECK4),
        MODE=DUAL,OPVERIFY=YES
```

Perform an all-member warm start and enter a reconfiguration dialog as follows:

1. enter

   ```
   S JES2,PARM='WARM,RECONFIG'
   ```

   JES2 initialization processing continues:
2. `$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - JES2 INITIALIZATION`
3. `$HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED... VALID RESPONSES ARE...`
4. `$HASP272 ENTER RESPONSE`
5. Operator response:

   ```
   R nn,CKPTDEF CKPT1=(DSN=SYS1.EMERGNCY.HASPCKPT,VOL=SPOOL1,INUSE=YES),
   CKPT2=INUSE=NO
   ```
6. JES2 displays the original CKPTDEF statement with changes made in the previous step:

   ```
   $HASP829 CKPTDEF CKPT1=(DSN=SYS1.EMERGNCY.HASPCKPT,VOL=SPOOL1,INUSE=YES),
   $HASP829         CKPT2=(DSN=SYS1.JES2CKPT2,VOL=CHECK2,INUSE=NO),
   $HASP829         NEWCKPT1=(DSN=SYS1.JES2REP1,VOL=CHECK3),
   $HASP829         NEWCKPT2=(DSN=SYS1.JES2REP2,VOL=CHECK4),
   $HASP829         MODE=DUAL,OPVERIFY=YES
   ```
7. `$HASP289 CKPT1 AND/OR CKPT2 SPECIFICATIONS ARE REQUIRED... VALID RESPONSES ARE...`
8. `$HASP272 ENTER RESPONSE`
9. Operator response: `R nn,END`

At this point, you are using the emergency checkpoint only. Continue reconfiguration processing as in the following examples to achieve the required permanent checkpoint configuration.

1. Enter

   ```
   $T CKPTDEF,NEWCKPT1=(DSN=SYS1.JES2CKPT1,VOL=CHECK1),RECONFIG=YES
   ```
2. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
3. $HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS OPERATOR REQUEST

4. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY MEMBER SY5
5. $HASP271 CHECKPOINT RECONFIGURATION OPTIONS...VALID RESPONSES ARE...
6. $HASP272 ENTER RESPONSE
7. Operator response: R nn,1
8. $HASP273 JES2 CKPT1 DATA SET WILL BE ASSIGNED TO NEWCKPT1 ...
9. $HASP272 ENTER RESPONSE
10. Operator response: R nn,CONT
11. $HASP256 FUTURE AUTOMATIC FORWARDING OF CKPT2 IS SUSPENDED UNTIL NEWCKPT2 IS RESPECIFIED. ISSUE $T CKPTDEF,NEWCKPT2=(...) TO RESPECIFY

    In this example, $HASP256 is a highlighted message and remains on the screen until you issue $T CKPTDEF,NEWCKPT2=(...) in step 13.
12. $HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE
13. Enter

    ```
    $T CKPTDEF,NEWCKPT1=(DSN=SYS1.JES2REP1,VOL=CHECK3)
    ```

At this point, you have returned CKPT1 and NEWCKPT1 to their original definitions. The last task is to return CKPT2 and NEWCKPT2 to their original definitions as well.

1. Enter

   ```
   $T CKPTDEF,RECONFIG=YES
   ```
2. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
3. $HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS OPERATOR REQUEST
4. $HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY MEMBER SY3
5. $HASP271 CHECKPOINT RECONFIGURATION OPTIONS...VALID RESPONSES ARE...
6. $HASP272 ENTER RESPONSE
7. Operator response: R nn,8
8. $HASP273 JES2 CKPT2 DATA SET WILL BE ASSIGNED ...
9. $HASP272 ENTER RESPONSE
10. Operator response: R nn,CONT
11. $HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE

At this point, you have resumed using CKPT2. NEWCKPT2 is left unchanged as originally specified. We have now returned all checkpoint specifications to their original definitions.

Making the checkpoint reconfiguration definitions permanent

JES2 honors the checkpoint configuration definition across any JES2 restart and forwards data sets based on the CKPTn and NEWCKPTn definitions. However, for consistency between your initialization data set and the new checkpoint definitions, you should update your initialization data set as soon as possible.

During JES2 checkpoint reconfiguration processing, JES2 tries to add a pointer in each replaced checkpoint data set to provide a 'forwarding trail'. (See "Using the NEWCKPTn data sets" on page 220 for a discussion of JES2 checkpoint data set

forwarding.) You cannot delete this series of old checkpoint data sets until you have updated your initialization data set to point to your current checkpoint. If JES2 was unable to write the forwarding information into the old checkpoint (possibly caused by an a I/O error) you should update the CKPTDEF statement in all your initialization data sets as soon as possible. This effectively makes your current checkpoint definition permanent and eliminates the forwarding trail. Otherwise, whenever you warm start JES2, JES2 will enter a reconfiguration dialog at initialization.

Attention: If, on a JES2 restart, you specify RECONFIG as an option in response to the $HASP426 SPECIFY OPTIONS message or you have previously added RECONFIG on an S JES2 command, JES2 will ignore all checkpoint data set forwarding previously defined. JES2 will override all forwarding and use the checkpoint data set you specified from the console. If this is inappropriate to your restart situation, be certain to have either redefined your checkpoint data sets in your initialization data set or **do not** use the RECONFIG option. See "Specifying the Start Options" on page 28 for information on specifying JES2 start options and *z/OS MVS JCL Reference* for specifying the JES2 PROC statement.

Moving a JES2 checkpoint to a coupling facility using system-managed rebuild

Managing a JES2 checkpoint in a coupling facility is different from managing a JES2 checkpoint on DASD. The main differences is coupling facilities store data in temporary memory versus the permanent storage of a DASD. The data JES2 stores in a coupling facility is needed for JES2 restart. If CKPT2 is placed on DASD, it can be used by JES2 restart processing if the data in the coupling facility is lost. However, if both checkpoints are on coupling facilities and the coupling facility data is lost (for example because of an extended power outage), then JES2 must be cold started, resulting in the loss of all Jobs and SYSOUT in the JES2 work queues. Because of this, special care must be taken with any coupling facility that contains a JES2 checkpoint structure.

In general, to prevent a total loss of the job queue, you should use 2 checkpoints, specify CKPT DUPLEX=ON on all members, and place CKPT2 on a DASD. You can also use nonvolatile coupling facilities (with battery backup in case of power failure) to minimize the impact of a power failure. However, data on a coupling facility can be lost in ways other than a power failure. The most common way JES2 checkpoints structures in a coupling facility are lost is when the coupling facilities are reinitialized while JES2 structures are still active in them. To prevent this, the JES2 structures should be moved out of a coupling facility before reinitializing it. Depending on the availability of an alternate coupling facility, this can be done in one of two ways.

If there are no suitable coupling facilities available to hold the checkpoint data set, or if this is a planned extended power outage, then the checkpoint data set should be moved to a DASD. This can be done by an operator initiated checkpoint reconfiguration. See "Operator-initiated entrance into a checkpoint reconfiguration dialog" on page 233 for details on how to do this.

If there is a suitable alternative coupling facility available, then a system managed rebuild can be used to move the checkpoint structure to the new coupling facility. To accomplish this do the following:

1. Ensure system managed processes are enabled in your SYSPLEX. See "System-Managed Processes Considerations" in *z/OS MVS Setting Up a Sysplex* for a complete list of the requirements.

2. Ensure that the structure definitions for the JES2 checkpoint data sets in the active CFRM policy will direct the checkpoint data set to a suitable coupling facility. If not, update and activate the appropriate policy.
3. Use the appropriate XCF command to either rebuild only the JES2 checkpoint structure (SETXCF START,REBUILD,STRNAME=) or all structures on a coupling facility (SETXCF START,REBUILD,CFNAME=).
4. When the rebuild completes, use the D XCF,STRUCTURE,STRNAME= to display the checkpoint structure and verify it is now on a suitable structure.

During the rebuild process, you may notice that JES2 is not responding to commands and not starting new work. Other MAS members may issue messages about not being able to access the checkpoint. JES2 will return to normal operations when the rebuild completes. To minimize the time required to complete the rebuild, you can request the JES2 structures be rebuilt individually by using the SETXCF START,REBUILD,STRNAME= instead of specifying CFNAME=. Alternatively, the structures can be rebuilt when JES2 is not active.

Note: JES2 supports rebuild of its structures only for the SETXCF START,REBUILD command. In the event of any other failure of the coupling facility (such as loss of connectivity), the I/O form of checkpoint reconfiguration will be used for recovery.

Recovering from member failures on other JES2 members

In a JES2 MAS, when a JES2 member fails, recovery depends on whether the checkpoint data set(s) reside on a Coupling facility structure or on a DASD volume, and whether the failing member involves only the JES2 member or involves MVS.

Checkpoint on DASD

- If only the JES2 member fails, and it holds the lock of a checkpoint data set that resides on DASD, recovery requires releasing the software checkpoint lock to make the checkpoint data set available to other JES2 members in the MAS.
- If MVS fails (which includes the JES2 member with a checkpoint data set residing on DASD), jobs executing on that member must be requeued before another JES2 member can restart those jobs. To make jobs eligible for restart:
  - The hardware lock holding the checkpoint data set must be reset if it is held.
  - The software checkpoint lock must be reset if it is held by the failed JES2 member.
  - The jobs that were executing on the failed member must be requeued for processing.

Checkpoint on Coupling facility structure

If a JES2 member fails and the checkpoint data resides on a Coupling facility structure, the Coupling facility lock that JES2 uses to control access to the checkpoint is released automatically. However, an installation still must requeue jobs that had been executing on the failed member so that they can be selected for execution on an active member.

You can recover from a member failure through use of operator commands or, in a sysplex, through use of the AUTOEMEM option.

Using operator commands

When using commands to requeue the jobs of a failed member, an operator must first determine whether the failure involves only the JES2 member or involves the MVS system. Also, if the checkpoint data set resides on a DASD volume, the operator has to determine if checkpoint locks are held and reset them before requeuing the jobs.

JES2 failure only

If the failed member (JES2 member) holds the JES2 checkpoint data set, the operator must:

1. Reset the software checkpoint lock (if the checkpoint is on DASD) so that other JES2 members of the MAS can access it. Use the $E CKPTLOCK,HELDBY= command to reset the lock.
2. Restart the failed JES2 member through a hot start.

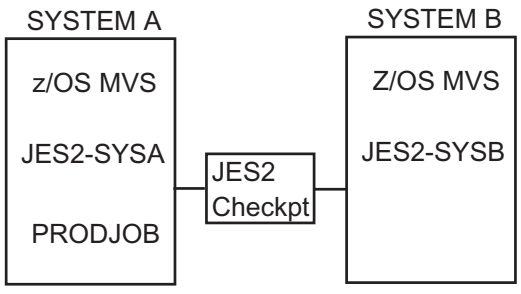
If the JES2 checkpoint data set is not held, the operator only restarts JES2 to provide JES2 services to the jobs executing under MVS.

MVS and JES2 failure

If the failed member holds the JES2 checkpoint data set, the operator must:

1. If the checkpoint resides on DASD, reset the hardware lock if held (as indicated by message $HASP263).
2. If the checkpoint resides on DASD, reset the software checkpoint lock by using the $E CKPTLOCK,HELDBY= command.
3. Requeue the jobs for restart by entering the $E MEMBER command, which makes jobs eligible for restart on another JES2 member. Note that jobs with member affinity to the failed system will not be restarted until the failed member is available or member affinity for those jobs is modified.

Figure 55 shows a sequence of events in a 2-member MAS configuration where the software checkpoint lock is held by the failed JES2 member (SYSA). The operator must respond to the $HASP264 message by entering the $E CKPTLOCK,HELDBY= command, even though the MVS system (SYSTEMA) continues to process jobs.

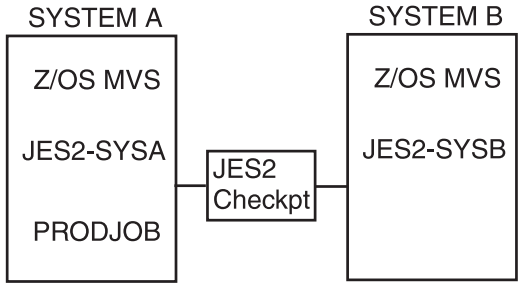



```
On SYSTEMA:                          On SYSTEMB:

JES2 Fails
                                       $HASP264 WAITING FOR ACCESS TO
                                            JES2 CHECKPOINT VOLUME
vvvvvv

                                       $E CKPTLOCK,HELBY=SYSA
Restart JES2
```

Figure 55. Two-member MAS with JES2 failure (checkpoint on DASD)

In a 2-member MAS configuration where the checkpoint resides on a Coupling facility structure and the Coupling facility lock that JES2 uses to control access to the checkpoint was held by the failed JES2 member (SYSA), the operator does not have to release it; the Coupling facility lock is released without operator intervention while the MVS system (SYSTEMA) continues to process jobs.

Note that the $HASP263 message contains a modified text that is informational.

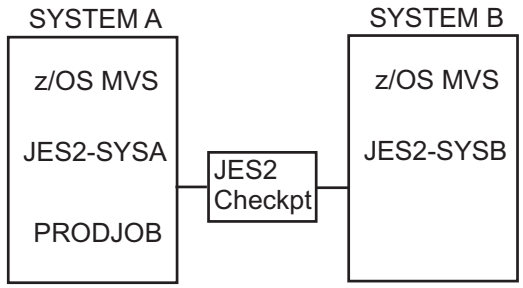


On SYSTEMA: *On SYSTEMB:*

```
JES2 Fails

                                    $HASP263 WAITING FOR ACCESS
                                     TO JES2 CHECKPOINT.  LOCK
                                     HELD BY MEMBER member_name

Restart JES2
```

Figure 56. Two-member MAS with JES2 failure (checkpoint on Coupling facilities)

Figure 57 on page 244 shows a sequence of events in a 2-member MAS configuration where the checkpoint data set resides on DASD and both the hardware lock and the software checkpoint lock are held by the failed JES2 member (SYSA). In this instance, both the JES2 member and the MVS system (SYSTEMA) have failed.




```
On SYSTEMA:          On SYSTEMB:

  SYSTEMA Fails
  (For example,
  the system failed
  and entered
  a nonrestartable
  wait state.)
                                    $HASP263 WAITING FOR ACCESS
                                     TO JES2 CHECKPOINT VOLUME...


SYSTEM RESET
 (Operator)

                                   $HASP264 WAITING FOR RELEASE
                                          OF CKPT LOCK BY SYSA

                                    $E CKPTLOCK,HELDBY= SYSA

                                    $E MEMBER(SYSA)

                                   $HASP493 SYSB MEMBER SYSA
                                      RESTART IS IN PROGRESS

                                   $HASP406 PRODJOB RESTARTED

                                   $HASP492 SYSB MEMBER-SYSA
                                       RESTART HAS COMPLETED

Re-IPL SYSTEMA
```

Figure 57. Two-member MAS with JES2 and MVS failure (DASD)

Using the AUTOEMEM option

IBM suggests that multiple JES2 members in the same sysplex use the AUTOEMEM option. The AUTOEMEM option makes jobs eligible for restart if another JES2 member (MVS and JES2) in a sysplex has failed.

The AUTOEMEM option has no effect on the DASD hardware lock. If it is held by a failed member, the operator must reset the lock.

The AUTOEMEM option has no effect on the Coupling facility lock, because that lock is released automatically when the JES2 address space ends.

Through the MASDEF initialization statement, you indicate whether:

- A JES2 member can have other members make its jobs eligible for restart if it fails (by specifying AUTOEMEM=ON).

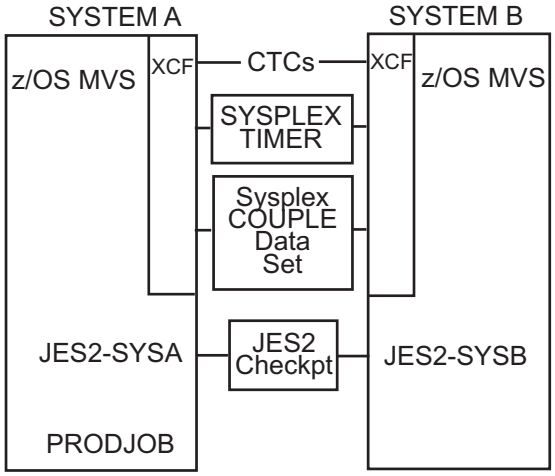
- A JES2 member can make jobs from any other failed member eligible for restart (by specifying RESTART=YES).
- A JES2 member can have other members make its jobs eligible for restart, but **not** allow this member to make jobs from any other failed member eligible for restart (by specifying AUTOEMEM=ON and RESTART=NO).

  Note: If all members specify RESTART=NO, no jobs can be made eligible for restart automatically.

If only the JES2 member fails, the operator must restart the JES2 member so that it can provide services for the jobs that continue to execute on the active MVS system.

As shown in Figure 58 and Figure 60 on page 247, JES2 uses the cross-system coupling facility (XCF) to communicate status among JES2 members in a sysplex. When a participating MVS system fails, other JES2 members are notified through XCF. When a JES2 member fails, other JES2 members are notified through XCF. For information about XCF and the sysplex, see *z/OS MVS Setting Up a Sysplex*.

Figure 58 shows a sequence of events in a 2-member MAS configuration where the checkpoint resides on DASD, the software checkpoint lock is held by the failed JES2 member (SYSA), and the AUTOEMEM option is enabled. In this instance, only the JES2 member has failed; the MVS system (SYSTEMA) continues to process jobs. Note that the $HASP784 message appears in the SYSLOG only.



On SYSTEMA:

```
  JES2 Fails
```

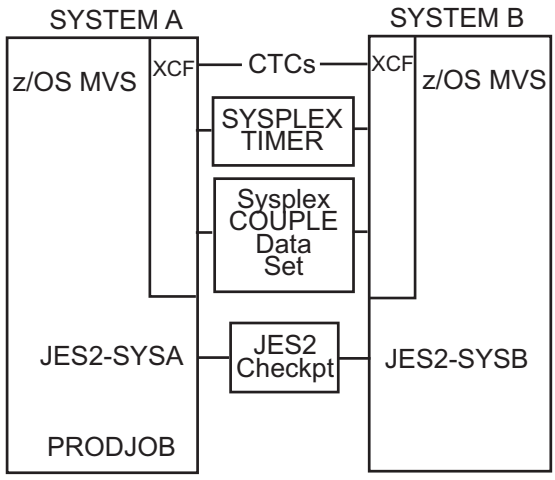
On SYSTEMB:

```
                                $HASP264 WAITING FOR RELEASE
                                        OF CKPT LOCK BY SYSA

                                $HASP784 ATTEMPTING AUTOMATIC RESET
                                        OF JES2 MEMBER SYSA
                               (JES2 automatically resets checkpoint lock)
Restart JES2
```

Figure 58. Sysplex with the AUTOEMEM option enabled, a JES2 failure (DASD)

Figure 59 on page 246 shows a sequence of events in a 2-member MAS configuration where the checkpoint resides on a Coupling facility structure, the

Coupling facility lock that JES2 uses to control access to the checkpoint is held by the failed JES2 member (SYSA), and the AUTOEMEM option is enabled. In this instance, only the JES2 member has failed; the MVS system (SYSTEMA) continues to process jobs. The Coupling facility lock is released from SYSA and passed to the next waiting member of the queue. Note that the $HASP784 message appears in the SYSLOG only.

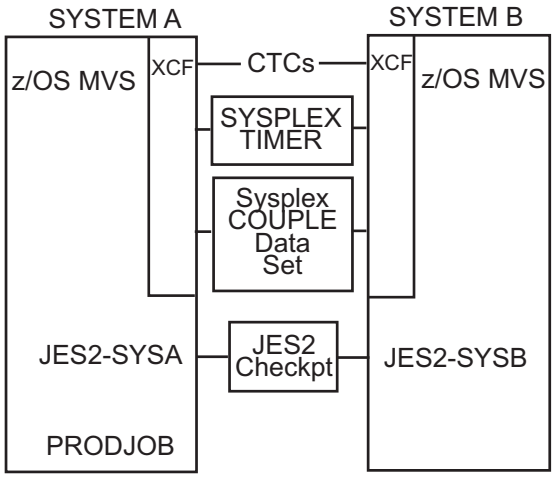


On SYSTEMA: *On SYSTEMB:*

```
  JES2 Fails

                                    $HASP784 ATTEMPTING AUTOMATIC RESET
                                             OF JES2 MEMBER SYSA
                                    (JES2 automatically resets checkpoint lock)
Restart JES2
```

Figure 59. Sysplex with the AUTOEMEM option enabled, a JES2 failure (Coupling facilities)

Figure 60 on page 247 shows a sequence of events in a 2-member MAS configuration where the hardware lock and the software checkpoint lock are held by the failed JES2 member (SYSA), the MVS system (SYSTEMA) has failed, and the AUTOEMEM option is enabled. Note that the $HASP784 message appears in the SYSLOG only.



On SYSTEMA:
On SYSTEMB:

```
  SYSTEMA Fails
  (For example, the
  system failed
  and entered
  a nonrestartable
  wait state.)
```

```
                                  $HASP263 WAITING FOR ACCESS
                                           TO JES2 CHECKPOINT VOLUME...
```

SYSTEM RESET
(Operator)

```
                                  $HASP264 WAITING FOR RELEASE
                                           OF CKPT LOCK BY SYSA

                                  IXC402D SYSTEMA LAST OPERATIVE
                                           AT hh:mm:ss ...
                                           REPLY DOWN IF MVS IS DOWN ...
```

R nn,DOWN

```
                                  $HASP784 ATTEMPTING AUTOMATIC RESET
                                           OF JES2 MEMBER SYSA
                                 (JES2 automatically resets checkpoint lock)

                                  $HASP493 SYSB MEMBER SYSA RESTART IS
                                           IN PROGRESS
                                           (JES2 automatically makes jobs
                                           eligible for restart)

                                  $HASP492 SYSB MEMBER SYSA RESTART
                                           HAS COMPLETED
Re-IPL SYSTEMA
```

Figure 60. Sysplex with the AUTOEMEM option enabled, an MVS system failure

Enabling the JES2 AUTOEMEM option at initialization

IBM suggests that each member enable the AUTOEMEM option (AUTOEMEM=ON on the MASDEF initialization statement), however, if you use automatic restart management to restart batch jobs and started tasks, automatic restart management can restart jobs regardless of whether the AUTOEMEM option is enabled.

The following scenario and figure show a sample configuration enabled for the AUTOEMEM option. The configuration consists of two JES2 members in the same sysplex. In the 2-member sysplex (SYSPLEX1), the AUTOEMEM option is enabled for both members. For information on configuring a sysplex, see *z/OS MVS Setting Up a Sysplex*.

Enable the AUTOEMEM option by coding JES2 MASDEF initialization statements on the JES2 members in SYSPLEX1. The following two MASDEF initialization statements enable the option:

```
MVSa:  MASDEF  AUTOEMEM=ON,RESTART=YES,OWNMEMB=HAS1, ...

MVSb:  MASDEF  AUTOEMEM=ON,RESTART=YES,OWNMEMB=HAS2, ...
```

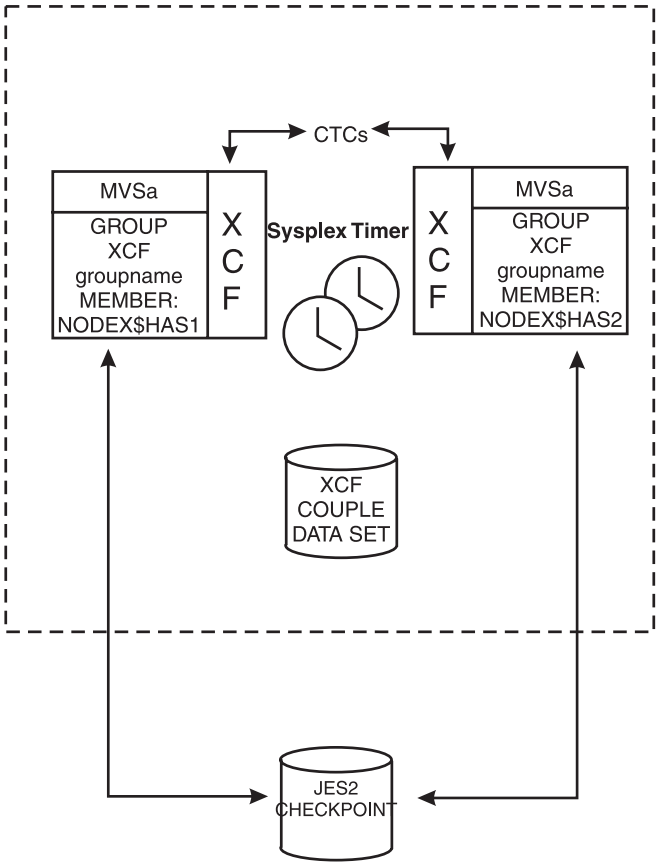


Figure 61. The JES2 AUTOEMEM option in a two-member sysplex configuration

Providing copies of the JES2 checkpoint to application programs

Certain authorized applications have a need to read information in the checkpoint data set. JES2 meets that need by maintaining and periodically updating a copy of the checkpoint data set. When an application requests checkpoint data by using the subsystem interface request macro (IEFSSREQ), JES2 makes the most recent copy of a subset of the checkpoint data set available to the requesting application. To ensure that the copy of the checkpoint data set remains constant while the application is working with it, JES2 does not make updates to this copy until the application releases the copy.

If another application then requests checkpoint data, JES2 determines if the copy being used is the latest copy. If it is, JES2 makes this copy available to the second application. (Therefore, multiple applications can use the same copy of the checkpoint data set, and the copy exists until the last user accessing the data releases the copy.) If the copy being used by the first application is not the latest copy, JES2 will give the application access to the latest copy.

JES2 controls the maximum number of copies that can exist at one time. JES2 will not create a new copy if the number of copies in use is equal to that maximum, but will make the most recently created copy available to the application. However, the copy might not contain the most current information, because it was not recently updated. Therefore, to avoid obtaining outdated information, applications should obtain whatever information they require and release the version as quickly as possible.

JES2 will determine an appropriate maximum number of versions at initialization. The installation can, however, control the copies made available by providing input on the VERSIONS= parameter on the CKPTDEF initialization statement. For information about the IEFSSREQ macro, see *z/OS MVS Using the Subsystem Interface*.

Using Versions= to provide copies of the checkpoint data set for use by applications

Use the VERSIONS= parameter on the CKPTDEF initialization statement to specify the maximum number of copies, or versions, of the checkpoint data set JES2 can create. This method of providing copies of the checkpoint data set is similar to allowing JES2 to manage the copies, except that the installation explicitly specifies the maximum number of copies that can exist at one time.

Chapter 5. Network job entry (NJE)

This chapter presents an overview of an NJE network, how to design a network, and how to connect a network. This chapter also discusses initializing NJE functions in a JES2 environment for various configurations and defining a multi-access spool node.

NJE Installation explains these concepts in greater detail.

Overview of NJE

Geographically dispersed processors communicate using the network job entry (NJE) facility for the purpose of transmitting commands, messages, jobs, and job output between the host systems (System i® JES2, JES3, VM/RSCS, and VSE/POWER). The facility can also provide communication among processors at the same location as an alternative to a JES2 multi-access spool configuration. An NJE network can consist of up to 32,767 nodes. Each of the nodes can be either a single processor or a multi-access spool (MAS) configuration. An MAS configuration is one in which as many as 32 members share one set of JES2 spool and checkpoint volumes. NJE can also communicate between primary and secondary JES2 members on the same processor.

NJE is compatible across all releases of System i, JES2, JES3, VM/RSCS, and VSE/POWER.

Differences between Networking Protocols

There are three ways to send and receive data in NJE. One is binary synchronous communication (BSC), the second is Systems Network Architecture (SNA), and the third is Transmission Control Protocol/Internet Protocol (TCP/IP).

A network can consist of any combination of SNA, BSC, and TCP/IP connections. Each of these has its advantages. Your choice depends on the available hardware and software.

Initialization statements

A JES2 complex can use BSC, SNA or TCP/IP protocol, or all. A user submitting an NJE job is not aware of whether JES2 is using BSC, SNA or TCP/IP.

Table 54. Comparison of SNA and NJE/TCP Initialization Statements

Structure	SNA	NJE/TCP
Connection to networking product	LOGON(nnn) – open APPLID to VTAM	NETSRV(nnn) – open SOCKET to TCP/IP and a NJE/TCP server address space
Maps NJE node to networking construct	APPL(name) – VTAM APPLID associated with a node	SOCKET(name) – IP address and port associated with an NJE node
Logical JES2 networking line	LINE(nnnnn) UNIT=SNA – connection to another node	LINE(nnnnnn) UNIT=TCP – connection to another node
Networking sub-devices	Lnnnnn,SRn, Lnnnnn.STn, Lnnnnn.JRn, Lnnnnn.STn	Lnnnnn.SRn, Lnnnnn.STn, Lnnnnn.JRn, Lnnnnn.STn

Logical and physical configurations

The difference between logical and physical configurations is the most significant distinction between SNA, BSC, and TCP/IP protocols. In BSC NJE, the physical configuration is the logical configuration. In SNA or TCP/IP NJE, the logical and physical configurations can be quite different. Various routing tables in the host and the communication controllers and routers determine the logical configuration.

In Figure 62, Node A can have a direct session with Node C even though no direct line exists between them. Node A and Node C view themselves as being directly connected to each other. ACF/VTAM* and Multi-Systems Networking Facility (MSNF) support this connection through the communications controllers (cc) and the network control programs (NCP), which are transparent to the nodes. Figure 62 represents a simple SNA NJE configuration.

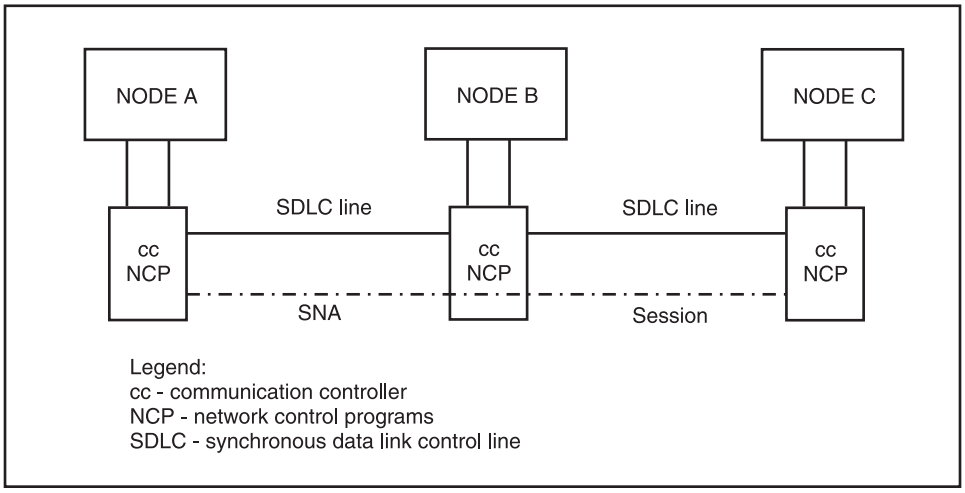


Figure 62. Simple SNA NJE Configuration

Hardware considerations for NJE

BSC lines, channel-to-channel (CTC) adapters, SNA or TCP/IP network lines connect JES2 systems participating in an NJE environment.

BSC lines attached to IBM 3704, 3705, or 3725 Communication Controllers operate in emulation mode. Generate NJE 3704s and 3705s to emulate a 2701 communication controller (see both *IBM 3704 and 3705 Communications Controllers* and *IBM 3725 Model 1 Communications Controller*).

During system installation, use hardware configuration definition (HCD) to define the attached NJE lines as follows:

- On the 'Primary Task Selection' panel, select '1. Define, modify, or view configuration'. On the resulting panel, select '5. I/O devices'
- On the 'Add Device' panel, specify the device type as `BSC1`.
- On the 'Define Parameter / Feature' panel, specify the TCU parameter as `2701`

See *z/OS HCD User's Guide* for a complete list of the steps involved in defining a device.

Channel-to-channel adapters supported as high-speed communication lines for NJE must be on block multiplexers because they can "lock up" selector channels. Use

the LINE(nnnn) JES2 initialization statement to define these adapters. (See *z/OS JES2 Initialization and Tuning Reference* for details concerning the LINE(nnnn) parameters.) If you do not code these specifications and I/O errors occur when establishing the network, you might find it very difficult to analyze those errors. Connections using internal or external CTC devices only operate in 360 compatibility mode.

Use the following steps to add the control units and CTC:

- Add the control units:
  - Follow the steps as described in *z/OS HCD User's Guide*.
  - On the 'Add Control Unit' panel (accessed from the 'Select Processor / Control Unit' panel), specify the 'I/O concurrency level' as `2`. (This corresponds to SHARED=NO in I/O Configuration Program (IOCP)).
- Add the CTC:
  - Use HCD and follow the steps as described in *z/OS HCD User's Guide*.

    On the 'Define Parameter / Feature' panel, specify `Yes` or `No`, as applicable, for the LOCANY, DYNAMIC and OFFLINE keywords.
  - Add UNIT=unit-addr to the JES2 LINE(nnnn) initialization statement.

Note that multiple systems through multiple CTCs on the same channel path could cause a deadlock condition.

You can also use CTC adapters to support an SNA NJE environment; however, ACF/VTAM controls these CTC adapters. Consult *z/OS Communications Server: SNA Resource Definition Reference* for VTAM-CTC requirements.

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).

TCP/IP NJE sessions take advantage of TCP/IP's hardware independent layered stack to establish connections over a number of existing hardware protocols such as Ethernet and token ring. To use TCP/IP sessions, z/OS Communications Server TCP/IP requires z/OS UNIX System Services and ACF/VTAM to be configured and active. If there is support on the node connected, TCP/IP NJE sessions support IPv6 and TLS/SSL.

SNA NJE application-to-application sessions using SDLC lines require that the Advanced Communication Function (ACF) for the Network Control Program/VS (NCP/VS) or the partitioned emulator program (PEP) resides in the IBM 3705. Also, the processors participating in these sessions must have ACF for VTAM and the Multi-System Networking Facility VTAM installed.

Many parameters affect the performance of BSC NJE traffic on a channel-to-channel adapter. At times, increasing or decreasing buffer sizes can result in a decrease of performance because of varying channel loads, hardware limitations, and the speed of the CTC adapter itself. Experienced system programmers (with their system engineers) should tune JES2 parameters. While adjusting the parameters, carefully consider the effects of the following on performance:

- Teleprocessing (TP) buffer sizes

- The number of buffers per spool page
- The loads on the channel
- Other demands on the BSC processes (such as RJE devices).

Data flow through a network

You can enter a job on one node, route it to another node for execution, and send the output to one or more nodes for output. Figure 63 on page 256 illustrates the data flow and shows the JES2 devices that participate in the transmission, routing, and receipt of jobs.

To determine the SYSOUT classes for job output within your network, see "Output class assignment" on page 110. Coordination of SYSOUT is important for consistency of print and punch classes among the network nodes.

How the network processes jobs, commands, and messages

Jobs and associated in-stream data entering the network from a local input device, internal reader, or remote workstation can execute on the entry node or any other node in the network. The entry node queues jobs and in-stream data sets destined for another node. The target node must have the proper resources (for example, the necessary tape drives, direct-access devices, and specific data sets) and must be the job's designated execution node. If you lose or change a path during work flow, the node currently processing the work will either hold the job or, when possible, select an alternate transmission path.

JES2 queues commands to a remote console processor for transmission to the command's final destination. If JES2 loses the path to the final destination, JES2 attempts to return a diagnostic message to the entry console. Severe errors can cause JES2 to discard commands without issuing unique diagnostic messages.

Messages flow through the network like commands; however, JES2 discards messages if it loses the path to the final destination.

How JES2 processes jobs from TSO/E user IDs in NJE

When JES2 first reads in the job, the TSO/E user ID is stored in the job header record if NOTIFY= was specified on the JOB statement. The TSO/E user ID can differ from the RACF userid specified in USER= on the JOB statement. The originating node name is also stored in the job header.

If the job header contains a NOTIFY userid and the job is transmitted from its origin node for execution on another node, the job transmitter issues a message to the NOTIFY user ID indicating that the job was transmitted for execution.

When the job completes execution, the JES2 output processor on the execution node examines the NOTIFY user ID in the job header. If the NOTIFY userid is present, a notification message is issued and directed to the userid on the origin node. If the origin node and execution node are the same, the result is an issuance of an MVS SEND command specifying the userid. If the origin node and execution node are different, the notification is sent back to the origin node through the JES2 remote console processor and RTAM; the JES2 system on the origin node issues the SEND command.

When a job's system output (or any part of it) reaches its ultimate destination, the SYSOUT receiver on the destination node examines the userid in the job header. If NOTIFY= was specified on the job card, the SYSOUT receiver issues a message to the specified user indicating that the job's SYSOUT was received and where it was

received. JES2 sends the message to the job's origin node (as explained previously for the end-of-execution message). For transmissions that entered the network through the TSO/Extensions interactive data transmission facility, NOTIFY= is automatically specified. For further discussion, see the "Other Programming Considerations" section of the description of Exit 40 in *z/OS JES2 Installation Exits*.

If an installation wants local TSO/E userids notified when mail is placed in the output queue (from either another node or a spool reload procedure), specify MAILMSG=YES on the NJEDEF initialization statement.

How the network determines the execution node

At the system on which jobs are submitted, JES2 determines the execution node for a job in one of three ways:

- The job enters the system on a device that has a defined default execution node. You can define the default execution node by using the XEQDEST= parameter on RDR(nn) or R(nnnnn).RD(m) initialization statements.
- A /*XMIT, /*XEQ, or /*ROUTE XEQ control statement, in the JCL for the job, explicitly defines the execution node (overriding the default).
- An operator command (entered when the job is queued for execution or transmission to another node) explicitly defines the execution node (overriding the default or control statement routing).

How the network transmits input jobs and SYSOUT

With NJE, JES2 can transmit a job intended to execute at another node directly to that node. If intermediate nodes are present, JES2 receives, spools, and retransmits the job through each of these. NJE handles job transmission on a store-and-forward basis. That is, a node must completely receive a job before it can take any action to either execute the job or transmit the job to another node. When the receiving node has completely received and spooled the job, the transmitting node frees the resources that it allocated to transmit the job.

JES2 can send both job and SYSOUT output data sets through a series of devices called transmitters, which select files destined for nodes reachable through the NJE line to which these transmitters are assigned. Job and SYSOUT transmitters select files from the transmission ($XMIT) and Network queues respectively, which are ordered FIFO (first-in-first-out) within priority. There can be multiple transmitters (and receivers) on a line as defined on the NJEDEF or LINE initialization statements. For recommendations on how to specify transmitters and receivers in your initialization stream, see "Specifying transmitters and receivers" on page 272.

SYSOUT files can range in size from very small (for example, one record, such as a Netmail acknowledgment) to very large (millions of records). When a transmitter has selected a file, it will not interleave it with other files or voluntarily interrupt the transmission until that file is completely transmitted. If the line drops or the transmission is otherwise interrupted, the entire file must be re-transmitted from the beginning. For recommendations on how to set up transmitters to send SYSOUT files, see "Specifying work selection values for NJE transmitters" on page 273.

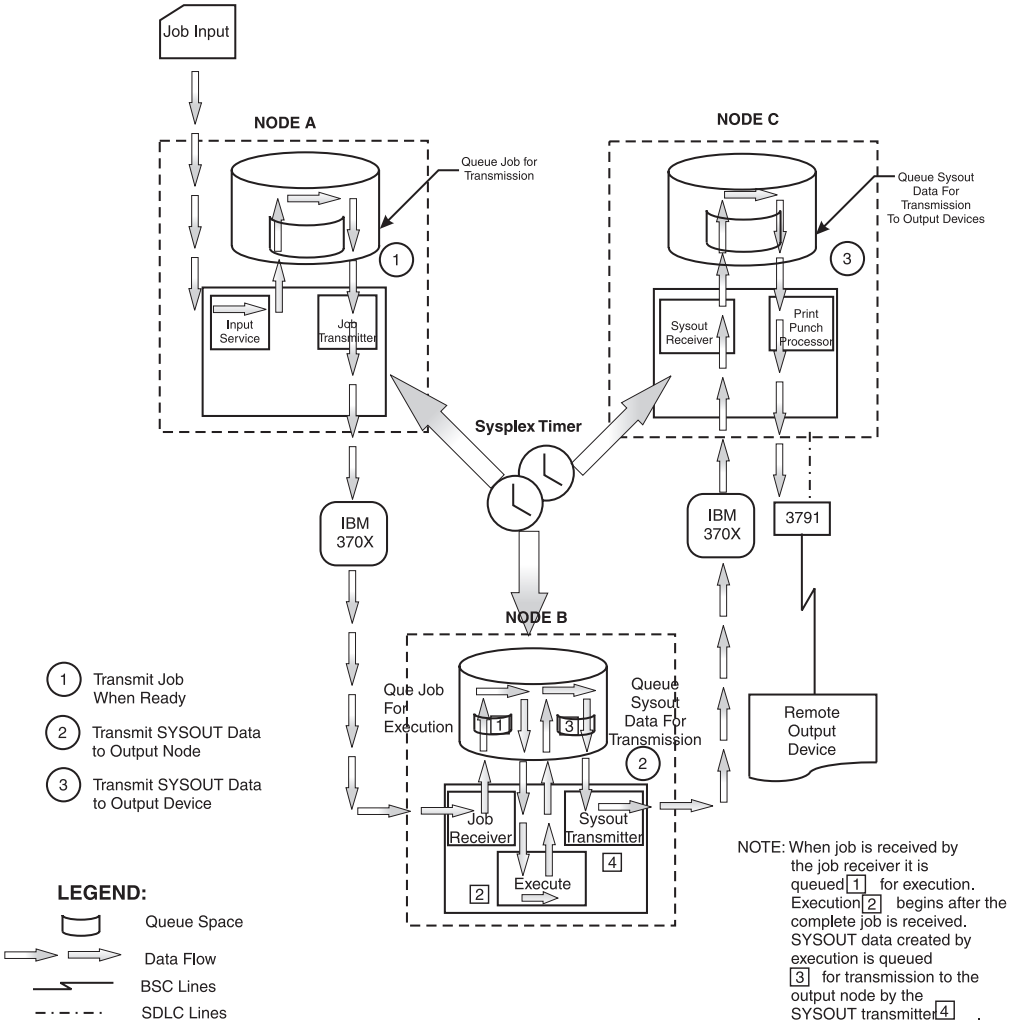


Figure 63. Processing Flow Through a Network

Assignment of JES2 job identifiers

When a JES2 system transmits a job to another system, the receiving system attempts to assign the original job identifier (from the NJE job header) to the received job. The original job ID is used if the job number is not already assigned, and either the job number is inside the JOBDEF RANGE or RASSIGN=YES. (The default is RASSIGN=YES.) Otherwise a new job ID is assigned from available job numbers within the JOBDEF RANGE. If this number is currently in use on the receiving system, the receiver will try to assign the next available job identifier. The newly assigned job identifier is not placed in the job header. The job header always contains the job identifier assigned at the node where the job originated. This job identifier assignment procedure is the same for both jobs and system output.

The input service processor, the job receiver processor, or the SYSOUT receiver processor assigns a unique JES2 job identifier to a job as it first enters a system in the network. This job identifier number is unique within a JES2 system; while the job remains in a system, JES2 will not assign the same number to another job. In the NJE environment, the NJE job header contains the original job identifier assigned by the input service processor at the entry node. Note that the same job

could receive a different job identifier in each system through which it travels. The job header maintains this identifier while the job is in the NJE network.

You can specify a range of numbers (up to 999999) that JES2 assigns to jobs originating at the local node. Use the RANGE= parameter on the JOBDEF statement to specify this range. Installations participating in the JES2 network can coordinate the specification of these parameters, so each node uses a different range of numbers for job identification. Thus, jobs read in at one node and sent to another for execution would always retain the original job identifiers. This is easy to accomplish and useful in a small network.

Determining the destination of output

JES2 queues output produced by a job, started task (STC), or TSO/E user either for processing on this node or transmission to another node. JES2 treats a transmission entering the network through the TSO Extensions interactive data transmission facility as an output data set. JES2 uses the TSO/E userid of the intended receiver as the destination userid. JES2 determines the destination node for output in the following ways:

- The originating node is the destination node if you have not explicitly specified a destination.
- The PRTDEST= and PUNDEST= parameters of the RDR(nn) and R(nnnnn).RD(m) initialization statements explicitly define the destination.
- A JES2 /*ROUTE control statement (submitted as part of the job) redefines the destination.
- The DEST= parameter on
  - A JCL OUTPUT statement
  - A JCL DD statement
  - A JES2 /*OUTPUT control statement

  for a specific SYSOUT data set (or groups of data sets) specifies the destination. (DEST=LOCAL specifies that the origin node is the destination node.)
- The DEST= text unit in a dynamic output request by an OUTADD macro specifies the destination. See *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for information about the OUTADD macro.
- JES2 operator commands explicitly define the destination (for example, $R PRT,J=J5,D=N1R2 or $T O J5,D=WASH).

JES2 then sends the SYSOUT directly to the destination or the closest intermediate node for store-and-forward transmission.

JES2 queues spin data sets (FREE=CLOSE or SPIN=UNALLOC specified on the DD statement) for transmission as soon as the data set closes, without waiting for the job to end.

JES2 attempts to transmit as one job, any non-spin SYSOUT which are created under a single job and directed to a common destination (or along a common path). This processing is performed even if multiple SYSOUT data sets are grouped into different output groups with different output priorities.

A receiving node processes any output received from other nodes as it would normal output. The system prints the received output when a device that matches that job's characteristics becomes available. (This assumes no other higher-priority work is waiting for the device.) You can also direct data to a particular user. That output will remain on the spool until the intended receiver requests it (or the

operator removes it). Use the RECEIVE command to receive "notes" or copy files sent using the TSO/E TRANSMIT command.

Designing the network

Before you code the initialization statements that define your network, you should design it. Designing your network can minimize the errors resulting from incorrect or incomplete initialization statements for the network. Take the time to draw out the network and determine what physical and logical connections you need to get jobs or output to other nodes. Careful coordination between each remote location is important to avoid problems caused by duplicate node names, incorrect line specifications, unclear resistance values, and other inconsistencies. Make sure that the nodes with which you need to communicate are accessible to your node either directly, or through other nodes in the network.

General considerations

Consider the following topics and questions when designing your network:

Network topology

(See "Network topology" on page 259.)

- What does the network look like?
- How does your node fit into the entire network?
- Do you need alternate paths to particular nodes?
- What connections do you need?
- Is each installation aware of all planned connections?

Performance

Performance issues includes the following topics:

- What should you do to keep work flowing through the network?
- How can you limit the amount of data going through the network?
- What is the best way to get data to a particular node?
- What are the predominant routes in the network, and how can they be optimized to reduce intermediate node store-and-forward activity?

Security

(See "Security considerations" on page 263.)

- How do you keep an unauthorized node from signing onto your node, or sending and receiving data both to and from your node?
- How can you protect the data as it travels through the network?
- Does the data require SSL/TLS encryption?
- Is there agreement on the use of nodal passwords?
- When and how will the passwords be changed?
- What kind of protection will the passwords have?
- What command authority should participating nodes have on the individual installations?
- Should that authority be changed by operator commands?

Node names

(See "Symbolic node names" on page 266.)

- What are the naming restrictions?
- Are there similar names in the network now?

- Which names does JES2 know?
- Do all the installations participating in an SNA NJE network agree on **unique** VTAM application names as specified on the APPL(avvvvvvv) initialization statement?

Resistance

(See “Determining path resistance” on page 311.)

- What is the criteria for assigning resistance values?

The following two considerations are discussed more generally.

Accounting: Can the job accounting exit routines process different job accounting fields in use across the network?

Cost: Are there any ways to limit the physical resources to reduce the cost of the network?

Each node's design should be a compromise of all the considerations to best meet your needs.

Network topology

Network topology is the physical and logical configuration of host systems and their interconnections. The attachment of host processors, communication controllers and lines, and other mechanisms (such as channel-to-channel adapters or shared DASD) determines the physical configuration of the network.

Simple configurations

When designing your network, be aware of the building blocks of networks. Figure 64 on page 260 illustrates five simple configurations used in constructing networks. The following discusses the configurations presented in Figure 64 on page 260 and some advantages and disadvantages of each.

Simple 2-node: This is the basis of all networks. It provides communication between two nodes but offers no recovery should the link fail.

2-Node with 2 parallel links: This configuration is the same as the simple 2-node but adds the ability to recover because of an additional line between the two nodes.

4-node ring: You can use this configuration to join four adjacent nodes. Three nodes can continue to communicate should any one node or link fail. You should be careful implementing this configuration because, if a path definition to any one node is not unique, data could loop through the network if a node fails. For more information on how to avoid the problems of looping in a network, see “Preventing looping when using connect statements” on page 308 later in this chapter.

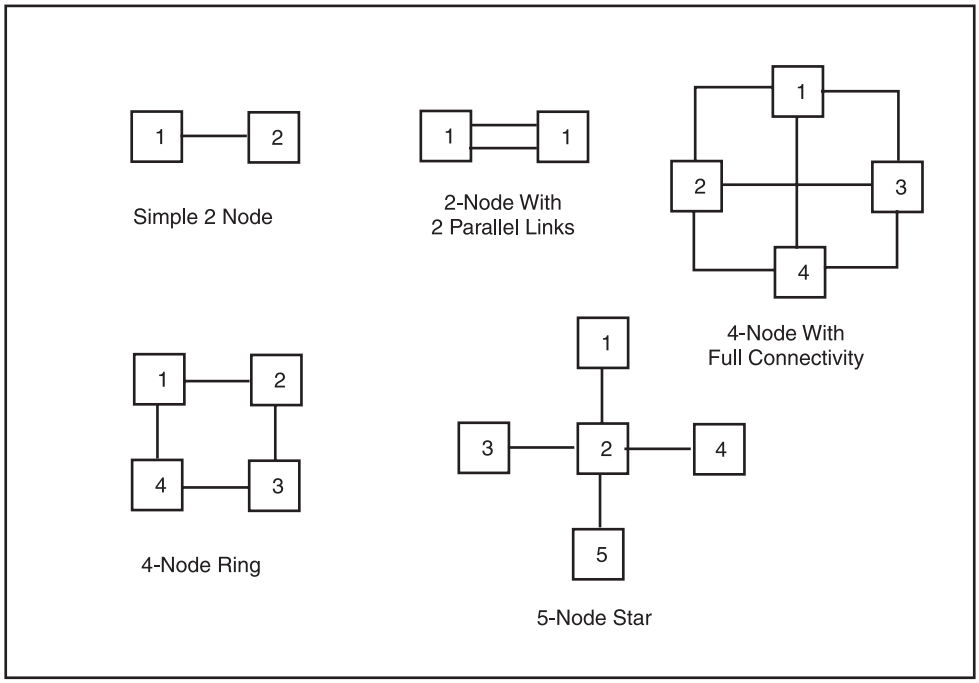


Figure 64. Basic Network Configurations

4-node ring with full connectivity: This configuration allows full communication capability between four nodes. If any one node should fail, the other three nodes will continue to communicate because of the cross communication between nodes 1 and 4, and nodes 2 and 3. This makes the network more reliable and minimizes the queuing of data for the failed node at one particular node.

5-node star: If most NJE traffic is between a central node and remote nodes, you might consider using this design. For example, you can place a central data base at node 2, which is also the center of the network. If the central data base becomes unavailable for any reason, node 2 can then shut down the network until the data base becomes available.

Complex configurations

As your network grows, so do the problems with managing the network. Cost considerations also begin to grow. It is not practical to have links between every node in a large network, and even logical links between every node might not be a good solution. NJE support for store-and-forward traffic makes it easier to manage growth without adding physical or logical overhead.

When you design a complex NJE network, start with your existing network and use the already-established links to help you build the larger one. Figure 65 on page 261 and Figure 66 on page 262 illustrate two complex configurations.

Subnets with gateways: As networks grow in size and complexity, and as multiple networks interconnect, it is practical to manage networks in pieces with a limited number of control points. For this reason, divide large networks into **subnets** with **gateway** nodes. For a definition of subnets and an example of how gateway nodes can enhance network productivity, see “Defining and communicating between subnets” on page 309. The gateway nodes provide the interconnections between the subnets, as you can see in Figure 65 on page 261. These gateways can also provide security, accounting, and control.

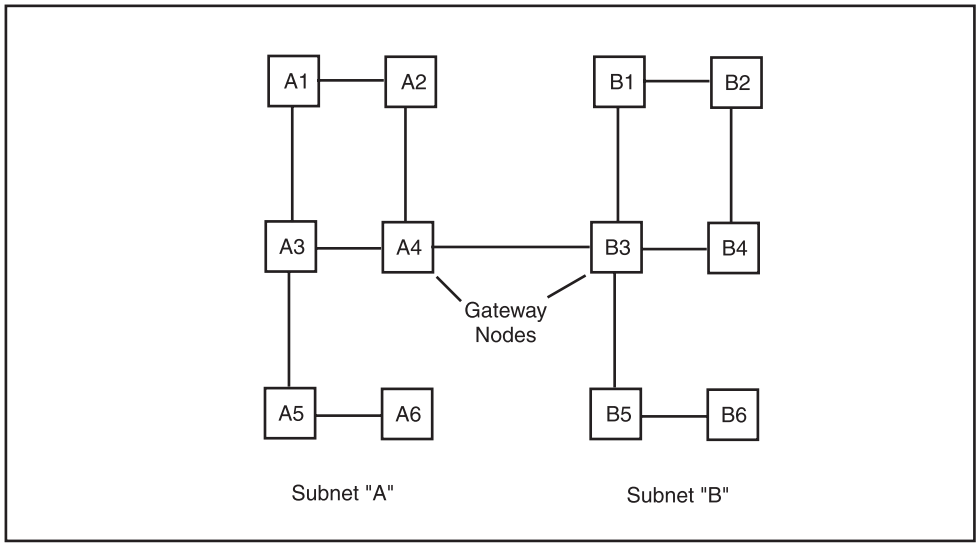


Figure 65. Gateway Configuration

Backbone network configuration: As the number of subnets and interconnections grow, the concept of a **hub** or **backbone** becomes useful in managing the interconnection of NJE traffic. Figure 66 on page 262 shows three subnets “A”, “B”, and “C” connected through gateway nodes A5, B3, and C3 which comprise the backbone of the network. Using matching CONNECT initialization statements to connect nodes A5 and C3 directly further increases the availability of the backbone. For example, if the lines in Figure 66 on page 262 that connect node A7 to node B1 are down but CONNECT statements have been specified on all nodes, the data collects at gateway node A5, rather than waiting at A-6. When the path becomes active, the data reaches its destination more quickly.

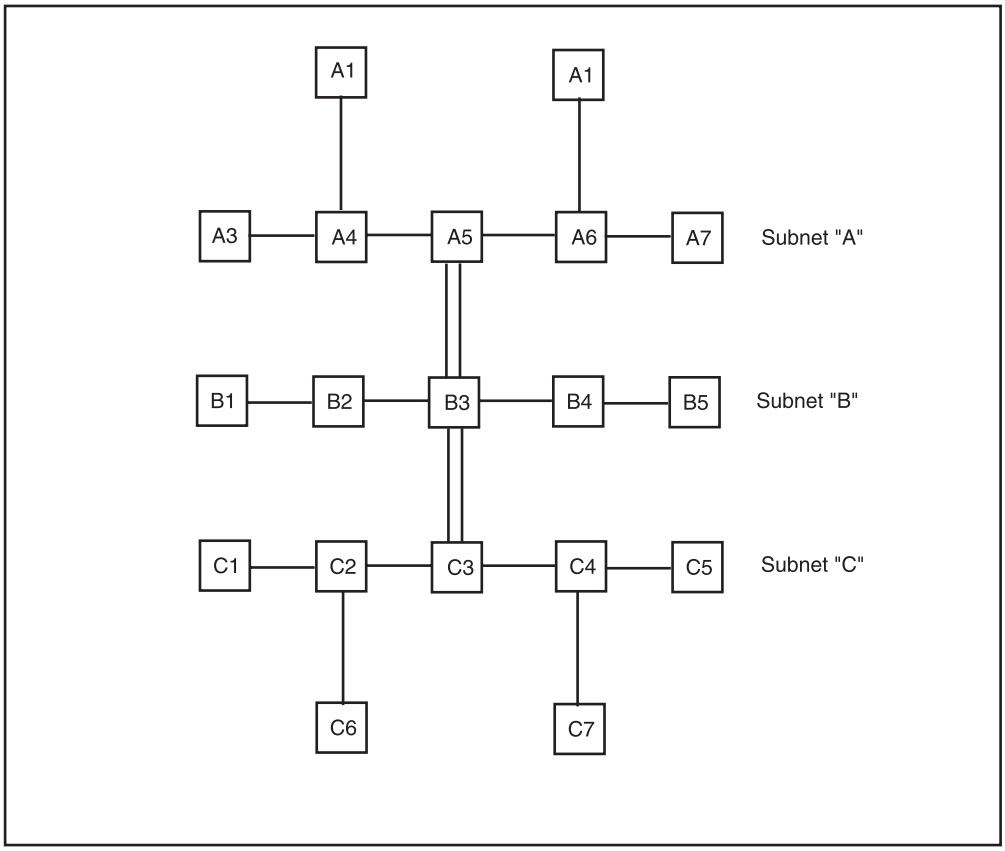


Figure 66. Backbone Network Configuration

Performance considerations

Line speeds, processor speeds, length of the path between nodes, and other factors determine how fast your network gets data from one node to another. The amount of data can also delay the data in reaching its final destination. For more information about JES2 initialization parameters that can impact performance greatly, see "Queuing messages to a multi-access SPOOL node" on page 321.

JES2 uses **compression** and **compaction** to minimize the amount of data sent through the network.

Compression

Compression is the method of reducing the size of a file by removing all blanks and duplicate characters which is applicable for all NJE connections. JES2 automatically compresses data sets sent through an NJE network. On a BSC line, you can specify COMPRESS=NO on the LINE(nnnn) initialization statement to turn off compression.

Compaction

In an SNA environment, you can further reduce the data sent through an NJE network by compacting the data. JES2 uses compaction to represent 8 bits of data in 4 bits. You reduce the data by carefully building a compaction table (using the COMPACT initialization statement) and using that table for the SNA session.

Note: Carefully consider the amount of data you are shipping through the network before using compaction. Compaction adds considerable CPU overhead to the session and might not be beneficial when transmitting binary data streams. Compactions is most useful when transmitting text-string data streams. If you are transmitting data to non-JES2 nodes, ensure that they support data compaction.

Security considerations

In networking, you must:

- Secure the node, which involves ensuring the proper nodes and users have access to the network
- Secure the data, which involves ensuring that if an unauthorized node or userid intercepts the data, that data is not usable by the interceptor.

You should protect your resources at different levels. Security mechanisms include:

- RACF (or an equivalent security product) -- Each node in the network should use RACF to protect its local resources. See Chapter 7, "Providing security for JES2," on page 349 for information on implementing security for your network.
- SSL (secure socket layer) -- NJE/TCP network can have SSL or Transport Layer Security (TLS) defined for nodes.
- Encryption.
- JES2 passwords.
- JES2 exits.
- Security authorization facility (SAF) exits.
- SMF exits.
- MVS exits and modifications.

In larger networks, implementing security is most commonly done at the points where subnets interconnect, besides the individual node security. Therefore, it is important that the gateway nodes in the network enforce the proper level of security. *NJE Installation* contains a complete discussion of security in large networks.

Password processing

JES2 defines passwords associated with NJE lines and with nodes in the NJE network during initialization. Use the PASSWORD= parameter of the LINE(nnnn) initialization statement for BSC lines to specify line passwords. Do not specify a password for SNA lines used for NJE; use the PASSWORD= parameter of the NODE(nnnn) initialization statement to specify a node password for each node with which another node can communicate in your network.

As an alternative to traditional NODE passwords, you can use NODE(nnnn) SIGNON=SECURE along with an APPCLU profile as a secure form of NJE signon. It does not exchange nodal passwords in clear text. With TCP/NJE you can provide passwords as a process of creating digital certificates to do SSL secure networking. You can use SECURE= parameter on Socket definitions to exploit the SSL security facility of TCP/IP. See "Extending network capability" on page 318 for additional information.

Installations can encrypt passwords for jobs sent through the network. Use RACF on the submitting node to encrypt a password that is then verified by RACF on the job's execution node. Before sending jobs with encrypted passwords through the network, ensure that the execution node supports encrypted passwords. Use

the PENCRYPT= parameter on the NODE(nnnn) statement to indicate the nodes that support encryption. See *z/OS JES2 Initialization and Tuning Reference* for a description the PENCRYPT= parameter.

NJE secure signon: SSL and TLS provide excellent security, from a TCP/IP standpoint, to encrypt data on unsecure links and ensure that the peer node at the other end of the connection is who it claims to be from a TCP/IP standpoint; however, from an NJE standpoint, you might need additional security to ensure that the peer node is who it claims to be. You can specify NODE and LINE parameters for connections, but these passwords are exchanged in clear text in signon records and might potentially be compromised if they are sent into an unsecure network.

An additional signon protocol allows encrypted keys to be exchanged by peer nodes rather than clear text passwords. This protocol uses the SAF APPLCU class to define an encryption key. You need to define the following items in order to make use of this protocol:

1. Sspecify NODE(nnnn) SIGNON=SECURE for the peer node at both ends of the connection.
2. Activate the APPCLU class with SETROPTS CLASSACT(APPCLU).
3. Define a profile to define the encryption key. Then entity name is NJE.localnode.othernode, and the session key is associated with it through the SESSION(SESSKEY(key)) parameter.

For example, to set up a secure signon between nodes POK and WSC, with a session key of "WILMA", the following definitions are required:

Table 55. Definitions required to set up a secure signon

JES2 Init Statements at node POK	**JES2 Init Statements at node WSC**
NODE(WSC) SIGNON=SECURE	NODE(POK) SIGNON=SECURE
RACF Definitions at node POK	**RACF Definitions at node WSC**
RDEFINE APPCLU NJE.POK.WSC SESSION(SESSKEY(WILMA)) UACC(NONE) SETROPTS CLASSACT(APPCLU)	RDEFINE APPCLU NJE.WSC.POK SESSION(SESSKEY(WILMA)) UACC(NONE) SETROPTS CLASSACT(APPCLU)

This secure protocol is not restricted to TCP/IP connections and can also be used by SNA and BSC connections.

For more information, see "Extending network capability" on page 318.

Encryption

You can encrypt a particular line (in which case everything sent on that line is encrypted) or a particular transmission. End-to-end encryption is the process of encrypting a teleprocessing line. You encrypt a teleprocessing line by using cryptographic modems. When using ACF/VTAM, there are software products you can use to encrypt individual sessions. You can also use software products to encrypt specific transmissions before sending them through the network. In this case, the receiver must have the same product and the encryption key to decode the data. When using TCP/IP for doing NJE, you can define a policy agent for the network and exchange digital certificates between nodes in network.

Initializing the network job entry functions

The following is a guide for the system programmer establishing an NJE network for the first time. This section describes and provides examples for the following tasks:

Initialization statements

The following initialization statements define a network. (For detailed information about parameter formats, default values, and other coding considerations, see *z/OS JES2 Initialization and Tuning Reference*.)

Table 56. JES2 Initialization Statements that Define a Network

Statement	Purpose
APPL(jxxxxxxx)	Associates a JES2 node with a VTAM application identifier and specifies session characteristics for SNA connections. An installation does not need this statement if adjacent nodes have the same names (NAME= parameter on the NODE(nnnn) statement) as their VTAM application identifiers.
NETSERV(nnn)	Defines a server address space for JES2 to facilitate networking using TCP/IP protocol.
CONNECT	Defines a static connection between two nodes.
DESTID(jxxxxxxx)	Defines the symbolic name for a JES2 route code, which may define a specific device on a node.
LINE(nnnn)	Defines the characteristics of a telecommunication line.
LOGON(nnn)	Specifies the VTAM application identifiers this node uses to communicate with VTAM for an SNA connection. See *z/OS Communications Server: SNA Resource Definition Reference* for information about defining application identifiers. Other nodes that want to communicate with this node must sign on to one of these application identifiers.
SOCKET	Defines the characteristics of TCP/IP connection.
MASDEF	Specifies the characteristics of a multi-access spool configuration.
MEMBER(n)	Specifies the name for each member of a multi-access spool configuration.
NETACCT	Permits network installations to define network account numbers that correlate to local account numbers.
NJEDEF	Defines the NJE characteristics for this JES2 node.
NODE(nnnn)	Defines the characteristics of each node in the network.

Table 56. JES2 Initialization Statements that Define a Network (continued)

Statement	Purpose
TPDEF	Defines the JES2 teleprocessing characteristics: • Buffer size and placement • VTAM sessions • Message limits for remote consoles

Naming the nodes in a network

The following provides the JES2 initialization parameters to use when naming nodes, routing output, and establishing the signon protocol for all nodes communicating throughout a network.

Default names

If you do not specify NAME= on the NODE(nnnn) initialization statement or omit the NODE(nnnn) statement, JES2 generates a default symbolic name, in the form Nnnnn, where the nnnn comes from the statement subscript.

Naming restrictions

In all communications with other nodes, a node uses its node name (the symbolic name, not the number). These communications include network connection control signon records. When verifying a signon, the receiving node checks the name received against its list of valid nodes.

Installations should specify both node names and numbers consistently, or risk having operators receive confusing JES2 operational messages. Because JES2 messages often display only the node number, an operator could be confused by the situation in Table 57, where N1 is NODEA at one node and NODEB at another.

Table 57. Confusing JES2 Node Naming Conventions To Avoid.

```
Node A                                  Node B

NJEDEF   OWNNODE=1                      NJEDEF   OWNNODE=1
NODE(1)  NAME=NODEA,...                 NODE(1)  NAME=NODEB,...
NODE(2)  NAME=NODEB,...                 NODE(2)  NAME=NODEA,...
```

JES2 propagates NODE(nnnn) initialization statement names throughout an NJE network, so these names must be unique throughout the network.

Symbolic node names

Use symbolic node names (rather than node numbers) to make individual nodes in your network easier to remember. For example, Honolulu is more memorable to you and your operators than N4358. However, it is not a good idea to use an unqualified city name (for example, DALLAS) because the city name could cause duplicate node name problems when you join two networks together. You should always use some manner of qualification (such as DALLAS1) when naming your nodes. Specify the names you want to give to each node in the NAME= parameter of the NODE(nnnn) initialization statement. For example, NODE(1) corresponds to POKIPSY1 in the following statement:

```
NODE(1)      NAME=POKIPSY1
```

Any following initialization statements or commands can then see the name, rather than the number of the node, for example: This eliminates the need to remember

the association between NODE(1) and the symbolic name, POKIPSY1. Furthermore, for display purposes ($D NODE) or for changing NODE characteristics ($T NODE), you can specify the NODE subscript as a generic form. For example, NODE(POK*) indicates all nodes with a name beginning with POK.

```
$T   NODE(POKIPSY1),AUTH=(JOB=NO)
```

The symbolic name assigned to a node through the NAME= parameter on the NODE(nnnn) statement should not be of the form NxRx, or Nxxx, which JES2 might incorrectly resolve as a different node.

Defining nodes using generics

You can minimize the number of statements needed to define your network by using generics to define your nodes. You use generics to define :

1. Common characteristics of **all** nodes
2. Unique characteristics for individual nodes
3. Unique characteristics for groups of nodes.

To clarify this point, study Figure 67 and its explanation immediately following.

```
NODE(*) REST=125
NODE(1) NAME=WASH1,AUTH=(DEVICE=NO,NET=YES)
NODE(2) NAME=WASH2,AUTH=(SYSTEM=NO,JOB=NO)
NODE(3) NAME=WASH3,AUTH=(SYSTEM=NO,NET=YES),HOLD=JOBS
NODE(4) NAME=NYORK1,AUTH=(JOB=NO,NET=YES)
NODE(5) NAME=NYORK2,AUTH=(DEVICE=NO,SYSTEM=NO)

⋮

NODE(WASH*) REST=250,AUTH=(JOB=NO)
NODE(NYORK*) HOLD=JOBS
```

Figure 67. Defining Nodes Using Generics

- The NODE(*) statement initially assigns a resistance value of 125 as the default for **all** nodes (NODE(1) through NODE(32767)) during initialization statement processing.

  When initialization processing completes, JES2 limits the set of affected nodes (1-32767) by the maximum specified in the NODENUM= parameter on the NJEDEF initialization statement.

  Note: This might significantly elongate JES2 initialization processing. You should use a more realistic number such as NODE(1-500) rather than NODE(*).
- Specific definitions for individual nodes (NODE(1) through NODE(5)) follow to add to or override the previous generic assignment statement.
- The last two statements (NODE(WASH*) and NODE(NYORK*)) assign values for groups of nodes.
  - The NODE(WASH*) statement overrides the values for REST= and AUTH= for all nodes whose names begin with WASH.
  - The NODE(NYORK*) statement overrides the values for HOLD= for all nodes whose names begin with NYORK.

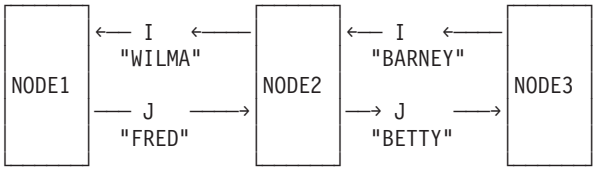
  These values replace those previously defaulted or defined in previous generic statements.

Defining nodes in the network requires careful planning and coordination among participating installations to ensure that JES2 does not encounter duplicate node names. **You** must ensure that a name known to any node is reachable from all nodes.

Specifying multiple passwords for NJE connections

JES2 allows a node to specify different passwords for each adjacent node. Also, an installation can specify a value for the password it receives from an adjacent node to verify the identity of the node signing on.

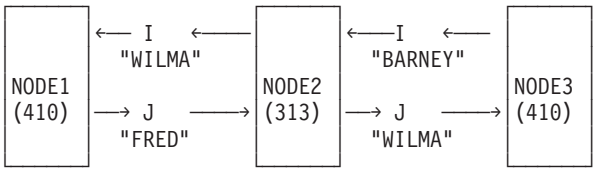
To implement these security functions, use the subparameters for the PASSWORD parameter on the NODE(*nnnn*) initialization statement for each directly connected node. Figure 68 shows a JES2 node sending different passwords to different directly-connected nodes. The figure also shows the PASSWORD subparameters that specify the passwords the nodes send and the passwords the nodes expect to receive to establish the connection.




```
N(1) PASSWORD=       N(1) PASSWORD=        N(1) PASSWORD=
                          (SEND=WILMA,
                           VERIFY=FRED)
N(2) PASSWORD=       N(2) PASSWORD=        N(2) PASSWORD=
     (SEND=FRED,                                (SEND=BARNEY,
      VERIFY=WILMA)                              VERIFY=BETTY)
N(3) PASSWORD=       N(3) PASSWORD=        N(3) PASSWORD=
                          (SEND=BETTY,
                           VERIFY=BARNEY)
```

Figure 68. JES2 NJE Signon Password Verification

Using multiple passwords, of course, affords your system greater security; however, you are not required to do so and can set up any or all nodes as shown in Figure 69. That is, notice that NODE3 is not using the 2-password feature with NODE2. The same password, "WILMA" is used to communicate with both NODE1 and NODE3. The trade-off is a more simplistic security configuration, but this allows **all** adjacent nodes to know and potentially misuse the single password.




```
N(1) PASSWORD=       N(1) PASSWORD=FRED    N(1) PASSWORD=

N(2) PASSWORD=       N(2) PASSWORD=WILMA   N(2) PASSWORD=WILMA
     (SEND=FRED,
      VERIFY=WILMA)
N(3) PASSWORD=       N(3) PASSWORD=BARNEY  N(3) PASSWORD=BARNEY
```

Figure 69. NJE Signon Password Verification for Mixed Levels of JES2

Using secure signon protocol for NJE signon

Using nodal passwords as specified previously still has some drawbacks:

- The passwords are exchanged across the network in clear text, which could compromise the security of the password.
- The passwords are defined and maintained in the JES2 initialization stream, by the JES2 system programmer rather than in the system's security policy (that is RACF), by a security administrator.

The secure signon protocol allows greater password security. In order to take advantage of this protocol, you need to specify the following:

- On the local node, specify **NODE(node2n) SIGNON=SECURE** to indicate that the secure protocol is to be used when signing on to node *node2*
- Specify **RDEFINE APPCLU NJE.node1.node2 SESSION(SESSKEY(key)) UACC(NONE)**, where
  - ***node1*** is the name of the local node
  - ***node2*** is the name of the adjacent node
  - ***key*** is an agreed upon session key for the connection
- **SETROPTS CLASSACT(APPCLU)** to activate the APPCLU security class.

The node at the other end of the connection must define this setup as well. Figure 70 illustrates an example of the definitions required on both nodes.

```
+---------------+                    +---------------+
|               |                    |               |
|  NODE1        |                    |  NODE2        |
|               |                    |               |
+---------------+                    +---------------+

JES2 Init Statements                 JES2 Init Statements
NODE(NODE32) SIGNON=SECURE           NODE(NODE1) SIGNON=SECURE

RACF Definitions                     RACF Definitions
RDEFINE APPCLU NJE.NODE1.NODE2       RDEFINE APPCLU NJE.NODE2.NODE1
  SESSION(SESSKEY(FRED))               SESSION(SESSKEY(FRED))
  UACC(NONE)                           UACC(NONE)
```

Figure 70. NJE Signon Password Verification for Mixed Levels of JES2

Dynamic changes of network information

You can change the node name at any time if the old node name is specified as confirmation on the subscript. For example, $T NODE(17),NAME=WILMA still fails as this is considered a re-definition of the node, but $T NODE(FRED) NAME=WILMA is allowed as this is considered a rename only. The node name change to WILMA is propagated throughout the MAS and DESTIDs for the old node name of FRED is retained on each member. Another effect of this change is that you can also alter the local node name with single-system warm start or operator command. Additionally, you can increase the maximum number of nodes allowed in the network (NJEDEF NODENUM=) dynamically with operator command.

Using destination identifiers (DESTIDs)

The DESTID(jxxxxxxx) initialization statement allows TSO/E users and batch jobs at a given node to see symbolic names for local output devices, special local output devices, remote workstations, and userids attached to network nodes. Specify the

symbolic name in the DESTID subscript as either a specific name or a generic name (similar to the NODE(nnnn) subscript).

You can specify any characters on DESTID subscripts except LOCAL and ANYLOCAL if you specify Ndest=USER, R|RM|RMTdest=USER, and Udest=USER on the DESTDEF initialization statement.

If you do not specify USER for the previously-mentioned parameters on the DESTDEF initialization statement, IBM suggests that you avoid potential confusion when routing SYSOUT from one destination to another by **not** specifying destids in the form of an explicit JES2 destination (route code):

- Nnnnn
- Rnnnnn
- RMnnnn
- RMTnnnn
- Unnnn
- NnnnnRnnnnn

Do not code an explicit node number or node name (NAME= value defined on a different NODE(nnnn) statement) for the subscript on a DESTID initialization statement. That is, if you allow the DESTDEF statement parameters to default, you cannot define DESTID(N1) DEST=N2R1, but you can define DESTID(X) DEST=N2R1.

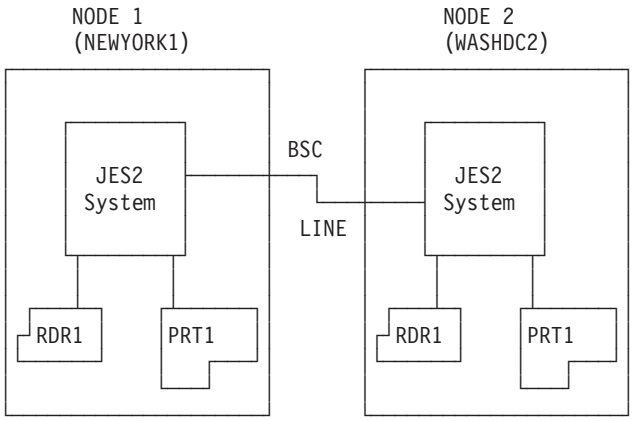
Do not specify a symbolic DEST= parameter which has the same value as a DESTID subscript, unless the DESTID has been previously created in the initialization stream. That is, you cannot define DESTID(TOM) DEST=FRED unless DESTID(FRED) has been previously created in the initialization stream.

The JES2 node that defines the DESTID(jxxxxxxx) name is the only node that knows this subscript name. For an example of how JES2 resolves destination identifiers at both the sending node and the receiving node, see "Routing output to other nodes" on page 108, "Using destination identifiers to route output" on page 142, and "How JES2 resolves destinations from node to node" on page 145.

Defining a minimum configuration for BSC NJE

During JES2 initialization, you can easily request the NJE feature. The default configuration (shown in Figure 71 on page 271) is a very simple BSC-only network. Include four JES2 initialization parameters, NODENUM=, LINENUM=, and OWNNODE= from the NJEDEF statement and the UNIT= parameter from the LINE(nnnn) statement, in your initialization data set. The LINE(nnnn) statement specifies the device address of the line (UNIT=) and enables the text transparency feature (TRANSPAR=), that is, the feature that allows JES2 to send all 256 characters in a data stream without the data being interpreted as control characters.

NODE 1 in Figure 71 on page 271 defines NEWYORK1 as the first node in the network. NODE 2 in the figure does the same for the second node (WASHDC2). The simple network illustrated in Figure 71 on page 271 shows how to use the NJEDEF statement parameters: NODENUM=, LINENUM=, and OWNNODE=. JES2 uses default values for the remaining NJE parameters.




```
NJEDEF NODENUM=2,                NJEDEF NODENUM=2,
       OWNNODE=1,                       OWNNODE=2,
       LINENUM=1                        LINENUM=1
NODE(1) NAME=NEWYORK1            NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2             NODE(2) NAME=WASHDC2
LINE(1) UNIT=xxx,                LINE(1) UNIT=xxx,
        TRANSPAR=YES                     TRANSPAR=YES
PRT(1) UNIT=013                  PRT(1) UNIT=233
                                 RDR(1) PRTDEST=LOCAL
RDR(1) PRTDEST=LOCAL
```

Figure 71. Minimum NJE Configuration (BSC only)

NODENUM

The value of NODENUM= specifies the maximum number of nodes in the network to which a particular system belongs. The network in Figure 71 has two nodes. If your network is growing, you may want to specify a value greater than the actual number of nodes. This way, you avoid changing this parameter every time you add new nodes.

LINENUM

The value of LINENUM= specifies the number of lines JES2 uses for NJE communications. For BSC NJE to use a line, the BSC adapter must have the text transparency feature. Therefore, the LINE(nnnn) statement must specify TRANSPAR=YES to enable the text transparency feature. Specifying LINENUM alone does not dedicate lines to NJE.

OWNNODE

The value of OWNNODE= specifies the node number assigned to this node. In a multi-access spool configuration, each JES2 member must specify the same value of OWNNODE=, because JES2 considers the entire configuration as one node (see "Defining a multi-access SPOOL node (SNA considerations)" on page 322). Installations should coordinate the specifying of OWNNODE= values to ensure that each node has a unique symbolic name.

Default NJE parameters (BSC, SNA, and TCP/IP)

Referring again to Figure 71, including the NJEDEF parameters LINENUM=, NODENUM=, and OWNNODE= in the JES2 initialization streams of these two systems created an NJE network. Because only these NJE options appear on the statements, JES2 uses default options for the remaining NJE parameters. These

options provide the capability to send and receive data, detect lockout situations, route output, specify the difference between TOD clocks at each node, and provide command authority.

Sending and receiving jobs and SYSOUT

All networks have the following capabilities; each node can:

- Send one job and one SYSOUT data set to the other node simultaneously.
- Receive a single job and SYSOUT data set from the other node simultaneously.

To send and receive multiple jobs and SYSOUT data sets (up to 8 in a single transmission) concurrently, override the default values for the JTNUM=, JRNUM=, STNUM=, and SRNUM= parameters on the NJEDEF initialization statement described in "Specifying transmitters and receivers." To control the flow of jobs and SYSOUT data sets in a network, see "Specifying work selection values for NJE transmitters" on page 273.

Specifying transmitters and receivers: An installation can specify a maximum of 16 devices (8 transmitters and 8 receivers) on a line. Each transmission, however, can consist of no more than 8 data sets (limited to a maximum of 7 jobs and 7 SYSOUT). The number of transmitters active on a line must match the number of receivers on the other end of the line; JES2 will drain excessive (unmatched) transmitters and receivers.

You can specify the number of transmitters and receivers on NJE lines globally through the following NJEDEF parameters, which apply to each line generated by the NJEDEF LINENUM parameter:

- JRNUM - Number of Job Receivers for each NJE line
- JTNUM - Number of Job Transmitters for each NJE line
- SRNUM - Number of SYSOUT Receivers for each NJE line
- STNUM - Number of SYSOUT Transmitters for each NJE line.

You can specify the value DEFAULT on the corresponding LINE(nnnnn) statement to take the value specified on the NJEDEF statement. However, if the use of DEFAULT results in more than eight transmitters (job and SYSOUT combined) or more than eight receivers (job and SYSOUT combined), JES2 reduces the defaulted value so that the combined number of transmitters and receivers equals 8.

Installations can commit lines to NJE through either the JRNUM=, JTNUM=, SRNUM=, and STNUM= parameters on the LINE(nnnn) statement, or through the L(nnnn).ST(m) or L(nnnn).JT(m) statements, or using the $T LINE command. (These lines do not count toward the limit specified on the LINENUM= parameter of the NJEDEF initialization statement.) In addition to committing these lines to specific devices, IBM suggests that each installation dedicate all lines for an NJE facility.

To dedicate these lines, specify them on the LINE= parameter on either the APPL(avvvvvvv) initialization statement (for SNA networks only), the SOCKET(xxxxxxxx) statement (for TCP/IP networks), or the NODE(nnnn) initialization statement (for SNA, BSC, and TCP/IP networks). If you specify different values for the LINE= parameter on these statements, JES2 uses the value on the APPL(avvvvvvv) statement (for SNA) or the SOCKET(xxxxxxxx) statement (for TCP/IP).

NJE transmitters can also be specified explicitly through the following statements:

- L(nnnn).JT(m) - Job transmitters for line nnnn - parameters WS=(LIMIT= can be used for selection criteria.
- L(nnnn).ST(m) - SYSOUT transmitters for line nnnn - parameters WS=, LIMIT=, and PLIM= can be used for selection criteria.

If this statement is specified for a line, then a full set of transmitters and receivers will be assigned according to the NJEDEF or LINE xxNUM parameters, where xx indicates SYSOUT transmitter (ST), SYSOUT receiver (SR), job transmitter (JT), or job receiver (JR). If the number (m) is higher than the number specified by the corresponding JTNUM or STNUM parameter, the specific transmitter specified through (m) is not created.

Specifying work selection values for NJE transmitters: You can use the work selection (WS=) parameter on the L(nnnn).JT(m) or L(nnnn).ST(m) statement to cause an NJE transmitter to select files based on their size. This allows you to optimize NJE transmissions. For example, by dedicating some transmitters to large files and others to small files, you can ensure that small files are not always forced to wait for completion of large file transmissions. You can set the limits based on the size in cards, lines, or pages, depending on whether the file is cards or page-mode or line-mode data sets. If you specify both page mode and line mode, a job must fit within both limits before JES2 will select it. For example, if you specify

```
L21.ST1 WS=(LIM/),LIMIT=0-5,PLIM=0-5
```

JES2 selects the job only if the output consists of a minimum of 0 to a maximum of 5 pages and lines.

The following example shows how to set up two transmitters, the first one for large files, and the second for small files:

```
L23.ST1  WS=(LIM/),LIMIT=0-*,PLIM=0-*     /* all files can use  */
L23.ST2  WS=(LIM/),LIMIT=0-100,PLIM=0-5   /* reserved for small */
```

Considerations when specifying NJE transmitters and receivers: Because there are several ways to specify NJE transmitters and receivers, you must coordinate these statements and parameters to maintain normal NJE functions:

- If you do not want to use work selection characteristics, and you want the same number of transmitters or receivers on each NJE line, then use the NJEDEF parameters.
- If you do not want to use work selection characteristics, but you want to specify different numbers of transmitters or receivers on each NJE line, then use the parameters on the LINE statement.
- If you want to specify work selection characteristics for a job transmitter, then use the L(nnnn).JT(m) statement.
- If you want to specify work selection characteristics for a SYSOUT transmitter, then use the L(nnnn).ST(m) statement.
- If you use the L(nnnn).JT(m) or L(nnnn).ST(m) statement, ensure that you have specified all possible ranges of both line and page sizes for your installation. If you do not, you could leave some jobs without transmitters. To avoid such omissions, you can define one transmitter with an infinite size range and then define others with descending size ranges, for example:

  ```
  L23.ST1  WS=(LIM/),LIMIT=0-*,PLIM=0-*
  L23.ST2  WS=(LIM/),LIMIT=0-100,PLIM=0-2
  L23.ST3  WS=(LIM/),LIMIT=0-1000,PLIM=0-20
  L23.ST4  WS=(LIM/),LIMIT=0-10000,PLIM=0-200
  ```

This ensures that there will always be one transmitter capable of transmitting any size SYSOUT data sets over the line.

- On L(nnnn).JT(m) or L(nnnn).ST(m) statements, use asterisks (*) only to represent any pre-defined lines or transmitters. Asterisks do not create transmitter definitions; they only change existing definitions.

  For example, assume that lines 2, 3, and 4 have already been defined by the following LINE statements:

  ```
  LINE 2 TRANSPAR=YES,UNIT=00E
  LINE 3 TRANSPAR=YES,UNIT=338
  LINE 4 TRANSPAR=YES,UNIT=261
  ```

  To define SYSOUT transmitters for these lines, you must code an explicit line number and transmitter number (or range of numbers) on the L(nnnn).ST(m) statement, for example:

  ```
  L(2).ST(1) WS=...
  L(3).ST(1-3) WS=...
  L(3-4).ST(4-7) WS=...
  ```

  After the transmitters are explicitly defined, you can use asterisks to change the definitions, for example:

  ```
  L(2).ST(*) WS=...    (changes all SYSOUT transmitters on Line 2)
  L(3).ST(*-3) WS...   (changes SYSOUT transmitters 1-3 on Line 3)
  L(3-*).ST(3-*) WS... (changes SYSOUT transmitters 3-7 on Lines 3-4)
  ```

- Be careful not to define transmitters for lines meant for RJE use. RJE cannot use lines with transmitters or receivers assigned. Use the $DU command to determine if a line has transmitters or receivers assigned:

  ```
  $DU,LINEn
  ```

Controlling jobs looping in the network

You can control the number of intermediate nodes through which a job (or SYSOUT) will be routed before JES2 determines that the data is looping in the network. The MAXHOP= parameter on the NJEDEF statement specifies the maximum number of nodes (0-65535) through which a job can travel before it is placed in hold. This parameter applies only to jobs arriving at this node from another node, not to jobs submitted locally.

JES2 begins the node count when the job is submitted and continues incrementing the count until the job prints. An installation can reset the count through both the $R and the $T O commands. To understand how to reset the count and alter the characteristics of output routed across a network, see *z/OS JES2 Commands*.

Specify a MAXHOP= value greater than twice the number of nodes in the longest path in your network. This allows for a job to traverse the network and output to return, plus an additional allowance if necessary for alternate path routing. If you do not specify a value for MAXHOP=, the default value is 0. This parameter does not count traffic travelling through intermediate nodes in the network.

Routing output

Node 1's PRT1 (in Figure 71 on page 271) processes all normal print output produced by jobs that entered the network at node 1's RDR1. For jobs entering the network at node 2's RDR1, node 2's PRT1 will receive all normal print output. To alter the default routing of output to some other node in the network, use either the PRTDEST= or the PUNDEST= parameter on the RDR(nn) initialization statement. You can modify both parameters by using the $T RDR1 operator command.

Command authority

In Figure 71 on page 271, each node in the figure has sufficient command authority to affect jobs that it owns at the other node. Use the AUTH= parameter on the NODE(nnnn) statement to limit or expand this command authority. To understand how individual systems maintain security, see Chapter 7, "Providing security for JES2," on page 349 and *z/OS Security Server RACF Security Administrator's Guide*.

Network time tolerance

Specifies, in approximate minutes, the time differential (0 - 1500) allowed between the TOD clock of this node and any adjacent node to allow successful signon. For example, if you specify TIMEtol=60 on an NJE statement and an adjacent node attempts a signon to this node with a clock that differs from this one by 60 minutes or less, the signon is successful. If the adjacent node's clock differs from yours by significantly more than 60 minutes, the signon fails.

Supply a TIMEtol= value (or accept the default value) so that JES2 can determine whether to accept status information about any node in the network or permit a signon.

During JES2 processing, if a record is received with a future timestamp, but the time does not differ from the TOD clock value by more than the TIMETOL=value, JES2 accepts the record. If the timestamp is in the future by an amount that is significantly greater than the TIMETOL= value, JES2 rejects the record. For network topology records, rejection results in the record being ignored (with an error message). For network signon records, the signon to the adjacent node fails.

IBM suggests that you use GMT time and correct time zone offset for your system's TOD clock setting.

Using different operating systems in a network

Consider the following features of JES2 processing when running in a network composed of operating systems other than z/OS JES2 (VSE/POWER, RSCS/VM, System i, z/OS JES3). In particular, there are processing considerations for execution nodes and job output nodes.

Execution node considerations

User access authority (RACF): If your installation uses RACF or any other access control facility to secure data, the user identified on the JOB statement or in the job header must be defined at the execution node with the proper password supplied.

Job account numbers: If your installation uses the JES2 4-character account numbers, you can use translation services to map from one set of local account numbers to another. Use either the NETACCT initialization statement or the /* NETACCT JECL statement to make an account number available to all nodes in a network. Installations can use the network job header to transmit and receive an accounting section that is composed of the complete accounting string used on the JCL JOB statement to create the job.

When specifying multiple NETACCT initialization statements to determine how network account numbers are converted to JES2 (local) account numbers, ensure that any ranges specified do not overlap. If they do, the second NETACCT statement overlays the first; JES2 does not convert network account numbers that were only specified in the first NETACCT statement into local (JES2) account numbers.

In the following example, the second NETACCT initialization statement specifies a range (A33 to B0) that overrides the range (A11 to B0) specified by the first statement. All jobs received from the network with network account number from A11 to A32 would not receive a JES2 (local) account number.

```
NETACCT NACCT=A11,NTHRU=B0,JACCT=AAA,TYPE=BOTH
NETACCT NACCT=A33,NTHRU=B0,JACCT=3AA,TYPE=BOTH
```

A local job that specified 'AAA' would still be sent into the network account number in the range A11 to B0.

Job accounting information: The system management facility (SMF) gathers and records information that installations can use to evaluate system usage. Specifically, the SMF Type 26 record records system identifiers that indicate which system in the configuration performed each major function of processing for a job such as input, conversion, execution, post-execution, division into output elements, and purging. Although job accounting information is completely written to type 5 and type 35 records, those records are not created unless the job runs.

JES2 records the entire contents (up to 142 bytes) of the JOB accounting field into the accounting section of the network job header. This header is transmitted to other nodes with jobs and SYSOUT. The accounting information for a job is available to all nodes in the SMF Type 26 record.

Procedure libraries: JES2 converts and interprets all jobs on the execution node. You may need to copy commonly used procedures to other nodes for use by a larger set of installations.

Unit names: You should standardize unit names across the network to avoid confusion when transmitting jobs to execute on remote nodes. For example, specifying UNIT=3400-16 on the JCL DD statement for all devices in the network avoids confusion.

Output node considerations

Held Output: JES2 transmits data sets held for Time Sharing Option Extensions (TSO/E) retrieval to their destination node and then places the data sets in the hold queue. (See “Defining held data sets” on page 156 for more information about held data sets.)

The TSO/E OUTPUT command allows a user to view, print, or copy held output if the job name is the same as the receiving TSO/E userid plus one or more characters. Your Installations can replace the job name restriction by specifying RACF check of the JESSPOOL class. For more information on how to replace this restriction, see *z/OS TSO/E Customization*.

Lines per page: If your output must have a specific number of lines per page, you must use the LINECT parameter in your JCL. The default used for this parameter when processing the output is the value of the LINECT= parameter on the PRINTDEF initialization statement of the printing subsystem.

FORMDEF and PAGEDEF: JES2 permits both FORMDEF and PAGEDEF names to be associated with any SYSOUT data set. Use the OUTPUT JCL statement specify form characteristics (FORMDEF) or to format the data itself (PAGEDEF). For more information on specifying either parameter, see *z/OS MVS JCL User's Guide*.

JESNEWS: The JESNEWS data set, used to share news among all processors in a multi-access spool configuration, is sent to other nodes with the JESMSGLG data set. The JES2 print processor only prints this data set on local or RJE printers attached to the node at which the JESNEWS data set is defined. To create a JESNEWS data set, see "Creating a JESNEWS data set" on page 136.

SYSOUT Classes: SYSOUT classes should be carefully coordinated throughout a network. All characteristics specified on the transmitting system are in effect on the system to which JES2 routes the output. Ensure that the following forms of SYSOUT are known to all nodes in the network:

- Dummy classes
- Held classes
- Print classes
- Punch classes
- OUTDisp= specifications

Defining a minimum configuration for SNA NJE

Figure 72 on page 278 shows five different ways of configuring SNA sessions. These are:

1. Over one or more synchronous data link control (SDLC) lines through a pair of communication controllers. A single transmission group defines any set of multiple lines.
2. Over multiple SDLC lines through intermediate communication controllers. A single transmission group defines these lines.
3. Through a single communication controller channel attached to two systems.
4. Through a CTC adapter controlled by ACF/VTAM
5. Through VTAM between two copies of JES2 in the same processor.

   When 2 copies of JES2 in the same processor are communicating through VTAM (intra-domain NJE session), there is no need for a communication controller or lines to establish the link. Testing new versions of networking software is the most common reason for using this configuration.

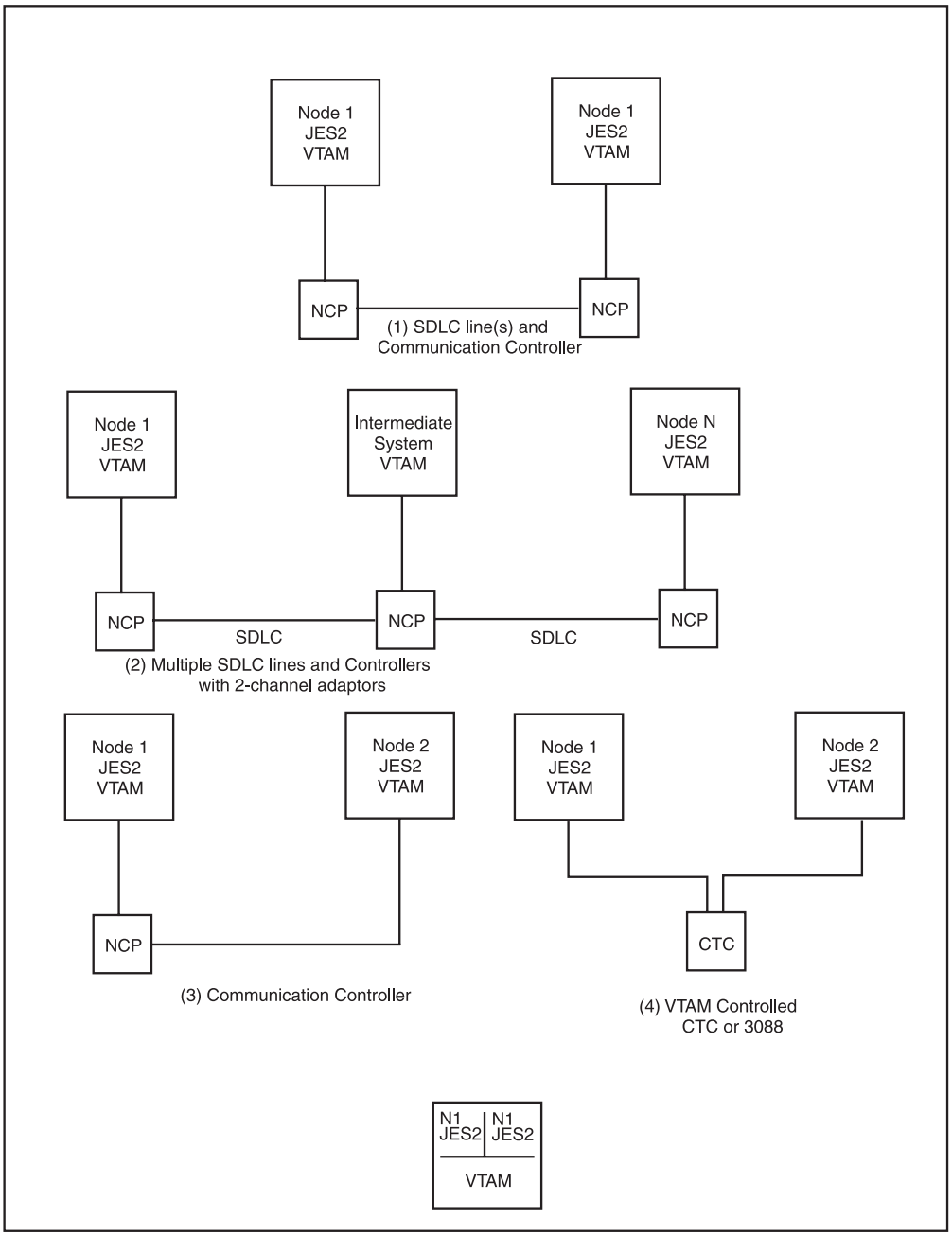
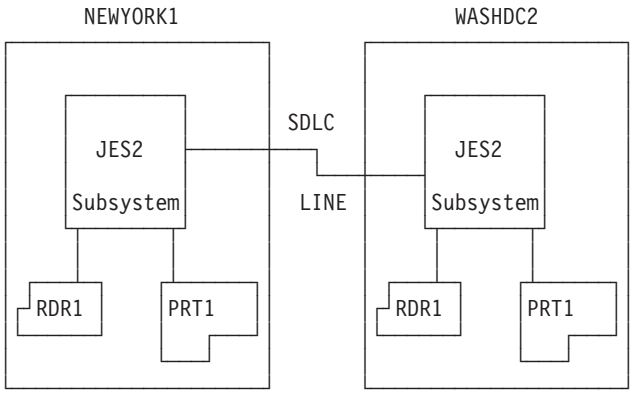


Figure 72. Basic SNA Configurations

A minimum SNA configuration is similar to the minimum BSC configuration depicted in Figure 71 on page 271. Notice that the SNA configuration (Figure 73 on page 279) also requires the four JES2 initialization parameters required in the minimum BSC configuration (NODENUM=, LINENUM=, and OWNNODE= on NJEDEF and the UNIT= on the LINE(nnnn) initialization statement). The capabilities of an SNA network are the same as those previously discussed for a BSC network. When using an SNA session to connect the nodes, supply a LINE(nnnn) initialization statement specifying the UNIT=SNA parameter in the initialization data set. You also need the LOGON(1) statement shown in Figure 73 on page 279 to specify the application name VTAM uses for this node. (In this case, it is set to the node name.)




```
NJEDEF NODENUM=2,                NJEDEF NODENUM=2,
       OWNNODE=1,                       OWNNODE=2,
       LINENUM=1                        LINENUM=1
NODE(1) NAME=NEWYORK1            NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2             NODE(2) NAME=WASHDC2
LOGON(1) APPLID=NEWYORK1         LOGON(1) APPLID=WASHDC2
LINE(1) UNIT=SNA                 LINE(1) UNIT=SNA
APPL(WASHDC2) NODE=2             APPL(NEWYORK2) NODE=1
```

Figure 73. Minimum NJE Configuration (SNA)

Special considerations for SNA NJE networking

Controlling SNA NJE networking is similar to controlling binary synchronous communication (BSC) networking. However, because VTAM controls SNA NJE application-to-application sessions, you must meet certain VTAM requirements before establishing a session between two JES2 nodes. See *z/OS Communications Server: SNA Resource Definition Reference* for descriptions of the VTAM definitions needed. To establish a session:

1. You must start VTAM before you establish a session between 2 nodes. VTAM must have the application names defined for the starting node and for all nodes eligible for application-to-application sessions with other nodes in the network.
2. The VTAM network must complete its initialization. This includes activating the IBM 3704, 3705, 3720, 3725, or 3745 Network Control Programs, synchronous data link control (SDLC) links, and all participating VTAM cross-domain resource managers.
3. LOGON(nnn) initialization statements must uniquely define each JES2 subsystem to VTAM.

   In the networking environment, each JES must have a unique application name. JES2 uses a default name if APPLID= is not specified on the LOGON(nnn) statement. You should always code the APPLID= parameter on the LOGON(nnn) statement because only one subsystem in the network can use the default name of JES2.

   The APPL(avvvvvvv) initialization statement defines the VTAM application name for each node with which an installation plans to connect. In addition, the APPL(avvvvvvv) statement specifies the specific characteristics of an application session. If an application name matches the NAME= parameter on the NODE(nnnn) statement, JES2 can create an APPL(avvvvvvv) statement dynamically during sign on through a $S N command.

Installations should ensure that the APPL(avvvvvvv) initialization statement is properly specified before attempting to initiate sign on with another node. If the APPL(avvvvvvv) provides an incorrect node name (NODE=), line (LINE=) or compaction table (COMPACT=), installations can often establish a VTAM session, but are unable to activate the NJE session.

4. LINE(nnnn) JES2 initialization statements must define (with the UNIT=SNA parameter) the paths for each NJE session.
5. NODE(nnnn) initialization statements define all NJE applications participating in the network.
6. The operator must issue a $S LOGON(nnn) (start logon) command to connect the NJE application with VTAM.

When you meet the VTAM session requirements, you can start a session with another eligible JES2 NJE application by issuing the $S N command with the appropriate application or node name specified.

Multiple parallel SNA NJE sessions between two nodes

SNA NJE protocols in JES2 prohibit multiple SNA sessions between two VTAM APPLIDs. However, using multiple APPLIDs per node, each with a unique logon identifier, you can start multiple SNA NJE sessions between two adjacent nodes. You can define up to 999 unique APPLIDs and logon identifiers for each node. In the following example Figure 74, each node has three unique LOGON(n)s and three APPLID statements which can be started in parallel on three different SNA lines.

```
+---------------+                    +---------------+
|               |                    |               |
|  NODE1        |                    |  NODE2        |
|               |                    |               |
+---------------+                    +---------------+

NJEDEF  OWNNODE=1                    NJEDEF  OWNNODE=2

LOGON1  APPL=APN1A                   LOGON1  APPL=APN2A
LOGON2  APPL=APN1B                   LOGON2  APPL=APN2B
LOGON3  APPL=APN1C                   LOGON3  APPL=APN2C

APPL(APN1A)  NODE=1                  APPL(APN1A)  NODE=1,LOGON=1
APPL(APN1B)  NODE=1                  APPL(APN1B)  NODE=1,LOGON=2
APPL(APN1C)  NODE=1                  APPL(APN1C)  NODE=1,LOGON=3
APPL(APN2A)  NODE=2,LOGON=1          APPL(APN2A)  NODE=2
APPL(APN2B)  NODE=2,LOGON=2          APPL(APN2B)  NODE=2
APPL(APN2C)  NODE=2,LOGON=3          APPL(APN2C)  NODE=2

LINE15  UNIT=SNA                     LINE25  UNIT=SNA
LINE16  UNIT=SNA                     LINE26  UNIT=SNA
LINE17  UNIT=SNA                     LINE27  UNIT=SNA
------------------                   -------------------
$S LOGON1-3                          $SLOGON1-3
$S LINE15-17                         $SLINE25-27

$S N,A=APN2A      (generates a connection between APN1A and APN2A)

$S N,A=APN2B      (generates a connection between APN1B and APN2B)

$S N,A=APN2C      (generates a connection between APN1C and APN2C)
```

Figure 74. Parallel SNA NJE sessions between two nodes

VTAM definitions for SNA NJE

JES2 uses ACF/VTAM services to communicate with other NJE nodes through SNA sessions. The APPL and MODETAB members are described in the following sections.

VTAM APPL statement

The NJE application (the JES2 subsystem) requires a few values from the VTAM APPL statement:

Table 58. VTAM APPL statement parameters necessary for a JES2 connection

Parameter	Purpose
AUTH=ACQ	Allows the NJE member to acquire the LU session.
DLOGMOD=	Points to the logmode table entry.
VPACING=	Prevents one node from overrunning another with SNA buffers by controlling the rate of transmission at both the primary and the secondary LU. **Note:** For the primary LU (the node that issued the $SN command), you must also use a non-zero SSNDPAC value from the secondary LU's logmode table entry. The SSNDPAC value provides the primary LU's inbound pacing rate.

NJE bind image

JES2 uses a standard bind image for communication with other NJE nodes. The following bind image illustrates the standard NJE session parameters. It is not necessary to specify or to include most of these parameters in the input bind image, but all are shown here for completeness.

```
MODEENT  LOGMODE=(name),TYPE=X'01',FMPROF=X'03,'TSPROF=X'03',  C
         PRIPROT=X'72',SECPROT=X'72',COMPROT=X'4020',          C
         SSNDPAC=X'03',SRCVPAC=X'03',RUSIZES=X'0000',          C
         PSNDPAC=X'03',PSERVIC=X'000000000000000000000000'
```

TYPE=X'01'
Specifies the type of bind image. In this case, X'01' indicates a non-negotiable bind. JES2 provides this default value.

FMPROF=X'03'
Specifies the FM (function management) profile. JES2 provides this default value.

TSPROF=X'03'
Specifies the TS (transmission services) profile. JES2 provides this default value.

PRIPROT=X'72'
Specifies the FM primary LU (logical unit) protocols. JES2 provides this default value.

SECPROT=X'72'
FM Secondary LU protocols. JES2 provides this default value.

COMPROT=X'4020'
Specifies the common LU protocols. JES2 provides this default value.

SSNDPAC=X'03'
Specifies the secondary LU send pacing count. Set this to a non-zero value to allow VPACING from the APPL to be used for the secondary LU to primary LU pacing count.

SRCVPAC=X'03'
Specifies the secondary LU receive pacing count. This parameter is not used for APPL-to-APPL sessions.

RUSIZES=X'0000'
Specifies the maximum RU (response or request unit) size for both inbound and outbound buffers.

The LOGMODE table entry value is not used. Instead, JES2 uses the value from the TPDEF SNABUF=SIZE= parameter, after rounding down to a valid RU size (if necessary). Acceptable RU sizes are within the range 512–32512 (in increments of 128).

PSNDPAC=X'03'
Specifies the primary LU send pacing count. This value is used for the primary LU to secondary LU pacing.

PSERVIC=X'000000000000000000000000'
Specifies the LU presentation services profile. Because no profiles are specified for NJE, you should leave the value at zeros or omit this parameter.

SNA buffer size

Each teleprocessing buffer for SNA is prefixed in storage with an RPL (request parameter list). The size of buffers for SNA RJE and SNA NJE is determined by the following:

1. Start with the SIZE= subparameter (within a range of 256-32512) on the SNABUF parameter of the TPDEF initialization statement.
2. If the SIZE= subparameter is less than any BUFSIZE= parameter value on an SNA RMT(nnnn) initialization statement, SIZE= is increased to the BUFSIZE value.
3. If this number is less than 280 and the LINENUM= parameter on the NJEDEF initialization statement is greater than zero, SIZE=280.
4. The size of the RPL prefix (256) is added to the buffer size, rounded up to a multiple of 8, and rounded down to a maximum of 4096.
5. The 'usable' teleprocessing (TP) buffer size is calculated by subtracting the RPL prefix area. This value can be displayed through a $D TPDEF operator command, which results in the $HASP839 message display.

Valid RU sizes: The actual transmission RU size for SNA NJE buffer can be further limited (rounded down) by the SNA architecture which only allows RU sizes that can be described by the formula

```
M * 2**E
```

where M (mantissa) and E (exponent) are integers between 1 and 15.

For NJE connections, the actual transmission data buffer size used is equal to the smaller buffer size of the two adjacent nodes. This is negotiated independently for each NJE session through the functional management header (FMH-4) records. Some NJE systems require that a valid RU size be used in the negotiation. Therefore, you should pick an acceptable RU sizes within the range 512–32512 (in increments of 128).

Specifying unique log-mode table entries for specific SNA connections

JES2 allows an installation to specify a different VTAM log–mode table entry for each node. The installation specifies the entry to use on the LOGMODE= parameter on the NODE(*nnnn*) or APPL(*xxxxx*) initialization statement. If you include both statements, VTAM uses the entry specified on the APPL(*xxxxx*) initialization statement for the node to locate the corresponding log mode entry. Figure 75 shows the relationship between the APPL(*xxxxx*) statement and the log mode entries.

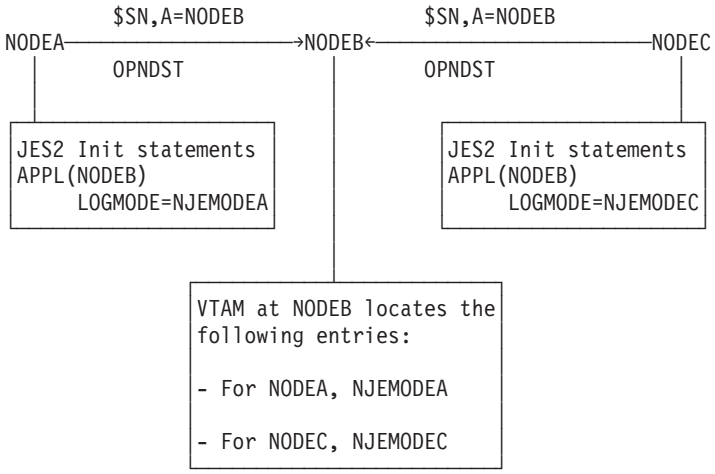


Figure 75. Using Multiple SNA Log Mode Table Entries

If you determine that specifying different log modes for adjacent nodes can improve your network performance, specify the value for the log–mode table entry for the adjacent nodes on the LOGMODE= parameter on either the NODE(*nnnn*) or APPL(*xxxxx*) initialization statement.

If you do not specify the LOGMODE= parameter on a JES2 initialization statement, the default log–mode table entry name is specified on the DLOGMOD= parameter of the VTAM APPL statement at each node.

Defining a minimum configuration for TCP/IP

A minimum TCP/IP configuration is similar to the minimum BSC or SNA configuration in Figure 71 on page 271 and Figure 73 on page 279. The TCP/IP configuration (Figure 76 on page 284) also requires the four JES2 initialization parameters required in the minimum SNA and BSC configurations (NODENUM=, LINENUM=, and OWNNODE= on NJEDEF and the UNIT= on the LINE(nnnn) initialization statement). The capabilities of a TCP/IP NJE are the same as those previously discussed for a BSC network. When using a TCP/IP session to connect the nodes, supply a LINE(nnnn) initialization statement specifying the UNIT=TCP parameter in the initialization data set. You also need the NETSRV(1) statement shown in Figure 73 on page 279 to specify the socket name that defines the IP address and ports the local node listens on.

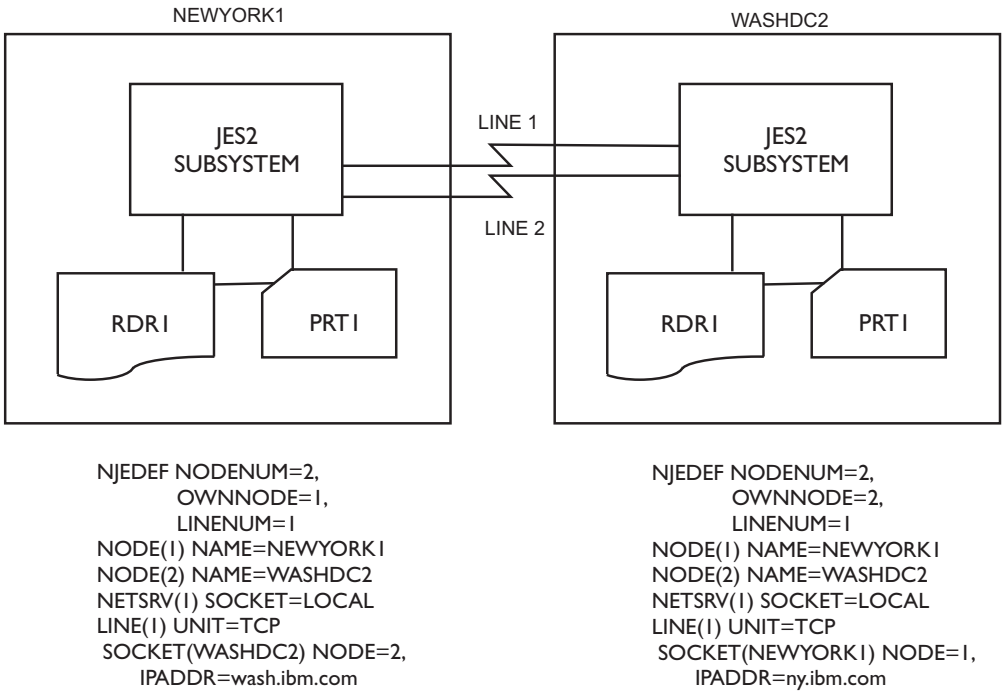


Figure 76. Basic TCP/IP Configurations

Setting up the JES2 initialization deck

You need to set up your JES2 initialization deck to include the definitions for your NJE/TCP connections. The NJE/TCP parameters work similarly to those for SNA NJE.

Table 59. Comparison of NJE/SNA parameters and NJE/TCP parameters

NJE/SNA statement or parameter	Corresponding NJE/TCP parameter
LOGON(nnn) – defines interface between JES2 at this node and VTAM	**NETSRV(nnn)** – defines interface between JES2 at this node and TCP/IP
APPL(xxxxxxxx) – maps an NJE node name to a VTAM application name. Used for both local and remote nodes. xxxxxxxx corresponds to an actual VTAM application name	**SOCKET(xxxxxxxx)** – maps an NJE node name to an IP address and port. Used for both local and remote nodes. xxxxxxxx is an abstract value used to reference the IP address and port combination.
LINE(nnnnn) UNIT=SNA – defines a logical connection	**LINE(nnnnn) UNIT=TCP** – defines a logical connection
$SN,A=xxxxxxxx – initiates an NJE connection with the specified VTAM application.	**$SN,S=xxxxxxxx** – initiates an NJE connection with the IP address and port defined on the SOCKET(xxxxxxxx) statement.

Other definitions that define your network (NJEDEF, NODE statements, additional parameters on LINE(nnnnn), etc.) are the same as for SNA and BSC connections. You can use TCP/IP connections in conjunction with SNA and BSC connections to talk to different nodes, or even as parallel connections to the same node.

NETSRV(nnn): The NETSRV statement defines an interface between JES2 and TCP/IP at this node. The code for each NETSRV device runs in its own address

space that is separate from JES2. The NETSRV statement includes a pointer to a corresponding SOCKET statement for the local node that defines the IP address and port that should be used. If you do not specify any IP address on the SOCKET, the default is to listen on all IP addresses. If you do not specify any port on the SOCKET, the default is to listen on either port 175 or 2252, depending on the setting of the SECURE= parameter. See the description of the SOCKET statement in *z/OS JES2 Initialization and Tuning Reference* for details.

You can also set the NETSRV device up to listen on a specific TCP/IP stack with the NETSRVnnn STACK= parameter when the Common INET (CINET) configuration is being used. If you do not specify this, the default is to listen on all TCP/IP stacks on this system. When only an INET configuration is being used, a specification of a stack name is ignored.

JES2 allows up to 999 NETSRV devices per JES2 image. Each device must point to a unique socket that defines a unique IP address and port combination. If you plan to use sysplex distributor or shared ports, multiple NETSRV devices within the same sysplex can listen on the same IP address and port combination. Normally, you only need a single NETSRV device to drive all of your NJE/TCP traffic; however, if you have many adjacent connections defined, you might need multiple NETSRV devices. There is a limit to the number of connections allowed per NETSRV device. That limit is based on the amount of available private storage and normally allows several hundred connections per NETSRV device. Because each connection is driven by a separate task in the NETSRV address space, the practical limit at your installation can be lower. You can also use multiple NETSRV devices to connect to individual stacks, or to isolate your SSL/TLS connections from other connections.

If you have not defined any NETSRV devices in your initialization deck, you can use the $ADD NETSRVnnn command to create them.

SOCKET(xxxxxxxx): The SOCKET statement defines an IP address and port combination that is being used for TCP/IP over NJE at some node in the network, including the local node. You need to define the following two types of SOCKETs:

- Sockets defined for the local node (for example, specifying NODE=ownnode) that are used by NETSRV devices to listen for incoming NJE connections
- Sockets defined for other nodes, which will be used on a $SN command to initiate a connection between this node and a specific IP address and port at another node.

You can specify the IP address (IPADDR=) in many different formats. You can specify an explicit IPv4 address in dotted-decimal format (for example, 9.57.1.33); you can specify and explicit IPv6 address in hex=colon format (for example 1234::5678), or you can use a symbolic name (for example, pok.ibm.com). You can specify the port (PORT=) as any number in the range of 1-32767; typically you can use the default. You can use the SECURE= parameter to define whether SSL/TLS is to be used. When SECURE=NO is used (no SSL or TLS), the default PORT= value is the value associated with the VMNET protocol, typically 175. When SECURE=YES is specified, the default is the value associated with the NJE-SSL protocol, typically 2252. If you have not defined any sockets in your init deck, you can use the $ADD SOCKET(xxxxxxxx) command to create them.

LINE(nnnn): The LINE statement defines a logical line that is used by a NJE connection. Various job and transmitter subdevices that represent data streams are associated with the line. The use of the LINE is identical to its use for SNA and BSC NJE, except that the line should specify UNIT=TCP. If you plan to define both

SNA and TCP/IP connections to adjacent nodes while testing NJE/TCP, you might need to define additional lines for this. The maximum number of lines is 65535 so you can define additional lines for this purpose. You can also use the $TLINE(nnn),UNIT= command to switch lines from SNA lines to TCP/IP lines, or the $ADD LINE(nnn) command to define additional lines.

For more details about the JES2 initialization statements, see *z/OS JES2 Initialization and Tuning Reference*.

Commands

You can use the following commands to define a minimum configuration for TCP/IP. For more details, see *z/OS JES2 Commands*.

$S NETSRV(nnn): When you start the NETSRV device with the $S NETSRV1 command, a new address space is created, and then listens for incoming connections on the IP address (IPADDR=) and port (PORT=) specified on the corresponding SOCKET statement. You can also specify START=YES on the NETSRV initialization statement so the device will be started automatically. If IPADDR= is not specified, the default is to use all IP addresses defined on this MVS image. If PORT= is not specified, the default is either 175 or 2252, depending on whether TLS is to be used (this is defined by the SECURE= parameter on the SOCKET statement). By default, JES2 provides two pre-defined sockets. SOCKET(LOCAL) uses the default IP address and port with SECURE=NO; SOCKET(LOCALTLS) uses the default IP address and port with SECURE=YES.

If TCP/IP or OMVS is not yet active, the NETSRV address space starts but waits very early in its initialization process for TCP/IP and OMVS to become active. One of the following highlighted messages displays in this case

```
IAZ0541I NETSRV1 NJETCP SERVER WAITING FOR TCP/IP
$HASP5001 NETSRV1 IS WAITING FOR OMVS INITIALIZATION
```

If you specify START=YES on your NETSRV initialization statement, you typically see one or both of these messages because OMVS and TCP/IP do not start until JES2 is active. When the NETSRV device is available to connect to other nodes, the following highlighted message displays:

```
IAZ0537I NETSRV1 NJETCP SERVER WAITING FOR WORK
```

The address space name associated with the NETSRV device is jesxSnnn, where jesx is the JES2 subsystem name and nnn is the NETSRV device number. Therefore, for example, if your subsystem name is “JES2” and you start NETSRV1, the address space name is JES2S001. Displaying the NETSRV device with the $D NETSRV command displays the address space name and address space number being used by the NETSRV device. For example:

```
$dnetsrv1
$HASP898 NETSRV1
$HASP898 NETSRV1 STATUS=ACTIVE,ASID=0026,NAME=JES2S001,
$HASP898 SOCKET=N1P1,STACK=,TRACEIO=(JES=NO,
$HASP898 COMMON=NO,VERBOSE=NO)
```

You can also request a more detailed display of the NETSRV device with the $DNETSRV,LONG command. In addition to the status of the device, it also displays the line, socket name, and internal socket ID of all active connections associated with the device on the SESSIONS= parameter.

```
$dnetsrv1,long
$HASP898 NETSRV1
$HASP898 NETSRV1 STATUS=ACTIVE,ASID=0026,NAME=JES2S001,
```

```
$HASP898 SOCKET=N1P1,STACK=,
$HASP898 SESSIONS=(LNE2020/N2P1/S4),
$HASP898 TRACEIO=(JES=NO,COMMON=NO,
$HASP898 VERBOSE=NO),
```

$SLNE(nnnnn): This command activates the logical line and makes it usable for NJE/TCP processing. There is no difference from SNA and BSC processing other than the specification of UNIT=TCP on the line definition.

$SN,SOCKET=xxxxxxxx: This commands initiates networking with the IP address and port specified on the SOCKET(xxxx) statement. It also verifies that the node being connected to is the node specified on the SOCKET(xxxx) NODE= parameter.

The $SN command selects any started TCP/IP line unless a line has been dedicated to this particular node or socket. To dedicate the lines, specify the LINE= parameter on either the NODE statement or the SOCKET statement. When a line is dedicated to a connection with either of these parameters, the line is started automatically if a $SLINE command has not already been issued. Dedicated lines are also honored on inbound connections.

You can also specify a specific line to use on the $SN command itself, for example, $SN,LNE1,SOCKET=xxxxx.

The $SN command uses NETSRV1 for the NJE connection by default. To use a different NETSRV, you can specify the NETSRV=nnn parameter on either the NODE statement or the SOCKET statement. Inbound connections are always associated with the NETSRV on which the request was received.

JES2 commands: For other commands, see *z/OS JES2 Commands* for more details.

Example

Here is a simple two node network and the JES2 initialization statements required to communicate between the two nodes using NJE/TCP:

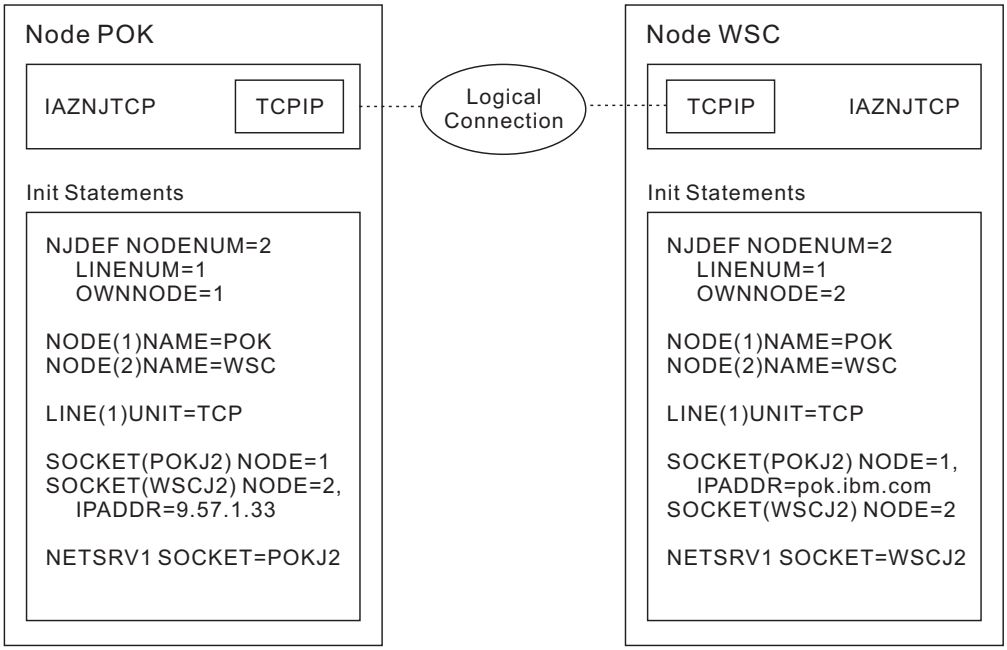


Figure 77. Example of a two node network and the JES2 initialization statements required to communicate between the two nodes using NJE/TCP

This example defines two nodes, POK and WSC. Each node has similar NJEDEF and NODE definitions, with NODE(1) defined as POK and NODE(2) defined as WSC. Each node then also needs to define a NETSRV device, a local socket defining the IP address and port that NETSRV is to listen on, and a non-local socket defining the IP address and port of each connection to be established with other network nodes. In this case, there is only one of those.

On node POK, a single NETSRV device is defined, pointing to socket POKJ2. SOCKET(POKJ2) specifies the local node number (1) on the NODE= parameter, and takes the default values for IPADDR= and PORT=. Therefore, when NETSRV1 is eventually started, it listens on all IP addresses, and because SECURE=YES is not specified, it uses the port associated with VMNET (175). The non-local socket for the connection to node WSC, defined by the SOCKET(WSCJ2) statement, indicates that the connection is associated with node 2 (per the NODE=2 keyword), and specifies an IP address of 9.57.1.33. In this case, the explicit IP address is specified in dotted-decimal format. Because PORT= is not specified, the default value (VMNET, or 175) is also used.

On node WSC, a single NETSRV device is defined, pointing to socket WSCJ2. SOCKET(WSCJ2) specifies the local node number (2) on the NODE= parameter, and takes the default values for IPADDR= and PORT=. The non-local socket for the connection to node POK, defined by the SOCKET(POKJ2) statement, indicates that the connection is associated with node 1 (per the NODE=1 keyword), and specifies an IP address of pok.ibm.com. In this case, a symbolic address is specified rather than an explicit address; either can be used. Because PORT= is not specified, the default value (VMNET, or 175) is also used.

To connect the two nodes, processing is very similar to SNA and BSC NJE. You can use the $SLINEnnn command to activate the logical line that is to be used for the connection, and the $SN command to start networking on the connection.

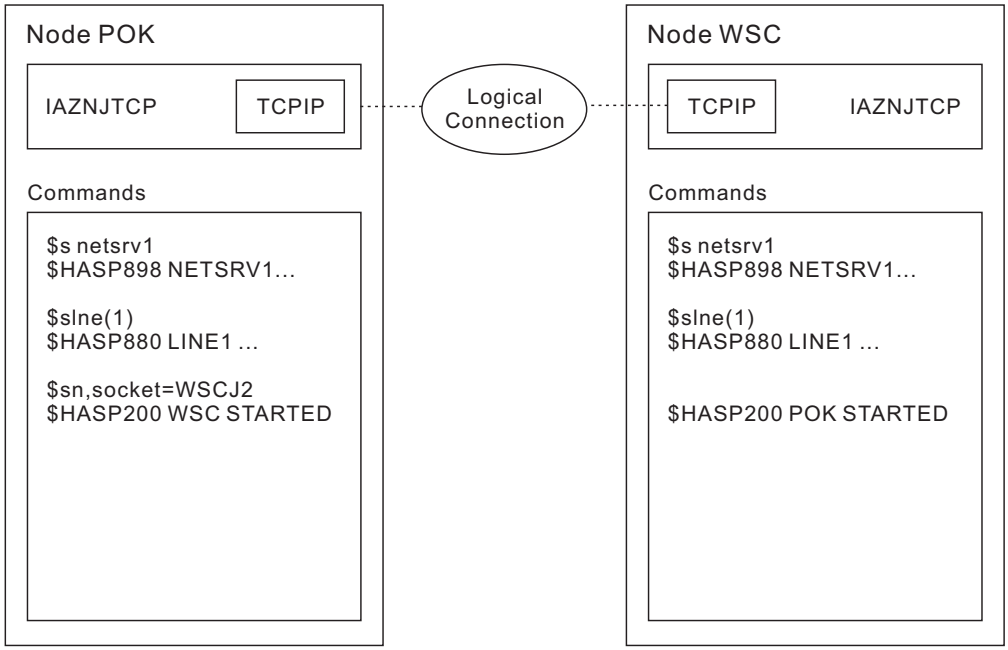


Figure 78. Example of a two node network and the JES2 commands required to start networking

First, on each end of the connection, you must start the NETSRV device and logical line with the $SNETSRV1 and $SLNE1 commands, respectively.

Next, when you issue the $SN,SOCKET=WSCJ2 command, this tells NETSRV1 (by default) to initiate a connection between itself and the IP address that is provided on the SOCKET(WSCJ2) statement. Because NETSRV1 on node WSC is listening on this IP address and port, you can establish a connection successfully. You can also establish the connection by issuing $SN,SOCKET=POK from node WSC.

Displaying the attributes of the NETSRVs, lines, and sockets on both sides of the connection results in the following displays:

```
On Node POK:

16.27.24           $dnsv1
16.27.24           $HASP898 NETSRV1
$HASP898 NETSRV1    STATUS=ACTIVE,ASID=0024,NAME=JES2S001,
$HASP898            SOCKET=LOCAL,STACK=,TRACEIO=(JES=YES,
$HASP898            COMMON=YES,VERBOSE=YES)
16.27.42           $dsock(pokj2,wscj2)
16.27.42           $HASP897 SOCKET(POKJ2)
$HASP897 SOCKET(POKJ2)     STATUS=ACTIVE/NETSRV1,
$HASP897                   IPADDR=::9.57.1.235,
$HASP897                   PORTNAME=VMNET,SECURE=NO,LINE=0,
$HASP897                   NODE=1,REST=0,NETSRV=0
16.27.42           $HASP897 SOCKET(WSCJ2)
$HASP897 SOCKET(WSCJ2)     STATUS=ACTIVE/LNE1/NETSRV1,
$HASP897                   IPADDR=9.57.1.161,PORT=175,
$HASP897                   PORTNAME=VMNET,SECURE=NO,LINE=0,
$HASP897                   NODE=2,REST=0,NETSRV=0,SOCKID=S3

On Node WSC:

16.27.24           $dnsv1
16.27.24           $HASP898 NETSRV1
```

Note the following in the previous display:

- First, even though the local sockets (POKJ2 on node POK and WSCJ2 on node WSC) did not specify an IP address in the initialization deck, a value is now displayed for each, rather than the *LOCAL value that was displayed before the NETSRV device was started. This is the primary IP address associated with the NETSRV device.
- Second, the PORT= value is now filled in for the remote socket associated with each node. On POK, the value for WSCJ2 is now displayed as PORT=175 (this is the value associated with VMNET). On WSC, however, the value for POKJ2 is displayed as PORT =1038, which is not associated with anything specified in the JES2 parameters. The reason for this is that TCP/IP can assign an ephemeral port number to the socket on the receiving node. If the connection drops, the value reverts to the value you specified, in case the connection needs to be re-established.
- Finally, the STATUS= parameter on the SOCKET display shows the line or NETSRV associated with the SOCKET, or both. For local sockets, only the NETSRV that is listening on that socket is displayed; for remote sockets, both the associated line and associated NETSRV are displayed.

Note: When associating an incoming request with a SOCKET, you can only use the NODE= and SECURE= values to make that association; the IPADDR= and PORT= values are ignored. This is because the inbound IP address might not match what is specified on the SOCKET statement because of the presence of firewalls, NAT translation, and so on. The inbound port might not match as TCP/IP assigns an ephemeral port value to the inbound socket, which in general cannot be predicted. It is also possible that for an inbound connection, neither the IP address nor the port matches the value that would have to be specified if the connection were initiated from this end of the connection. You must specify the correct values, though, to initiate the connection successfully from either end of the connection.

Special considerations for TCP/IP NJE

Controlling TCP/IP NJE networking is similar to controlling SNA networking. In addition to VTAM, it is necessary to start TCP/IP in order to communicate using TCP/IP. TCP/IP NJE configurations also require that a JES2 SOCKET statement be defined which specifies the IP address and port (typically 175 or 2252) that will be used by both the local node and the partner node. The IP address and port used by the local node must then be specified on a SOCKET statement associated with the NETSRV in order for JES2 to listen for TCP connections. The sockets LOCAL and LOCALTLS are defined automatically by JES2 initialization and default to listening on all IP addresses and TCP/IP stacks associated with the MVS image. For definitions regarding the partner node, it is necessary to know the IP name or address of the partner node, either as a symbolic name (such as wash.ibm.com) or as an explicit address in either dotted decimal format (for IPv4) or hex-colon format (for IPv6). You can also increase security in your network by using SSL or TLS across this NJE link. For more information, see the TLS data protection information in *z/OS Communications Server: IP Configuration Guide*.

Setting up TCP/IP

Before implementing NJE/TCP, you need to set up TCP/IP on your MVS system. If you are already using TCP/IP for other applications, there are a few additional definitions required to enable JES2 to perform NJE/TCP. Your security administrator should determine the actual values to use. For more information, see *z/OS Security Server RACF Security Administrator's Guide* and *z/OS UNIX System Services Planning*. The additional definitions follow:

Check the buffer sizes defined in your TCP/IP definitions. In particular, you must set the TCPCONFIG MAXRECVBUFRSIZE and RECVBUFRSIZE parameters to at least 512, and ideally set them to a value of 32768 or higher.

The NJE/TCP implementation also supports virtual IP addresses (VIPA addresses), and the use of sysplex distributor to share IP addresses throughout a SYSPLEX. There is no additional JES2 setup to take advantage of either of these features other than to specify the appropriate IP address or host name on the SOCKET statement.

For more information, see *z/OS Communications Server: IP Configuration Guide*.

Defining a NETSRV

When a NETSRV device is started, an address space is created to interface with JES2 and TCP/IP. The address space name is jesxSnnn where jesx is the name of the owning JES2 address space and nnn is the subscript on the NETSRV(nnn) statement. For example, NETSRV(1) in subsystem "JESA" results in an address space named JESAS001. This address space uses z/OS UNIX systems services to communicate with TCP/IP. Therefore, the userid that is associated with the address space needs to have an OMVS segment defined in RACF. Here is an example:

1. Define or alter a UserID with an OMVS segment:
   - Add a new UserID

     ```
     ADDUSER started-userid NAME('netsrv address space') DFLTGRP(sys1) OMVS(UID(0))
     ```
   - Alter an existing UserID

     ```
     ALTUSER started-userid OMVS(UID(0))
     ```
2. Complete the following items through SETROPTS if they are not already done:
   - Use RACLIST for the STARTED class.
   - Activate the STARTED class.
   - Enable generic processing for the STARTED class to the GENERIC and GENCMD resource class.
3. Define The STARTED Profile for NETSRV:

   ```
   RDEFINE STARTED jesas001.* STDATA(USER(started-userid))
   ```

   Be sure to refresh the STARTED class after the RDEFINE.

   Note: The prior example uses UID(0). NETSRV does not require UID(0). If non-UID(0) is used, after performing the prior three steps, perform the following two steps. Your security administrator should determine the actual values to use for your environment. For more information, see *z/OS Security Server RACF Security Administrator's Guide* and *z/OS UNIX System Services Planning*.
4. Ensure that the NETSRV UserID has access to the TCP/IP Ports that are used by NETSRV:
   - For information about PORT statement (SAF option) and TCPCONFIG statement (RESTRICTLOWPORTS parameter), see chapter TCP/IP profile in *z/OS Communications Server: IP Configuration Reference*.
   - If SAF is used to protect the port, RACLIST and activate the SERVAUTH class. If generic profiles are used, add SERVAUTH to GENERIC and GENCMD resource classes. For more information, see *z/OS Security Server RACF Security Administrator's Guide*.
5. Ensure that the NETSRV UserID has access to the following BPX resource:

   ```
   FACILITY BPX.CONSOLE (Read)
   ```

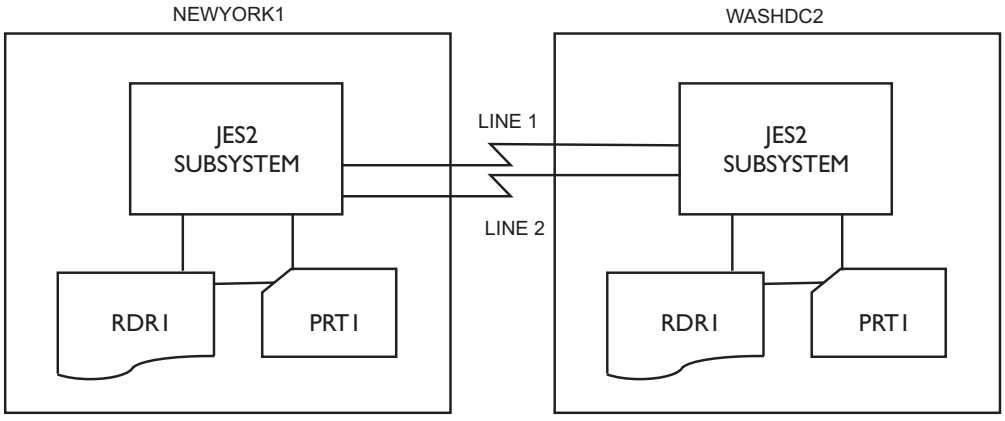
Finally, you also need to consider adjusting the service class that is assigned to the address space. The address space uses separate subtasks for each connection. If it is supporting a large number of connections, the address space can use as much CPU resource as the system allows it to transmit and receive data. If its priority is too high, it could lock out other work in the system.

NETSRV security considerations

NETSRV can support both secure and non-secure connections. You must meet the following minimum requirements to establish a secure connection:

- On the initiating (client/sending) node side, SECURE=YES must be coded on the socket statement that is associated with the listening (server/receiving) NODE.
- On the listening (server/receiving) node side:
  - A NETSRV or equivalent must exist that can accept secure connections from a remote JES2 node.
  - If the listening (server/receiving) node is a JES2 node, you must use the following settings:
    - SECURE=YES on the local node socket that is associated with the NETSRV statement.
    - SECURE=YES on the remote node socket or sockets that the connections will be initiated from.

Note: Meeting the previous requirements ensures that a successful secure connection is established only when started from the initiating (client/sending) node. See Figure 79 for an example of this configuration.




```
NJEDEF NODENUM=2,
       OWNNODE=1,
       LINENUM=1
NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2
LINE(1) UNIT=TCP
SOCKET(WASHDC2) NODE=2,
    IPADDR=wash.ibm.com,
    SECURE=YES
```

```
NJEDEF NODENUM=2,
       OWNNODE=2,
       LINENUM=1
NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2
NETSRV(1) SOCKET=LOCALTLS
LINE(1) UNIT=TCP
SOCKET(LOCALTLS) NODE=2,
    IPADDR=wash.ibm.com,
    SECURE=YES
SOCKET(NEWYORK1) NODE=1,
    IPADDR=ny.ibm.com,
    SECURE=YES
```

Figure 79. Secure connection from client node

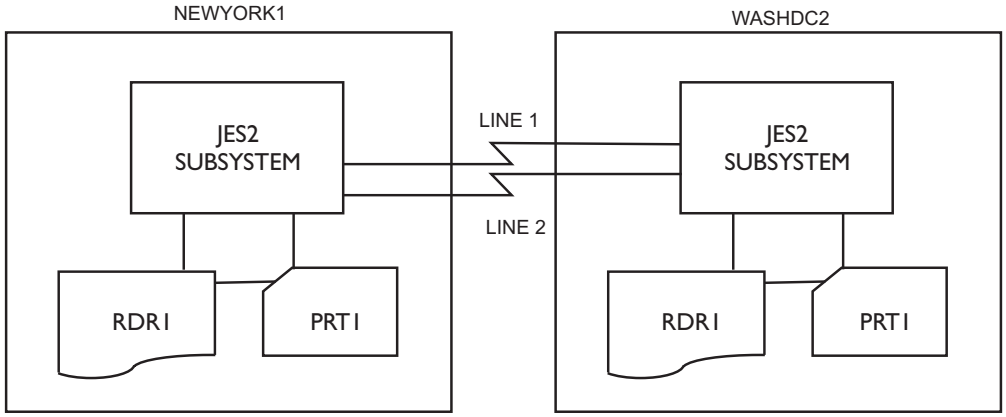
The previous configuration, Figure 79, can initiate a secure connection only when started from node NEWYORK1. To enable a secure transmission, the PORT

name/number must match between the local socket on the listening side (WASHDC2) and corresponding remote socket on the sending node (NEWYORK1). Using LOCALTLS allows JES2 to automatically utilize the default port for secure transmission.

To establish a secure connection from either node, the following symmetrical configuration of the sockets at both ends is required:

- Each node must have a local NETSRV with the associated socket statement specifying SECURE=YES.
- Each node must have a defined socket for the remote node also specifying SECURE=YES.

See Figure 80 for an example of this configuration.



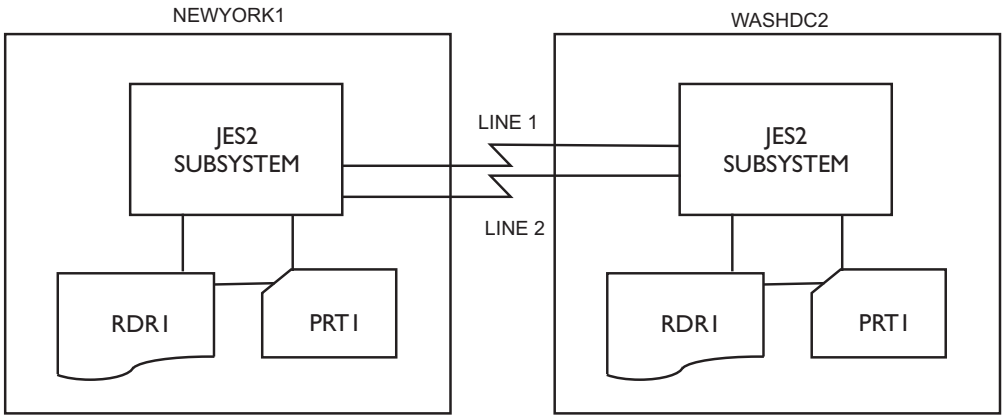

```
NJEDEF NODENUM=2,
       OWNNODE=1,
       LINENUM=1
NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2
NETSRV(1) SOCKET=LOCALTLS
LINE(1) UNIT=TCP
SOCKET(LOCALTLS) NODE=1,
     IPADDR=ny.ibm.com,
     SECURE=YES
 SOCKET(WASHDC2) NODE=2,
     IPADDR=wash.ibm.com,
     SECURE=YES
```

```
NJEDEF NODENUM=2,
       OWNNODE=2,
       LINENUM=1
NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2
NETSRV(1) SOCKET=LOCALTLS
LINE(1) UNIT=TCP
SOCKET(LOCALTLS) NODE=2,
     IPADDR=wash.ibm.com,
     SECURE=YES
 SOCKET(NEWYORK1) NODE=1,
     IPADDR=ny.ibm.com,
     SECURE=YES
```

Figure 80. Secure connection from either node

The previous configuration, Figure 80, can initiate a secure connection when started from either side. To enable a secure transmission, the PORT name or number must match between the local socket on the listening side (WASHDC2) and corresponding remote socket on the sending node (NEWYORK1). Using LOCALTLS allows JES2 to automatically utilize the default port for secure transmission.

You can also define a NETSRV that can tolerate both secure and non-secure connections, depending on which sockets are utilized in starting the NETSRV connection. See Figure 81 on page 294 for an example of this configuration.




```
NJEDEF NODENUM=2,
       OWNNODE=1,
       LINENUM=1
NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2
NETSRV(1) SOCKET=LOCALTLS
LINE(1) UNIT=TCP
SOCKET(LOCALTLS) NODE=1,
     IPADDR=ny.ibm.com,
     SECURE=YES
 SOCKET(WASHDC2A) NODE=2,
     IPADDR=wash.ibm.com,
     SECURE=YES
SOCKET(WASHDC2B) NODE=2,
     IPADDR=wash.ibm.com,
     SECURE=NO, PORT=2252
```

```
NJEDEF NODENUM=2,
       OWNNODE=2,
       LINENUM=1
NODE(1) NAME=NEWYORK1
NODE(2) NAME=WASHDC2
NETSRV(1) SOCKET=LOCALTLS
LINE(1) UNIT=TCP
SOCKET(LOCALTLS) NODE=2,
     IPADDR=wash.ibm.com,
     SECURE=YES
 SOCKET(NEWYORK1A) NODE=1,
     IPADDR=ny.ibm.com,
     SECURE=YES
SOCKET(NEWYORK1B) NODE=1,
     IPADDR=ny.ibm.com,
     SECURE=NO, PORT=2252
```

Figure 81. Secure and non-secure connections

In Figure 81, if node NEWYORK1 issues a $SN,SOCKET=WASHDC2A request, a secure connection would be established. However, if node NEWYORK1 issues a $SN,SOCKET=WASHDC2B request, a non-secure connection would be established via conventional secure port 2252.

SSL and TLS

Use of SSL and TLS by NJE/TCP is through Application Transparent-Transport Layer Security (AT-TLS). All of the SSL/TLS definitions are defined in the TCP/IP and security profiles, rather than in JES2 parameter statements. The only definition required in JES2 is SECURE=YES on the appropriate SOCKET(xxxxxxxx) statements as discussed in "NETSRV security considerations" on page 292.

To exploit this functionality, you need to code the TCPCONFIG TTLS parameter in your TCP/IP configuration, and issue RACF commands to configure AT-TLS. For more information, see *z/OS Communications Server: IP Configuration Guide*.

Displaying information about a network

Use the following commands to display attributes of both the local node and remote nodes throughout the network. A brief overview of the console display created by each command follows. For more information on using each command, see *z/OS JES2 Commands*.

$D NJEDEF
Display NJE global parameters

$D NODE
Display node attributes

$D SOCKET
Display socket attributes

$D APPL
Display JES2 applications for JES2 NJE

$D DESTID
Display symbolic destination identifiers

$D PATH
Display active paths in a network

$D CONNECT
Display NJE connections

$N,D=node,'$D *xxx*'
Send a display command to another (JES2) node

Displaying NJE global parameters

Specify the display NJEDEF ($D NJEDEF) command to display the global parameters that apply to all NJE sessions.

```
$d njedef
$HASP831         OWNNAME=POK,OWNNODE=1,DELAY=120,
$HASP831         HDRBUF=(LIMIT=10,WARN=80,FREE=10),JRNUM=1,
$HASP831         JTNUM=3,SRNUM=4,STNUM=3,LINENUM=3,MAILMSG=NO,
$HASP831         MAXHOP=0,NODENUM=5,PATH=1,RESTMAX=79992000,
$HASP831         RESTNODE=100,RESTTOL=0,TIMETOL=1440
```

Displaying node attributes

Specify the display node ($D NODE) command without any parameters to display all the attributes specified on a NODE(nnnn) initialization statement (and operator command settings) and its status (such as what line can be used to reach that node).

```
$d node(wscvm)
$HASP826 NODE(2) 844
$HASP826 NODE(2)     NAME=WSCVM,STATUS=(VIA/LINE36),AUTH=(
$HASP826             DEVICE=NO,JOB=YES,NET=NO,SYSTEM=NO),
$HASP826             TRANSMIT=BOTH,RECEIVE=BOTH,HOLD=NONE,
$HASP826             PENCRYPT=NO,ENDNODE=NO,REST=0,SENTREST=ACCEPT
$HASP826             COMPACT=0,LINE=0,LOGMODE=,PASSWORD=(
$HASP826             VERIFY=(NOTSET),SEND=(NOTSET)),
$HASP826             PATHMGR=NO,PRIVATE=NO,SUBNET=VNET,TRACE=NO
```

If you use filtering, the display is more tabular:

```
$d node(pok*),name,status

$HASP826 NODE(3172)  NAME=POK,STATUS=(VIA/LINE36)
$HASP826 NODE(3178)  NAME=POKCAD1,STATUS=(VIA/LINE36)
$HASP826 NODE(3182)  NAME=POKMCR,STATUS=(VIA/LINE36)
$HASP826 NODE(3184)  NAME=POKMVSSP,STATUS=(VIA/LINE36)
$HASP826 NODE(3188)  NAME=POKVMCR3,STATUS=(VIA/LINE36)
⋮
```

Displaying JES2 applications for SNA NJE

Use the display SNA NJE applications ($D APPL) command to display applications known to JES2 (through the APPLID initialization statement):

```
$D APPL(*),NODE=1
$HASP821 APPL(WZ1JNJE)   NODE=1,COMPACT=0,LINE=0,LOGMODE=,REST=0
```

Displaying TCP/IP sockets

Specify the $D SOCKET command to display the TCP/IP addresses and ports known to JES2:

```
$d socket(*)|$dsock
$HASP897 SOCKET(LOCAL)
$HASP897 SOCKET(LOCAL)    IPADDR=*LOCAL,PORTNAME=VMNET,
$HASP897                  SECURE=NO,LINE=0,NODE=1,REST=0,
$HASP897                  NETSRV=0
$HASP897 SOCKET(LOCALTLS)
$HASP897 SOCKET(LOCALTLS) IPADDR=*LOCAL,PORTNAME=NJENET-SSL,
$HASP897                  SECURE=YES,LINE=0,NODE=1,REST=0,
$HASP897                  NETSRV=0
$HASP897 SOCKET(POK)
$HASP897 SOCKET(POK)      IPADDR=9.117.234.87,PORTNAME=VMNET,
$HASP897                  SECURE=NO,LINE=0,NODE=1,REST=0,
$HASP897                  NETSRV=0
$HASP897 SOCKET(WSC)
$HASP897 SOCKET(WSC)      IPADDR=9.117.234.95,PORTNAME=VMNET,
$HASP897                  SECURE=NO,LINE=0,NODE=2,REST=0,
$HASP897                  NETSRV=0
```

Display symbolic destination identifiers

Specify the display destid ($D DESTID) command to display the symbolic destination identifiers that have been defined at a particular node.

```
$d destid(*)
$HASP822 DESTID(POK)       DEST=N1,STATUS=NODENAME+DESTID,
          $HASP822 DESTID(REMOTE20)  DEST=R20,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE21)  DEST=R21,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE22)  DEST=R22,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE23)  DEST=R23,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE24)  DEST=R24,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE25)  DEST=R25,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE26)  DEST=R26,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE27)  DEST=R27,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE34)  DEST=R34,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(REMOTE5)   DEST=R5,STATUS=DESTID,PRIMARY=NO
          $HASP822 DESTID(SANJOSE) 052
 $HASP822 DESTID(SANJOSE)   DEST=N3,STATUS=NODENAME+DESTID,
 $HASP822                   PRIMARY=NO
          $HASP822 DESTID(WSC) 053
 $HASP822 DESTID(WSC)       DEST=N2,STATUS=NODENAME+DESTID,
 $HASP822                   PRIMARY=NO
```

Displaying active paths in a network

Use the display path ($D PATH) command to display all active paths to the node (or list of nodes), the total resistance of each path, and the status of each path. You can use this command first to determine if there is a connection to another path and what path(s) or route(s) exist. Note that a comma between nodes indicates a direct path, while an ellipsis (...) indicates that nodes must travel through a subnet to transmit and receive data.

```
$d path(*)
$HASP231 PATH(SANJOSE)  STATUS=(THROUGH LINE11),REST=120,
$HASP231                PATH=(POK,FISHKILL,BEACON,NEWBURGH,
$HASP231                CHICAGO,SANFRAN...SANJOSE)
$HASP231                STATUS=(THROUGH LINE20),REST=150,
$HASP231                PATH=(POK,NYC,SANDIEGO...SANJOSE)

$HASP231 PATH(SANDIEGO) STATUS=(THROUGH LINE11),REST=120,
```

```
$HASP231               PATH=(POK,FISHKILL,BEACON,NEWBURGH,
$HASP231               CHICAGO,SANFRAN...SANDIEGO)
$HASP231               STATUS=(THROUGH LINE20),REST=150,
$HASP231               PATH=(POK,NYC,SANDIEGO)

$HASP231 PATH(MOSCOW)  STATUS=(THROUGH LINE20),REST=300,
$HASP231               PATH=(POK,NYC,LONDON,PARIS,ROME...VENICE,
$HASP231               BRUSSELS,BONN...BERLIN,GLASGOW,LENINGRD
$HASP231               ...MOSCOW)

$HASP231 PATH(BAGHDAD) STATUS=(UNCONNECTED)
```

Displaying NJE connections

Use the display connection ($D CONNECT) command to display the connection between a pair of nodes or all of the connections from a given node. You can also use the $D CONNECT command to display the type of connection; the time and date the connection was made; and when this node received the information.

```
$d connect,na=wsc

$HASP815 CONNECT NODEA=WSC,MEMBERA=1,NODEB=POK,
$HASP815         MEMBERB=1,REST=100,STATUS=ACTIVE,STATE=ACTIVE
$HASP815         STATIC=NO,PATHMGR=YES,CES=(2003/045,12:45:23),
$HASP815         TIME=(2003/045,13:20:18)

$HASP815 CONNECT NODEA=WSC,MEMBERA=1,NODEB=SANJOSE,
$HASP815         MEMBERB=1,REST=100,STATUS=ACTIVE,STATE=ACTIVE
$HASP815         STATIC=NO,PATHMGR=YES,CES=(2003/045,12:42:12),
$HASP815         TIME=(2003/045,13:20:19)

$HASP815 CONNECT NODEA=ENDICOTT,MEMBERA=1,NODEB=WSC,
$HASP815         MEMBERB=1,REST=8000,STATUS=ACTIVE,STATE=PENDING
$HASP815         STATIC=YES,PATHMGR=YES,TIME=(2003/045,12:05:10)
```

The $HASP815 message displays node WSC's connections to nodes POK, SANJOSE, and ENDICOTT. The NA= and NB= display parameters are interchangeable: JES2 displays 'WSC' on the third connection, even though it is the node B connected to ENDICOTT, not the Node A specified on the command. Only the connection between ENDICOTT and WSC is a static connection made through a CONNECT initialization statement.

JES2 displays the time when NodeA and NodeB established a connection through the CES= parameter, and displays when this node last obtained new status information about the connection.

Sending a display command

Use the $N command to send a command to another node and get a display of the NJE parameters that apply to all nodes on a network. For example, the NJEDEF and NODE definitions on adjacent nodes:

```
$N,D=adjacent_nodename,'$D NJEDEF'
$N,D=adjacent_nodename,'$DNODE(our_nodename)'

$N,D=WSCMVS,'$D NJEDEF'
N4   M1 $HASP831 NJEDEF  OWNNAME=POK,OWNNODE=1,DELAY=120,
N4   M1 $HASP831         HDRBUF=(LIMIT=10,WARN=80,FREE=10),JRNUM=1,
N4   M1 $HASP831         JTNUM=3,SRNUM=4,STNUM=3,LINENUM=3,MAILMSG=NO,
N4   M1 $HASP831         MAXHOP=0,NODENUM=5,PATH=1,RESTMAX=79992000,
N4   M1 $HASP831         RESTNODE=100,RESTTOL=0,TIMETOL=1440

$N,D=WSCMVS,'$DNODE(MVSJES2)'
N4   M1 $HASP826 NODE(9)     NAME=MVSJES2,STATUS=(VIA/LINE5),AUTH=(
N4   M1 $HASP826             DEVICE=NO,JOB=YES,NET=NO,SYSTEM=NO),
```

```
N4   M1 $HASP826            TRANSMIT=BOTH,RECEIVE=BOTH,HOLD=NONE,
N4   M1 $HASP826            PENCRYPT=NO,ENDNODE=NO,REST=O,
N4   M1 $HASP826            SENTREST=ACCEPT,COMPACT=0,LINE=0,
N4   M1 $HASP826            LOGMODE=,PASSWORD=(VERIFY=(SET),SEND=(SET)),
N4   M1 $HASP826            PATHMGR=YES,PRIVATE=YES,SUBNET=,TRACE=NO
```

Connecting the network

After the nodes are defined, each node must be connected to form a network. You connect a node to the network using initialization statements and operator commands.

There are two ways to connect nodes to a network:

- Dynamic connections (see "Starting BSC dynamic connections", "Starting SNA dynamic connections," and "Starting TCP/IP dynamic connections" on page 299.)
- Static connections (see "Static connections" on page 306)

Dynamic connections are created when JES2 nodes sign on to one another through the $S N command, while static connections are defined during initialization through the CONNECT statement and dynamically by operators through the $ADD CONNECT command.

The JES2 network resource monitor can assist you in starting and restarting NJE and RJE networking devices and NJE connections. For details, refer to "Using the network resource monitor" on page 299.

JES2 also keeps an internal representation of the network, which is controlled by the JES2 path manager.

Starting BSC dynamic connections

At connection time, operators at both nodes involved in the connection must issue the start line command. A $S LINE(nnnn) command instructs JES2 to prepare a line for transmission. For a bisynchronous (BSC) NJE connection, the UNIT= parameter on the LINE(nnnn) initialization statement specifies the address of the line you want to start.

After the line starts, the operator at either NJE node must issue the $S N command to specify the name of the BSC line and any password information required to make the connection. If a dedicated line exists but is drained, it will be started automatically and a connection attempted by the $S N,N= node command. For a BSC connection, the $S N command directly invokes the NJE network path manager. If the other end of the line is started, the connection is made.

Starting SNA dynamic connections

For an SNA NJE session, you must define a logical SNA line; that is, you must have added UNIT=SNA on the LINE(nnnn) initialization statement. If the line is not dedicated, (no LINE= parameter specified on either the NODE(nnnn) or APPL(avvvvvvv) initialization statement), an operator must enter a $S LINE(nnnn) command to start the line before attempting to establish any SNA connections.

After the line starts, an operator at either NJE node issues a start networking ($S N) command to initiate an SNA NJE application-to-application session. When entering the $S N command, the operator can specify one of two names to identify the other node:

- The unique application name (on the A= parameter) by which VTAM knows JES2 at the other end. (The JES2 LOGON(1) initialization statement at the other node uses the application name to identify JES2 as a cross-domain resource to VTAM.)
- The node name if only one application exists at that node.

If a dedicated line exists but is drained, it will be automatically started by the $S N command. For SNA application-to-application sessions, JES2 invokes the path manager after VTAM establishes the SNA session.

Starting TCP/IP dynamic connections

For a TCP/IP NJE session, you must define a logical TCP/IP line; that is, you must have added UNIT=TCP on the LINE(nnnn) initialization statement. If the line is not dedicated, (no LINE= parameter specified on either the NODE(nnnn) or SOCKET(avvvvvvv) initialization statement), an operator must enter a $S LINE(nnnn) command to start the line before attempting to establish any TCP/IP connections.

You need to match sockets on each participating node that corresponds to the TCP lines being used. Socket connection is needed by the $SN command, therefore, a socket connection needs to be established before it can be issued.

After the line starts and the socket is established, an operator at either NJE node issues a start networking ($S N) command to initiate a TCP/IP NJE session. When entering the $S N command, the operator can specify one of two names to identify the other node:

- The unique socket name (on the S= parameter) for the SOCKET(xxxxxxxx) statement that defines the IP address and port at the other end of the connection.
- The node name if at least one socket exists at that node.

If a dedicated line exists but is drained, it will be automatically started by the $S N command. For TCP/IP sessions, JES2 invokes the path manager after Communications Manager establishes the TCP/IP session.

Using the network resource monitor

The JES2 network resource monitor allows you to automatically start and restart JES2 networking devices (NJE and RJE) and NJE connections, without supplying any additional code. The network resource monitor also supports JES2 network commands, which can be run to control the entire JES2 network and to facilitate a transition to TCP/IP usage.

Starting and restarting networking devices automatically

The START parameter allows you to automatically start networking devices during JES2 initialization. Specifying the START=YES|NO parameter on the LINE, LOGON, and NETSERV initialization statements indicates whether (YES) or not (NO) these networking devices will be started when JES2 starts processing. The default value is NO.

The RESTART parameter allows you to automatically start networking devices after JES2 initialization, and to restart them if they become drained. Specifying the RESTART=(YES|NO,*interval*) parameter on the LINE, LOGON, and NETSERV initialization statements indicates whether (YES) or not (NO) these networking devices will be started by the network resource monitor, and restarted if they become drained for longer than the specified time interval. The default value is

NO. The RESTART=(YES | NO,*interval*) parameter can also be specified with the JES2 $ADD, $P, and $T commands to set or modify this networking device attribute. Specifying RESTART=(YES) overrides a specification of START=(NO).

The RESTART parameter *interval* value is the approximate time, in minutes, between attempts to restart a networking device. The valid range for the *interval* value is 0-1440 minutes. A value of 0 (default) indicates that the interval value from the NJEDEF CONNECT parameter is used. The valid range for the NJEDEF CONNECT parameter *interval* value is 1-1440 minutes, and the default value is 10 minutes.

When JES2 attempts to automatically start or restart a networking device, JES2 issues a $HASP568 message similar to the following example:

```
$HASP568 LNEn - AUTOMATIC RESTART
```

When an automatic attempt to start or to restart a networking device fails, JES2 issues a $HASP569 message similar to the following example:

```
$HASP569 LNEn - LINE COULD NOT BE ALLOCATED, RC=07
                NEXT ATTEMPT: 13 APR 2011 AT 10:16
```

If RESTART=YES has been specified for a networking device that is currently drained, the $D command displays the projected restart time, in the following format:

```
RESTART=(YES,0,2011.103,10:16)
```

For more information on using each initialization statement, refer to *z/OS JES2 Initialization and Tuning Reference*. For more information on using each command, refer to *z/OS JES2 Commands*.

Starting and restarting NJE connections automatically

The CONNECT parameter allows you to automatically start NJE connections after JES2 initialization, and restart them if they become disconnected. Specifying the CONNECT=(YES|NO|DEFAULT,*interval*) parameter on the APPL, LINE, and SOCKET initialization statements indicates whether (YES) or not (NO) these NJE connections will be initially started by the network resource monitor, and restarted if they become disconnected for longer than the specified time interval. The CONNECT=(YES|NO|DEFAULT,*interval*) parameter can also be specified with the JES2 $ADD and $T APPL, LINE, and SOCKET commands, and with the $E LINE command, to set or to modify this NJE connection attribute. The default value is CONNECT=DEFAULT.

Specifying CONNECT=DEFAULT on the APPL, LINE, and SOCKET initialization statements indicates that processing defers to the CONNECT=(YES|NO) parameter value that is specified for the associated NODE initialization statement. The APPL and SOCKET statements use the NODE= parameter to refer to their associated NODE. A line can become associated with a node in two ways:

1. The NODE= parameter on the LINE statement can refer to the node
2. The LINE= parameter on the NODE statement can refer to the line.

The NODE default value for CONNECT is NO. The JES2 $T NODE command can be used to modify the CONNECT attribute.

The APPL, LINE, NODE, and SOCKET CONNECT *interval* value is the approximate time, in minutes, between attempts to restart a connection. The valid range for the *interval* value is 0-1440 minutes. A value of 0 (default) indicates that the *interval* value from the NJEDEF CONNECT parameter is being used. The valid

range for the NJEDEF CONNECT parameter interval value is 1-1440 minutes, and the default is 10 minutes. To prevent simultaneous connection attempts from both ends, JES2 randomly modifies the specified *interval* value by adding up to 45 seconds to it

The NJEDEF initialization statement CONNECT=(YES|NO,*interval*) parameter specifies whether (YES) or not (NO) any NJE connections will be automatically started after JES2 initialization, or restarted if they are disconnected for longer than the applicable time interval. The default value is YES. If NJEDEF CONNECT=NO is specified, the $D NJEDEF command displays COMMAND=($TNJEDEF,CONNECT=NO), and the $JDSTATUS command displays the following JES2 NOTICES:

```
$HASP9150 JES2 NOTICES
$HASP9173 NJE AUTOMATIC CONNECT PROCESSING DISABLED
          (NJEDEF CONNECT=NO)
```

When JES2 attempts to automatically start or restart an NJE connection, JES2 issues a $HASP568 message similar to the following example:

```
$HASP568 LNEn - AUTOMATIC CONNECT
```

When an automatic attempt to start or to restart a NJE connection fails, JES2 issues a $HASP569 message similar to the following example:

```
$HASP569 SOCKET=Sn - NO IDLE TCP/IP LINE IS AVAILABLE, RC=08
         NEXT ATTEMPT: 13 APR 2011 AT 16:05
```

Using the $D command to display an NJE connection (APPL, LINE, NODE, or SOCKET) that is currently disconnected will display the projected connection restart time if CONNECT=YES is defined:

```
CONNECT=(YES,2,2011.103,16:05)
```

For more information on using each initialization statement, refer to *z/OS JES2 Initialization and Tuning Reference*. For more information on using each command, refer to *z/OS JES2 Commands*.

Starting NJE connections manually

When the NODE parameter is specified for a LINE, the $SN,LINE*nnnn* command can be used to start an SNA or TCP/IP connection for that line. The UNIT parameter specified for the LINE determines the type of connection: BSC, SNA or TCP/IP. For SNA and TCP/IP connections, the first available APPL or SOCKET, respectively, is chosen for the specified node. The NODE parameter also specifies the node to connect to when a LINE is defined to automatically start or restart an NJE connection. For more information on using the $SN command, refer to *z/OS JES2 Commands*.

For a BSC LINE, specifying the NODE parameter has no effect because the $SN,LINE*nnnn* command will connect to the node that is physically attached to the other end of the line.

With the NODE parameter specified for a LINE, the SDSF SN prefix command can also be used from the SDSF LINE display to start an SNA or TCP/IP connection. For information on the SN prefix, refer to *z/OS SDSF Operation and Customization*.

Using the JES2 network commands

Use the following JES2 network commands to display or affect the status of all JES2 networking devices (NJE and RJE) and NJE connections. For more information on using each command, refer to *z/OS JES2 Commands*.

$D NETWORK
Displays the status of all active networking devices and NJE connections.

$E NETWORK
Runs the $E command for all active lines, JES2/VTAM interfaces (LOGONs), and network servers (NETSERVs), and causes NJE connections to be immediately reset.

$P NETWORK
Runs the $P command for all active lines, JES2/VTAM interfaces (LOGONs), and network servers (NETSERVs), causing the networking devices to be drained after their NJE connections are reset.

The $P NETWORK command also prevents the automatic restarting of networking devices and NJE connections. Afterwards, the $D NJEDEF command displays COMMAND=($PNET) and the $JDSTATUS command displays the following JES2 NOTICES:

```
$HASP9150 JES2 NOTICES
$HASP9173 NJE AUTOMATIC CONNECT PROCESSING DISABLED
          (NJEDEF CONNECT=NO)
$HASP9174 NJE AUTOMATIC RESTART PROCESSING DISABLED
          ($ZNET/$PNET)
```

$S NETWORK
Allows the automatic restarting of networking devices and NJE connections previously disallowed by $P NETWORK, $Z NETWORK or NJEDEF CONNECT=(NO) and immediately starts networking devices and NJE connections that are restartable.

$Z NETWORK
Performs $E NETWORK and $P NETWORK processing, halting automatic networking device and NJE connection restart processing and draining networking devices. Afterwards, the $D NJEDEF command displays COMMAND=($PNET) and the $JDSTATUS command shows the following JES2 NOTICES:

```
$HASP9150 JES2 NOTICES
$HASP9173 NJE AUTOMATIC CONNECT PROCESSING DISABLED
          (NJEDEF CONNECT=NO)
$HASP9174 NJE AUTOMATIC RESTART PROCESSING DISABLED
          ($ZNET/$PNET)
```

SMF record summary: NJE (network job entry) processing

The following lists the SMF records written during JES2 NJE processing. For a complete list of the record contents, see *z/OS MVS System Management Facilities (SMF)*.

Table 60. SMF Records Used by JES2 NJE

Record Number	Action that Causes the Record to be Written
55	A signon record is written at each node ($S N(n) commands).
56	Whenever a signon attempt contains an incorrect line or node password.
57	To record SYSOUT information.
58	The records written at each node whenever a networking session is ended.

The path manager

JES2 systems have a feature called the network path manager; other IBM networking packages (for example, JES3 and RSCS) do not. The JES2 network path manager maintains path information for all connections between NJE members (nodes). Each JES2 member of the NJE network has a copy of the path manager. The path manager determines the best paths to the various nodes and maintains information about the status of the network.

Path manager processing is essentially the same for SNA, BSC, or TCP/IP NJE. Each path manager can have a consistent picture of the network at any given point in time. As nodes make and break connections in the network, the path managers at the various nodes broadcast their views of the network through network connection control (NCC) records to directly attach nodes and update internal routing tables based on the information they receive. The path manager issues message $HASP500 when various NJE signon and connectivity problems occur. See *z/OS JES2 Messages* for the list of associated reason codes and their meanings.

Network connection control records

The path manager sends network connection control (NCC) records between JES2 nodes to establish NJE connections and keep track of non-adjacent connections. The path manager uses the NCC records to update the path manager routing tables that establish the path manager's view of the network. These network updates occur through the **M** and **N** records that add and subtract members from a network, respectively.

The path manager determines the *best path* based upon the *path of least resistance* between the local node and each other node in the network. The resistance is derived from NCC records, CONNECT initialization statements, and parameters on the NJEDEF, NODE(nnnn), APPL(avvvvvvv), and LINE(nnnn) initialization statements. For more information on how the path manager uses these specifications for both adjacent and non-adjacent connections, see "Determining path resistance" on page 311.

The following is a list of the network connection control records. Only JES2 uses those records denoted by an asterisk (*).

Record	Purpose
I	Initial Signon
J	Response Signon
K	Reset Signon*
L	Concurrence Signon *
M	Add Connection *
N	Subtract Connection *
B	Signoff

Note that the path manager does not send records about the following connections:

- Predefined or static connections (using a CONNECT initialization statement)
- Non-path manager connections
- Private connections

Non-JES2 nodes (such as JES3, RSCS, and System i) discard NCC **M** and **N** records.

See “Displaying information about a network” on page 294 for an example of how to display all active paths to a node.

Non-path manager connections

You can identify nodes that do not have a path manager (non-JES2 nodes) by coding PATHMGR=NO on the NODE(nnnn) initialization statement. You can also specify that an adjacent node is not to receive any NCC add and subtract records by specifying PATHMGR=NO on NODE statement for that node.

Note that JES2 can establish a connection between two NJE nodes using either path manager protocols or non-path manager protocols. However, **both nodes must use the same protocols or the connection will not complete.** In these instances, JES2 issues a $HASP500 message at one or both ends of the connection to explain why the connection failed.

Non-path manager protocol: In the network illustrated in Figure 82, the JES2 WASH node initiates a non-path manager signon with the non-JES2 NEWYORK node. (Either node can initiate non-path manager signon.)

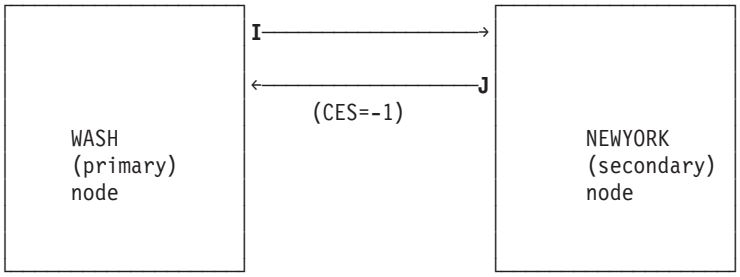


Figure 82. Signon Between Two Nodes Using Non-Path Manager Protocol

1. WASH attempts to signon to NEWYORK by sending an **I** (initial signon) record.
2. NEWYORK responds to WASH by sending a **J** (response signon) record.

(The **I** and **J** records can contain both node and BSC line passwords.)

Full path manager protocol within a network: The example in Figure 83 shows two path manager nodes connecting using SNA protocols, or BSC protocols when the "high end" (node name closer to "Z" in the alphabet) initiates the protocol.

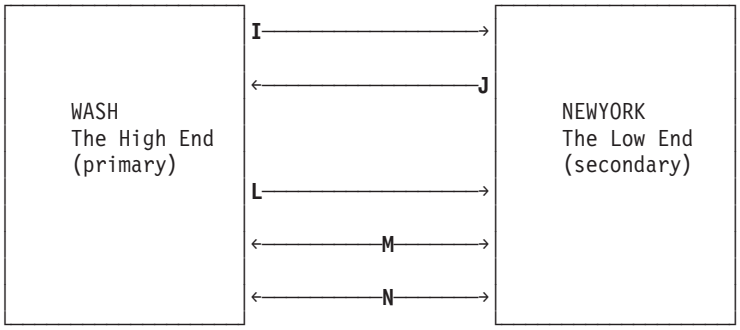


Figure 83. Two Path Manager Nodes Signon Protocol

1. WASH attempts to sign on to NEWYORK by sending an **I** (initial signon) record.

2. NEWYORK responds to WASH by returning a **J** (response signon) record.
3. WASH responds to NEWYORK by returning an **L** (concurring signon) record to indicate that both systems will communicate across the established line.
4. When connected, both WASH and NEWYORK send **M** (add connection) and **N** (subtract connection) records to each other to share information about their other NJE connections. They also send **M** records to other path manager nodes to which they are connected to notify them of this connection.

When this connection is dropped by either WASH or NEWYORK, both nodes send **N** records to any path manager nodes with which they are connected to inform them that the connection no longer exists.

In the BSC case where the "low end" initiates the signon, the "low end" responds to the **J** record with a **K** record and the "high end" sends the **L** record.

Private connections

A private connection exists when two nodes in a network do not pass NCC records to other nodes to inform them of the connection. You might want to define a private connection if your node has limited spool space and you would like to reserve this spool space only for communication with an adjacent node.

You can suppress information from reaching the rest of the network about a connection between your node and an adjacent JES2 node by specifying PRIVATE=YES on the NODE(nnnn) initialization statement.

Note that you must specify both ends of a connection as PRIVATE=YES on the NODE(nnnn) initialization statements to ensure that the path manager does not send NCC records.

In Figure 84, only nodes A6 and B5 know about their 'private' connection.

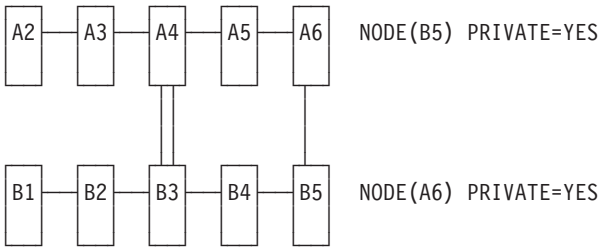


Figure 84. Two Private Nodes Across Subnets

Summary of PATHMGR= and PRIVATE= parameters

The following table summarizes the possible settings of the PATHMGR= and PRIVATE= parameters on the NODE(nnnn) initialization statements and their effects.

Table 61. NODE(nnnn) Statement - PATHMGR= and PRIVATE= Parameter Specifications

Node Location	Operating System	PATHMGR=	PRIVATE=
Own Node	JES2	Not applicable	NO YES to prevent sending NCC records about your member-to-member connections within a MAS.

Table 61. NODE(nnnn) Statement - PATHMGR= and PRIVATE= Parameter Specifications (continued)

Node Location	Operating System	PATHMGR=	PRIVATE=
Adjacent	JES2	YES NO to make private and keep from sending Path Manager NCC records to it.	NO YES to keep other nodes from finding out about this connection
Non-adjacent	JES2	YES should be specified	not applicable
Non-adjacent	non-JES2	NO should be specified	not applicable

Static connections

Static connections, which are always private connections, are predefined through a CONNECT statement. These connections are useful to supplement or override the dynamic connections created by the JES2 path manager. A CONNECT statement can be used to:

- Specify remote connections that exist but the NCC M and N records cannot reach the local node because of intervening non-path manager nodes.
- Indicate that a connection should be assumed to be there by the path manager even if it is not really there.
- Override the resistance of the connection calculated by the path manager. (See "Determining path resistance" on page 311 for details.)

A connection involves 2 points in a network. Each point is the smallest addressable piece of the network and a single JES2 member. A CONNECT statement specifies 2 points that are to be connected. Each point is identified by its node with a member number of 1.

Use the MEMBerA= and MEMBerB= parameters to see various members of the shared spool (on the nodes defined by the NodeA= and NodeB= parameters) involved in the connection. The MEMBerB= parameter has the same meaning for NodeB as the MEMBerA= has for NodeA. You should always specify a value of 1, unless you know that the node to which you are connecting has specified a different value.

Figure 85 on page 307 shows that nodes 1 and 2 have a static connection because of the CONNECT statements in their initialization data sets.

```
CONNECT NODEA=BOST1,MEMBERA=1,NODEB=NYC2,MEMBERB=1,REST=150
```

The NodeA= and NodeB= designations provide a means of specifying the two nodes that make up the connection (node 1 and node 2 in the example).

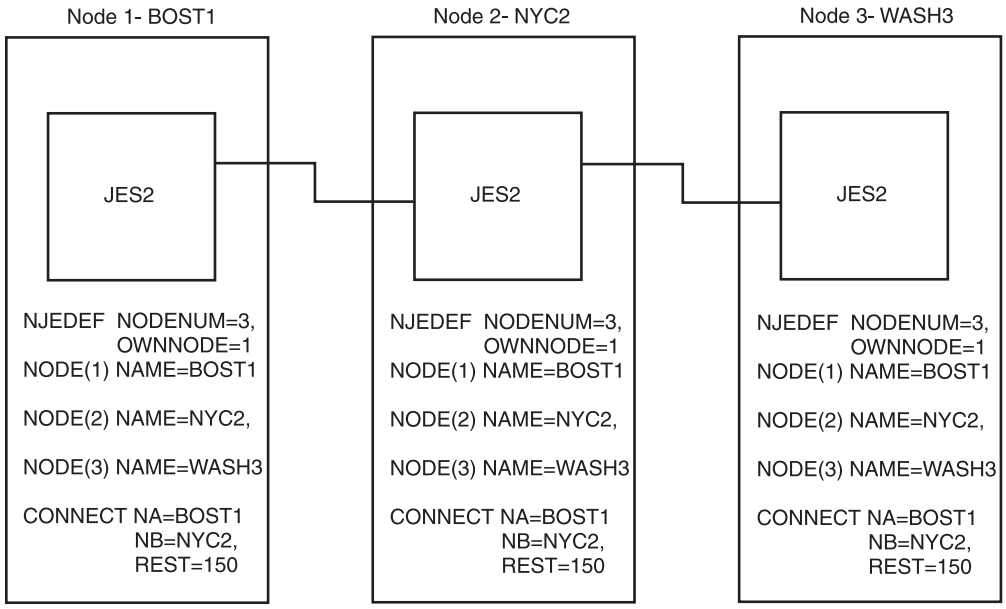


Figure 85. Static Connections

The REST= parameter specifies the total resistance of the connection between BOST1 (node 1) and NYC2 (node 2).

Static connections are not propagated to other nodes, and are therefore private. A private connection is known only to those systems comprising the connection. If, in the example in Figure 85, the CONNECT statement appeared only in the initialization data sets for nodes 1 and 2, the connection is private and node 3 cannot reach node 1. Because the CONNECT statement describing the node 1-to-node 2 connection does appear among the initialization statements of some of the other nodes in the network, in this example, the connection is known to the nodes that have the CONNECT statement.

Specifying the CONNECT statement places an entry describing the connection in the path manager routing tables.

Using the CONNECT statement

The following table summarizes when it is appropriate to use a CONNECT initialization statement:

Table 62. CONNECT Statement

Type of Connection	**CONNECT in the Local JES2 Version 4 or a Later Release Initialization Data Set**
To an adjacent JES2 node	Use to override dynamic connection; otherwise omit
To an adjacent non-JES2 node	Use to override dynamic connection or override resistance; otherwise omit
Remote connection between two path manager nodes	Use to define connection if the other node cannot send this node NCC records through other JES2 nodes. (that is, There is a non-path manager node between this node and the other node.) Otherwise omit
Remote connections between two non-path manager nodes	Required if they are in the same remote subnet.

Controlling static connections

Static connections can be added, deleted, modified, and displayed using the following commands.

- $ADD CONNECT
- $DEL CONNECT
- $T CONNECT
- $D CONNECT

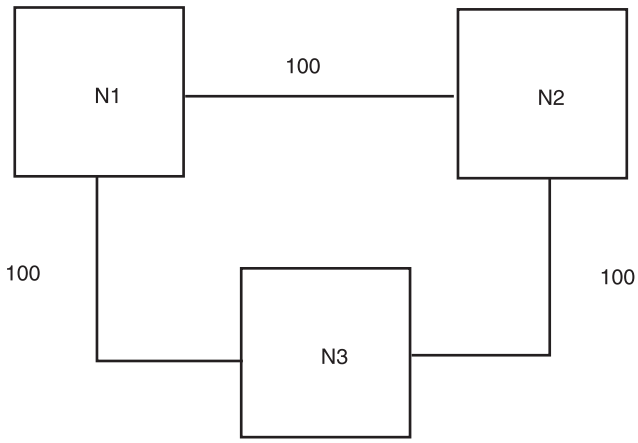
For examples of how to use these commands, see "Displaying information about a network" on page 294.

Preventing looping when using connect statements

Figure 86 shows definitions for a 3-node network with one non-JES2 node (N3). Each of these connections has a resistance value defined to prevent transmitted data from looping through the system if multiple outages occur.

To understand the effect of simultaneous multiple outages on incorrectly defined connections in a network involving a non-JES2 node, assume that:

- The N1 to N3 and N2 to N3 connections are unavailable.
- N3 is a non-JES2 node.
- Only the connections listed exist.




```
N1
NODE(3) NAME=N3,PATHMGR=N0
          CONNECT NODEA=N2,NODEB=N3, REST=100
NJEDEF RESTMAX=150

N2
NODE(3) NAME=N3, PATHMGR=N0
          CONNECT NODEA=N1, NODEB=N3,REST=100
NJEDEF RESTMAX=150
```

Figure 86. Three-Node Network Configuration CONNECT Statements to Avoid Looping

The path managers at both N1 and N2 discover that the path to N3 is down. Both path managers attempt to route data to the alternate path. When data arrives at N2, the path manager there knows the N2 to N3 path has failed. Therefore N1 sends the data to N1 to make use of the N1 to N3 path. When data arrives at N1, the path manager there knows the N1 to N3 path has failed. Therefore, N1 sends the data to N2 to make use of the N2 to N3 path. The path managers continue to

send the data back and forth (looping) until one of the unavailable links restarts or the NJEDEF MAXHOP= limit at one of the sites has been reached.

Note: In the case where both paths are prone to failure and require alternate paths, IBM suggests a low setting, such as NJEDEF MAXHOP=2. In the event of a failed connection, data travels back-and-forth only one time.

An installation can avoid looping by defining a NJEDEF RESTMAX= parameter value that eliminates all alternate paths, as shown in Figure 86 on page 308. That is, the path from N1 through N2 to N3 has a total resistance of 200. Data from N1 will not take this path to N3 because the resistance exceeds the 150 maximum.

If one of the paths is particularly prone to failure, an installation can add static connections between both N1 and N3 and N2 and N3 as illustrated in Figure 87.

```
CONNECT  NODEA=N1,NODEB=N3,REST=100
CONNECT  NODEA=N2  NODEB=N3,REST=100
```

Figure 87. CONNECT Statements that Avoid Looping in 3-Node Network

The CONNECT statements advise the path managers at both N1 and N2 when the N3 connection is down, so both path managers hold the data.

Note: If both paths require alternate paths but CONNECT statements like those in Figure 87 do not exist, IBM suggests a low setting for the NJEDEF MAXHOP= parameter (such as MAXHOP=2). In the event of a failed connection, data travels back-and-forth only one time.

Defining and communicating between subnets

A subnet is a set of nodes identified by name to simplify NJE routing tables by reducing NCC record traffic and path manager overhead. Networks consisting of many nodes become more manageable when you group nodes into subnets.

For example, when two corporations merge, it might be easier to manage nodes from each of the two corporations as separate subnets, with access to each other through nodes attached to each other through CONNECT statements. Such nodes are called **gateway nodes**. The path manager presumes that if one node in another subnet can be reached, all nodes in that subnet can be reached without additional resistance.

Figure 88 on page 310 provides an example of two subnets named MD and NY. In this example, NYC and WASH are the gateway nodes for each of the subnets. Figure 89 on page 310 shows the initialization statements that define these subnets.

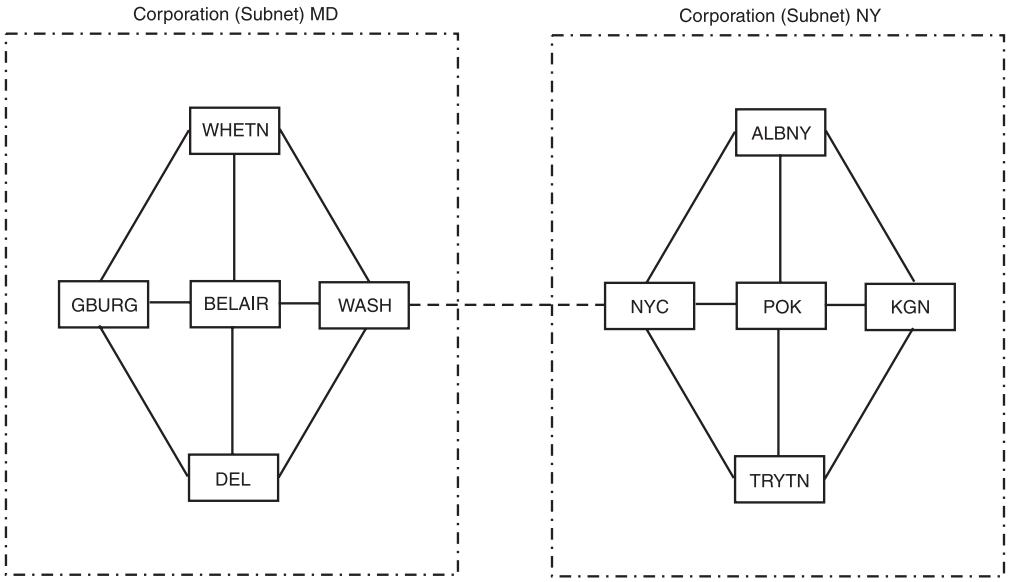


Figure 88. Example of Defining Subnets

```
/*******  CONNECT STMT TO SUBNETS  *********************/
/******************************************************/
CONNECT NODEA=0005,NODEB=0006,MEMBA=1,MEMBB=1,REST=10
/*******  NJE NODE DEFINITIONS  DEFINING SUBNETS ********/
/******************************************************/
NODE(1-10) AUTH=(NET=YES),TRACE=YES
NODE(01) NAME=WHETN,PATHMGR=YES,SUBNET=MD
NODE(02) NAME=GBURG,PATHMGR=YES,SUBNET=MD
NODE(03) NAME=BELAIR,PATHMGR=NO,SUBNET=MD
NODE(04) NAME=DEL,PATHMGR=NO,SUBNET=MD
NODE(05) NAME=WASH,PATHMGR=NO,SUBNET=MD
NODE(06) NAME=NYC,PATHMGR=NO,SUBNET=NY
NODE(07) NAME=POK,PATHMGR=NO,SUBNET=NY
NODE(08) NAME=KGN,PATHMGR=NO,SUBNET=NY
NODE(09) NAME=TRYTN,PATHMGR=NO,SUBNET=NY
NODE(10) NAME=ALBNY,PATHMGR=NO,RECEIVE=JOBS,SUBNET=NY
/******************************************************/
```

Figure 89. Example of Subnets Defined in the Initialization Data Set

These definitions must be in the initialization data set of each node involved in order to define a subnet. To prevent confusion, IBM suggests using different naming conventions for subnets than those used for nodes. For example, if all nodes in a network begin with the prefix 'ND', use a different prefix for the names of subnets in your network.

Use the SUBnet= parameter on the NODE(nnnn) initialization statement to define the subnet to which a node belongs. Then, use one (or more) of the nodes in a subnet as the gateway to that subnet. That node communicates to the gateway node in another subnet. Any node with which you establish a connection to a node in another subnet becomes a gateway node. You might want to use more than one node as the gateway depending on factors such as the number of nodes in a subnet, or the communication needs between subnets. Note, however, that the fewer gateway nodes that are used, the more manageable the communication between subnets becomes.

Because you cannot define the subnets on all nodes in a network simultaneously, you must specify them carefully. Ensure that nodes outside your subnet have your subnet defined before nodes inside your subnet define it. Then define all subnets consistently throughout the network, so that all nodes have a valid view of routing.

You can exercise control over subnets through JES2 operator commands, which allow you to:

- Add nodes to a subnet or change nodes from one subnet to another by using the $T NODE(nnnn) command.
- Change the static connections using the $ADD, $DEL, and $T CONNECT commands.
- Display a list of nodes in one or more subnets using the display subnet(s) ($D SUBNET) command.

```
$D SUBNET(M*)

$HASP816 SUBNET(MD) WHETN, GBURG, BELAIR, DEL, WASH
```

For more information about using these commands, see "Displaying information about a network" on page 294 and *z/OS JES2 Commands*.

Determining path resistance

Path resistance is a value set by an installation to control the flow of work between nodes in a network. The *best path* is determined by the JES2 path manager facility based on the path of least resistance. The resistance is derived from NCC records, CONNECT initialization statements and parameters on the NJEDEF, NODE(nnnn), APPL(avvvvvvv), SOCKET(vvvvvvvv) and LINE(nnnnn) initialization statements.

When defining resistance for a path, you should base the values on the line speed (the faster the line, the lower the line resistance) and the expected workload (the lighter the workload, the lower the work resistance).

The following describes how to:

- Calculate resistance for adjacent connections
- Calculate resistance for non-adjacent connections
- Limit store-and-forward transmissions.

Calculating resistance for adjacent connections

Table 63 summarizes how JES2 calculates path resistance for adjacent connections. The parameters that the JES2 path manager uses to calculate resistance varies depending on the connection type as follows:

Table 63. Resistance calculation for adjacent connections. The second data cell in the final row spans the second and third columns.

Connection type	Partial resistance from local definitions	Partial resistance from non-local definitions (Unless SENTREST=IGNORE is specified on the NODE (nnnn) statement.)
Dynamic SNA Connection to JES2	NJEDEF RESTNODE + APPL or NODE REST	NJEDEF RESTNODE + APPL or NODE REST
Dynamic TCP/IP Connection to JES2	NJEDEF RESTNODE + SOCKET or NODE REST	NJEDEF RESTNODE + SOCKET or NODE REST

Table 63. Resistance calculation for adjacent connections (continued). The second data cell in the final row spans the second and third columns.

Connection type	Partial resistance from local definitions	Partial resistance from non-local definitions (Unless SENTREST=IGNORE is specified on the NODE (nnnn) statement.)
Dynamic BSC Connection to JES2	NJEDEF RESTNODE + LINE REST	NJEDEF RESTNODE + LINE REST
Dynamic Connection to JES3 or POWER® Node	NJEDEF RESTNODE + LINE, APPL or NODE REST	0
Dynamic Connection to VM or System i	NJEDEF RESTNODE + LINE, APPL or NODE REST	200
Static Connection	CONNECT REST= value (overrides any dynamic resistance as previously described)	

Use the following formula to calculate the resistance of a connection to an adjacent node without a CONNECT statement:

```
From Node A    R = RLa + RWa + NCCIREST

From Node B    R = RLb + RWb + NCCIREST


KEY:

 R        - represents total resistance
 RLx      - represents the resistance of the line (BSC) or session (SNA)
 RWx      - represents the resistance through the node
 NCCIREST - represents the partial resistance sent by the other node
```

Figure 90. Formula for Calculating the Resistance of Adjacent Nodes.

Note:

1. NCCIREST consists of the combined resistance of the line plus the JES2 workload as determined by the node at the other end of the connection.
2. Neither node can control the line or workload resistance specified by the other node. For ways of limiting the impact of another node's resistance, see "Path selection considerations" on page 317.
3. Two nodes must first exchange their partial resistance values to establish the total resistance of a connection. The partial resistance is the sum of the resistance of the line plus the resistance caused by workload as determined by each node. The exchange occurs as follows:
   a. The node that initiates the connection sends its partial resistance to the other node using an NCC I (initial signon) record.
   b. The other node returns its partial resistance using an NCC J (response signon) record.
   c. If path manager is active, the total resistance of the connection is sent back and forth in the NCC K (reset) and the NCC L (concurrence) records, respectively.

For example, when defining the 3-node connection shown in Figure 91 on page 314, an installation would base the resistance of the BSC connection between NEWYORK and BOSTON using line 3 using these variables:

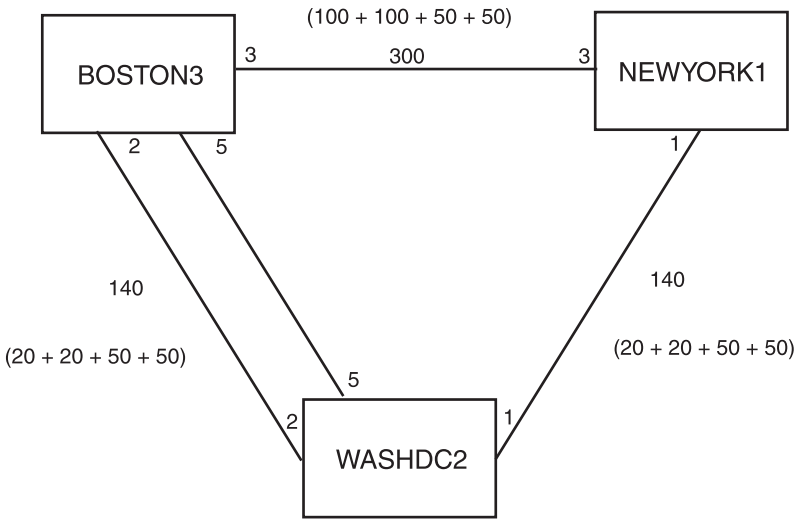
Variable
Description

RLa Represents the resistance of the line as evaluated by the installation manager at node "a" (NEWYORK in the following example). This is the value specified on NODE A's REST= parameter on the LINE(3) statement.

RLb Represents the resistance of the line as evaluated by the installation manager at node "b" (BOSTON in the following example). This is the value specified on NODE B's REST= parameter on the LINE(3) statement.

RWa Represents the resistance because of workload on JES2 as evaluated by the installation manager at node "a". This is the value specified on NODE A's RESTNODE= parameter on the NJEDEF statement.

RWb Represents the resistance because of workload on JES2 as evaluated by the installation manager at node "b". This is the value specified on NODE B's RESTNODE= parameter on the NJEDEF statement.



NEWYORK1 (NODE 1)

```
NJEDDEF NODENUM=3, LINENUM=2, OWNNODE=1, PATH=2, RESTNODE=50, RESTMAX=250,
        REESTTOL=30
NODE(1) NAME=NEWYORK1
NODE (2) NAME=WASHDC2, REST=20
NODE (3) NAME=BOSTON3
LINE   (1) UNIT=SNA
LINE   (3) REST=100, TRANSPAR=YES
LOGON (1) APPLID=NEWYORK1
```

WASHDC2 (NODE 2)

```
NJEDDEF NODENUM=3, LINENUM=3, OWNNODE=2, PATH=2, RESTNODE=50, RESTMAX=250,
        RESTTOL=10
NODE(1) NAME=NEWYORK1, REST=20
NODE (2) NAME=WASHDC2
NODE (3) NAME=BOSTON3
LINE   (1) UNIT=SNA
LINE   (2) REST=20, TRANSPAR=YES
LINE   (5) REST=20, TRANSPAR=YES
LOGON (1) APPLID=WASHDC2
```

BOSTON3 (NODE 3)

```
NJEDDEF NODENUM=3, LINENUM=3, OWNNODE=3, PATH=2, RESTNODE=50, RESTMAX=250,
        RESTTOL=10
NODE(1) NAME=NEWYORK1
NODE (2) NAME=WASHDC2, REST=20
NODE (3) NAME=BOSTON3
LINE   (2) UNIT=SNA
LINE   (3) REST=100, TRANSPAR=YES
LINE   (5) REST=20, TRANSPAR=YES
```

Figure 91. Path Resistance Values

For the SNA connection in Figure 91, the resistance between NEWYORK1 and WASHDC2 (using line 1) is calculated with the following variables:

Variable
Description

RLa Represents the resistance of the session as evaluated by the installation manager at node “a” (NEWYORK1 in the previous example). This is the value specified on the REST= parameter of NEWYORK1's APPL(WASHDC2) initialization statement. If APPL(WASHDC2) is omitted, resistance is obtained from the REST= parameter in the NODE(nnnn) initialization statement for WASHDC2.

RLb Represents the resistance of the session as evaluated by the installation manager at node “b” (WASHDC2 in the previous example). This is the value specified on the REST= parameter of WASHDC2's APPL(NEWYORK1) initialization statement. If APPL(NEWYORK1) is omitted, resistance is obtained from the REST= parameter in the NODE(nnnn) initialization statement for NEWYORK1.

RWa Represents the resistance because of workload on JES2 as evaluated by the installation manager at node “a”. This is the value specified on NODE A's RESTNODE= parameter on the NJEDEF statement.

RWb Represents the resistance because of workload on JES2 as evaluated by the installation manager at node “b”. This is the value specified on NODE B's RESTNODE= parameter on the NJEDEF statement.

Using values from Figure 91 on page 314, you would calculate the total resistance for these connections as follows:

```
NEWYORK1 to WASHDC2 (line 1):      R = 20 + 20 + 50 + 50 = 140
WASHDC2 to BOSTON3  (line 2):      R = 20 + 20 + 50 + 50 = 140
NEWYORK1 to BOSTON3 (line 3):      R = 100 + 100 + 50 + 50 = 300
```

If parallel lines with different resistance values connect two nodes (for example, lines 2 and 5 in Figure 91 on page 314), JES2 uses the lower resistance value when determining the best path.

You can use the SENTREST= parameter on the NODE(nnnn) initialization statement to ignore the partial resistance sent by another node (in the NCC I and NCC J records). SENTREST= allows you to specify whether another node's resistance should be added to the local node's to determine resistance at the local node. If you specify SENTREST=IGNORE, the local node ignores the resistance sent from the connecting node. This value allows the local node to control the resistance sent during sign on processing. You can specify SENTREST=IGNORE for only non-path manager nodes.

For example, if NEWYORK1 in Figure 91 on page 314 were a VM node, BOSTON3 can maintain a local resistance of the NEWYORK1-to-BOSTON3 connection of 150 by refusing to accept the partial resistance from NEWYORK1.

Calculating resistance for non-adjacent nodes

Use Table 64 to calculate total resistance between non-adjacent nodes.

Table 64. Resistance Determination for Non-Adjacent Connections

Connection Type	Total Resistance
Path to a node within our subnet (or not in any subnet)	Sum of the individual connections in that path
Path to a node not in the local subnet.	Sum of the connections between subnets in that path
Static Connection	CONNECT REST= value (overrides any dynamic resistance described previously)

Table 64. Resistance Determination for Non-Adjacent Connections (continued)

Connection Type	Total Resistance
Two nodes in the same remote subnet	0

In Figure 91 on page 314, the resistance from NEWYORK1 to BOSTON3 through lines 1 and 2 would be calculated as follows:

```
RESISTANCE FROM NEWYORK1 to WASHDC2 (line 1) =  140
RESISTANCE FROM WASHDC2 to BOSTON3  (line 2) =  140

_____________________________________________________
TOTAL RESISTANCE NEWYORK1 to BOSTON3         =  280
```

Although line 3 is a direct path between NEWYORK1 and BOSTON3 as shown in Figure 91 on page 314, the path through WASHDC2 (over lines 1 and 2) actually have a lower resistance, making it the best path. No data will be sent over line 3 unless you instruct JES2 to use multiple paths by setting the RESTTOL= and PATH= parameters on the NJEDEF initialization statement as described in "PATH" on page 317.

Controlling store-and-forward transmissions

You can force JES2 to:

- Use only direct connections to send data to another node
- Use only direct connections to send data to a specific node
- Prevent the use of a specific node as an intermediate node.

Using only direct connections to send data to another node: To restrict all NJE transmissions to use only direct connections and not use any indirect routing paths, specify RESTMAX=0 on the NJEDEF initialization statement. Specifying RESTMAX=0 limits path selection to adjacent nodes and nodes whose total path resistance is 0.

Using only direct connections to send data to a specific node: If you have files destined for an adjacent node that are too large or highly sensitive, you might want to prevent any alternate path routing if the link is down or backed up with other work. To prevent files from being sent through intermediate nodes, you can do one of the following:

- Specify REST=0 on a CONNECT initialization statement that defines the connection. This will override the path manager's attempt to use an alternate path unless you have NJEDEF PATH greater than 1 and the alternate path's resistance is less than RESTTOL.
- Specify DIRECT=YES on the NODE(nnnn) statement for that node. When JES2 determines paths to that node, JES2 will be assume paths to be unconnected if an adjacent connection is not active.

Preventing the use of a specific node as an intermediate node: If you have adjacent nodes that have limited capacity or that are not trustworthy to safely store and forward NJE files, you might want to restrict their ability to be an intermediate node to other nodes in your network:

- Specify a high resistance for the node (a value higher than RESTMAX).
- Isolate the node from the other nodes through subnets and non-path manager protocols.
- Specify ENDNODE=YES on the NODE(nnnn) initialization statement.

Specifying ENDNODE=YES prevents a node from being eligible to store-and-forward work.

Path selection considerations

You can use the NJEDEF initialization statement to affect how JES2 selects paths throughout a network. Use the following parameters to specify alternate paths to a node and influence how JES2 determines those paths:

- PATH=
- RESTTOL=
- RESTMAX=
- SENTREST=

PATH

Specify this parameter if you maintain more than one path to a destination. For example, in Figure 91 on page 314, you must specify PATH=2 on the NJEDEF statement if NEWYORK1 uses both paths to BOSTON3. The network path manager maintains the number of paths to a given destination that is specified on the PATH= parameter. If an installation specifies two paths to a node, the network path manager first checks to see if both paths are within the limits of the RESTTOL= parameter. If one is not, the path manager cannot select that path. However, if both paths are within the limits of RESTTOL=, an installation cannot predict which path the path manager will choose.

RESTTOL

This parameter specifies the maximum difference in resistance between the best possible path and an acceptable alternate path. For example, based on the calculation of the resistances of the two possible paths between NEWYORK1 and BOSTON3 (Figure 91 on page 314), the path through lines 1 and 2 (with a total resistance of 280) is the best path. Because the RESTTOL= value is set to 30, data cannot travel through line 3 because the resistance of 300 exceeds the RESTMAX= parameter value. (See the following description of the RESTMAX= parameter.)

The value of RESTTOL= must be less than the resistance between any two nodes in the network; otherwise, jobs and messages will transmit in an erratic fashion.

RESTMAX

This parameter allows you to control the storing-and-forwarding of data throughout your network. RESTMAX= specifies the maximum resistance through which JES2 attempts to transmit work for both adjacent and non-adjacent nodes. To avoid inconsistencies in the views of the network maintained by the various path managers, specify the same value for RESTMAX= at all nodes.

To prevent store-and-forward through any nodes in a network, specify a low value on this parameter. This value restricts all NJE connections to direct connections, eliminating direct routing paths. RESTMAX=0 ensures that data is sent to an adjacent node only, and never to an intermediate node from which it can be forwarded to a final destination. If you specify RESTMAX=0, JES2 considers the direct path to any adjacent node usable, no matter what resistance is coded. Nodes beyond the adjacent node are reachable only if the total resistance to them is zero, or they exist in the same subnet as an adjacent node.

In Figure 91 on page 314, the value specified for RESTMAX= at WASHDC2 (RESTMAX=250) prevents WASHDC2 from transmitting to NEWYORK1 through BOSTON3, even if line 1 is down, because the total resistance of the connection is

440. Because the RESTMAX= parameter applies to adjacent and non-adjacent nodes, NEWYORK1 cannot transmit to BOSTON3 over line 3 because the resistance is 300.

Extending network capability

The network illustrated in Figure 92 on page 319 shows how you can extend the capability of basic BSC and SNA NJE configurations. These initialization statements enable an installation to determine the security access one node has to another, the job numbers specified at each node, and the destinations for all output jobs and SYSOUT. Individual nodes must share definitions for the capabilities defined through these initialization parameters to ensure the regular flow of traffic throughout a network.

See Figure 92 on page 319 as you read through the following explanation of the various initialization statements that control the networking connections between NEWYORK and WASHDC. Preceding each of the following statements is a highlighted number that correlates to a number in parentheses in the figure.

1 NODE(1) NAME=NEWYORK, PASSWORD=SECRET

This statement assigns the name NEWYORK to this node. When node 1 attempts to sign on to another node in the network, it will identify itself as NEWYORK. The receiving node will look up the name NEWYORK to determine if NEWYORK is a member of the network. If the receiving node does not find an entry for NEWYORK, the receiving node will reject the signon.

PASSWORD=SECRET specifies that if no SEND= password value is specified for NODE(1) to send to another node to which it is signing on, NODE(1) will send this password to that node instead. Note that this option is less secure than two passwords. If you have defined all your nodes using PASSWORD=(SEND=word1,VERify=word2), then it would not be necessary to specify a PASSWORD= parameter on the NODE(1) definition in this initialization data set.

2 NODE(2) NAME=WASHDC, PASSWORD=(SEND=SECRET,VER=HIDDEN)

PASSWORD=(SEND=SECRET,VER=HIDDEN) specifies that the password that NEWYORK will send to WASHDC is SECRET. The node WASHDC will use this password for verification during signon. In WASHDC's initialization data set, the NODE(1) initialization statement must specify PASSWORD=(SEND=HIDDEN,VER=SECRET).

If NEWYORK does not supply the password SECRET when attempting to signon to node WASHDC, node WASHDC rejects the signon attempt. WASHDC will place the password HIDDEN in the response to the signon record that it sends to NEWYORK. If WASHDC does not supply the password HIDDEN, NEWYORK stops signon record processing. If the connection is an SNA session over line 2, the processing described takes place after VTAM establishes a session between the two NJE nodes.

Note: The initialization data set should be protected to prevent disclosing passwords contained on NODE(nnnn) initialization statements. If RACF is used, provide protection by specifying UACC=NONE for the data set.

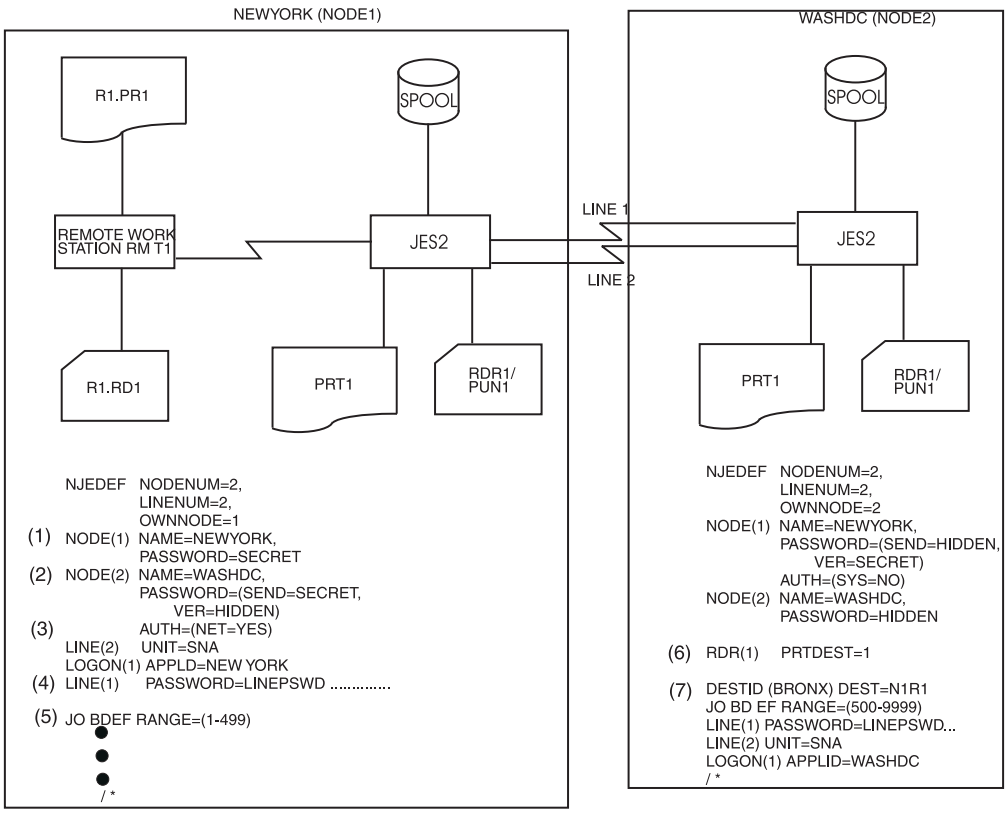


Figure 92. An Expanded Network

3 AUTH=(NET=YES)

Specifies one of the four authority categories. In the example, the specification of NET=YES in the AUTH= parameter on the NODE(2) initialization statement on NEWYORK for WASHDC indicates WASHDC has the same authority as consoles attached locally to NEWYORK. WASHDC also has (by default) the ability to affect jobs (JOB=YES), the ability to affect devices (DEVICE=YES), and the ability to affect system function (SYSTEM=YES) at NEWYORK. You can override these defaults. For example, NODE2's initialization data set restricts NEWYORK from affecting system function (AUTH=SYS=NO) at WASHDC.

4 LINE(1) PASSWORD=LINEPSWD

Specifies the password associated with the line connecting nodes 1 and 2. Notice that both nodes include the password specification in their line definition initialization parameters. NEWYORK will include the line password in the initial signon record sent to WASHDC. WASHDC verifies the line password and sends a response record to NEWYORK that also includes the line password that NEWYORK verifies before continuing signon record processing.

5 JOBDEF RANGE=(1-499)

Node 1 will assign job identifiers beginning at 1 and assign no identifier greater than 499 for jobs originating locally. For jobs received from node 2 for execution at node 1, node 1 attempts to assign the JOBID that appears in the NJE header received from node 2. Because of the JOBDEF RANGE= specification at node 2, jobs originating at node 2 will have job identifiers beginning at 500 up to and including 999999. Because node 1 does not assign identifiers higher than 499, it can

assign numbers beginning at 500 and ending at 999999 (the JES2 system default for job identifiers) to jobs received from node 2. Consequently, each job identifier is unique, and a job transmitted from node 1 to node 2 will have separate job identifiers at each node.

6 RDR(1) PRTDEST=NEWYORK

Indicates that NEWYORK will print normal print output produced by jobs read in at RDR1 in WASHDC. The PRTDEST=NEWYORK specification corresponds to the NODE 1 node definition statement as defined at node WASHDC. If no NODE 1 statement has been specified at node WASHDC, a printer attached to the system specified by OWNNODE= is the default printer (WASHDC in this example).

7 DESTID(BRONX)

The statement assigns the name BRONX to the remote workstation numbered 1 (RMT1) attached to NODE1. You can now use BRONX to send jobs or data to the remote workstation.

Adding a node to an existing network

The network illustrated in Figure 91 on page 314 is more complex than those examined thus far. It introduces some of the initialization parameters that influence path selection. The addition of BOSTON (node 3) to the network requires you to update the initialization statements for NEWYORK and WASHDC as follows:

- Add a NODE(nnnn) NAME=BOSTON statement to define BOSTON, and on this initialization statement, include a PASSWORD=(SEND=word1,VER=word2) parameter to define unique passwords between BOSTON and the other two nodes.
- Add 1 to the NODENUM= parameter on the NJEDEF statement.

In the network in Figure 91 on page 314, there are three possible connections:

- NEWYORK to WASHDC (line 1)
- NEWYORK to BOSTON (line 3)
- WASHDC to BOSTON (lines 2 and 5)

The parallel lines between WASHDC and BOSTON do not represent a separate connection. This link is an alternate line to an adjacent node.

Multi-access SPOOL configuration considerations for NJE

A multi-access spool configuration (MAS) can participate in the JES2 network. JES2 considers the spool and its attached members to be one NJE network node. However, the parameters used to define JES2 members that share a spool can affect the performance of the network (see “Queuing messages to a multi-access SPOOL node” on page 321).

The following describes:

- An example of JES2 specifications required on an MAS node (BSC, SNA, or TCP/IP)
- JES2 specifications that can affect network performance
- An example of JES2 specifications particular to an MAS SNA node.

Example of a multi-access SPOOL node

All JES2 members of a multi-access spool node must specify the same value for the OWNNODE= and NODENUM= parameters on the NJEDEF statement in each of their initialization data sets. They must also have identical node definition parameters as specified on the NODE(nnnn) initialization statement, including the node name.

In the network in Figure 93, node 2 is a multi-access spool configuration with the symbolic name WASHDC. There are 2 members active on node 2 (SYSA and SYSB), each of which has an active NJE line to node 1 (NEWYORK).

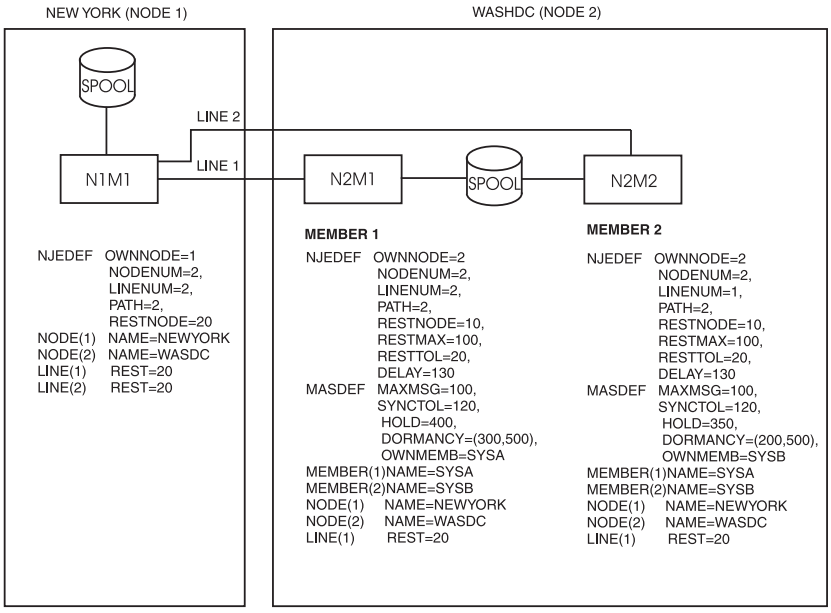


Figure 93. Multi-Access Spool Configuration (BSC)

In this configuration, there is one NJE connection between node 1 and node 2. This connection connects NEWYORK member 1 with WASHDC member 1. There are two active NJE links for one NJE connection; this is considered a multi-trunk connection. The link with the lower resistance is considered to be the primary trunk. In Figure 93, line 2 is the primary trunk because its resistance (60) is less than line 1's resistance (70). The resistance of the primary trunk (60) is associated with the NJE connection that is sent into the network. Data is sent on whatever link is available.

Queuing messages to a multi-access SPOOL node

Optionally, you can direct commands and messages to a particular JES2 member attached to the spool. Use $NnnnMmm to route a command to a specific member, where mm is the member number as defined on the MEMBER(mm) statement.

The MASMSG= parameter on the CONDEF statement specifies the maximum number of commands or messages that can queue between any two members of

the multi-access spool configuration before the network discards messages. This count also applies to the number of input messages or commands a member can queue to itself.

When a member of a multi-access spool node queues a message to another member of that node, there is no interrupt mechanism to inform the receiving member that a message exists. The receiving member periodically reads the shared job queue record and examines queue information for new messages. The following JES2 initialization parameters control the length of time it takes for the receiving member to recognize the queuing. You specify the parameters (HOLD= and DORMANCY=) on the MASDEF initialization statement, and specify (DELAY=) on the NJEDEF statement. The $T operator command can modify each of these parameters to make it easier to tune your multi-access spool node.

You should experiment to determine the parameter values that yield satisfactory responsiveness without causing adverse increases in system overhead for the individual nodes. Figure 93 on page 321 shows examples of these parameters. These values are for illustrative purposes and are *not* recommendations.

MASDEF queuing parameters

The MASDEF statement defines various default values for a multi-access spool configuration. When a member of a multi-access spool configuration participates in a network, that member must have control of the shared queues before it can transmit or receive network data. The HOLD= and DORMANCY= parameters on MASDEF control the access to the shared queues by a member of the multi-access spool configuration. Careful consideration when specifying these parameters will help the performance of your communications within the network.

HOLD: Specifies the length of time the sending system holds the hardware reserve lock on the job queue. If the posting of work occurs early in the time slot represented by the HOLD= value, then the receiving system might not retrieve the job queue in its attempt to read because the sending system still holds the lock. Small values increase system overhead while increasing responsiveness to queuing.

DORMANCY (mmmm,nnnn): The mmmm value specifies the minimum length of time the receiving system will not consider requests to the job queue. This will also delay the sending system because it must own the job queue to queue the message. The nnnn value specifies the maximum length of time between requests for the job queue by the receiving system. Small values for both increase system overhead while increasing responsiveness to queuing.

NJEDEF queuing parameter

The DELAY= parameter, on the NJEDEF statement, specifies the maximum delay time for intra-nodal message or command transmission. JES2 considers the value of the DELAY= parameter when transmitting commands and messages across the shared spool. If a JES detects that the time specified in DELAY= is exceeded, it assumes a lockout has occurred. Too small a value can have adverse affects on spool use.

Defining a multi-access SPOOL node (SNA considerations)

For SNA sessions, an installation must define each JES2 member attached to the MAS configuration as an application to VTAM. In addition, JES2 must supply unique application names that VTAM will use to identify JES2 through APPL(avvvvvvv) initialization statements. These considerations do not apply to BSC networks.

The application ID subscript (avvvvvvv) on the APPL statement that defines the JES2 member reading the initialization statement should correspond to the APPLID= on the LOGON(1) statement for that JES2. You can use a form of the APPLID= that is a variation of the node name qualified by multi-access spool member number.

Figure 94 shows samples of the initialization statements required for an SNA network, while adding a Node 3 (LA) to the two nodes (NEWYORK and WASHDC) shown in Figure 93 on page 321 for three NJE nodes, all of which are eligible for SNA sessions and one of which (WASHDC) is an MAS configuration. Notice that all nodes specify the APPL initialization statements defining the multi-access spool members at WASH specified. The APPL(avvvvvvv) initialization statements build identical application tables for each JES2.

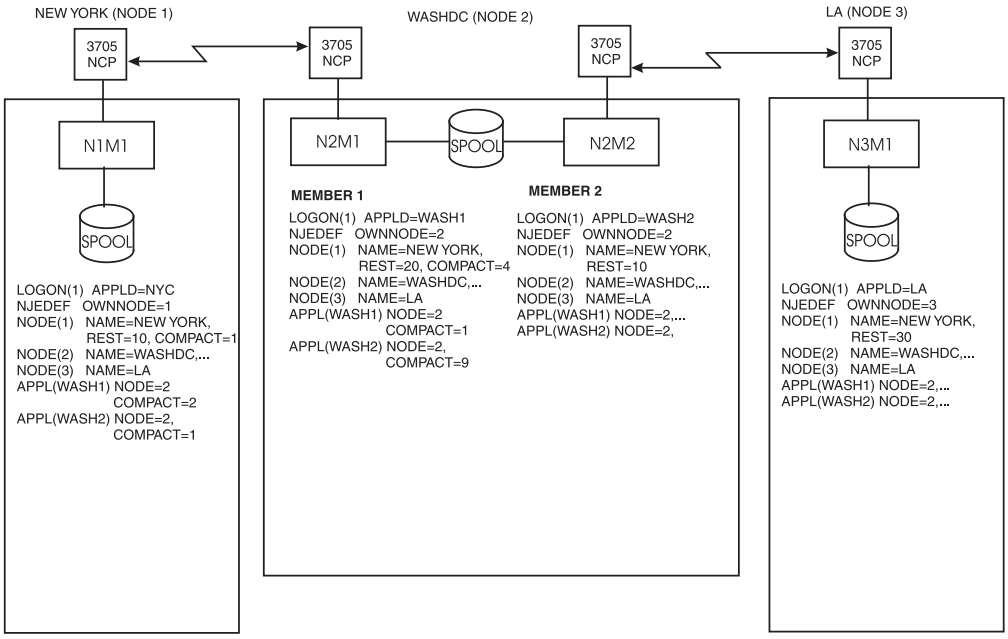


Figure 94. Multi-Access Spool Configuration (SNA)

Defining a multi-access SPOOL node (TCP/IP considerations)

For TCP/IP connections, an installation has a number of options when multiple MAS members connect to the network. An installation can define a unique IP address and port for each member of the MAS or for each NETSRV(nnn) in the MAS. In this way, connections into this node can be directed to a specific NETSRV(nnn) instance within the MAS. One advantage is that if you configure TCP/IP correctly, you can ensure there are multiple redundant connections between 2 nodes and lessen the impact of a failure (of a system or a piece of networking hardware) on your network; However, you might not be able to make a connection when there is a problem with the target NETSERV(nnn).

Another method uses SYSPLEX distributor to provide a single VIPA address for a SYSPLES. In this case, multiple NETSRV(nnn) address space (on multiple MVS images) will listen to the same IP address (VIPA). The connecting system only knows one address for the target system and Communication Server with the assistance of WLM determines which NETSRV handles the connection. An outage of a system or some network hardware does not impact the ability to establish an

NJE connection; however, if there is a system outage, the active NJE connection that was assigned to that system will be terminated. New connections can then be established. Using this method, you cannot ensure that two parallel connections between two nodes will use different NETSRVs on different members, thus losing some redundancy. An advantage to this method is that it makes a simpler network setup and ensures that if at least one NETSRV is active, an NJE session can be established.

Consistency of networking information across an MAS

Node definition information is shared across all members of an MAS, and all nodes within the network are forced to have the same attributes. It applies to the following NODE(nnnnn) parameters:

- NAME=
- SUBNET=
- PATHMGR=
- PRIVATE=
- ENDNODE=
- DIRECT=

If inconsistencies are detected in the definition of any of these parameters on a restart of JES2, a $HASP563 message is issued for each node that is different. Depending on the start type, the init deck changes may be ignored, the operator may be queried as to whether the changes should be honored, or the start may be denied.

Chapter 6. Remote job entry (RJE)

The JES2 **remote job entry (RJE)** function provides the ability to submit jobs and receive system output (SYSOUT) at remote facilities as if the jobs had been submitted at a local facility. In this chapter, the term **RJE workstation** or **remote** is used to indicate these RJE facilities. JES2 supports both **systems network architecture (SNA)** and **binary synchronous communication (BSC)** RJE workstations. JES2 RJE communication can use any of the following transmission protocols:

- BSC - that only allows the transmission of a single print or punch stream
- BSC multileaving - that simulates the transmission of multiple print and punch streams simultaneously
- SNA - that allows multiple print and punch streams simultaneously and accesses JES2 through VTAM.

The RJE workstations can be attached to JES2 by synchronous data link control (SDLC) or by a (point-to-point) BSC link. The RJE workstation becomes a logical extension of the local computer facility, and JES2 expects the RJE workstation to be controlled by a remote operator.

RJE workstations can include a processor such as a System/390®. RJE workstations also support hardware terminals, multi-leaving BSC workstations, and SNA logical devices and can include **remote devices** (printers, punches, card readers, and a console).

An RJE workstation can also consist of a **remote terminal** that does not have a processor. A remote terminal, for example, a 2780 or 2770, can be used for entering jobs into and receiving data from JES2.

This chapter describes the following:

- "Defining RJE workstations to JES2" on page 326.

  The JES2 initialization statements required to define an RJE workstation to JES2.
- "SNA RJE considerations" on page 330.

  The JES2/VTAM connection that is required to use systems network architecture (SNA) protocols for RJE processing.
- "BSC RJE processing" on page 331.

  The BSC protocols used in RJE processing, including a definition of multileaving and RMT generation.
- "SMF record summary: RJE processing" on page 333.

  The SMF records written during RJE processing.
- "Defining lines for RJE workstations" on page 334.

  How lines must be defined at initialization for RJE processing.
- "Starting and stopping remote job entry" on page 337.

  The JCL statements and their syntax required to begin and end RJE processing.
- "JES2 RJE bind image" on page 342.

  The VTAM statements and the corresponding JES2 parameters that create the bind image that JES2 uses to communicate with remote devices.
- "Altering the sequence of operations from an RJE workstation" on page 344.

How a remote operator or the JES2 systems programmer can control how SYSOUT is printed at an RJE workstation.

- "Pooling RJE workstations" on page 345.

  How JES2 can route output from a series of RJE workstations to one device to either expand or consolidate installation resources.
- "Remote message spooling" on page 347.

  How JES2 spools messages that cannot be immediately sent to a remote console.

Defining RJE workstations to JES2

RJE workstations are defined to JES2 through the following initialization statements and operator commands. (For detailed information about parameter formats, default values, and other coding considerations, see *z/OS JES2 Initialization and Tuning Reference* and *z/OS JES2 Commands*.

Table 65. JES2 Initialization Statements and Commands Used to Define a Network

Statement	Purpose	BSC or SNA
APPL(jxxxxxxx)	Associates a JES2 node with a VTAM application identifier and specifies session characteristics for SNA connections. See *z/OS Communications Server: SNA Resource Definition Reference* for information about defining application identifiers.	SNA
COMPACT	Defines a compaction table to be used by JES2 in SNA RJE workstation communications.	SNA
DESTID(jxxxxxxx)	Defines the symbolic name for a JES2 route code, which may define a specific RJE workstation.	Both
LINE(nnnn)	Defines the characteristics of a telecommunication line, such as whether it is a logical SNA line (UNIT=SNA) or an actual BSC line (UNIT= the line's unit address).	Both
LOGON(nnn)	Specifies the VTAM application identifier this node uses to communicate with VTAM for an SNA connection.	SNA
RMT(nnnn) $ADD RMT(nnnn)	Defines an RJE workstation to JES2. For SNA RJE workstations, these definitions must specify DEVTYPE=LUTYPE1. LUTYPE1 indicates that this RJE workstation requires a logical line to communicate with JES2. Also, SNA RJE workstations can specify CONS=YES if the workstation contains a remote console. This remote console is considered as one of the eight devices an RJE workstation can possess.	Both

Table 65. JES2 Initialization Statements and Commands Used to Define a Network (continued)

Statement	Purpose	BSC or SNA
TPDEF	Defines the JES2 teleprocessing characteristics: • Buffer size and placement • VTAM sessions • Message limits for remote consoles • Highest number RJE workstation allowed	Both

You can define an RJE workstation on one or more members of a multi-access spool (MAS) configuration, but an RJE workstation can be active on only one member at a time. If an RJE workstation is not defined on a particular member, then no device control blocks (DCBs) are allocated for that RJE workstation on that member. The processor control elements (PCEs) required to support the devices attached to RJE workstations are not allocated until the RJE workstation becomes active and are not freed until the RJE workstation is no longer connected. Also, note that necessary control blocks for BSC RJE workstations on dedicated lines are built when the line is started.

An installation must specify a RMT(nnnn) statement or a $ADD RMT(nnnn) command to define an RJE workstation. Specify the highest number for an RJE workstation allowed on a member through the RMTNUM= parameter on the TPDEF initialization statement. The RMTNUM= parameter defines the virtual storage (68 bytes below 16 Mb in virtual storage for each RJE workstation) for RJE workstations. If RMTNUM=99, JES2 provides enough virtual storage for 99 RJE workstations whether those RJE workstations have been defined or not. If, at initialization, your highest-numbered RJE workstation is numbered above the RMTNUM= value, for example, you define RJE workstation 106, JES2 accepts the definition, increases RMTNUM= to 106, and responds with the following informational message:

```
$HASP510 TPDEF RMTNUM PARAMETER INCREASED TO 106
```

where *106* is the highest–numbered RJE workstation specified at initialization. By defining RJE workstation (RMT106) be aware that you have left a ‘gap’ of six (RMT100 through RMT105) RJE workstations that are available for future definition.

Planning for RJE workstation system growth

Define the RMTNUM= parameter carefully to minimize the amount of unused virtual storage on your JES2 member. However, if you expect to add additional RJE workstations, you should consider setting up your RJE workstation definitions by either:

- Setting this number somewhat higher than your current configuration requires to create a pool of available RJE workstation definitions or
- Leave ‘gaps’ in your RJE workstation definitions that although specified at initialization are only defined when you need them at some future time. Remember the use of $ADD RMT is limited by the highest–numbered RJE workstation.

If you do not specify the RMTNUM= parameter, it defaults to the highest-numbered RJE workstation specified at initialization. If you do specify RMTNUM=, as noted above, it can be overridden, but **only** during initialization.

After initialization, you cannot add an RJE workstation with a number higher than that specified on the RMTNUM= parameter. If you enter a $ADD RMT(nnnn) command for an RJE workstation greater than the RMTNUM= parameter value, JES2 responds with an error message.

Defining RJE devices

If you specify a RMT(nnnn) statement or a $ADD RMT(nnnn) command without any parameters, JES2 provides the default of 1 remote printer, 1 remote reader, and 0 remote punches. During initialization, if the NUMPRT= parameter defaults to 1 and an installation defines two remote printers, JES2 recognizes only the first remote printer.

You can specify a RMT(nnnn) statement as follows:

```
RMT(6) NUMPRT=0,NUMPUN=1...
```

to instruct JES2 to provide 1 punch and, by default, 1 reader.

If you specify a RMT(nnnn) statement explicitly defining no devices,

```
RMT(5) NUMPRT=0,NUMPUN=0,NUMRDR=0...
```

the RJE workstation is not defined to the JES2 member. If you enter a $ADD RMT(nnnn) command or a $T RMT(nnnn) command explicitly defining no devices, JES2 issues an error message.

Note: For SNA RJE workstations, CONS=Yes specifies a console as a device, and this device is counted as 1 of the 8 allowed total devices that can be attached to the RJE workstation (readers are not included in this 8-device limitation). You must define at least one device (printer, punch, reader, or a console) for JES2 to consider the RJE workstation definition to be valid.

After initialization, if you need to add RJE workstations, you can use the $ADD RMT command to do so if:

- The RJE workstation is numbered less than or equal to the RMTNUM= parameter on the TPDEF statement. (Remember, a number higher than RMTNUM= is not valid.)
- You define RJE workstations with numbers from a pool of numbered but yet undefined RJE workstations. (Because you planned ahead and set RMTNUM= somewhat higher than you needed when you initialized JES2 or left gaps in your RJE workstation definition as noted in "Planning for RJE workstation system growth" on page 327.)

Be aware that any RJE workstation added by the $ADD RMT command is lost across a JES2 restart. Therefore, if you want to permanently retain this remote definition, either when convenient or on the next JES2 restart, add the appropriate RMT(nnnn) initialization statements to your initialization data set.

Although you cannot add remote devices (R(nnnnn).PR(m), R(nnnnn.PU(m), and R(nnnnn).RD(m)) directly, you can add them by defining them on the RMT(nnnn) NUMPRT=, NUMPUN=, and NUMRDR= parameters. This defines them to JES2 with all their defaults. When defined here, use the $T R(nnnnn).PR(m), $T R(nnnnn).PU(m), and $T R(nnnnn).RD(m) commands, to reset the default values as you require. For example, at initialization your initialization data set contained:

```
TPDEF RMTNUM=206...
RMT(1)...
⋮
RMT(200)...
```

This provided a pool of RJE workstations numbered 201 through 206 for future use. You can now enter, for example:

```
$ADD RMT(201-206) NUMPRT=4,NUMRDR=2...
```

This command adds 4 printers and 2 readers to each of the 6 new RJE workstations (RMT201 through RMT206). Then use the $T R20n.PRm and $T R20n.RDm commands to select printer and reader specifications appropriate to your installation. Such as:

```
$tr201.pr1,cl=a,forms=letter...
$tr201.pr2,cl=b,forms=legal...
⋮
$tr206.pr4...
```

Conversely, when you need to decrease the number of remote devices, do so by decreasing the NUMPRT=, NUMPUN=, or NUMRDR= values. JES2 deletes the devices in descending order starting with the highest numbered device (of the specified type) first. Continuing with the previous example that added four printers to RMT(206), issuing

```
$T RMT(206) NUMPRT=2
```

causes JES2 to delete the two highest-numbered printers (PR4 and PR3) on RMT(206).

Modifying RJE workstations

Some parameters on the RMT(nnnn) definition can be modified regardless of whether the RJE workstation is active or not. Most parameters, however, can be modified through a $T RMT(nnnn) command only if the RJE workstation is not active. You can determine whether an RJE workstation is active or inactive by either:

- Entering the $D RMT(nnnn) command - the STATUS= value displays as ACTIVE or INACTIVE.
- Entering the $D ACTRMT(nnnn) command - $HASP137 displays all the active RJE workstations throughout the MAS, on a particular member, or a particular workstation on a particular member (based on how you use the command's filtering capabilities).

Note:

1. If you modify an RJE workstation using the $T command, for most parameters the RJE workstation need only be inactive on the member on which you are changing the definition. If a response from a $D ACTRMT command shows the RJE workstation as active on some other member, all RMT parameter definitions can still be modified.
2. If you enter a $T RMT(nnnn) command that includes one of the parameters that requires that the RJE workstation be inactive against an active RJE workstation, JES2 responds with an error message.

Recovering RJE workstations from failed members

If a member of your MAS fails while an RJE workstation was active, the RJE workstation can sign on to another member of the MAS.

Because JES2 automatically resets the record of active RJE workstations when a member fails, you can activate an RJE workstation on a member if:

- It is not active elsewhere in the MAS, and
- At least one member of the JES2 MAS is active or made active before reactivating the failed member (and its remote workstations).

To determine the active RJE workstations in a MAS, enter the $D ACTRMT(nnnn) command. JES2 responds with the following console display:

```
$HASP137 ACTRMT(nnnn) MEMBER=memname
```

where *nnnn* specifies the number of the active RJE workstation and *memname* indicates the name of the member (derived from the MEMBER(nn) initialization statement) on which this RJE workstation is active.

SNA RJE considerations

RJE workstations use systems network architecture (SNA) to gain access to JES2 through VTAM. During VTAM system definition, these RJE workstations (for example, 3790 terminals, 3770 terminals, and System/32 workstations) are defined to VTAM as logical units (LUs) and physical units (PUs) by means of the VTAM LU and PU definition statements, respectively. At VTAM system definition, each installation must decide how to tailor the JES2/VTAM support for its SNA stations. The parameters of the LU and PU statements affect not only how JES2 operates, but also how the remote operator uses the workstation.

VTAM LU and PU parameters that affect SNA RJE

Some of the LU parameters affect both the logon syntax and the handling of logon requests. For a description of what the following parameters specify and how to code the LU macro instruction, see *z/OS Communications Server: SNA Resource Definition Reference*. The teleprocessing links (the physical connections) are transparent to JES2; they are managed exclusively by VTAM and the network control program (NCP).

Table 66notes several LU parameters that are a small subset of those that affect the VTAM session with JES2.

Table 66. VTAM Parameters that Affect JES2

Unit Type	Name	Purpose
LU	USSTAB	Specifies a table to be used in generating a logon sequence.
LU	SSCPFM	Specifies whether an RJE workstation supports formatted or unformatted systems services.
PU	DISCNT	Specifies how the teleprocessing link is to be disconnected.

VTAM APPL definitions for SNA RJE

A VTAM APPL definition statement defines an application program to VTAM. JES2 requires the following values from the VTAM APPL statement:

Table 67. VTAM APPL Statement Parameters Necessary for a JES2 Connection

Parameter	Purpose
AUTH=ACQ	Specifies the setting (ACQ) if this RJE workstation uses automatic logon.
APPLNAME=	Corresponds to the APPLID= parameter on the JES2 LOGON(nnn) initialization statement. (The LOGON(2) is used as the default APPLID for JES2 initiated logons.)
PRCT=	Corresponds to the PASSWORD= parameter on the JES2 LOGON(nnn) initialization statement.

SNA RJE buffer size

Each teleprocessing buffer for SNA is prefixed in storage with a request parameter list (RPL). JES2 uses the values you have specified to determine the buffer size for RJE processing. JES2:

1. Starts with the SNABUF=(SIZE=) parameter value (within a range of 256-3840) on the TPDEF initialization statement.
2. Increases SIZE= value to BUFSIZE= parameter value on the SNA RMT(nnnn) initialization statement if BUFSIZE= is larger than SIZE=.
3. Increases SIZE= to 280 bytes if SIZE= is less than 280, and the LINENUM= parameter on the NJEDEF initialization statement is greater than 0.
4. Adds 256 (the size of the RPL prefix) to the buffer size, rounded up to a multiple of 8, and rounded down to a maximum of 4096.

You can calculate the 'usable' teleprocessing (TP) buffer size by subtracting the RPL prefix area. You can display this value with a $D TPDEF operator command, which results in the $HASP839 message.

Valid RU sizes

The actual transmission response/request unit (RU) size for SNA RJE buffer is limited further (rounded down) by the SNA architecture, and JES2 enforces the restriction through the BUFSIZE= parameter on the RMT(nnnn) definition.

After initialization, if you use the $ADD RMT(nnnn) command to add an SNA RJE workstation, or you use the $T RMT(nnnn) command to modify one, the BUFSIZE= parameter value cannot be larger than the SNABUF=(SIZE=) parameter value on the TPDEF initialization statement. The BUFSIZE= parameter is rounded down for RU bind limitations before it is checked against the SNABUF=(SIZE=) parameter value on the TPDEF initialization statement.

BSC RJE processing

Reading, printing, and punching between the host processor and the RJE workstation are asynchronous events. For example, the processor is either transmitting print data or transmitting punch data or reading an input stream. For an explanation of how the remote operator can influence the order of these events, see "Altering the sequence of operations from an RJE workstation" on page 344.

Communication between the local processor and BSC RJE workstations uses a JES2 facility, called **multileaving**, which simulates the transmission of multiple print and punch streams. Multileaving allows JES2 to receive multiple console messages and input streams. With multileaving, several operations can take place concurrently. Operators at RJE workstations that have no console can enter commands into the

input stream in the normal manner. Command replies are scheduled back to the RJE workstation for printing on a remote printer.

RMT generation is the JES2 procedure for generating multileaving remote terminal processor programs for RJE from programmable RJE workstations. An RJE workstation is established by a JES2 program, RMTGEN, during or after system installation. These programs allow multileaving workstations (see "BSC multileaving workstations supported by JES2") to operate as RJE workstations with JES2. RMT generation requires other procedures for its processing; for example, procedures for allocating space and cataloging. It also requires certain spool data sets for job processing after generation. These procedures and data sets, also required by JES2 generation, are described in Chapter 2, "Controlling JES2 processes," on page 71.

BSC multileaving workstations supported by JES2

JES2 provides RJE workstations with multileaving support for the following programmable workstations:

- IBM System/360* Model 20 (Submodels 2, 4, 5, and 6) and the 2922 Remote Workstation RTP program with the following selectable options:
  - 1403 Printer
  - 1442 Card Read Punch
  - 2152 Printer-Keyboard
  - 2203 Printer
  - 2501 Card Reader
  - 2520 Card Read Punch
  - 2560 Multi-Function Card Machine
- IBM System/360 (Models 22 and up) and System/370 (Models 115, 125, 135, 145, 148, 155, 158, 165, 168, and 195) with the following selectable options:
  - 1052 Printer-Keyboard
  - 1403 Printer
  - 1442 Card Read Punch
  - 1443 Printer
  - 2501 Card Reader
  - 2520 Card Read Punch
  - 2540 Card Read Punch
  - 3203 Printer (supports 3203-1 through 3203-4)
  - 3210/3215 Printer-Keyboard (supported as a 1052)
  - 3211 Printer
  - 3504 Card Reader (supported as a 2501)
  - 3505 Card Reader (supported as a 2501)
  - 3525 Card Punch (supported as a 1442)
  - 5203 Printer
  - 5313 Console for the Model 125 (requires 1052 compatibility feature)

BSC teleprocessing buffer considerations

The BSCBUF=(SIZE=) parameter on the TPDEF initialization statement specifies the size of BSC teleprocessing buffers that JES2 allocates below 16 megabytes in virtual storage. This value is specified in bytes (520-3960). If you do not specify this parameter during initialization, JES2 provides the minimum of 520 bytes.

After initialization, if you use the $ADD RMT(nnnn) command to add a BSC RJE workstation, or you use the $T RMT(nnnn) command to modify one, the BUFSIZE= parameter value cannot be larger than the BSCBUF=(SIZE=) parameter value on the TPDEF initialization statement. If you specify a larger BUFSIZE value, JES2 responds with an error message.

If you specify the following values for the TPDEF initialization statement, the SIZE= subparameters on BSCBUF= can be overridden.

```
TPDEF SNABUF=(LIMIT=200,SIZE=1840),
      BSCBUF=(LIMIT=200,SIZE=1792),
      MBUFSIZE=1840
```

After initialization, a $D TPDEF command displays the following:

```
$HASP839 AUTOINTV=32,BSCBUF=(LIMIT=200,WARN=90,SIZE=1840)
$HASP839 MBUFSIZE=1840,RMTMSG=150,RMTNUM=50,SESSION=65535,
$HASP839 SNABUF=(LIMIT=200,WARN=80,SIZE=1840)
```

To avoid having the SIZE= subparameters for BSCBUF= overwritten by the MBUFSIZE= parameter, ensure that the value of the SIZE= subparameter is greater than that specified by MBUFSIZE=.

Note: If a RMT(nnnn) initialization statement specifies a higher BUFSIZE= value, JES2 increases the BSCBUF=(SIZE=) parameter.

SMF record summary: RJE processing

The following lists the SMF records written during RJE processing. For a complete list of the record contents, see *z/OS MVS System Management Facilities (SMF)*.

Table 68. SMF Records Used by JES2 RJE

Record Number	Action that Causes the Record to be Written
47	• A BSC line is started [$S LINE(n) command]. • A BSC line is restarted [$E LINE(n) command]. • A BSC RJE workstation user signs on.
48	• A BSC line is stopped [$P LINE(n) command]. • A BSC line is restarted [$E LINE(n) command]. • A BSC RJE workstation user signs off.
49	A BSC RJE workstation user signs on with an incorrect password.
52	• An SNA line is started [$S LINE(n) command]. • An SNA line is restarted [$E LINE(n) command]. • An SNA RJE workstation user signs on.
53	• An SNA line is stopped [$P LINE(n) command]. • An SNA line is restarted [$E LINE(n) command]. • An SNA RJE workstation user signs off.
54	An SNA RJE workstation user signs on with an incorrect password.

Defining lines for RJE workstations

The following explains how to define RJE lines for both BSC and SNA protocols (TCP/IP lines cannot be used for RJE). Define lines that connect remote devices to RJE workstations at initialization on LINE(nnnn) initialization statements. Note the following restrictions:

- You cannot use a line for network job entry (NJE) or have NJE devices dedicated to it.
- You cannot add (through the $ADD RMTnnnn command) or modify (through the $T RMTnnnn command) an RJE workstation unless the lines intended for its use have been predefined during initialization.
- You cannot use channel-to-channel (CTC) lines for RJE workstations.

There are two ways an RJE workstation can specify lines during initialization:

- Nondedicated
- Dedicated (often referred to as a **leased line**)

Nondedicated lines

A nondedicated line is a line that has been defined, but is not specified on the LINE= parameter of a RMT(nnnn) statement, a $T RMT(nnnn) command, or a $ADD RMT(nnnn) command.

To begin an RJE processing session with nondedicated lines, each RJE workstation must provide a /*SIGNON JES2 control statement. The central operator, who controls BSC line activation for all RJE workstations connected to a member, is aware of workstation use through the /*SIGNON and /*SIGNOFF JES2 control statements entered through the console.

For both SNA and BSC RJE workstations, nondedicated RJE workstations can be configured to meet the needs of your installation. For example, one physically connected RJE workstation can be initialized as multiple nondedicated RJE workstations for use by different groups at different times.

When the RJE workstation is not signed on, it cannot process jobs, output, messages, or queue commands to JES2. Although you can queue work to such an RJE workstation, it remains on the JES2 spool until the RJE workstation again signs on. Also, one RJE workstation can be used as a backup for an inoperable RJE workstation. You must sign on the backup RJE workstation with the inoperable RJE workstation's ID.

RJE BSC dial-up connections

RJE workstations that require the operator to dial a number to connect them to the JES2 processor (through **dial-up** lines) normally do not specify a dedicated line. Therefore these dial-up RJE workstations can connect to any available nondedicated line.

However, RJE workstations connected through dial-up lines might be considered a data security risk at your installation. Because the line remains active, the RJE workstation operator must not disconnect the line before signing off the RJE processing session.

Any RJE workstation can lease nondedicated lines through either the $T RMT(nnnn) command or the $ADD RMT(nnnn) command causing the line to become dedicated.

Dedicated lines

A dedicated line is defined and specified on the LINE= parameter of a RMT(nnnn) statement, a $T RMT(nnnn) command, or a $ADD RMT(nnnn) command.

An installation can configure dedicated lines and RJE workstations that must be connected by dial-up facilities; however, typically, dial-up RJE workstations do not use dedicated lines.

Dedicated lines can be **shared** if two or more RMT(nnnn) initialization statements specify the same line on the LINE= parameter. However, only one RJE workstation can sign on to the line at a time. To begin an RJE processing session with shared lines, each RJE workstation must provide a /*SIGNON JES2 control statement or VTAM LOGON.

Defining BSC lines for RJE workstations

JES2 allocates resources for BSC lines only when the lines are active. These resources can include a substantial amount of storage that is required by the processor control elements (PCEs) to support devices attached to RJE workstations.

A BSC dedicated line is **non-shared** if the SHARABLE=NO parameter is specified on the RMT(nnnn) initialization statement. However, setting SHARABLE=YES defines the line as capable of being shared; not that it is being shared, and when specified as SHARABLE=YES, the line will require sign on. If you enter a $D LINE command, a non-shared line displays RMTSHARE=NO and does not require sign on.

Dedicated lines that are non-shared do not require a /*SIGNON JES2 control statement to begin an RJE processing session. If you specify a /*SIGNON statement on a dedicated line, it is ignored because the workstation is considered active when the line is started. Note that lines are started by the central operator, who might not be aware of station use. Because specifying a /*SIGNON statement is a manual process at all RJE workstations, dedicating lines as non-shared conserves resources.

BSC lines that have been defined as dedicated can be **sharable** by specifying SHARABLE=YES on the RMT(nnnn) initialization statement, $T RMT(nnnn) command, or a $ADD RMT(nnnn) command. You can choose this option if you want to restrict particular lines to particular RJE workstations, but want the security provided by a /*SIGNON JES2 control statement.

Consider the following RJE workstation and line definitions to illustrate how RJE workstations can be defined as sharable or non-sharable, your use of a /*SIGNON statement, and how JES2 deals with conflicting shared line definitions:

Example 1

```
     LINE(3)    UNIT=3099,...
⋮
     RMT(16)    LINE=3,SHARABLE=YES,...
```

Example 1 defines LINE3 and a single RJE workstation (RMT16) that will use that line. However, SHARABLE=YES indicates that if another RJE workstation also defines LINE3 it can do so. In any case, a /*SIGNON JES2 control statement is required.

Example 2

```
      LINE(27)   UNIT=2799,...
⋮
      RMT(17)    LINE=27,SHARABLE=YES,...
      RMT(18)    LINE=27,SHARABLE=YES,...
```

Example 2 defines LINE27 and two RJE workstations (RMT17 and RMT18) that will use that line. However, if you mistakenly defined RMT17 by specifying SHARABLE=NO (with the intention of wanting it to share the line with RMT18), JES2 forces the RMT17 definition to SHARABLE=YES and you still obtain the result you want. Therefore, if two RMT(nnnn) statements specify the same line, the line is shared. To warn you of such an error, JES2 issues $HASP456 LINEnnnn REQUEST TO BE NON-SHARED BY REMOTE DENIED.

Now consider Example 3:

Example 3

```
      LINE(27)   UNIT=2799,...
⋮
      RMT(17)    LINE=27,SHARABLE=YES,...
      RMT(18)    LINE=0,SHARABLE=YES,...
```

Although both RMT17 and RMT18 are defined as SHARABLE=YES, the RMT18 definition now specifies LINE=0 rather than LINE=27 as it did in Example 2. JES2 defines RMT18 as capable of using any **nondedicated** BSC line. A /*SIGNON JCL statement is required to start the RJE workstation. Although LINE27 is no longer shared, JES2 will indicate RMTSHARE=YES in response to a $D RMT17 command.

Example 4

```
      LINE(27)   UNIT=2799,...
⋮
      RMT(16)    LINE=3,SHARABLE=YES,...
      RMT(17)    LINE=27,SHARABLE=NO,...
      RMT(18)    LINE=27,SHARABLE=NO,...
```

However, if you intended each RJE workstation (RMT17 and RMT18) to use a different line and mistakenly defined each to the same line (as shown in Example 4), JES2 forces both definitions to SHARABLE=YES and both RJE workstations use LINE27, different from your expectations and likely having availability implications. As in Example 2, JES2 issues $HASP456 LINEnnnn REQUEST TO BE NON-SHARED BY REMOTE DENIED to warn you of such an error.

Defining SNA lines for RJE workstations

When using SNA protocols for RJE, note that all RJE workstations (except those defined as AUTOLOG=YES) specify a LOGON initialization statement to begin an RJE processing session. SNA workstations never use the /*SIGNON JES2 control statement. SNA workstations end a session through the VTAM LOGOFF command.

RJE line passwords

An installation can establish a relationship between RJE workstations and lines through line passwords. A line password, defined by the PASSWORD= parameter on the LINE(nnnn) initialization statement, guarantees either a user or a group of

users the use of a particular line and prevents unauthorized RJE workstation operators from using that line to gain access to JES2.

To increase security checking, in addition to standard password processing, your installation can use JES2 Exit 17 for BSC RJE signon and signoff and Exit 18 for SNA RJE logon and logoff. See *z/OS JES2 Installation Exits* for more information.

Changing an RJE workstation from BSC to SNA

While your JES2 member is active, you might want to upgrade your installation's processor by changing an RJE workstation from BSC to SNA for RJE processing. To accomplish this task, perform the following steps on an inactive RJE workstation:

1. Set the LINE= parameter on your RMT(nnnn) initialization statement to zero through a $T RMT(nnnn) command:

   ```
   $T RMT4,LINE=0
   ```

2. Redefine the predefined teleprocessing line as a logical line:

   ```
   $T LINE11,UNIT=SNA
   ```

3. Redefine the terminal type to indicate that it is a logical terminal that can be accessed only through a logical line.

   ```
   $T RMT4,DEVTYPE=LUTYPE1,LINE=11
   ```

   Review all other RJE workstation parameters (such as BUFSIZE=) to ensure they are appropriate for an SNA RJE workstation.

Changing an RJE workstation from SNA to BSC

You might choose to change an SNA remote workstation to a BSC RJE workstation. Perform the following steps to change an inactive RJE workstation from using an SNA line to using a dedicated BSC line:

1. Modify the LINE= parameter on the RMT(nnnn) initialization statement through a $T RMT(nnnn) command:

   ```
   $T RMT4,LINE=0
   ```

2. Redefine the predefined logical line:

   ```
   $T LINE4,UNIT=30A
   ```

3. Redefine the terminal type to indicate the type of terminal or processor (multileaving or hardware) being used at this RJE workstation location.

   ```
   $T RMT4,DEVTYPE=2770,LINE=4
   ```

   Review all other RJE workstation parameters (such as BUFSIZE=) to ensure that they are appropriate for a BSC RJE workstation.

Starting and stopping remote job entry

Because teleprocessing lines are never considered active at JES2 initialization, each line must be activated using a JES2 start command ($S). An RJE workstation cannot be started on a member if it is already active on another member of the MAS. Use the $D ACTRMT(nnnn) command to determine the remote's status before you attempt to start it. The $S command can be issued by the operator through a command stream (for example, through the JES2 initialization data set) or through the automatic command processor. However, JES2 ignores any JES2 command statement within a job. A line is dynamically allocated when activated. A line can be deactivated and deallocated using the JES2 stop command ($P).

A remote device is considered active when its RJE workstation becomes active, provided that the device is specified for automatic start by the START= parameter on the following initialization statements:

- R(nnnnn).RD(m)
- R(nnnnn).PR(m)
- R(nnnnn).PU(m)

Otherwise, the device is considered inactive and must be started either by the RJE workstation or through local operator command.

Options for disconnecting remote lines

To minimize data security problems when using BSC dial-up lines, always specify AUTODISC=YES on the LINE(nnnn) statement for each dial-up line. When you specify AUTODISC=YES, JES2 disconnects a line automatically by simulating a $E command sequence when it detects a not-ready data set condition. (The circumstances in which JES2 detects a not-ready data set condition depends on different factors, including line configuration.)

If you specify AUTODISC=NO or do not specify this parameter during initialization, the line remains active; JES2 waits for either the data set to be made ready or the operator to take action.

Ensure that remote operators sign off before disconnecting a line. If a line-drop or a disconnect occurs on a dial-up line before signoff, another user might connect to the same port and receive data intended for the prior user of that port. (To ensure the transmission of data to the proper authorized remote terminals, use SNA or dedicated lines.)

For both SNA and BSC RJE workstations, you can also cause JES2 to automatically disconnect an inactive RJE workstation by coding a nonzero value for the DISCINTV= parameter on the RMT(nnnn) statement at JES2 initialization, the $ADD RMT(nnnn) command, or the $T RMT(nnnn) command while the member is inactive. When the amount of time specified in the DISCINTV= parameter elapses with no data sent or received on the line, JES2 disconnects the line by simulating a $E command sequence.

BSC workstations

The first action taken at nondedicated or dedicated and sharable RJE workstation is the submission of a signon statement. This statement must be submitted through the input stream as a single-image statement, and cannot be immediately followed by the first job. (Signon is ignored for dedicated, non-shared lines.)

When an RJE workstation signon is successfully completed, JES2 sends a message to the RJE workstation verifying the successful signon.

Format and definitions for the /*SIGNON statement

Column	Description
1	/*SIGNON
16	REMOTEnn | RMTnnnn
25	password1
35	newpass
73	password2

RMTnnnn|REMOTEnn
Defines the RJE workstation requesting signon.

Note: The long form (REMOTEnn) is only valid for RJE workstations 1 through 99.

password1
Defines the password established at initialization or changed by the operator for that line. If the line has a password, then password1 is required.

To establish password1, specify the LINE(nnnn) JES2 initialization statement. You can change this password through the $T LINE(nnnn) command.

newpass
Specifies the password used by the Resource Access Control Facility (RACF) to change the password of this remote. If RACF is not active, you cannot change the remote's password in this way. See Chapter 7, “Providing security for JES2,” on page 349 for information about using RACF to control RJE workstations.

If you are using RACF to control the RJE workstations, you must specify newpass the first time this remote issues a logon or signon. Otherwise, RACF will reject the access attempt. See Chapter 7, “Providing security for JES2,” on page 349 for information about defining an RJE workstation to RACF.

Password2
Defines the password maintained by RACF unless you use JES2 remote password control. If you use JES2 remote password control, JES2 establishes this password through a RMT(nnnn) initialization statement, or through a $ADD RMT(nnnn) command or a $T RMT(nnnn) command.

If the terminal has a password, then password2 is required to ensure that the RJE workstation signing on is a valid workstation.

Format for the /*SIGNOFF statement

A signoff statement is submitted to end BSC job stream processing. This statement, too, should be submitted through a card reader as a single-image statement, as is the /*SIGNON statement.

Column	Description
1	/*SIGNOFF

SNA RJE workstations

There are three ways that the operator can initiate an automatic terminal connection:

1. Entering the VTAM LOGON command that identifies the VTAM application ID to JES2.
2. Specifying AUTOLOG=YES on the RMT(nnnn) definition to make the RJE workstation eligible for automatic logon whenever work exists.

   This parameter causes JES2 to connect to an RJE workstation when SYSOUT is queued for processing.
3. Entering the $S RMT(nnnn) command to force a connection whether or not any work exists.

   This command starts the JES2/VTAM interface, which allows JES2 to begin processing logons from the RJE workstations and terminals through the following command:

   ```
   $S LGN(2)
   ```

If you use either method 2 or 3, you must also specify the logical unit name (LUNAME=) on the RMT(nnnn) statement, $ADD RMT(nnnn) command, or $T RMT(nnnn) command.

Initiating an automatic terminal connection

The operator can issue an automatic logon command before the JES2/VTAM interface is started. However, JES2 does not act upon the automatic logon command until the operator starts the logical lines and the JES2/VTAM interface. If, however, the RJE workstation is a multi-logical unit (MLU) device, JES2 automatically initiates one session only when AUTOLOG=YES

JES2 automatic logon capability: The automatic logon capability allows JES2 to contact and connect an RJE terminal automatically. This can occur either when output (print/punch) or messages are queued for transmission for that terminal or when an authorized console operator issues the appropriate JES2 command. Automatic logon allows maximum use of communication facilities such as lines, adapters, modems, and controllers. For example, a line can be dedicated to a terminal using automatic logon. The efficiency is increased because RJE workstations need not remain logged on while waiting for output. This frees facilities for use with other terminals and applications.

Although JES2 automatically calls (actually dials-out in some cases) an RJE workstation to produce output or messages, the need for a terminal operator is still present. *Automatic logon does not provide for unattended operation of the terminal*. The operator must continue to control the terminal and take the necessary error recovery actions.

The automatic logon capability is available only to terminals operating in a VTAM network using SDLC (synchronous data link control) or RECORD-mode communications. *BSC RJE workstations are not supported by automatic logon.*

Making the RJE Workstation Eligible When Work Exists: Specifying AUTOLOG=YES causes JES2 to check the output queues periodically for work. The RJE workstation will be eligible for automatic connection whenever work exists.

The RJE workstation cannot connect immediately if there is a resource shortage. However, the member retries the connection request until it is successful or until a permanent error is recorded. The RJE workstation will remain connected until it has been signed off by its operator. If an appropriate disconnect interval has been specified (the DISCINTV= statement of the RMT(nnnn) initialization statement), the RJE workstation will be disconnected automatically when it becomes idle.

Making the RJE Workstation Eligible Whether or Not Work Exists: An authorized (system authorization) operator can force a remote terminal to be eligible for connection regardless of whether work is queued for it by issuing a $S RMT(nnnn) command. The RJE workstation cannot connect immediately if there is a resource shortage. However, the member retries the connection request until it is successful or until a permanent error is recorded.

Automatic logon considerations: To initialize the automatic logon capability for use with SNA LUTYPE1 terminals, the RMT(nnnn) statement, $ADD RMT(nnnn) command, or $T RMT(nnnn) command for the RJE workstation must specify the name of the VTAM logical unit (LU) to be contacted through the LUNAME= operand. Because the automatic logon is performed internally, JES2 does not receive any logon message or password data. Therefore, an RJE workstation connected by automatic logon never uses a communication line that has been

assigned a line password. Sufficient lines (without line passwords) should be initialized through the LINE(nnnn) initialization statement to allow for expected automatic logon activity.

For normal logon procedures, issue the $S LGN(n) command should only after VTAM has been started. (When VTAM has been started, both the network controllers or communication links needed to establish a path to the remote station and the physical unit and logical unit associated with the station must be activated.) To allow an SNA RJE workstation to log on, the operator must also issue $S LNE(n) to activate a line to VTAM for the SNA RJE workstation.

Creating an SNA RJE connection

JES2 attempts to contact a terminal only when there are sufficient resources to allow connection. To establish a session, certain conditions must be met. If any of these conditions are not met, the RJE workstation is not logged on immediately. The member retries the automatic logon attempt periodically until it is successful. If the RJE workstation is currently in contact with another application (prevents logon), JES2 retries the connection request periodically.

- The RJE workstation must not be already connected to the system.
- A JES2/VTAM application interface must be available (logon DCT started).
- The RJE workstation must not be connected to another processor in the same MAS configuration.
- A JES2 logical line that does not have a line password defined must be available (started and not in use by any other RJE workstation).
- The system must not have reached the session limit specified by the SESSION= parameter on the TPDEF initialization statement.
- Enough storage must be available for the processor control elements (PCEs) required to support the devices attached to the RJE workstations. This storage requirement can be substantial depending upon the device configuration.

Using the VTAM LOGON command for SNA RJE

The SNA RJE workstations use the LOGON command to request a session with JES2. When the operator issues the LOGON command, VTAM notifies JES2 that a logon has been received, and passes to JES2 the data sent with the LOGON command. If the data is acceptable, JES2 establishes a session with the logical unit (LU) associated with the RJE workstation.

The syntax of the LOGON command for each LU is established at JES2 and VTAM system definition. For example, the password included in the data sent with LOGON might or might not be required; the operator must be told what syntax to use.

The default VTAM syntax:

```
LOGON APPLID (JES2)  [LOGMODE(name)]
      DATA (RMTnnnn,password1,password2,newpassword2)
```

JES2
Identifies JES2 as an application program.

LOGMODE(name)
Indicates a mode table entry that JES2 uses to determine some of the characteristics of the session with an SNA terminal. If LOGMODE is not specified, VTAM selects a default bind image. See “JES2 RJE bind image” on page 342 for the statements that alter the JES2 bind image.

DATA
Specifies the RJE workstation and any valid passwords.

RMTnnnn|REMOTEnn|RMnnnn|Rnnnnn|NnnRnnnnn
Specifies the RJE workstation.

Note: The long form (REMOTEnn) is only valid for RJE workstations 1 through 99.

password1
Authorizes the use of the SNA line associated with it.

password2
Authorizes the use of the SNA RJE workstation associated with it.

newpassword2
If using RACF, provides a new *password2* for the RJE workstation.

Using the VTAM LOGOFF command for SNA RJE

SNA RJE workstations can use the JES2 SIGNOFF statement or the VTAM LOGOFF ($P LGN(n) is used to stop the JES2/VTAM interface). The VTAM LOGOFF command, however, has an option which is not provided by the JES2 SIGNOFF statement. As with the LOGON command, the installation defines the syntax of LOGOFF at VTAM system definition time. The default syntax is:

```
LOGOFF TYPE({COND | UNCOND})
```

TYPE
Indicates whether a conditional or unconditional logoff is requested.

COND|UNCOND
Indicates that the session will be disconnected at the end of any current data transmission (COND) or immediately, regardless of the data transmission (UNCOND).

For more information about LOGON and LOGOFF, see *z/OS Communications Server: SNA Resource Definition Reference*. For a description of $S LGN(n), $P LGN(n), and $S LNE(n), see *z/OS JES2 Commands*.

JES2 RJE bind image

JES2 uses a standard bind image for communication with remote devices. You can only alter this bind image with the VTAM statements listed in Table 69 when JES2 sends the bind image to a remote device.

Table 69. Variable Statements Used to Alter the JES2 RJE Bind Image

Statement	Source of Value
Compression*	Can be set on by the user's input bind image. This value defaults to the PRIPROT parameter specified in the user's input bind image. Can be set on by the COMPRESS=YES parameter in the RMT(nnnn) initialization statement, the $ADD RMT(nnnn) command, the $T RMT(nnnn) command, or the LINE(nnnn) initialization statement. Will be reset if alternate code is specified in the user's input bind area.

Table 69. Variable Statements Used to Alter the JES2 RJE Bind Image (continued)

Statement	Source of Value
Alternate Code*	Can be set on by the user's input bind image COMPROT byte 5 bit 4.
Pacing Count	Can be set to value of 0 or 1 by the user's input bind image SSNDPAC, byte 7.
SLU Send*	
Pacing Count	Can be set to a value (0 to 3F) by the user's input bind image SRCVPAC, byte 8.
SLU Receive*	
RU	Will be set by the BUFSIZE= parameter on the RMT(nnnn) initialization statement, the $ADD Rmt(nnnn) command, or the $T RMT(nnnn) command.
C	Can be set on by the COMPACT parameter on the RMT(nnnn) initialization statement, the $ADD Rmt(nnnn) command, or the $T RMT(nnnn) command.
Media Flag	Can be set on by the SELECT parameter on the Rnn.PRm and Rnn.PUm R(nnnnn).PU(m) initialization statements, or the $T R(nnnnn).PR(m) and $T R(nnnnn).PU(m) operator commands.
PDIR	Can be set on by the SETUP= parameter on the RMT(nnnn) initialization statement, the $ADD Rmt(nnnn) command, or the $T RMT(nnnn) command.
PDIR Flag	Can be set on by the SETUP= parameter on the RMT(nnnn) initialization statement, the $ADD Rmt(nnnn) command, or the $T RMT(nnnn) command.
Media	Can be set on by the SELECT= parameter on the R(nnnnn).PR(m) and R(nnnnn).PU(m) initialization statements, or the $T R(nnnnn).PR(m) and $T R(nnnnn).PU(m) operator commands.
Presentation	Can be set on by the users input bind image.
Service*	PSERVIC byte 4 bit 4.

*Indicates a bind statement which can be set from a user's input bind image.

VTAM has an IBM-supplied logon mode table which provides generally accepted session protocols for a basic list of IBM device types. This logon mode table can be found in *z/OS Communications Server: SNA Resource Definition Reference*. VTAM presentation services can be specified by the SELECT= parameter on the R(nnnnn).PR(m) and R(nnnnn).PU(m) initialization statements or the $T R(nnnnn).PR(m) and $T R(nnnnn).PU(m) commands. However, some options are not compatible with all device types.

Although consideration need only be given to the variable statements, the bind image shown below illustrates the JES2 standard session statements. It is not necessary to include these statements in the input bind image.

```
MODEENT  LOGMODE=(name),TYPE=X'01',FMPROF=X'03'TSPROF=X'03',   C
         PRIPROT=X'B1',SECPROT=X'A3',COMPROT=X'7080',          C
         SSNDPAC=X'00',SRCVPAC=X'00',RUSIZES=X'0000',          C
         PSNDPAC=X'00',PSERVIC=X'011000009100C00000010040'
```

Setting line density for RJE devices

You can select the line density for a remote printer by:

1. Specifying the FCBLOAD parameter on either the R(nnnnn).PR(m) initialization statement or the $T R(nnnnn).PR(m) operator command.

2. Turning on bit 4 of byte 4 in the PSERVIC field of the JES2 RJE bind image.

This causes JES2 to select the proper density (6 lines per inch or 8 lines per inch).

Certain older printers (for example, the 3777-1 and the 3776-1) cannot have their densities set with PSERVIC's bit 4 of byte 4 set on and require the standard JES2 bind image. If you receive message $HASP190 at your remote printer, you **must** manually advance the form in the printer to line 1 of the page before starting the printer. This will cause the PSERVIC byte to look like this:

```
PSERVIC=X'011000009900C00000010040'
```

Altering the sequence of operations from an RJE workstation

JES2 provides two parameters that allow the RJE workstation operator to control the sequence of operations at the RJE workstation. One parameter (SUSPEND) allows JES2 to interrupt job output processing, while the other (WAITIME) causes JES2 to wait until it is between print or punch processing. For transmitting either jobs or JES2 commands to the JES2 member that is attached to this RJE workstation, a remote operator can specify:

- The WAITIME= parameter on the $T RMT(nnnn) command
- The SUSPEND= parameter on either the $T R(nnnnn).PR(m) command or the $T R(nnnnn).PU(m) command

The delay time specified on the WAITIME= parameter takes effect between printing or punching the output of each job. This delay gives the operator the opportunity to ready the card reader and change the terminal status to transmit data. JES2 reads the input stream before resuming printing or punching.

When each BSC hardware terminal printer or punch is defined with SUSPEND=YES on either the R(nnnnn).PR(m) statement, the $T R(nnnnn).PR(m) command, the R(nnnnn).PU(m) statement, or the $T R(nnnnn).PU(m) command, you can interrupt the printing or punching of jobs by stopping the output device. When the device is again readied by the operator (after transmitting jobs or JES2 commands to the JES2 member), JES2:

1. Simulates an interrupt situation by flushing its current I/O buffers and printing the remote separator page, if one exists.
2. Determines whether the remote card reader is ready.

   If the remote card reader is ready, input is read in. If it is not ready, the highest priority output is selected. This output can be the resumption of the current operation or the start of another data set. The delay must be sufficiently long for the terminal to notify JES2 of the stopped device state. (The time depends on the terminal type.)

If SUSPEND=NO on the remote device, JES2 resumes the current operation after the device is readied again.

For both multileaving terminals and SNA devices, JES2 ignores the SUSPEND= parameter on the R(nnnnn).PR(m) and R(nnnnn).PU(m) statements; both input and output operations can occur at the same time. For SNA remote devices, operations are not suspended automatically on a time interval. For example, an operator can suspend a data stream being sent to an RJE workstation by issuing the JES2 $I (interrupt) command.

Pooling RJE workstations

Remote pooling associates several remotes with a single route code. JES2 supports logical pooling of remote devices and operator consoles in order that the RJE workstations can be used more efficiently. The following explains how installations can use remote pooling to:

- Expand RJE workstations
- Consolidate RJE workstations

JES2 does not allow and cannot resolve multiple levels of remote redirection, that is, you cannot pool a remote to a remote that is already pooled to a second remote. If you attempt to do so, purposely or in error, JES2 issues messages:

1. $HASP242 - to flag the error
2. $HASP238 - to indicate the specific remotes that you defined improperly
3. $HASP441 - to prompt you to terminate JES2 initialization processing and correct the improperly coded RMT(nnnn) ROUTECDE= initialization statements.

Expanding RJE workstations

Remote pooling can benefit an installation with an expanding workload by balancing that workload across multiple RJE workstations.

Consider an insurance company office that contains a single printer that receives all the workload. Because of an increasing workload, the company decides to purchase two new RJE workstations (71 and 72), each containing its own printer. The insurance company wants to add these RJE workstations to its installation quickly, without rewriting all their input JCL applications. The solution is remote pooling as shown in the following example.

```
RMT(45)     RMT(71)  ROUTECDE=45     RMT(72)  ROUTECDE=45
```

Three RJE workstations are pooled by adding the ROUTECDE= parameter on the new workstation RMT definitions, thus allowing each workstation to send output to RMT45. Output routed to route code 'R45' can be controlled and printed at any of the three workstations (45, 71, or 72).

Using Remote Pooling to Accommodate an Expanding Workload

Consolidating RJE workstations

Remote pooling can benefit an installation by allowing the elimination of remote workstations by routing the workload from several remotes to a single remote. In the following example RMT definition, two departments of an organization are merging into one.

```
RMT(15)     RMT(24)  ROUTECDE=15
```

RJE workstation 24 has been removed, so all the work for both departments has been redirected to the printer at RJE workstation 15. By adding the route code 15 to RJE workstation 24, the installation can avoid changing JCL. As the example illustrates, the RMT(24) initialization statement remains in the initialization stream, even though RJE workstation 24 no longer exists.

Using Remote Pooling to Consolidate Resources

When previously existing JCL with /*ROUTE PRINT RMT24 or /*ROUTE PUNCH RMT24 is submitted, JES2 processing treats the R24 as R15. Similarly, when operators specify the route code R24 in any command, JES2 treats the R24 as R15. Accordingly, before pooling the remotes, systems programmers at this installation must ensure that:

- All jobs have completed execution processing so that output groups have been created.
- All output is routed to the new pooled destination by entering the command.

  ```
  $R ALL,R=R24,D=R15
  ```

  Note that this command will not work when the remotes have been pooled.

If you use a $T RMT(nnnn) command to modify the ROUTECDE= parameter on RJE workstation 24, you must have previously completed the following procedures:

1. Change all DESTID statements that had been equated with RJE workstation 24 to RJE workstation 15.

   By making these changes through the $T DESTID(jxxxxxxx) command and the DESTID(jxxxxxxx) initialization statement, you can avoid having to alter JCL.
2. Change the Routecde= specification on all remote devices at your installation to the new RJE remote routing that matches the Routecde= parameter on the RMT statement. (If you want different routing for individual remote devices than that required for the RJE workstation, you can provide Routecde= parameters that need not match.)
3. Use the $R command to redirect output and the $DO command to display the jobs that have gone through execution (with output groups created).

Using CONDEST= to specify the RJE operator console

Responses to operator commands are made to the RJE workstation that enters the command unless the response is spooled (message spooling). If one of the RJE workstations does not have a console, all responses can be directed to the RJE workstation with a console through the CONDEST= parameter on the RMT(nnnn) initialization statement, the $ADD RMT(nnnn) command, or the $T RMT(nnnn) command. The following example shows initialization data set specifications if the console is at RJE workstation 45.

```
RMT(71)   CONDEST=45
```

Note: On a $ADD RMT(nnnn) command or a $T RMT(nnnn) command, you cannot specify a CONDEST= value higher than the RMTNUM= value on the TPDEF initialization statement. If you specify a higher CONDEST= value, JES2 responds with an error message.

Specifying CONDEST= also allows the operator at either remote workstation to control both devices and jobs whose output is routed back to either RJE workstation. However, the remote that receives the responses to the commands is the remote specified in the CONDEST= parameter.

If the RJE workstation has pooled remote printers, JES2 routes spooled responses to either remote printer.

Remote message spooling

When JES2 cannot send a message directly to a remote console device, the member writes the message to spool for later printing by a remote printer. JES2 might not be able to send a message to a remote console because:

- A remote console is not defined for that RJE workstation

  [CONS=NO on the RMT(nnnn) initialization statement, $ADD RMT(nnnn) command, or the $T RMT(nnnn) command.]
- The number of messages already queued for an RJE workstation exceeds the RMTMSG= parameter on the TPDEF initialization statement.
- The RJE workstation is not currently signed on.

When JES2 spools remote messages, it creates a special **job** to represent the spooled remote message data set. This **job** is created as a started task with a job name equal to the remote name. Therefore, this data set can be displayed or purged by operator commands.

The following limitations apply to operator commands entered against remote messages:

- The $R command cannot be used to route remote messages.
- The $DO command cannot be used to list the status of remote messages.
- A $C or $P command will be rejected if the remote message data set is currently being processed (that is, if messages are being queued or printed).

If additional messages are added to an existing spooled message data set, the same remote message job is used, and the messages are added to the end of the data set.

Note: Remote message jobs are automatically moved from a spool volume affected by the dynamic spool deletion commands, $P SPL and $Z SPL.

Chapter 7. Providing security for JES2

Security in a data processing environment involves controlling and auditing access to resources that are important to your installation. These resources include:

- JES2-owned data sets (spool, checkpoint, and module libraries)
- Input (nodes, RJE workstations, readers, internal readers, and offload devices)
- Job names
- Data sets residing on spool (SYSIN/SYSOUT)
- Output devices (nodes, printers, punches, RJE workstations, and offload devices)
- Command input

JES2 provides a basic level of security for some of these resources through initialization statements. Additionally, JES2 provides information to the security authorization facility (SAF) at various points during its processing. This interface to SAF allows you to use RACF to provide any additional control and auditing your installation might require. When SAF returns control to JES2, JES2 enforces any security decisions SAF returns.

You and your security administrator should plan your security as you would any additional facility you would add to your JES2 system. The plan should address questions such as the following:

- What resources must I protect?
- Should I restrict jobs and users from certain information depending on other criteria (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?

Minimally, use initialization statements to control the work that input devices can input to your JES2 system. JES2 exits can add another level of security to your JES2 system. However, you can implement a more complete security policy by using the SAF interface to RACF.

Each additional resource you protect could affect the overall performance of your JES2 system. Therefore you should discuss the resources you are protecting, the frequency of their use, and your performance goals with your security administrator to determine the best overall plan for your installation. Additional information about planning security appears in the *z/OS Security Server RACF Security Administrator's Guide*.

MVS Security Authorization Facility (SAF)

During its processing, JES2 will pass information to the security authorization facility (SAF) to perform password validation, to request authority to access a resource, to build or obtain a security token for a resource, or to determine security information in a certain environment. When SAF receives the request, it determines if RACF is active. If RACF is active, SAF passes the request to RACF; if RACF is not active, SAF returns immediately to JES2 with a response indicating that SAF could not validate the request and any existing JES2 security processing controls the resource. "Using JES2 to provide security" on page 380 describes the existing JES2 security processing. When SAF indicates a decision on a security request, JES2 bypasses its own security processing.

In your exit routines, you can use JES2 macros to place additional calls to SAF. The exit routines can do security processing. The JES2 exits available are:

- Exit 36 (Pre-security Authorization Call)
- Exit 37 (Post-Security Authorization Call)
- Exit 38 (TSO/E Receive Data Set Disposition)
- Exit 39 and 55 (NJE SYSOUT Data Set Disposition)

See *z/OS JES2 Installation Exits* for information about the security exits, macros, and invoking SAF in your exit routines.

Using RACF to provide security

The following sections discuss securing your system with RACF

JES2 access to resources

JES2 uses many resources. You must ensure that JES2 has enough authority to access every resource that it requires. **Your RACF support personnel must define JES2 in the RACF started procedures table with the trusted attribute so JES2 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.

Securing resources

You or your RACF administrator can secure your JES2 resources by creating RACF resource profiles. Each profile (or generic profile) contains:

- The name that identifies a resource or group of resources
- The class of the resource
- The availability of the resource to all users
- A list of user IDs or group IDs that can access the resource and their authorization level, if needed
- An optional security label for the resource
- Other security-related information.

The profile name identifies a resource or set of resources to RACF. The name that identifies a resource to JES2 (defined in the initialization data set) is the basis of the profile name your RACF administrator uses to define the RACF profile.

Your security administrator defines different types of resources (printers, nodes, and SYSOUT, for example) to different RACF classes. Table 70 shows the JES2 resource type and the class(es) your RACF administrator can use to define the resource.

Table 70. JES2 resources and associated RACF classes

JES2 resource	RACF profile name format	RACF classes	Class purpose
Commands from Network Job Entry (NJE) Nodes	NJE.nodename jesname.command[.qualifier] node.RUSER.userid	FACILITY OPERCMDS NODES	Allows a node to issue commands to your system
Commands from RJE Workstations	jesname.command[.qualifier]	OPERCMDS and FACILITY	Restrict commands to authorized users
Data sets JES2 uses • Initialization data set • Spool data sets • Checkpoint data sets • Procedure libraries • Module libraries • Parameter libraries • Spool offload data sets	'data set name'	DATASET (always active)	Prevents unauthorized access to data sets
Data Sets Residing on Spool • SYSIN • SYSOUT • JESNEWS • Trace data sets • SYSLOG	localnodeid.userid.jobname.jobid.dsidentifier.name localnodeid.userid.jobname.jobid.dsidentifier.name localnodeid.jesid.$JESNEWS.STCtaskid.Dnewslvl.JESNEWS localnodeid.jesname.$TRCLOG.taskid.dsidentifier.JESTRACE localnodeid.+MASTER+.SYSLOG.jobid.dsidentifier.?	JESPOOL	Restrict access to data on spool to authorized users
Input Sources • Readers • Internal Readers (INTRDR) • STCINRDR • TSUINRDR • Remote Job Entry (RJE) Workstations • Network Job Entry (NJE) Nodes[1] • Spool Offload Receivers **Note:** TSUINRDR and TSOINRDR are used interchangeably.	• RDRnn as defined on the RDR(nn) initialization statement • INTRDR • STCINRDR • TSUINRDR • OFFn.SR and OFFn.JR as defined in the initialization stream **Note:** TSUINRDR and TSOINRDR are used interchangeably.	JESINPUT (see note)	Restricts users submitting specific jobs to specific devices
Job submission	SUBMIT.localnodeid.jobname.userid	JESJOBS (see note)	Controls which jobnames and user IDs users can use when submitting jobs. Can also control which users can submit any jobs.
Job modification and cancellation	SUBMIT.localnodeid.jobname.userid HOLD.nodename.userid.jobname RELEASE.nodename.userid.jobname PURGE.nodename.userid.jobname CANCEL.nodename.userid.jobname START.nodename.userid.jobname RESTART.nodename.userid.jobname SPIN.nodename.userid.jobname MODIFY.nodename.userid.jobname REROUTE.nodename.userid.jobname	JESJOBS (see note)	Controls which jobnames and user IDs users can use when modifying or cancelling jobs.

Table 70. JES2 resources and associated RACF classes (continued)

JES2 resource	RACF profile name format	RACF classes	Class purpose
Local Commands	jesname.command[.qualifier]	OPERCMDS	Restricts commands to authorized users
Network Job Entry (NJE) Nodes	• nodeid.keyword.entity*. • NJE.ownnode.othernode SESSKEY(key)	• NODES • APPCLU	• Prevents processing of unauthorized jobs or sysout from another node • Extracts the encryption key to control NJE signons
Output Devices • Local and FSS devices • RJE devices • NJE devices • Spool Offload Transmitters	jesname.LOCAL.devicename jesname.RJE.devicename jesname.NJE.nodename	WRITER	Restricts processing of output to specific devices
Remote Job Entry (RJE) Workstations	RJE.workstation-id	FACILITY	Prevents unauthorized signon by remotes
Update JESNEWS	jesname.UPDATE.JESNEWS	OPERCMDS	Restricts ability to create, update, and delete JESNEWS.

Note: At least one profile that defines all jobs must exist in this class when this class is active or all jobs will fail.

[1] See "Authorizing networking jobs and SYSOUT (NJE)" on page 359.

When RACF is active, every user must have a RACF **user profile**, any class in use must be active, and all resources you want to protect must have a resource profile (except those in the JESSPOOL class). Before JES2 completes a request for a resource from a user, JES2 requests authorization from SAF. SAF passes the request to RACF which determines the authority based on the existing profiles. If RACF is not active or cannot determine the authorization for a resource, JES2 carries out its own security processing, if any, for that resource. The *z/OS Security Server RACF Security Administrator's Guide* has additional information about profiles and access.

SMF record summary: RACF security

The following lists the SMF record written when RACF provides security for your JES2 subsystem. For a complete list of the record contents, see *z/OS MVS System Management Facilities (SMF)*.

Table 71. SMF Records Used by JES2 with RACF Security

Record number	Action that causes the record to be written
80	• An unauthorized attempt to access the system. • Any attempt to access a RACF-protected source. • Any attempt to modify a RACF profile.

Defining and grouping your installation's support personnel

System programmers and operators (the support personnel) at your installation need to be defined to RACF. Your installation's security administrator must define any personnel who:

- Issues JES2 or MVS commands
- Updates data sets JES2 uses.

You can group your installation's support personnel into groups that are responsible for a particular area. For example, you might want to group your support personnel by shift, by functional area, or by both criteria. If you want, your security administrator can define your support personnel to RACF in groups. Therefore, you might want to consider combining users into groups and assigning a unique identifier to each group.

To group your installation's support personnel you should:

- List the userids (or names of proposed userids if not already defined) 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

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

Specifying access authority

RACF allows you to permit access to resources with:

- Universal access authority (UACC)
- Access lists

Universal access authority

The universal access authority assigned to each resource defines the access that all users have to a resource unless the users are in the access list. The meaning of the universal access authorities 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.

EXECUTE
Specifies a user can load and execute a program from a private program library but cannot read or copy the program.

READ Specifies the user can read from the resource.

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

ALTER
Specifies a user the maximum access RACF allows to the resource and permits a user to update a discrete profile.

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 or group in an access list associated with a resource. For example, you can assign a UACC value of READ to WORKLOAD and assign a 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* information about defining access authority.

Security labels

You can add another layer of control for your resources through security labels (through the RACF SECLABEL class). Security labels allow an installation to restrict personnel who can view data from copying that data to a less secure device or data set. Also, users with the authority to view a group of data could be restricted from viewing more vital pieces within that group through the use of security labels and profiles.

When the SECLABEL class is active, each user must specify the security label for a session or job (or use the default SECLABEL established in the USER profile) to be able to access data that has a security label. (A user can have more than one security label associated with a userid but can only specify one for a session or job.) RACF determines whether the user's security label for the session is equal to or greater than the security label of the data the user is accessing before verifying if the user has authority to view the data. The highest security label in the system is **SYSHIGH**, the lowest is **SYSLOW**. Your installation should define as many levels of security labels between SYSHIGH and SYSLOW as needed to implement your installation's security policy.

If you receive SYSOUT data sets that contain an unknown or blank security label band your node has an active SECLABEL class, JES2 assigns this SYSOUT the SECLABEL RACSLUNK (SECLABEL unknown). Because your installation has not predefined this security label, no one can access the SYSOUT.

You can make the SYSOUT accessible by:

- Defining RACSLUNK to RACF as a valid security label and authorizing users.
- Using the NODES class translation facility to translate RACSLUNK to a predefined security label.

Your installation's security policy could require security labels for all users and resources. Your security administrator enforces this policy by activating various RACF classes and options. For more information on security labels, see *z/OS Security Server RACF Security Administrator's Guide*.

Multilevel security support

Using RACF, JES2 can limit job selection based on "security label by system" and thus providing an additional layer of system security. Although the RACF database is shared by all MAS members and most profiles apply to all systems, you can specify a subset of members to which security labels apply. A RACF SETROPT

command option (SECLBYSYSTEM) controls whether security labels are active on all systems or only those you specify. Security labels control MAC (mandatory access control).

JES2 extracts security labels associated with JES2 devices and NJE nodes (associated with RACF profiles used by a device/NJE node) for use by SDSF. JES2 extracts these security labels at JES2 initialization and when devices are started. Therefore, only update security labels associated with a device when the device is not active. If the device is active, be certain to drain and re-started it to allow it to recognize and use the new security label.

The JES2 address space is a server address spaces that performs work for users running with different security labels. In particular, during conversion processing, JES2 anchors ACEEs with security labels matching the job being converted to the converter subtask. If multilevel security options are active, then the user ID associated with the JES2 started task should have a default security label of SYSMULTI. Failure to do so will result in jobs failing to convert with the following messages:

```
ICH408I USER(userid) GROUP(group)  NAME(programmername)
LOGON/JOB INITIATION - USER SECLABEL NOT COMPATIBLE WITH SERVER
$HASP313 jobname JES2 IS UNABLE TO CREATE THE SECURITY ENVIRONMENT
SECURITY PRODUCT RETURN CODE = 00000038 REASON CODE = 00000014
```

See “Using RACF multi-level security” on page 66for a more complete discussion of security label by system.

Controlling access to data sets JES2 uses

The JES2 spool and checkpoint data sets are critical for proper operation of your JES2 system. It is critical that JES2 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 JES2 uses. Make sure profiles exist for any data sets you might use for checkpoint reconfiguration.

Your RACF administrator (or you) would define profiles for the data sets involved to the DATASET class and permit the users that must have access to the data sets to the DATASET profile just created. Define **discrete profiles** for all your spool and checkpoint data sets. This protects them from unauthorized access even when not in use.

See *z/OS Security Server RACF Command Language Reference* for more information about discrete profiles.

Controlling input to your system

There are many ways a job can enter the system. RACF can:

- Validate the security information for job and data entering the system from any reader, internal reader (INTRDR, STCINRDR, and TSUINRDR), RJE workstation, NJE node, and spool offload receiver. If RACF is available, SAF propagates missing security information to the submitted job (or data) for validation.

  Note: TSUINRDR and TSOINRDR are used interchangeably.
- Limit the users authorized to submit jobs to the system.
- Restrict the job names a user can submit or cancel.

- Restrict the userids a user can use to submit or cancel jobs.
- Authorize a userid to submit or cancel any jobs.

How RACF validates users

Use RACF to prevent unauthorized users from accessing your system or running batch jobs. Your RACF security administrator can indicate to RACF that any batch job requires a valid userid, password, and, optionally, security label on the JOB statement or propagated security information. Your administrator can also define what security information must be present for any data or user to enter your system.

JES2 issues a request to RACF to validate a job's security label, userid, groupid, and password when a job enters the system from any source. This ensures that JES2 does not process any unauthorized work.

As a result, the submitter becomes aware of any authorization errors almost immediately (before JCL conversion) rather than waiting until JES2 selects the job for execution. For example, during a TSO/E session a user issues the TSO/E SUBMIT command to enter a job into the system, if the job fails an authorization check, JES2 notifies the user, queues the job for the JES2 output or purge processing and flushes the job from the system.

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 (TSO/E, executing batch job or INTRDR) 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 workstation 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, JES2 propagates the submitter's security information when the information is omitted. For example, you can have users omit their password from their job statements 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 userids” on page 363 for information about the default userid.)

“How JES2 propagates security information” on page 365 describes what security information RACF propagates for NJE jobs.

Controlling job submission

Using RACF, you can control the users who can submit jobs. By defining the jobname in the JESJOBS class, and a list of users authorized to submit the job to RACF, SAF determines if the user that is submitting the job has the authority to do so. The format of the RACF profile name for job submission is:

```
SUBMIT.nodename.jobname.userid
```

where:

SUBMIT
Controls which users can submit jobs.

nodename
The name of the node on which the job submission will occur.

jobname
The job name you are securing.

userid The user ID associated with the job that you are securing.

You do not need a profile for a user to submit jobs that have their own user ID on the JOB statement. When you grant a user the ability to submit a job, you should permit that user ID to the profile with an access level of READ.

The UACC and access lists associated with these profiles also allow you to control which user IDs a user can use on JOB statements or whether the user can submit any job. Refer to *z/OS Security Server RACF Security Administrator's Guide* for additional information on controlling job submission.

Controlling job modification and cancellation

Using RACF, you can identify users who have job control authority. By defining the *jobname* in the JESJOBS class, and a list of users authorized to control the job to RACF, SAF determines if the user has the authority to affect a job. The format of the RACF profile name for jobs is:

```
HOLD.nodename.userid.jobname
RELEASE.nodename.userid.jobname
PURGE.nodename.userid.jobname
CANCEL.nodename.userid.jobname
START.nodename.userid.jobname
RESTART.nodename.userid.jobname
SPIN.nodename.userid.jobname
MODIFY.nodename.userid.jobname
REROUTE.nodename.userid.jobname
```

where:

nodename
The name of the node on which the job modification or cancellation will occur.

jobname
The job name that you are securing.

userid The user ID associated with the job that you are securing.

The following access requirements control user control over each function:

```
HOLD.nodename.userid.jobname Update
RELEASE.nodename.userid.jobname Update
PURGE.nodename.userid.jobname Alter
CANCEL.nodename.userid.jobname Alter
START.nodename.userid.jobname Control
```

```
RESTART.nodename.userid.jobname Control
SPIN.nodename.userid.jobname Control
MODIFY.nodename.userid.jobname Update
REROUTE.nodename.userid.jobname Update
```

The UACC and access lists associated with these profiles also allow you to control which user IDs a user can use on JOB statements or whether the user can modify or cancel any jobs. See *z/OS Security Server RACF Security Administrator's Guide* for additional information on controlling job modification and cancellation.

Authorizing users to submit jobs for other users

Your RACF administrator can allow users to submit jobs on behalf of other users without compromising the password of the user whose jobs are being submitted. This feature is particularly useful when a person is assuming a production workload for someone going on vacation or a leave of absence.

Your RACF administrator must define the userid of the person who will be submitting the jobs in the RACF SURROGAT class. When this is done, the user just submits the job with the original userid and without a password on the JOB card. RACF then validates the job(s) as if the original user had submitted it and will use that userid's password and default security information when verifying the jobs authority for resources.

See *z/OS Security Server RACF Security Administrator's Guide* for information about the SURROGAT class.

Authorizing the use of input sources

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

To authorize the submission of work from specific input sources, ask your security administrator to activate the RACF JESINPUT class and define a profile for each input source. Provide your security administrator with the following information:

- The name of the device. Table 72 lists the names to use for each type of input device.

Table 72. Device names to use for RACF profiles

Input Source	Device Name
NJE (Jobs and SYSOUT)	adjacent node name (specified on NODE(nnnn) statement)
Internal readers (other than started tasks or TSO/E logon)	INTRDR
Internal readers for started tasks (STCs)	STCINRDR
Internal readers for TSO/E logon (TSUs)	TSUINRDR
Device Readers	Use same name as specified on the RDR(nn) initialization statement.
RJE workstations	Rnnnnn.RDm where devname is the name specified on the R(nnnnn).RD(m) initialization statement.

  Note: TSUINRDR and TSOINRDR are used interchangeably.

- The userid or groupid of the users you want to authorize or restrict.

- The universal access authority to associate with each device. Valid access values for input devices are:

  NONE
  Specifies that only those users explicitly permitted through the access list can use the input device.

  READ Specifies that only those users assigned an access authority of read, update, or control can use the input device.

A user must have READ access to a device (either through the UACC or explicitly in an access list) in order to use a input device. If the device has a UACC of READ, you can restrict users from a device by listing them in an access list and specifying NONE for their access.

Note: By default, JES2 accepts all work from an input source when the RACF JESINPUT class is inactive.

Authorizing work from remote job entry (RJE) workstations

Operators at RJE workstations must issue either a /*SIGNON (if BSC) or a LOGON (if SNA) statement before JES2 establishes a session with the remote. In RACF, the administrator defines the workstation ID used in the /*SIGNON statement in a profile name in the FACILITY class. When properly defined, this allows the remote access to your system. For example, the profile name for a workstation with the ID RMT1 is:

```
RJE.RMT1
```

The administrator must also define the remote workstation as a valid RACF userid for RACF to control the signon process and any commands from the remote.

JES2 passes the workstation ID and the password specified to RACF for userid validation during signon processing. RACF requires the password be present during signon processing for all remote workstations it controls. If RACF is active, RACF performs the userid validation for the remote. If RACF is not active, JES2 performs the userid validation for the remote using the line and remote password specified in the JES2 initialization stream or set by the system operator. The UACC of the remote should be READ.

The password for an RJE workstation must be changed the first time the workstation issues a logon or signon. Otherwise, RACF will reject the access attempt.

Authorizing networking jobs and SYSOUT (NJE)

This section contains information about how to 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.
- The extent of security validation performed at your node.
- Jobs and data destined for other nodes in a network.

JES2 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 at your node.

To provide security for Networking Job Entry (NJE), have your security administrator activate the RACF NODES class (for inbound work) or the RACF

WRITER class (for outbound work), and define the profiles needed to enforce your installation's networking security policy. As with other RACF-protected resources, you must provide your security administrator with specific information needed to define profiles.

Before you can provide this information, you must first 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 or groups of users you will allow to submit work, and the security labels that are valid for processing on your node.

If a WRITER class profile does not release SYSOUT to a device, JES2 does not issue a message. To issue messages, see Exit 36 in *z/OS JES2 Installation Exits*.

Authorizing inbound work

The following topics identify what functions RACF provides and the information that you should provide to your security administrator to activate those functions. This section provides information about the following concepts and tasks:

- ***Understanding Networking Profiles*** explains how the values you select for the RACF NODES profile determines what type of work and which users you want validated.
- ***Authorizing Jobs*** explains how you can use RACF to validate inbound jobs from other nodes in a network.
- ***Authorizing SYSOUT*** explains how you can use RACF to validate inbound job output (SYSOUT) from other nodes in a network.
- ***Translating Security Information*** explains how RACF can be used to replace inbound userids, groupids, or security labels with locally-defined values.
- ***Validating Work Based on the Submitter*** explains how RACF can be used to validate inbound work using the submitter's security information instead of the owner's security information.
- ***Understanding Default Userids*** describes how JES2 handles security information for work from incompatible nodes and explains how JES2 handles security information belonging to store-and-forward work. This section also describes how you can use RACF to manipulate default security information.
- ***How JES2 Propagates Security Information*** explains where JES2 obtains missing security information for NJE work.
- ***Defining Nodes as Local Input Sources*** explains how you can use RACF to treat work from another node as if the work originated at the home node.

Understanding networking profiles

The values that you specify in networking profiles determine how RACF validates inbound work. As with other RACF profiles, this profile consists of a profile name, profile class, and a universal access authority value. All networking profiles for inbound work are associated with the RACF NODES class. The profile name is a 3-part identifier that indicates the origin of the work and the type of security information you want validated. The universal access value determines the actions that RACF performs on the inbound work.

A NODES profile name has the following format:

```
nodeid.keyword.name
```

where:

nodeid
The name of the node from which you expect inbound work.

keyword
The type of work (job or SYSOUT) entering from the node and the security attributes (userid, groupid, or security label) you want RACF to examine. The last character, J or S, indicates the type of work entering your system. Allowable values are USERJ, USERS, GROUPJ, GROUPS, SECLJ and SECLS for userid, groupid, and security label examination respectively.

name The actual userid, groupid, or security label you are controlling. These values must either be defined in your RACF database or you can define profiles that translate these values into acceptable values for your system.

For example, the following profile:

```
BERMUDA.USERJ.WAYNE
```

controls whether jobs coming from userid WAYNE at node BERMUDA can execute here. You can optionally associate a local userid with userid WAYNE through the profile. Information about associating local security information with inbound work is contained in "Translating Security Information".

You can place an asterisk in any position of a profile 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. For information about using generic profiles see, *z/OS Security Server RACF Security Administrator's Guide*.

When you choose to activate the RACF NODES class, you must provide your security administrator with the appropriate profile information to control the work entering your system. The following sections identify the appropriate values for each type of work.

Authorizing jobs

You can control which network jobs are authorized for processing at your installation based on the userid, groupid, or security label associated with the inbound job.

To authorize or restrict jobs entering your system from another node, ask your security administrator to define a NODES profile that specifies the criteria upon which jobs are accepted. Provide your security administrator with the following information:

- The node name from which you expect jobs.
- The user ID or group ID from which you expect jobs.
- The security labels that you will accept.
- The universal access value, which determines how JES2 will process the job. Table 73 on page 362 lists the universal access values you can assign and defines the validation that RACF performs.

Table 73. NODE class keywords and the UACC meaning for inbound jobs. The Type of check (keyword) column header spans the two header rows. The top UACC column header row spans the four bottom column headers, which are NONE, READ, UPDATE and CONTROL or greater. The bottom row in the table contains the Notes and spans all table columns.

Type of check (keyword)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
userid (USERJ)	Fails the job	Verifies all security information available including password validation.	If non-default valid security information exists, uses the submitter's or translated security information as the owning security information without revalidation.	Same as UPDATE, but default security information is allowed.
groupid (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 SECLABEL to that specified in ADDMEM. If ADDMEM is not specified, uses the security label received.		

Note: If no profile exists for a job when the NODES class is active or if the RACF NODES class is inactive, RACF performs only userid, groupid, and password validation without performing any translation.

Authorizing SYSOUT

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

If when running in RACF warm mode, you receive message $HASP186 indicating an access violation [to devices, external writers, or SYSOUT Application Programming Interface (SAPI) requests], and you require further diagnostic information, set DEBUG SECURITY=YES and see *z/OS JES2 Diagnosis*.

To authorize or restrict SYSOUT entering your system from another node, ask your security administrator to define a profile that identifies the criteria upon which SYSOUT is accepted. Provide your security administrator with the following information:

- The node name(s) from which you expect SYSOUT.
- The userid(s), groupid(s), and security labels from which you expect SYSOUT.
- The universal access value, which determines how RACF assigns an owner to 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. Table 74 on page 363 shows the universal access values you can assign and defines the validation that RACF performs.

Validating SYSOUT based on the submitter

JES2 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 have your security administrator 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 execute under another userid, and allow the submitting user to view the output upon its return.

To translate inbound work ownership to the submitter, specify &SUSER as the value of the ADDMEM keyword of the RACF NODES profile for the inbound SYSOUT. For example, the profile:

```
RDEFINE NODES GOTHAM.USERS.NAPIER ADDMEM(&SUSER) UACC(UPDATE)
```

causes ownership of all SYSOUT created at node GOTHAM by userid NAPIER to be assigned to the userid that submitted the creating job.

Understanding default userids

RACF assigns a default userid to all work that enter your node when:

- SYSOUT enters from a non-JES2 system, unless translated to a valid userid by a NODES class profile
- 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. Your security administrator can assign a different userid using the RACF SETROPTS command. For example, if you want to process any of these jobs locally, you can define a single userid to which all of these undefined userids are translated. You can then permit that translated userid to specific resources. You can not directly permit the default userid (either IBM-supplied or installation-defined) to any resources. See *z/OS Security Server RACF Security Administrator's Guide* about defining default userids for NJE work that meet the criteria.

Table 74. NODES Class Keywords, UACC, and SYSOUT Ownership when Execution Node is Not Defined to &RACLNDE - User ID and node that created SYSOUT. The Type of check (keyword) column header spans the two header rows. The UACC column header row spans the four column subheaders, which are NONE, READ, UPDATE and CONTROL or greater. The bottom row in the table contains the Notes and spans all table columns.

Type of check (keyword)	UACC			
	NONE	**READ**	**UPDATE**	**CONTROL or greater**
userid (USERS)	**Check of User ID and Node that Created SYSOUT**			
	Purges the output.	If the translation value from ADDMEM specifies &SUSER, check submitting userid and node. Otherwise, assigns ownership of the output to ????????.	If default or no security information is available, processing is the same as a UACC of READ. If security information is valid, assigns the translation value from ADDMEM to the output. When ADDMEM is not specified, ownership is assigned to the userid that created the output.	Processing is similar to UACC(UPDATE) except RACF translates any available information. This allows RACF to assign local user IDs to output from pre-SP3.1.3 nodes.

Note:

1. If the node ID 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. Use the 'node.RUSER.userid' NODES class profile to provide command authority at your installation.

Table 75. NODES Class Keywords, UACC, and SYSOUT Ownership when Execution Node is Not Defined to &RACLNDE - Submitting user ID and node. The Type of check (keyword) column header spans the two header rows. The top UACC column header row spans the four bottom column headers, which are NONE, READ, UPDATE and CONTROL or greater. The bottom row in the table contains the Notes and spans all table columns.

Type of check (keyword)	UACC			
	NONE	**READ**	**UPDATE**	**CONTROL or greater**
userid (USERS)	**Check of Submitting Userid and Node (only when &SUSER is specified for ADDMEM in execution node profile)**			
	Assigns ownership of the output to ????????.	Assigns ownership of the output to ????????.	Assigns ownership of the output to ????????.	Assigns the translation value from ADDMEM to the output, if available. If translation value is &SUSER, assigns the submitter userid to the output. Otherwise, assigns ownership of the output to ????????.
groupid (GROUPS)	Purges the output	Translates GROUPID to that specified in ADDMEM. If ADDMEM is not specified, uses the groupid received.		
security label (SECLS)	Purges the output	Translates SECLABEL to that specified in ADDMEM. If ADDMEM is not specified, uses the security label received.		

Note:

1. When you specify &SUSER for ADDMEM and the submitting node **is** defined to &RACLNDE, ownership is assigned to the submitter.
2. 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.
3. Use the 'node.RUSER.userid' NODES class profile to provide command authority at your installation.

Translating security information

It might not be practical to maintain a RACF database that contains all of the userids, groupids, and security labels that might enter your system through the network. However, you can translate inbound userids, groupids, or security labels into predefined values known to your system.

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, the security administrator would specify the following:

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

If you do not define profiles that translate inbound userids, then those userids must be defined in your RACF database or the work will not pass RACF validation.

Authorizing inbound SYSOUT by translating the default userID

Rather than translating the default userid to a universal userid, you can use the submitter's userid to translate the default to a unique userid that is valid for your system. To translate the default userid to a unique userid, ask your security administrator to:

1. Define a NODES profile that assigns the submitter's userid (&SUSER) to the inbound SYSOUT.
2. Define a NODES profile that translates the submitting userid (&SUSER) to a valid userid known to your node.

See *z/OS Security Server RACF Security Administrator's Guide* for information about translating default userids.

How JES2 propagates security information

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
- Group
- Security label

If JES2 receives a request from an application or subsystem, JES2 propagates any security information provided by the application or subsystem to SAF.

Defining nodes as local SYSOUT sources

Your security administrator can have RACF treat SYSOUT received from other nodes the same as SYSOUT created by jobs running on local processors. To do this, the security administrator uses the &RACLNDE profile in the RACFVARS class to identify the nodes that you want RACF to consider as local SYSOUT sources. 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. Defining remote nodes as local SYSOUT sources is useful if your node receives and processes SYSOUT as a back-up to another node.

Note: All local nodes, including the one on which the RACF database resides, **must** be defined in the &RACLNDE profile in RACFVARS class.

Authorizing outbound work

You can use RACF to control whether a specific node can receive work (jobs or SYSOUT).

For NJE, RACF ensures that the originator of outbound work is authorized to send the work to the target node. To activate protection for a node, ask your security administrator to activate the RACF WRITER class and define a profile for each node that you want to protect. You should provide your security administrator with the following information needed to define profiles:

- The name of your JES2 system.
- The list of users to be authorized or restricted from sending work to a specific node.
- The node name as specified in the JES2 initialization stream (where the specified node name is the ultimate destination of the output).
- The security label if you want to limit which security labels can appear on output that is sent to a particular destination.

RACF will check all data leaving your system for another node except data destined for another node your system received from a different node. This data

RACF considers already validated and RACF protects the data with the default userid (????????) and the highest (SYSHIGH) security label possible while it resides on your spool. See “Authorizing the use of operator commands” on page 371 for details on authorizing commands.

For information on sending output through the network see “Controlling where output can be processed” on page 370.

Controlling access to data that resides on SPOOL

You can use RACF to provide security for your spool data including:

- SYSIN and SYSOUT data sets
- JESNEWS
- Trace data sets
- SYSLOG
- Spool offload data

The following sections identify what spool resources you can protect, and what information you must provide to your security administrator to control those resources.

Controlling access to SYSIN and SYSOUT

During job execution, JES2 creates data sets for a job's input (SYSIN) and output (SYSOUT). These data sets have profile names in the form:

```
localnodeid.userid.jobname.jobid.dsidentifier.name
```

where:

localnodeid
The name of the node on which the SYSIN or SYSOUT data set currently resides. The localnodeid appears in the JES2 job log of every job.

userid The userid associated with the job. This is the userid RACF uses for validation purposes when the job runs.

jobname
The name that appears in the name field of the JOB statement.

jobid The job number JES2 assigned to the job. The jobid appears in notification messages and the JES2 job log of every job.

dsidentifier
The unique 8-byte alphanumeric character identifier JES2 assigned to this data set. This identifier is an encoded printable representation of the internal data set number (data set key) of the SPOOL data set. The internal data set number and data set name (which includes the dsidentifier) are available from the Extended Status SSI (SSI 80), which is documented in *z/OS MVS Using the Subsystem Interface*.

Note: The first 10 million data sets that are created by a job can be sorted chronologically by data set name. The same is true for data sets created after the first 10 million data sets. When the two subsets are sorted together, however, the resulting sequence will not be in the order of data set creation.

name The name of the data set specified in the DSN= parameter of the DD statement. This name cannot be JESYSMSG, JESJCLIN, JESJCL, or JESMSGLG 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, JES2 uses a question mark (?).

For example, if user MYUSER submits a job named MYJOB to run on NODEA, and JES2 assigns a jobid 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
```

These data sets might exist after the job completes execution. Your RACF administrator can control access to these data sets by activating the JESSPOOL RACF class. If RACF is not active, JES2 always allows a job access to a SYSIN/SYSOUT data set the job creates. When RACF is active, RACF allows the userid (MYUSER in the preceding example) that creates a SYSIN/SYSOUT data set access to the data set, even though a profile might not exist for the data set. If any other users require access to a spool data set, the owner of the data set would have to define the data set profile (or a generic profile for the data set) and give access (through the RACF PERMIT command) to the users requiring the data.

Note: The data set name JES2 uses in its control blocks is the last 5 qualifiers used to build the RACF profile name.

SPOOL browse access to SYSIN and SYSOUT

Applications can use the SPOOL Browse interface to access data sets managed by JES2. SPOOL browse uses both the standard form of the JESSPOOL class profiles and a modified form for certain special system data sets. These special profiles are of the form:

```
localnodeid.userid.jobname.jobid.jes_dsname
```

where:

localnodeid
The name of the node on which the SYSIN or SYSOUT data set currently resides. The localnodeid appears in the JES2 job log of every job.

userid The userid associated with the job. This is the userid RACF uses for validation purposes when the job runs.

jobname
The name that appears in the name field of the JOB statement.

jobid The job number JES2 assigned to the job. The jobid appears in notification messages and the JES2 job log of every job.

jes_dsname
One of the following fixed names:

- JCL - This represents the jobs input JCL (with all SYSIN data sets)
- JESJCL - The JCL images data set as created by the conversion process
- JESMSGLG - The JES2 job log data set
- JESYSMSG - The MVS SYSTEM messages data set

To allocate the SYSLOG data set, the SPOOL browse interface defines a new data set name:

```
sysname.SYSLOG.SYSTEM
```

In the data set name, *sysname* is the name of the system whose SYSLOG the application examines. JES2 logically concatenates all the SYSLOG data sets (even if there are multiple jobs) for the specified system in chronological order. The application then reads the SYSLOG data sets by using the SPOOL browse.

In addition, a new form of relative byte address (RBA) can be used in the POINT request for the SYSLOG data set. This RBA format allows applications to position a SYSOUT data set (SYSLOG in this case) in time order. This is the format of the new RBA:

```
FxCC CCCC
```

RBAs that start with *FF* in the first byte indicate that this is a special case value (not the normal JES2 MTTR because an *M* of *FF* is not valid for SPOOL in JES2). The second byte *x* indicates that *CC CCCC* is the first 6 bytes of a STCKE value that indicates the time (and date) to which the application positions the log. The byte *x* can be one of the following values:

0 The first occurrence of a message that was returned at or after the specified time is returned.

1 The search starts at the current record and proceeds forward.

2 The search starts at the current record and proceeds backward.

Because messages in the log might not be in chronological order, the RBA allows the application to do repeat finds and search for messages that have a particular timestamp.

Protecting JESNEWS

JESNEWS is an informational data set that prints after the header separator page of a job, if required. Generally, this data set is of interest to all job submitters, so it is important that all users can read this information (UACC of READ).

The resource name for JESNEWS has the form:

```
nodeid.jesid.$JESNEWS.STCtaskid.Dnewslvl.JESNEWS
```

where:

nodeid
Is the name of the node that created the JESNEWS data set.

jesid Is the userid associated with your JES2 system.

taskid Is the name of the task that created the JESNEWS data set.

newslvl
Is the level of this copy of JESNEWS. The value of newslvl varies from 0000101 to 0065535.

The RACF profile name for JESNEWS on NODEB created by JOB01 is:

```
NODEB.SYSTEM.$JESNEWS.JOB01.D0000101.JESNEWS
```

To prevent unauthorized updating of JESNEWS, your RACF administrator must define another profile. This profile is in the RACF OPERCMDS class and any userids authorized to update JESNEWS must have CONTROL access to the profile. The JESNEWS update profile has the following form:

```
jesname.UPDATE.JESNEWS
```

If RACF is not active, JES2 requests authorization to update JESNEWS from the operator.

Note: If RACF and the SECLABEL class are active, RACF assigns the security label of the last job that updated JESNEWS to the JESNEWS profile. This could cause jobs with lower security labels than the updating job to miss important information and RACF will record security violations for jobs accessing JESNEWS that did not previously occur. To make JESNEWS accessible to all users, the job that creates it should have a security label of SYSLOW and the data set profile should have a UACC of READ. If the security label is greater than SYSLOW, JESNEWS will not print in the output of any jobs submitted with a lower security label.

Protecting trace data sets

Trace data sets could contain information that compromises your installation's security (userids and passwords, for example). You protect these data sets in the RACF JESSPOOL class by defining a profile for the data sets and permitting only those userids that need access to the data sets. The profile for all trace data sets on NODE1 could be specified as:

```
NODE1.JES2.*.*.*.JESTRACE
```

The security administrator can then grant access to data sets that fit this profile to userids or groupids for the system support group(s). If you also use security labels for controlling your data, the trace data set(s) should have the highest (SYSHIGH) security label in your installation.

Protecting SYSLOG

Your security policy might require that you protect SYSLOG because it is the record of your systems daily activities.

To control SYSLOG, your security administrator defines a RACF JESSPOOL profile for the data set, an appropriate universal access, and then grants access to the userids or groupids that need a different access. NODEB's SYSLOG profile could look like this:

```
NODEB.+MASTER+.*.*.*.?
```

If you also use security labels for controlling your data, note that the SYSLOG data set(s) also has the highest (SYSHIGH) security label in your installation, a label it inherited from the master address space when created.

Protecting SPOOL offload data sets

You should protect offloaded information by defining the offload data set to RACF with no (NONE) universal access authority. If you have security labels active, you should assign the offload data a security label of SYSHIGH, to prevent unauthorized access.

Offload: When you offload data from the spool to another device, JES2 copies the security information for the data to the offload job and data set headers. No validation is made of the security information written to the offload data set. JES2 calls RACF to ensure the operator starting the offload operation has sufficient authority to issue the command to start the offload.

During the offload process, JES2 calls RACF (using the WRITER class) to ensure the owner of the SYSOUT data set has access to the offload device by checking the security information associated with the data against the device's profile in the RACF WRITER class. The offload device profile for offload SYSOUT transmitter 1 would be:

```
jesx.LOCAL.OFF1.ST
```

During spool offload, jobs are not checked for access to the device.

Reload: When JES2 reloads the information from an offload data set, it performs any security validation necessary (similar to reading a job into the system or receiving a network SYSOUT data set) before writing the data to spool by checking the RACF JESJOBS class for reloaded jobs and the RACF NODES class. When RACF performs the NODES class check, if the node associated with the data is in &RACLNDE, RACF accepts the data.

The following profiles for the JESINPUT class apply to spool reload:

```
OFFn.JR     for jobs
OFFn.SR     for SYSOUT
```

As with offload, JES2 calls RACF to ensure the operator starting the reload operation has sufficient authority to issue the commands.

Attention: When using security labels, data offloaded from a JES2 SP3.1.3 or later member reloaded to a pre-SP3.1.3 member could cause a security exposure because the pre-SP3.1.3 member cannot validate security labels.

When reloading a data set that was offloaded on this node, the name of the node must be defined in &RACLNDE.

Controlling where output can be processed

JES2 normally processes output on any device that matches the output's selection criteria. You might, however, want to control the specific devices certain users can access. For example, you might want the users who process your payroll to be the only users of a particular printer.

By defining the device to RACF (as a profile in RACF class WRITER), you can limit access to that device to a user or group of users. When a defined device selects a piece of output for processing, JES2 passes an authorization request to SAF; access to the device depends on the response JES2 receives. RACF profile names for devices have the following formats:

- For local devices,

  ```
  jesname.LOCAL.devicename
  ```
- For RJE devices,

  ```
  jesname.RJE.devicename
  ```

  Note: RJE printers and punches are specified as Rnnnnn.PRm and Rnnnnn.PUm.
- For data destined to a node,

  ```
  jesname.NJE.nodename
  ```

where:

jesname
The JES2 system that defines the node

devicename
The name of the JES2 initialization statement that defines this device.

nodename
The name of the NODE(nnnn) statement that defines the node.

Your security administrator would then grant access to the devices to the userids and groupids you have identified.

Also, if security label verification is active, RACF ensures that the device can write this data depending on the security label specified. If your RACF options require the security label of the device to be equal to or greater than the security label of the output, the device will not be able to process the output.

If a device cannot select output because the device does not have sufficient access to that output, the output remains on the output queue until you:

- Change the security attributes of the device; you must then stop and restart that device for the security attributes to take affect.
- Authorize the userid associated with the output to the device.
- Change the work selection criteria of the output so that a device with the appropriate security attributes can select the output.

RACF also allows your installation to print security labels on printed output that PSF-managed printers process. See *z/OS Security Server RACF Security Administrator's Guide* for information about printing security labels on job output using RACF. Also, see *z/OS MVS JCL User's Guide* for information about using the OUTPUT JCL statement to print security labels.

Controlling external writers

An external writer is a started task used to process output. Because the external writer is a started task, it has a userid associated with it. You process output with an external writer by naming the writer on the DD statement (or OUTPUT JCL statement) that defines the output as follows:

```
//MYOUTPUT  DD  SYSOUT=(A,XTWTR)
```

For the writer to be able to process that output, the writer's userid must be in an access list that permits the writer's userid to access the SYSOUT data set. The minimum access required for external writers is ALTER. The writer's userid is the userid specified in the started procedure table for the writer task. Alternately, you can define external writers with the trusted attribute in the started procedures table. However, the trusted attribute might be contrary to your installation's policy and allows the external writer access to data you might not want it to process.

Also, if your policy requires security labels, the security label associated with the external writer must be equal to or greater than the security label associated with the SYSOUT.

Authorizing the use of operator commands

Your installation can require that only certain operators issue certain JES2 commands. There could also be a requirement to restrict commands coming in from RJE workstations, NJE nodes, or any other device.

To control commands, you should review the commands and your installation's security policy with your RACF administrator and determine:

- The list of commands you need to control
- The command profile names
- The appropriate the authority level for any restricted commands
- The authority level your policy requires to issue a particular command
- the list of users or groups authorized to issue each command.

The command resource names have the format:

```
jesname.command[.qualifier]
```

where:

jesname
The name of the JES2 system requesting the command validation.

command
The name of the command.

qualifier
The type of object the command specifies. (For example, JOB or SYS.)

Table 77 on page 373 contains a list of commands, a sample resource name for each command, and the minimum required authority level for each command. (If you give an operator access to a command with a lower authority than the authority in the table, RACF will not allow the operator to issue the command.) The authority levels give you a means of controlling commands that can affect your installation's processing.

Ensure that your RACF administrator activates the OPERCMDS class and defines each command profile to that class. In order to minimize the number of definitions, commands should be defined with generic profile name unless you are restricting specific commands. Supply your RACF administrator with a generic name that covers a group of commands and then group restricted commands in that group with a more specific name. For example,

```
jesx.*
```

would define all commands as a group. Then,

```
jesx.START.*
```

defines all start commands.

The first definition, jesx.*, could restrict a group of operators to only the commands that need READ authority. Then your RACF administrator would define any operators who would need access to all the start commands with CONTROL authority. The operators with CONTROL authority are able to issue all commands based on the RACF authority hierarchy shown in Table 76.

Table 76. RACF command authority hierarchy. The following assumes a specific profile does not exist that would prevent the operator from issuing a command.

Access Level	Commands Operator Can Issue
CONTROL	All commands
UPDATE	All commands specified with UPDATE or READ
READ	Only commands specified with READ

If you need to restrict commands to specific consoles, your RACF administrator should define an operator's access conditional to the console from which the operator enters the command. For more information on conditional access, see *z/OS Security Server RACF Security Administrator's Guide*.

If you expect commands to enter your system from local readers, you **must** have an operator issue the $S RDRx command rather than have JES2 start the reader. This ensures RACF has a specific userid to use when checking command authorization.

To control the commands entering from RJE workstations your security administrator must do the following:

- Define the RJE workstation profile in the RACF FACILITY class.
- Define the RJE workstation as a valid RACF userid using the remote name.
- Permit the RJE userid to the command profiles the node can issue.

The FACILITY class profile for RMT(1) has the form:

```
RJE.RMT1
```

The userid your administrator would define to RACF is RMT1.

Your administrator would use a similar process to define an NJE node in order to control commands.

- Define the node to the RACF NODES class
- Define the node to the RACF FACILITY class
- Define the node as a RACF userid
- Permit the node's userid to the command profiles the node can issue

The FACILITY class profile for node 1 named HYDEPARK is:

```
NJE.HYDEPARK
```

To control access to commands from other NJE nodes, use the following NODES class profile:

```
*.RUSER.*
```

Your security policy can require auditing of all commands issued, even if they are not valid on your system. You can audit commands that might be unknown to JES2 by defining a profile called JESx.UNKNOWN with a UACC of NONE and an AUDIT value of ALL. JES2 validates all unknown commands with the JESx.UNKNOWN profile.

If you need a finer level of control over your commands (for example, you want only certain operators to change the routing of a printer), you should use JES2 Exit 36. See *z/OS JES2 Installation Exits* for information about Exit 36.

Table 77. JES2 commands with profile names and minimum required authority

JES2 command	Example profile name	Minimum required user access level
$A A	jesx.MODIFYRELEASE.JOB	UPDATE
$A J	jesx.MODIFYRELEASE.BAT	UPDATE
$A JOBQ	jesx.MODIFYRELEASE.JST	UPDATE
$A S	jesx.MODIFYRELEASE.STC	UPDATE
$A T	jesx.MODIFYRELEASE.TSU	UPDATE
$ACTIVATE	jesx.ACTIVATE.FUNCTION	CONTROL
$ADD APPL	jesx.ADD.APPL	CONTROL
$ADD CONNECT	jesx.ADD.CONNECT	CONTROL

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$ADD DESTID	jesx.ADD.DESTID	CONTROL
$ADD FSS	jesx.ADD.FSS	CONTROL
$ADD JOBCLASS	jesx.ADD.JOBCLASS	CONTROL
$ADD LINE	jesx.ADD.LINE	CONTROL
$ADD LOADMOD	jesx.ADD.LOADMOD	CONTROL
$ADD LOGON	jesx.ADD.LOGON	CONTROL
$ADD NETSRV	jesx.ADD.NETSRV	CONTROL
$ADD PROCLIB	jesx.ADD.PROCLIB	CONTROL
$ADD PRTnnnn	jesx.ADD.DEV	UPDATE
$ADD REDIRECT	jesx.ADD.REDIRECT	CONTROL
$ADD RMT	jesx.ADD.RMT	CONTROL
$ADD SOCKET	jesx.ADD.SOCKET	CONTROL
$ADD SRVCLASS	jesx.ADD.SRVCLASS	CONTROL
$B device	jesx.BACKSP.DEV	UPDATE
$C A**	jesx.CANCEL.AUTOCMD	******
$C J	jesx.CANCEL.BAT	UPDATE
$C Lx.yy	jesx.CANCEL.DEV	UPDATE
$C O J	jesx.CANCEL.BATOUT	UPDATE
$C O JOBQ	jesx.CANCEL.JSTOUT	UPDATE
$C O S	jesx.CANCEL.STCOUT	UPDATE
$C O T	jesx.CANCEL.TSUOUT	UPDATE
$C S	jesx.CANCEL.STC	UPDATE
$C T	jesx.CANCEL.TSU	UPDATE
$C device	jesx.CANCEL.DEV	UPDATE
$C OFFn.JR	jesx.CANCEL.DEV	UPDATE
$C OFFn.JT	jesx.CANCEL.DEV	UPDATE
$C OFFn.SR	jesx.CANCEL.DEV	UPDATE
$C OFFn.ST	jesx.CANCEL.DEV	UPDATE
$D A	jesx.DISPLAY.JOB	READ
$D ACTIVATE	jesx.DISPLAY.ACTIVATE	READ
$D ACTRMT	jesx.DISPLAY.ACTRMT	READ
$D APPL	jesx.DISPLAY.APPL	READ
$D CKPTDEF	jesx.DISPLAY CKPTDEF	READ
$D CLASSGRP	jesx.DISPLAY.CLASSGRP	READ
$D CONDEF	jesx.DISPLAY.CONDEF	READ
$D CONNECT	jesx.DISPLAY.CONNECT	READ
$D DESTDEF	jesx.DISPLAY.DESTDEF	READ
$D DEStid	jesx.DISPLAY.DESTID	READ
$D F	jesx.DISPLAY.QUE	READ

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$D I	jesx.DISPLAY.INITIATOR	READ
$D INITINFO	jesx.DISPLAY.INITINFO	READ
$D J	jesx.DISPLAY.BAT	READ
$D JES2	jesx.DISPLAY.SYS	READ
$D JOBCLASS	jesx.DISPLAY.JOBCLASS	READ
$D JOBQ	jesx.DISPLAY.JST	READ
$D L(nnnn).JR(n)	jesx.DISPLAY.L	READ
$D L(nnnn).JT(n)	jesx.DISPLAY.L	READ
$D L(nnnn).SR(n)	jesx.DISPLAY.L	READ
$D L(nnnn).ST(n)	jesx.DISPLAY.L	READ
$D LINE	jesx.DISPLAY.LINE	READ
$D LOADmod	jesx.DISPLAY.LOADMOD	READ
$D M	jesx.SEND.MESSAGE	READ
$D MASDEF	jesx.DISPLAY.MASDEF	READ
$D MEMBer	jesx.DISPLAY.SYS	READ
$D MODULE	jesx.DISPLAY.MODULE	READ
$D N	jesx.DISPLAY.JOB	READ
$D NETSRV	jesx.DISPLAY.NETSRV	READ
$D NJEDEF	jesx.DISPLAY.NJEDEF	READ
$D NODE	jesx.DISPLAY.NODE	READ
$D O J	jesx.DISPLAY.BATOUT	READ
$D O JOBQ	jesx.DISPLAY.JSTOUT	READ
$D OPTSDEF	jesx.DISPLAY.OPTSDEF	READ
$D O S	jesx.DISPLAY.STCOUT	READ
$D O T	jesx.DISPLAY.TSUOUT	READ
$D PATH	jesx.DISPLAY.PATH	READ
$D PCE	jesx.DISPLAY.PCE	READ
$D PROCLIB	jesx.DISPLAY.PROCLIB	READ
$D PRT	jesx.DISPLAY.DEV	READ
$D PRTnnnn	jesx.DISPLAY.DEV	READ
$D Q	jesx.DISPLAY.JOB	READ
$D REBLD	jesx.DISPLAY.REBLD	READ
$D RDI	jesx.DISPLAY.RDI	READ
$D RDRnn	jesx.DISPLAY.DEV	READ
$D Rnnnnn.CON	jesx.DISPLAY.DEV	READ
$D Rnnnnn.PRm	jesx.DISPLAY.DEV	READ
$D Rnnnnn.PUm	jesx.DISPLAY.DEV	READ
$D Rnnnnn.RDm	jesx.DISPLAY.DEV	READ
$D REDIRect	jesx.DISPLAY.REDIRECT	READ

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$D S	jesx.DISPLAY.STC	READ
$D SOCKET	jesx.DISPLAY.SOCKET	READ
$D SPOOL	jesx.DISPLAY.SPOOL	READ
$D SPOOLDEF	jesx.DISPLAY.SPOOLDEF	READ
$D SRVCLASS	jesx.DISPLAY.SRVCLASS	READ
$D SSI	jesx.DISPLAY.SSI	READ
$D SUBNET	jesx.DISPLAY.SUBNET	READ
$D T	jesx.DISPLAY.TSU	READ
$D TRACE(x)	jesx.DISPLAY.TRACE	READ
$D U	jesx.DISPLAY.DEV	READ
$D init stmt	jesx.DISPLAY.initstmt	READ
$D PUNnn	jesx.DISPLAY.DEV	READ
$DEL CONNECT	jesx.DEL.CONNECT	CONTROL
$DEL DESTID	jesx.DEL.DESTID	CONTROL
$DEL JOBCLASS	jesx.DEL.JOBCLASS	CONTROL
$DEL LOADMOD	jesx.DEL.LOADMOD	CONTROL
$DEL PROCLIB	jesx.DEL.PROCLIB	CONTROL
$E CKPTLOCK	jesx.RESTART.SYS	CONTROL
$E J	jesx.RESTART.BAT	CONTROL
$E JOBQ	jesx.RESTART.JST	CONTROL
$E LINE(x)	jesx.RESTART.LINE	CONTROL
$E LOGON(x)	jesx.RESTART.LOGON	CONTROL
$E MEMBER()	jesx.RESTART.SYS	CONTROL
$E NETSRV	jesx.RESTART.NETSRV	CONTROL
$E OFFn.JT	jesx.RESTART.DEV	UPDATE
$E OFFn.ST	jesx.RESTART.DEV	UPDATE
$E device	jesx.RESTART.DEV	UPDATE
$F device	jesx.FORWARD.DEV	UPDATE
$G A	jesx.GMODIFYRELEASE.JOB	UPDATE
$G C	jesx.GCANCEL.JOB	UPDATE
$G D	jesx.GDISPLAY.JOB	READ
$G H	jesx.GMODIFYHOLD.JOB	UPDATE
$G R	jesx.GROUTE.JOBOUT	UPDATE
$G R (for execution)	jesx.GROUTE.JOBOUT	UPDATE
$H A	jesx.MODIFYHOLD.JOB	UPDATE
$H J	jesx.MODIFYHOLD.BAT	UPDATE
$H JOBQ	jesx.MODIFYHOLD.JST	UPDATE
$H S	jesx.MODIFYHOLD.STC	UPDATE
$H T	jesx.MODIFYHOLD.TSU	UPDATE

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$I device	jesx.INTERRUPT.DEV	UPDATE
$JD DETAILS	jesxMON.DISPLAY.DETAIL	READ
$JD HISTORY	jesxMON.DISPLAY.HISTORY	READ
$JD JES	jesxMON.DISPLAY.JES	READ
$JD MONITOR	jesxMON.DISPLAY.MONITOR	READ
$JD STATUS	jesxMON.DISPLAY.STATUS	READ
$J STOP	jesxMON.STOP.MONITOR	CONTROL
$L J	jesx.DISPLAY.BATOUT	READ
$L JOBQ	jesx.DISPLAY.JSTOUT	READ
$L S	jesx.DISPLAY.STCOUT	READ
$L T	jesx.DISPLAY.TSUOUT	READ
$M	jesx.MSEND.CMD	READ
$M SPL	jesx.MIGRATE.FUNCTION	CONTROL
$N	jesx.NSEND.CMD	READ
$N device	jesx.REPEAT.DEV	UPDATE
$O J	jesx.RELEASE.BATOUT	UPDATE
$O JOBQ	jesx.RELEASE.JSTOUT	UPDATE
$O S	jesx.RELEASE.STCOUT	UPDATE
$O T	jesx.RELEASE.TSUOUT	UPDATE
$P	jesx.STOP.SYS	CONTROL
$P I	jesx.STOP.INITIATOR	CONTROL
$P JES2	jesx.STOP.SYS	CONTROL
$P JOBQ	jesx.STOP.JST	UPDATE
$P LINE(x)	jesx.STOP.LINE	CONTROL
$P LOGON(x)	jesx.STOP.LOGON	CONTROL
$P NETSRV	jesx.STOP.NETSRV	CONTROL
$P O J	jesx.STOP.BATOUT	UPDATE
$P O JOBQ	jesx.STOP.JSTOUT	UPDATE
$P O S	jesx.STOP.STCOUT	UPDATE
$P O T	jesx.STOP.TSUOUT	UPDATE
$P OFFn.JR	jesx.STOP.DEV	UPDATE
$P OFFn.JT	jesx.STOP.DEV	UPDATE
$P OFFn.SR	jesx.STOP.DEV	UPDATE
$P OFFn.ST	jesx.STOP.DEV	UPDATE
$P OFFLOADn	jesx.STOP.DEV	UPDATE
$P RMT(x)	jesx.STOP.RMT	CONTROL
$P S	jesx.STOP.STC	UPDATE
$P SPOOL	jesx.STOP.SPOOL	CONTROL
$P SRVCLASS	jesx.STOP.SRVCLASS	CONTROL

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$P T	jesx.STOP.TSU	UPDATE
$P TRACE(x)	jesx.STOP.TRACE	CONTROL
$P XEQ	jesx.STOP.SYS	CONTROL
$P device	jesx.STOP.DEV	UPDATE
$PO JOBQ	jesx.STOP.JSTOUT	UPDATE
$PO JOB	jesx.STOP.BATOUT	UPDATE
$PO STC	jesx.STOP.STCOUT	UPDATE
$PO TSU	jesx.STOP.TSUOUT	UPDATE
$R ALL	jesx.ROUTE.JOBOUT	UPDATE
$R PRT	jesx.ROUTE.JOBOUT	UPDATE
$R PUN	jesx.ROUTE.JOBOUT	UPDATE
$R XEQ	jesx.ROUTE.JOBOUT	UPDATE
$S	jesx.START.SYS	CONTROL
$S A	jesx.START.AUTOCMD	CONTROL
$S I	jesx.START.INITIATOR	CONTROL
$S J	jesx.START.BAT	UPDATE
$S LINE(x)	jesx.START.LINE	CONTROL
$S LOGON(x)	jesx.START.LOGON	CONTROL
$S N	jesx.START.NET	CONTROL
$S OFFn.JR	jesx.START.DEV	UPDATE
$S OFFn.JT	jesx.START.DEV	UPDATE
$S OFFn.SR	jesx.START.DEV	UPDATE
$S OFFn.ST	jesx.START.DEV	UPDATE
$S OFFLOADn	jesx.START.DEV	UPDATE
$S RMT(x)	jesx.START.RMT	CONTROL
$S SPOOL	jesx.START.SPOOL	CONTROL
$S SRVCLASS	jesx.START.SRVCLASS	CONTROL
$S TRACE(x)	jesx.START.TRACE	CONTROL
$S XEQ	jesx.START.SYS	CONTROL
$S device	jesx.START.DEV	UPDATE
$T A(CREATE)	jesx.MODIFY.AUTOCMD	READ
$T A(OWNER)	jesx.MODIFY.AUTOCMD	READ
$T A(NOT OWNER)	jesx.MODIFY.AUTOCMD	CONTROL
$T APPL	jesx.MODIFY.APPL	CONTROL
$T BUFDEF	jesx.MODIFY.BUFDEF	CONTROL
$T CKPTDEF	jesx.MODIFY.CKPTDEF	CONTROL
$T CONDEF	jesx.MODIFY.CONDEF	CONTROL
$T CONNECT	jesx.MODIFY.CONNECT	CONTROL
$T DEBUG	jesx.MODIFY.DEBUG	CONTROL

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$T DESTDEF	jesx.MODIFY.DESTDEF	CONTROL
$T DEStid	jesx.MODIFY.DESTID	CONTROL
$T ESTBYTE	jesx.MODIFY.ESTBYTE	CONTROL
$T ESTIME	jesx.MODIFY.ESTIME	CONTROL
$T ESTLNCT	jesx.MODIFY.ESTLNCT	CONTROL
$T ESTPAGE	jesx.MODIFY.ESTPAGE	CONTROL
$T ESTPUN	jesx.MODIFY.ESTPUN	CONTROL
$T EXIT	jesx.MODIFY.EXIT	CONTROL
$T FSS	jesx.MODIFY.FSS	CONTROL
$T I	jesx.MODIFY.INITIATOR	CONTROL
$T INTRDR	jesx.MODIFY.INTRDR	CONTROL
$T J	jesx.MODIFY.BAT	UPDATE
$T JOBCLASS	jesx.MODIFY.JOBCLASS	CONTROL
$T JOBDEF	jesx.MODIFY.JOBDEF	CONTROL
$T JOBPRTY	jesx.MODIFY.JOBPRTY	CONTROL
$T JOBQ	jesx.MODIFY.JST	UPDATE
$T LINE	jesx.MODIFY.LINE	CONTROL
$T LOADMOD	jesx.MODIFY.LOADMOD	CONTROL
$T LOGON	jesx.MODIFY.LOGON	CONTROL
$T MASDEF	jesx.MODIFY.MASDEF	CONTROL
$T MEMBER(x)	jesx.MODIFY.SYS	CONTROL
$T NETSRV	jesx.MODIFY.NETSRV	CONTROL
$T NODE	jesx.MODIFY.NODE	CONTROL
$T NJEDEF	jesx.MODIFY.NJEDEF	CONTROL
$T NUM	jesx.MODIFY.NUM	CONTROL
$T O J	jesx.MODIFY.BATOUT	UPDATE
$T O JOBQ	jesx.MODIFY.JSTOUT	UPDATE
$T O S	jesx.MODIFY.STCOUT	UPDATE
$T O T	jesx.MODIFY.TSUOUT	UPDATE
$T OFFx.yy	jesx.MODIFY.OFF	CONTROL
$T OFFLOADx	jesx.MODIFY.OFFLOAD	CONTROL
$T OUTCLASS	jesx.MODIFY.OUTCLASS	CONTROL
$T OUTDEF	jesx.MODIFY.OUTDEF	CONTROL
$T OUTPRTY	jesx.MODIFY.OUTPRTY	CONTROL
$T PCE	jesx.MODIFY.PCE	CONTROL
$T PRINTDEF	jesx.MODIFY.PRINTDEF	CONTROL
$T PROCLIB	jesx.MODIFY.PROCLIB	CONTROL
$T RECVopts	jesx.MODIFY.RECVOPTS	CONTROL
$T REDIRect	jesx.MODIFY.REDIRECT	CONTROL

Table 77. JES2 commands with profile names and minimum required authority (continued)

JES2 command	Example profile name	Minimum required user access level
$T RMT	jesx.MODIFY.RMT	CONTROL
$T S	jesx.MODIFY.STC	UPDATE
$T SMFDEF	jesx.MODIFY.SMFDEF	CONTROL
$T SOCKET	jesx.MODIFY.SOCKET	CONTROL
$T SPOOLDEF	jesx.MODIFY.SPOOLDEF	CONTROL
$T SPOOL	jesx.MODIFY.SPOOL	CONTROL
$T SRVCLASS	jesx.MODIFY.SRVCLASS	CONTROL
$T SSI	jesx.MODIFY.SSI	CONTROL
$T STCCLASS	jesx.MODIFY.STCCLASS	CONTROL
$T TPDEF	jesx.MODIFY.TPDEF	CONTROL
$T TRACEDEF	jesx.MODIFY.TRACEDEF	CONTROL
$T T	jesx.MODIFY.TSU	UPDATE
$T device	jesx.MODIFY.DEV	UPDATE
$T init stmt	jesx.MODIFY.init stmt	CONTROL
$T TSUCLASS	jesx.MODIFY.TSUCLASS	CONTROL
$VS*	jesx.VS	CONTROL
$Z A	jesx.HALT.AUTOCMD	CONTROL
$Z I	jesx.HALT.INITIATOR	CONTROL
$Z OFFLOADn	jesx.HALT.DEV	UPDATE
$Z SPOOL	jesx.HALT.SPOOL	CONTROL
$Z device	jesx.HALT.DEV	UPDATE
$ZAPJOB	jesx.ZAP.JOB	CONTROL

Considerations for automatic commands

Automatic commands have the same types of controls as other commands. Profiles exist for the specific automatic commands ($TA, $CA, $ZA, $HA) but there are other things you must consider when issuing the automatic commands.

- The issuer of the automatic command must have the appropriate authority for the automatic command and all commands the automatic command issues. JES2 passes the security information of the originator of the automatic command with all the commands issued by the automatic command. RACF uses the originator's userid when deciding if the user can issue any particular command.
- To update an automatic command, your userid must either be the userid that originally issued the automatic command with READ access or have CONTROL access to automatic commands. You should consider giving CONTROL access to automatic commands to a limited number of operators who need to change the automatic commands on a regular basis.

Using JES2 to provide security

JES2 has a number of JES2 initialization statement parameters and installation exits that you can use to protect JES2. Some of the resources that you can protect using JES2-provided facilities include:

- NJE communication lines

- RJE communication lines
- Remote workstation SIGNON/LOGON
- VTAM sessions
- Commands

JES2 initialization statements

You can use JES2 initialization statements to control:

- Access to your system by:
  - RJE lines and workstations [LINE(nnnn) and RMT(nnnn)]
  - NJE lines and nodes [LINE(nnnn) and NODE(nnnn)]
- JES2 access to VTAM [LOGON(n)]
- Job transmission or forwarding to:
  - Another node with encrypted passwords [NODE(nnnn)]
  - Spool offload devices
- Output to:
  - Local devices
  - Remote workstation devices
  - Other nodes [NODE(nnnn)]
  - Spool offload devices [OFFLOAD(n)]
- Operator commands entering the system:
  - Through internal readers (INTRDR)
  - Through readers [RDR(nn)]
  - From other nodes [NODE(nnnn)]
  - From jobs, started tasks, or a TSO/E session [JOBCLASS(v)]

when RACF is not active or installed.

Table 78 shows the JES2 initialization statements and parameters that deal with security, the resources they affect, and a brief description of each parameter. For detailed information about each parameter, see *z/OS JES2 Initialization and Tuning Reference*.

Table 78. JES2 security-related initialization statements

Statement	Parameter	Resource affected	Parameter description
DEBUG	SECURITY=	device access	Specifies whether JES2 provides additional RACF security logging to include warnings for JES2 output work selection for JES2 devices and external writer (XWTR) requests and SYSOUT application programming interface (SAPI) requests.
INTRDR	AUTH=*	commands	Determines the types of commands accepted from any internal reader.

Table 78. JES2 security-related initialization statements (continued)

Statement	Parameter	Resource affected	Parameter description
JOBCLASS(n)	AUTH= *	commands	Determines the group of MVS commands accepted from a particular job class, started tasks, or TSO/E sessions.
	COMMAND= *	commands	Defines the action the system takes when encountering an MVS command in the job stream.
LINE(nnnn)	PASSWORD=	system access from line	Restricts access to the system from this line to workstations specifying the proper password.
LOGON(n)	PASSWORD=	VTAM access	Restricts access by JES2 to VTAM.
NODE(nnnn)	AUTH= *	commands	Defines the command authority another node has on your system.
	PASSWORD=	system access from node	Restricts access to the system from the node this statement defines.
	PENCRYPT=	password encryption	Enables encryption of passwords sent with data and jobs to this node.
	RECEIVE=	data	Defines the type of data your system can receive from an adjacent node.
	SIGNON=	system access from node	Restricts access to the system from the node this statement defines.
	TRANSMIT=	data	Defines the type of data your system can transmit to an adjacent node.
OFFLOAD(n)	PROTECT=	spool offload data set	Specifies whether the spool offload data set needs RACF protection.
RDR(nn)	AUTH= *	commands	Determines the types of commands accepted from a reader.
RMT(nnnn)	PASSWORD= *	system access from remote terminals	Restricts access to the system from this remote to workstations specifying the proper password.
SOCKET(xxxx)	SECURE=	NJE Connection	Specifies whether SSL/TLS is to be used on an NJE connection.
* This parameter takes effect only when RACF is not active or installed.			

JES2 security exit points

You can enhance standard JES2 security by performing your own validation at certain exit points. These exits are located at strategic places in the processing and using existing information, you can make decisions about access to certain resources. Table 79 on page 383 lists the JES2 exits related to security and a brief description of how you could use them.

Table 79. Security-related exits

Exit	Exit Name	Description
2	JOB Statement Scan	Perform password validation or restrict job names to certain users.
3	JOB Statement Accounting Field Scan	Validate the account number on the JOB statement.
4	JCL and JES2 Control Statement Scan	Restrict resources a job uses based on information in the JCL or JES2 Control statements. Inspect and validate commands imbedded in a job stream.
5	JES2 Command Preprocessor	Restrict commands processed by JES2. Allow or disallow commands based on the parameters specified.
6	Converter/Interpreter Text Scan	Restrict resources a job uses based on information in the JCL.
13	TSO/E Interactive Data Transmission Facility Screening and Notification	Restrict data received from nodes through the TSO/E RECEIVE command.
17	BSC RJE SIGN-ON/SIGN-OFF	Control the BSC RJE devices that access your system.
18	SNA RJE LOGON/LOGOFF	Control the SNA RJE devices that access your system.
20	END OF JOB INPUT	Perform a final check of a job's attributes after JES2 processes all the input.
22	CANCEL/STATUS	Restrict use of the TSO/E CANCEL and STATUS commands.
30	OPEN (and RESTART) of SSI Data Sets and Internal Readers	Control access to data sets residing on spool. Restrict use of internal readers.
31	Allocation of SSI Data Sets and Internal Readers	Control access to SSI data sets or internal readers.
32	SSI Job Selection	Control which jobs JES2 selects for execution.
33	SSI Data Set Close	Validate characteristics and destination of subsystem data sets.
34	SSI Data Set Unallocation	Validate characteristics and destination of subsystem data sets.
36	Pre-Security Authorization Call	Modify the information passed to SAF.
37	Post-Security Authorization Call	Perform other security checks before granting access to a resource.
38	TSO/E RECEIVE Data Set Disposition	Change the default processing for a data set that the user can never receive.
39	NJE SYSOUT Data Set Disposition	Change the default processing for an NJE SYSOUT data set that failed validation.
50	End of input	Perform a final check of a job's attributes after JES2 processes all the input.
52	JOB JCL Statement Scan	Perform password validation or restrict job names to certain users.

Table 79. Security-related exits (continued)

Exit	Exit Name	Description
53	JOB Statement Accounting Field Scan	Validate the account number on the JOB statement.
54	JCL and JES2 Control Statement Scan	Restrict resources a job uses based on information in the JCL or JES2 Control statements. Inspect and validate commands imbedded in a job stream.
55	NJE SYSOUT Data Set Disposition	Change the default processing for an NJE SYSOUT data set that failed validation.
60	Converter/Interpreter Text Scan	Restrict resources a job uses based on information in the JCL.

See *z/OS JES2 Installation Exits* for details on implementing these exits.

Multiple levels of a security product in a MAS

JES2 security support assumes that the same level of the security product is installed on all members of a MAS and that the security product's data base is the same on all members. Unexpected security failures and inconsistent audit records might occur if this is not the case.

For example, if the security product on member 1 requires SECLABELs and the security product on member 2 does not support SECLABELs, then a job submitted on member 2 but run on member 1 can fail because it has no SECLABEL. Security failures can also occur attempting to access from member 1, a job submitted on member 2 or SYSOUT created on member 2. Accesses that can fail include the JESSPOOL call made when a SPOOL data set is purged. The data set will be purged, but a security violation audit record and message might be generated.

Complete audit records might not be available in this environment. Audit records will only be created by systems with security products that support the appropriate classes. Access from other systems will not be audited.

For these reasons, it is recommended that classes/profiles related to JES2 should not be activated until all members have security products capable of supporting those classes/profiles. Also, those classes should be activated on all members at the same time.

Special consideration must be given to MASes in which one member has a security product and the other does not. Information placed in TOKENs by SAF on a system with no security product has not been authenticated. The security product on the other member must be aware of this and perform appropriate verification.

In this environment, if the security product installed on the one member is RACF, the following considerations apply:

- Jobs submitted on the member without RACF will be verified either when they are converted, or when they go into execution. In order for them to pass this reverification, USER= and PASSWORD= must be specified on the JOB card.
- If any installation exits change the userid associated with a JOB to something other than what was explicitly specified on a JOB card, the JES2 data set names might have inconsistent userids in them.
- Installation might experience problems activating any of the following classes:

- JESINPUT
- JESJOBS
- JESSPOOL
- SURROGAT
- WRITER

Controlling job class usage

To customize job class usage for a specific environment, you can grant access based on the submitter's profile, the owner's profile, or both.

Control of job class usage is based on the presence or absence of two SAF FACILITY class profiles:

- If the profile JES.JOBCLASS.OWNER is defined in the FACILITY class, job class profiles for owners are enforced.
- If the profile JES.JOBCLASS.SUBMITTER is defined in the FACILITY class, job class profiles for submitters are enforced.
- If both profiles are defined in the FACILITY class, a job class must pass both sets of profiles before it is considered valid.
- If neither profile is defined in the FACILITY class, any job class can be used.

Job class profiles are in the form: JOBCLASS.*localnodeid*.*jobclass*.*jobname*:

localnodeid
: Specifies the name of the node on which the job is located

jobclass
: Specifies the 1-8 character name of the job class that is being controlled

jobname
: Specifies the name that is defined in the name field in the JOB statement.

Appendix A. IBM devices supported by JES2 and how to use them

The following provides a list of the devices supported by JES2. JES2 supports all model numbers of devices listed, unless otherwise specified. See *z/OS Migration* for more information about these devices.

- Local Card Readers
  - 2501
  - 2540
  - 3505
  - 3525
- Local Card Punches
  - 2540
  - 3525
- Local Printers (JES2-driven)
  - 1403-N1
  - 3203
  - 3211
  - 3800
  - 4245
  - 4248
- DASD (for Checkpoint and Spool)
  - 3330
  - 3340
  - 3350
  - 3375
  - 3380
  - 3390
  - 9340/45
- Spool Offload Devices
  - Any device supported by DFP
- Remote Workstations
  - BSC Non-Programmable Remotes
    - 2770
    - 2780
    - 3780 and 3781
    - 3770 models which emulate 2780/3780
  - BSC Multi-Leaving Remotes
    - 3777-2
    - S/360 Model 20 (Submodels 2, 4, 5, 6)
    - S/360 Models 22 and subsequent models
    - S/370 Models 115 and subsequent models
    - Workstation Programs on S/36, S/38, AS/400®, and so on.

– Remote SNA Workstations (LUTYPE1)
  - 3771
  - 3776
  - 3777-3
  - 3790
  - 8100

Locally attached Print Services Facility (PSF) printers are specified as AFP1. See *AFP: Printer Information*, G544-3290, for a list of supported device types.

The following devices are not supported by JES2:

- 1403-3 Printer
- 1442 Card Reader
- 1443 Printer
- 2520 Card Reader
- 3540 Diskette Reader/Writer
- S/3 and 1130 as Remote Workstations

DASD utilization tables

Table 80 shows two things:

1. The storage capacity of various types of devices presented as MBTYPES/VOLUME, BTYES/TRACK, and TRACKS/VOLUME.
2. How JES2 assigns SPOOL devices based on each device's characteristics and where the SPOOLDEF initialization statement parameters have been set with TGSIZE=30 and BUFSIZE=3992. The values in the table represent what JES2 uses in calculating the available SPOOL Device storage.

   For example:

   a. JES2 uses BUFFERS per TRACK of 10 for either 3380s or 9340s and uses a BUFFERS per TRACK GROUP of 30.
   b. JES2 uses BUFFERS per TRACK of 12 for 3390s and sets the BUFFERS per TRACK GROUP to 36.
   c. For a mixture of devices (3380s, 9340s, and 3390s), JES2 starts with TGSIZE=30 and makes the appropriate adjustments for each device types (that is, 3380s and 9340s get 30 BUFFERS/TRACK GROUP and 3390s get 36 BUFFERS/TRACK GROUP).

When you have identified the device types for use as SPOOL devices and have set your SPOOLDEF initialization statements, you use this table to identify how JES2 utilizes these devices and how JES2 allows for their storage limitations.

Table 80. Calculated values for several DASDs. The Device attribute table header cell spans two table header rows, which are Spool device type characteristics and the specific model types. The final row in the table is a footnote, which spans all table columns.

Device attribute	Spool device type characteristics							
	3380	**3380E**	**3380K**	**3390-1**	**3390-2**	**3390-3**	**9340-1**	**9340-2**
MBYTES/VOLUME	630	1260	1890	946	1892	2838	1003	1502
BYTES/TRACK	47476	47476	47476	56664	56664	56664	46456	46456
TRACKS/VOLUME	13275	26550	39825	16695	33390	50085	21600	32340
Note: Characteristics of 3390 Model 2. For 3390 Model 1 capacities, divide the volume characteristics by 2.								

Table 81. Calculated values for several DASDs - BUFSIZE and TGSIZE. The Device attribute values table header cell spans two table header rows, which are Spool device type characteristics and the specific model types. The final row in the table is a footnote, which spans all table columns.

Device attribute values (BUFSIZE=3992, TGSIZE=30)	Spool device type characteristics							
	3380	3380E	3380K	3390-1	3390-2	3390-3	9340-1	9340-2
BUFFERS/TRACK	10	10	10	12	12	12	10	10
TRKCELL	4,5	4,5	4,5	3,4,6	3,4,6	3,4,6	4,5	4,5
TRACKS /TRACK GROUP	3	3	3	3	3	3	3	3
BUFFERS /TRACK GROUP	30	30	30	36	36	36	30	30
KBYTES /TRACK GROUP	120	120	120	144	144	144	120	120
TRACK GROUPS /VOLUME	4425	8850	13275	5565	11130	16695	7200	10780
Usable MBYTES/VOLUME	530	1060	1590	800	1600	2400	862	1291
Device Utilization(%)	84	84	84	85	85	85	85.9	85.9
Note: Characteristics of 3390 Model 2. For 3390 Model 1 capacities, divide the volume characteristics by 2.								

Table 82 provides device utilization percentages for selected DASD based on several JES2 initialization parameter BUFSIZE specifications.

The 3390-9 is supported for SPOOL. See "Specifying the storage of checkpoints on DASD" on page 199 for information on using the SPACE= parameter.

Table 82. Selected device and BUFSIZE parameter (on SPOOLDEF) values and resultant utilization percentages. The first table header row, Utilization (%), spans all six table columns. The DASD and BYTES/TRACK column data cells span multiple data rows. The final row in the table is a footnote, which spans all table columns.

UTILIZATION (%)					
DASD	**BYTES /TRACK***	**BUFSIZE**	**BUFFERS /TRACK**	**DEVICE****	**STORAGE*****
3380	**47476**	**3992**	**10**	**84**	**97**
		3856	**11**	**89**	**94**
		1944	**19**	**78**	**95**
3390	**56664**	**3992**	**12**	**85**	**97**
		3856	**12**	**82**	**94**
		3768	**13**	**86**	**92**
		1944	**22**	**76**	**95**
9340	**46456**	**3992**	**10**	**85.9**	**97**
		3672	**11**	**89.6**	**65**
		1944	**18**	**75.3**	**95**
* Values presented as "Maximum Block Size Per Track" in *MVS/DFP Managing non-VSAM Data Sets*					
** Device Utilization % = BUFSIZE x buffers/track bytes/track					
*** Storage Utilization % = BUFSIZE x buffers/page bytes/page (= 4096)					

Appendix B. Miscellaneous JES2 facilities

This appendix describes some optional JES2 facilities that can help an installation to operate and tune their JES2 member. Some of the facilities described here support other components of the operating system.

The JES2 facilities described are:

Automatic command processing

The operator can specify, from the console or through a local reader, that certain commands or strings of commands take effect automatically at specific times or at regular intervals. The procedures for using the following commands are described in *z/OS JES2 Commands*.

Note that automatic commands can be placed in the initialization deck.

- Start Automatic ($S A): Starts automatic command processing.
- Set Automatic ($T A): Displays, specifies, or modifies the strings of commands (the "automatic command elements").
- Cancel Automatic ($C A): Cancels all previously entered automatic commands. This command can also selectively cancel selected commands.
- Halt Automatic ($Z A): Stops all automatic command processing until it is restarted.

Automatic command processing can be used to provide status displays and to lessen the operator's work for common, preset routines or schedules. For example, if an installation normally does one specific kind of processing at 8 a.m., and another at 9 a.m. every day, it is possible to preset automatic command processing to issue the operator commands that would ordinarily be necessary at those times.

Writing a day's work scheduler

Establish the use of automatic command processing with the AUTOCMD parameter on the CONDEF statement at initialization. You can enter the commands with the JES2 $T command or write a program to put command statements through an internal reader.

Use the T= keyword on the $T A command to specify the time of day JES2 will issue automatic command(s). T= specifies a time in hours (hh) and minutes (mm) from the previous midnight. Therefore, if the specified time is earlier than the current time, the command is executed immediately. If the time is greater than the current time, the command will be executed hh hours and mm minutes after the last midnight. If the specified time is greater than 24 hours, (note the maximum value for T= is 99.59) a command can be executed up to as many as 3 days, 3 hours, and 59 minutes after the previous midnight. (The time is reset by JES2 each day (at midnight) by decreasing the value by 24 hours.) The following example

shows these three situations.

If the current time is 14:00 on Tuesday, then:

```
$TA,T=11.30,'$S LINE(1)'     /*LINE1 started immediately    */
$TA,T=15.00,'$S LINE(2)'     /*LINE2 started in one hour    */
$TA,T=27.00,'$S LINE(3)'     /*LINE3 started at 03:00 Wed   */
$TA,T=51.00,'$S LINE(4)'     /*LINE4 started at 03:00 Thurs */
$SA,ALL
```

Be aware that if these automatic commands are not added to your initialization data set, and JES2 experiences a failure, the commands are lost. You can prevent such situations by adding the commands to your initialization data set or by submitting a $TA,T=24.01,'$VS,'S D,JOB=jobname'' (where D is a reader procedure and jobname defines a job that contains the series of automatic commands your installation wants to execute daily) command to automatically recall the job each day. Note that the following example procedure provides a set of automatic commands that are continually refreshed daily.

```
$TA001,T=00.01,'$TRDI,A=0'                /*SET INTERNAL READERS' AUTH*/
$TA004,T=00.03,'$VS,'V 02C-02F,ONLINE''    /*VARY PRINTERS ONLINE   */
$TA115,T=00.05,'$OJOBQ,Q=B,CANCEL'        /*CANCEL CLASS B HELD OUTPUT*/
$TA116,T=07.58,'$VS,'S MANAGMT,JOB=BKUPCTLG''    /*  SUBMIT JOB THRU*/
$TA300,T=15.58,'$VS,'S MANAGMT,JOB=BKUPCTLG''    /*  MANAGMT PROC   */
$TA400,T=23.58,'$VS,'S MANAGMT,JOB=BKUPCTLG''    /*  EVERY 8 HRS    */
$TA995,T=24.01,'$VS,'S D,JOB=AUTOCMD1''    /*SUBMIT MEMBER TO INTRDR*/
```

The following statements represent sample command statements placed in the initialization data set:

```
$TA,T=10.30,'$SLINE(1-3)'
$TA,T=12.30,'$TI1,C=ABC;TINIT2,C=XBC',L=A
$TA,T=16.15,'$PLINE(1),LINE(3);DMR1-9,'PLEASE SIGN OFF ASAP'
$TA,I=90,'$DPRT2,L=CN3D8-A'
```

These four statements mean the following (and include times of the day set aside for the processing):

1. Start the three remote job entry lines defined.
2. Modify these initiators.
3. Prepare to stop the remote job entry lines and give a warning to users who are currently using the system.
4. Display the attributes of printer 2 every 90 minutes (through the Interval parameter) to out-of-line area A at console CN3D8.

These commands may be set up by a user-written program for scheduling the day's work. This program can use the internal reader to enter the commands in the subsystem. When writing such a program, consider the following:

- When more than one command is to execute at approximately the same time, you should combine them into one command text entry.
- The responses to the command within the text are normally directed to the in-line messages area and to any consoles receiving MCS route code 1, unless you use the L= operand as described in *z/OS JES2 Commands*.
- Multiple commands with responses to the out-of-line area on graphic consoles normally are automatically overlaid so rapidly that the operator cannot view them. Avoid this kind of command sequencing.
- The authority of the internal reader determines which of the commands entered through it will be valid. See *z/OS JES2 Commands* for a discussion of command authority relative to automatic command processing.

- The day's work scheduler program should limit the number of automatic command entries to a value that does not overload the system consoles or leave the operators insufficient resources for their interval status displays.
- An entire automatic command processing entry must fit on an 80-column statement image.

Limiting considerations

When automatic command processing is active, commands entered at system speed (rather than at operator speed) may congest the system. In turn, system response to the commands may tend to flood the console with the response messages.

If your installation is experiencing difficulty either with commands congesting the system or messages flooding the console, reevaluate the mix of commands submitted with automatic command processing and try some changes.

Automatic command processing can also end prematurely under the following conditions:

- If the operator enters the $Z A command to halt automatic command processing and then lets 24 hours or more elapse without restarting it.
- If the operator specifies a start time for automatic command processing that is earlier than the current time, the command will be issued and then canceled.
- If the operator specifies a start time that does not include an hour specification, but only a minute specification, the command will be canceled without being issued.
- If the system becomes so congested that the automatic commands are delayed approximately 5 minutes.

When an automatic command specifying T= has been executed, it must be reentered if it is to be executed at a later time. Make sure the operator fully understands the procedures for using automatic command processing. They are fully outlined in *z/OS JES2 Commands*.

Using the MVS message service (MMS) for JES2 messages

The MVS message service (MMS) provides for the display of JES2 messages in languages other than English, including languages that require a double-byte character set. For more information about MMS, see *z/OS MVS Programming: Assembler Services Guide*.

Managing storage in JES2

Most JES2 code in common storage resides above 16 megabytes in virtual storage, while all JES2 code in private storage resides below 16 megabytes in virtual storage. All JES2 code runs in 31-bit addressing mode, except those sections of JES2 code that deal with calls to restricted MVS services (such as the FSI). JES2 uses 31-bit addressing to reduce its use of the common storage area (CSA) below 16 megabytes in virtual storage.

Some JES2 control blocks reside below 16 megabytes in virtual storage, because these control blocks need to references other MVS control blocks that also reside there. (For a list of these control blocks, see *z/OS JES2 Installation Exits*.)

If you require further information on JES2 when operating in 31-bit addressing mode, to include coding specifics, control block residency, required linkage conventions, or the changes to several assembler instructions (to provide compatibility with the operating system code) see *z/OS JES2 Installation Exits* and *z/OS MVS Programming: Assembler Services Guide*.

Storage considerations

The careful specification of several initialization parameters enables you to optimize both central and virtual storage. The following sections can help you optimize storage use and aid performance of your system.

JES2 private area virtual storage

The checkpoint data set, NJE and RJE line control blocks, RJE device control blocks, and spool buffers can require large amounts of virtual storage. Also, teleprocessing buffers, offload transmitters and receivers, networking lines, JES2 code, and installation modifications consume additional virtual storage. Therefore, to minimize JES2 private area size below 16 megabytes in virtual storage, take care not to overspecify the initialization parameters listed in the following table.

Initialization Statement	Parameter
BUFDEF	BELOWBUF LIMIT
TPDEF	BSCBUF LIMIT
	SIZE

Central storage

Control use of central storage space by carefully specifying the value for the TRKCELL parameter on the SPOOLDEF initialization statement. The following notes should assist you, if you experience central storage constraints.

- Two copies of the checkpoint data set reside in virtual storage.
- If your installation has a 3800-1 printer operated by JES2 and track-cell despooling and double buffering are in effect, the number of central storage frames is two times the value of TRKCELL during printing activity. (If you set the BUFSIZE= parameter on the SPOOLDEF statement to its lower limit, 1944, you only need TRKCELL= number of frames.)

Also, note that as your installation adds printers to its configuration, JES2 central storage requirements increase proportionally.

JES2 obtains the buffer storage requirements for printers operated by a functional subsystem from the functional subsystem address space private area and page-fixes the storage. Therefore, these buffers do not affect JES2 virtual storage usage, but they increase the use of central storage.

Storage isolation

JES2 has specific storage requirements. If you do not set aside sufficient central storage for JES2, your system can suffer any combination of the following symptoms:

- The system runs in an extremely degraded state.
- The number of remote job entry (RJE) time-outs increases.
- System tasks initiate slowly.
- System lock-out messages occur in multi-access spool configurations.

If your system performance exhibits these symptoms, we recommend the use of storage isolation. Also, if your installation requires only certain functions (for example, RJE) during certain periods of the day, you should specify a fairly wide range of storage isolation values, in order to free up some central storage when those functions are not in use. The technique for specifying storage isolation for JES2 is the same as for any other MVS address space. You can adjust the appropriate level of storage isolation for your JES2 system by using parameters in the installation performance specifications (IPS) member of SYS1.PARMLIB. (See *z/OS MVS Initialization and Tuning Reference* for a complete description of this technique.)

Job journaling, SMF records and SMF exits

To reduce overhead, request job journaling explicitly only when needed and turn off unused JES2-generated SMF records and exits (unless you need them for analysis). Measurements have shown a significant (up to 34%) increase in overhead caused by journaling virtual I/O on a system with a workload of many steps.

To bypass journaling and turn off SMF records, specify the following parameters on the JOBCLASS(v) initialization statement: JOURNAL=NO, TYPE6=NO, TYPE26=NO, IEFUJP=NO, IEFUSO=NO, and LOG=NO. The default for each is YES; therefore, you must specify these to turn off journaling. Turning off SMF records has an impact on your ability to audit security-related events. Before suppressing SMF records, you should discuss the implications with your security administrator.

For more information about these parameters and the JOBCLASS(v) statement, see the JOBCLASS statement description in *z/OS JES2 Initialization and Tuning Reference*.

Spool utilization

Correct performance problems related to spool capacity or I/O efficiency by properly specifying the following initialization parameters: BUFSIZE=, TGSPACE=(MAX=), TRKCELL=, and TGSIZE= on the SPOOLDEF statement. See the individual description of each in *z/OS JES2 Initialization and Tuning Reference* and the device utilization statistics tables, "DASD utilization tables" on page 388.

The JES2 health monitor

The JES2 health monitor is a self-starting, self-diagnostic service aid that allows JES2 to monitor very severe performance problems within the JES2 address space. It resides in load module HASJ2MON and runs in its own address space, *jesx*MON (where *jesx* is the specific subsystem name). The health monitor automatically starts when JES2 is initialized and terminates when JES2 terminates cleanly, such as in response to a $P JES2 command. Do not consider it a performance monitor in the traditional sense but rather an overall subsystem status reporter. The health monitor samples JES2 processing, collects data, and reports situations where JES2 is not responding to commands, and you cannot easily diagnose the problem. Such situations can be as basic as a command that is taking an unexpected amount of time to complete, a legitimate "bug" in JES2, an exit routine problem, or other code running in the JES2 address space.

Also, JES2 provides a set of $J commands to allow you to both control the monitor itself and request JES2 report the data it has collected for the operator to view. To perform its intended function, the health monitor performs the following series of processing steps:

1. Sampler processing

2. Probe processing
3. Message issuing.

The following discusses the processing steps and how you can interact with the monitor to view its data.

Health monitor processing

Health monitor processing includes the following steps:

1. Sampler processing: The first step JES2 takes is one of sampler processing. At a rate of 20 times per second, JES2 examines all JES2 main task control blocks (TCBs) and request blocks (RBs) as it looks for main task waits, loops, and other processing delays. It also records all unexpected MVS waits. Further, the monitor examines resource usage at a rate of once per second and tracks low, high, and average usage. It resets this usage data at the beginning of each new hour, and maintains a 72-hour usage history of the same set of resources reported by $HASP050. See *z/OS JES2 Messages* for that resource list. With a baseline and history, JES2 feeds the probe processing task where JES2 can determine typical and abnormal processing trends.
2. Probe processing: With up to 72 hours of processing data available, JES2 can compare current data to that base and determine when problems arise. Based on specific condition duration, JES2 performs increasingly heightened degrees of monitoring as it begins to inform the operator of problem situations. JES2 also groups and tracks main task events and checkpoint-lock-held events separately.
3. Operator notification messaging: The monitor has the ability to issue messages based on length of a particular event. Generally, the longer a potential problem lasts the higher notification level JES2 assigns it and the more messages it provides the operator. JES2 responds and notifies the operator as events are detected and revealed through $HASP9nnn messages. Table 83 lays out the various message types and an indication of what information each type provides.

Table 83. JES2 health monitor message types (as determined by time). The last row in the table is the table footnote, which spans the three table columns.

Time range *	Message type	JES2 interpretation and handling
0 - n seconds	Notice	JES2 considers the condition within "normal" parameters (or just begun) and ignores it. JES2 gathers such information and displays them in response to $JDJES and $JDSTATUS commands. Examples include events such as JES2 TERMINATING, CKPT RECONFIGURATION IN PROGRESS, NOT ALL SPOOL VOLUMES ARE AVAILABLE. These messages are collected in one place to assist the operator get a more complete view of JES2 status, although such messages are issued elsewhere, outside the JES2 monitor message range ($HASP9*nnn*).
n - x seconds	Tracking	JES2 starts tracking the event. $JD JES command displays both real and potential problem events
x+ seconds	Alert	JES2 issues an alert message. The situation might require eventual operator involvement to correct
+30 or +120 seconds	Alert DOM'd	JES2 reissues the alert message with updated information and continues to do so until the condition has cleared
n/a	"All Clear"	JES2 issues $HASP9301 JES2 MAIN TASK ALERTS CLEARED
* *n* and *x* values vary because they are based on "normal" timing for the specific event. JES2 sets these dynamically based on individual processes and as compared to the 72-hour historical data the health checker maintains.		

Controlling the JES2 health monitor

To control the JES2 health monitor and request data it has recorded, you can issue appropriate JES2 $J commands. (The dollar sign ($), is your specific command prefix as assigned by the CONCHAR= parameter on the CONDEF statement and changeable as appropriate for your instance of JES2.) However, the J is unique to this set of commands and distinguishes them so they receive special JES2 processing. These commands are routed immediately to the monitor command subtask, and except for SAF Exits 36 and 37, bypass exit processing, including Exit 5, the command pre-processor routine. This can cause $J commands to be processed out of order from other JES2 commands issued by the same source.

Special command processing

$J commands cannot be issued from NJE, RJE, internal reader, or the JES2 initialization stream; they are restricted to the SVC34 subsystem interface (SSI) and as automatic commands. However, you can issue a $J command as the object of a $VS command, typically used to enter MVS system commands through JES2.

Because of the critical nature assigned to $J commands, JES2:

- Does not restrict several limits set by the CONDEF initialization statement as follows:
  - CMDNUM= the number queued commands
  - DISPMAX= the output produced by the command
  - RDIRAREA= the default out-of-line area
- Does not support the use of the semicolon (;) as a command separator. Only one $J command is allowed for each SVC 34 request
- Ignores spaces and comments (/* comment */) as in typical processing
- Support L= is supported to direct responses to out-of-line area of a specific console, although RDIRAREA= is ignored
- Calls RACF for all commands, similar to other JES2 commands noting that the first qualifier is determined by the monitor address space name. For example, *jesx*MON.*action.object* where jesx is the monitored subsystem name

$J (JES2 health monitor) commands

See *z/OS JES2 Commands* for complete syntax, descriptions, and example output returned for the following JES2 health monitor commands. Each is listed here with an overview of the data it reports and its function.

- $J D MONITOR - display the status of the health monitor
- $J STOP - halt the monitor, and JES2 will automatically restart it in several minutes. This commands corrects errors the monitor is having and clears the processing history it is currently holding. A hot start is required if the monitor code has been refreshed and you need JES2 to recognize those changes
- $J D STATUS - the primary command used to determine what problems JES2 is experiencing. This commands reports:
  - Alerts for which JES2 has already issued a message
  - Notice (or tracking) conditions which might be contributing to a problem, but not surfaced by JES2 with an alert message
- $J D JES - although similar to $J D STATUS, this command displays information JES2 is tracking that is not currently a problem, but based on duration might become an alert condition over time
- $J D DETAILS - displays a consolidated view of resource usage although also tracked by the $HASP050. This is particularly useful when JES2 commands can't be processed because of a serious CMB shortage, for example.

- $J D HISTORY - display a complete history (up to 72 hours) of data the monitor has collected, to include low, high, and average usage statistics.

Appendix C. The external writer

This chapter 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 400) 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. A large format spool data set can be shared.
- It supports basic format, extended format and large format sequential data sets on DASD.

- It supports checkpoint and spool on 3390 Model 9 and Enterprise Storage Server (ESS).
- 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,,'
//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='cxxxxxxxx[,seprname][,CERTIFY=Y|N]]'
[PARM='cxxxxxxxx[,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.

xxxxxxxx
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 LNKLSTxx 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.

Note:

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 407 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. See "Functions of the Output Writer Routine" on page 404 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=anyname,                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=anyname
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 84. External Writer Parameter List

Byte	Meaning
0	Used to describe the type of output device **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:

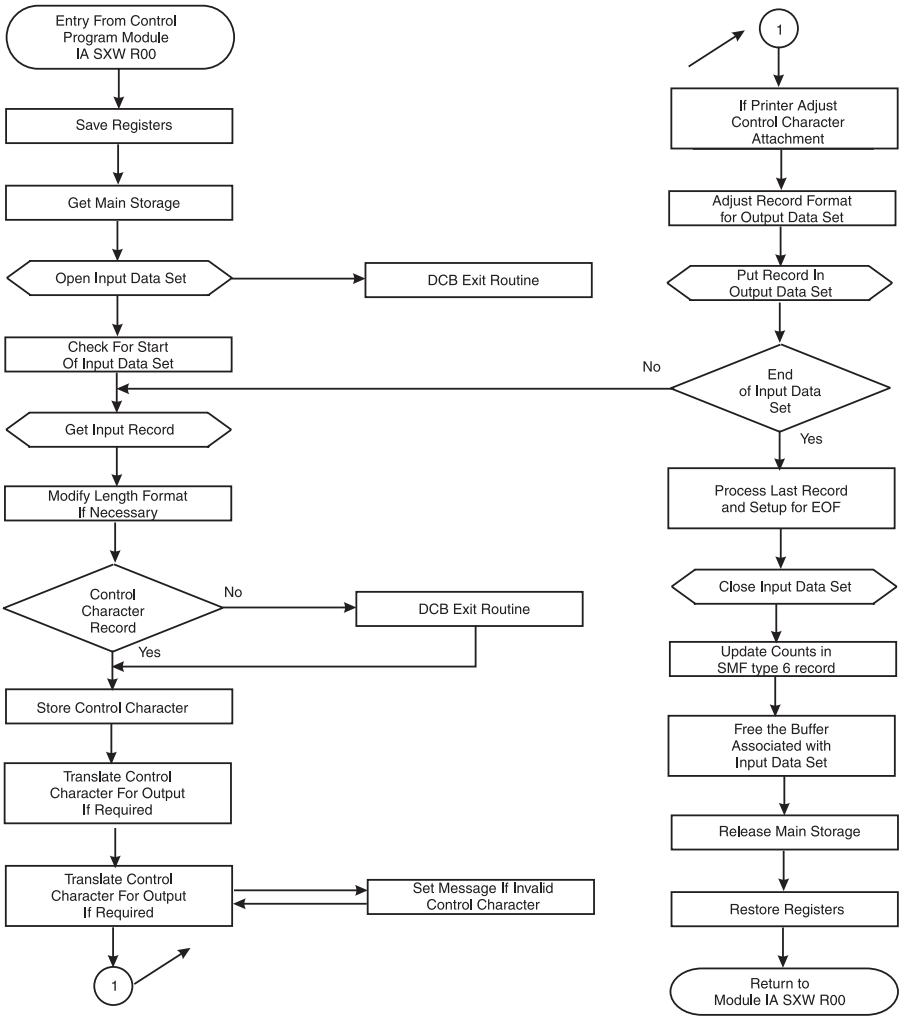


Figure 95. General Flow of IBM's Output Writing Routine

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 and 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 LNKLSTxx 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' (*x*37) 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 85. Separator Routine Parameter List

Byte	Meaning
0	Used to describe the type of output device **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 412.) 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 401 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 86. 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 D. Accessibility

Accessible publications for this product are offered through the z/OS Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrcfs@us.ibm.com or to the following mailing address:

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

Accessibility features

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 such products 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 Vol 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.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually

exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

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

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

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1!

(KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Note:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

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
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 PUBLICATION “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 publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication 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:

Site Counsel
IBM Corporation
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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface Information

JES2 Initialization and Tuning Guide primarily documents information that is NOT intended to be used as Programming Interfaces of z/OS.

JES2 Initialization and Tuning Guide 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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml (http://www.ibm.com/legal/copytrade.shtml).

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Product Number: 5650-ZOS

Printed in USA

SA32-0991-00