z/OS

JES2 Introduction

Version 2 Release 1

SA32-0994-00

z/OS

JES2 Introduction

Version 2 Release 1

SA32-0994-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 37.

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

Figures

About this document

This document provides an introduction to the job entry subsystem 2 (JES2). It is meant to provide an overview of JES2; it is not meant to be an instructional manual. This document is specifically designed for installations running z/OS. This document presents:

- JES2 as a unified set of related functions whose purpose is to accomplish certain data processing goals for MVS
- JES2 and its relationship to MVS, Time Sharing Option Extensions (TSO/E), and components of the MVS operating system
- An overview of JES2 functions, device control, network job entry (NJE), remote job entry (RJE), initialization, operations, customization techniques, and diagnostic tools.

Who should read this document

This document is intended for the MVS system programmer, operator, data processing manager, or student who is unfamiliar with JES2 and its functional relationship to the base control program of MVS.

This document is specifically designed for installations that are evaluating or running z/OS JES2. However, much of the function described is available in previous versions of the JES2 component in MVS/System Product Version 1, Version 2, Version 3, Version 4, and Version 5. Therefore, most of the material presented here applies to JES2 at those version/release levels because of the overview-level of detail. The document, however, makes no distinction between functions and in what version/release they were added to JES2.

How to use this document

To obtain the most from this document, you should read it sequentially. The information presented in each chapter and topic is built upon previous information. If you are unfamiliar with JES2 concepts and terminology, you will find that reading the document in this manner is most helpful.

Additional Information

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

Title	Order Number	Description
z/OS Introduction and Release Guide	GA32-0887	Describes the contents and benefits of z/OS as well as the planned packaging and delivery of this new product.
z/OS Planning for Installation	GA32-0890	Contains information that lets users: • Understand the content of z/OS • Plan to get z/OS up and running • Install the code • Take the appropriate migration actions • Test the z/OS system

Title	Order Number	Description
z/OS Information Roadmap	SA23-2299	Describes the information associated with z/OS including z/OS documents and documents for the participating elements.
z/OS Summary of Message and Interface Changes	SA23-2300	Describes the changes to messages for individual elements of z/OS. **Note:** This document is provided in softcopy only on the message bookshelf of the z/OS collection kit.

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 Introduction
  SA32-0994-00
- 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/).

Chapter 1. Introduction to JES2

This introduction answers the following questions:

- What is a JES?
- Why is JES required in an MVS™ system?
- What is the relationship between JES and the system users?
- Is there really any difference between JES2 and JES3?
- What are some of JES2's key functions?

An initial consideration when installing MVS is whether to select the JES2 or the JES3 component. This introduction provides some background if you are new to MVS, followed by a more detailed discussion of JES2. If you require further insight into the functional differences between JES2 and JES3, or are deciding which to install, you should also read *z/OS JES3 Introduction* which provides a similar overview of JES3.

What is a JES?

MVS uses a job entry subsystem (JES) to receive jobs into the operating system, schedule jobs for processing by MVS, and control job output processing. JES2 (job entry subsystem 2) is descended from HASP (Houston automatic spooling priority), which is defined as *a computer program that provides supplementary job management, data management, and task management functions such as: scheduling, control of job flow, and spooling*. HASP persists within JES2 as the prefix string for most module names and for all messages sent by JES2 to the operator.

JES2 is *a functional extension of the HASP II program that receives jobs into the system and processes all output data produced by the job*. JES2 is the component of MVS that provides the necessary functions to get jobs into, and output out of, the MVS system. It is designed to provide efficient spooling, scheduling, and management facilities for the MVS operating system. (See "Spool data sets and spooling" on page 7 for a full definition of spooling.)

By separating job processing into a number of tasks, MVS operates more efficiently. At any point in time, the computer system resources are busy processing the tasks for individual jobs, while other tasks are waiting for those resources to become available. In its most simple view, MVS divides the management of jobs and resources between the JES and the base control program of MVS. In this manner, JES2 manages jobs before and after running the program; the base control program manages them during processing. [1] Figure 1 on page 2 presents a diagram of the relationship between JES2 and MVS.

1. JES3, in contrast, manages jobs throughout the entire process cycle (before, during, and after running the programs).

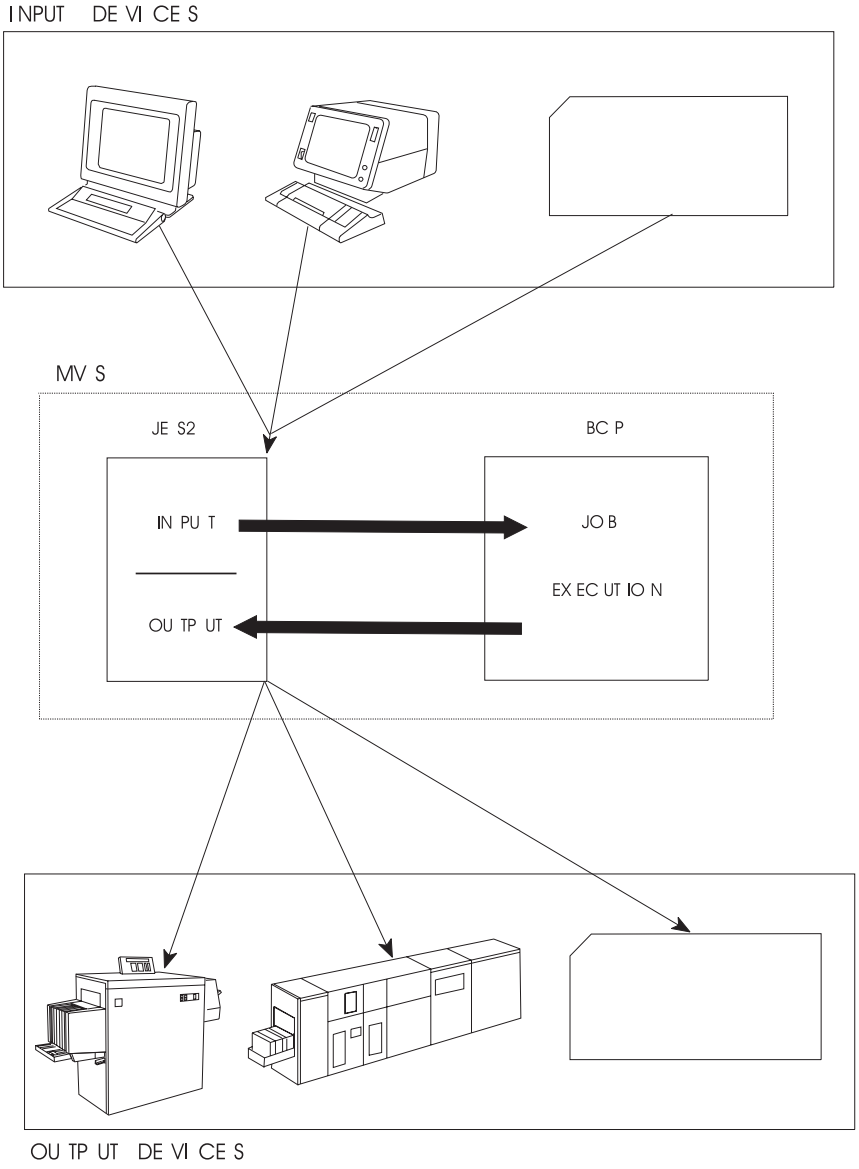

Figure 1. Relationship of JES2 to the Base Control Program

How JES2 fits into the z/OS system

During the life of a job, both JES2 and the base control program of z/OS® control different phases of the overall processing. Most jobs go through the following phases:

1. Input
2. Conversion
3. Processing
4. Output
5. Print/punch (hard copy)
6. Purge

With the exception of the processing phase, all job phases are controlled by JES2.

These phases are explained in more detail in Chapter 3, "JES2 job processing and functions," on page 15.

Relationship of JES2 to JCL and submitted jobs

JES2 initialization statements give the system programmer a single point of control to define an installation's policies regarding the degree of control users have over their jobs. If users were allowed to define their own job classes and priorities, users would tend to define their own jobs as top priority, resulting in no effective priority classifications. Although a user can use job control language (JCL) options to define a priority, a JES2 initialization statement (defined by the system programmer) determines whether JES2 acknowledges that priority.

This same type of control (validation of user specification and provision for a default) extends to many JCL definitions. For example, JES2 can specify the maximum time a particular job is allowed to run, the storage a job may consume, the number of copies of output a job can print, and the type of paper (form) on which the output prints.

Users can override system and JES2 specifications at three distinct levels: user-specified JCL, installation-specified JCL, and JCL defaults. The hierarchy of control is as follows, from highest to lowest priority:

1. JCL specification on a user job.

   This user JCL overrides:
2. JCL default, which z/OS uses if there is no user definition or the user specification is disallowed.

   This JCL default overrides:
3. JES2 default, which z/OS uses if:
   a. No JCL default exists
   b. The user specification is disallowed or undefined
   c. The JCL default definition is not supported.

Note that the user can override the JCL defaults and the JCL defaults can override the JES2 specifications, but the ability to override is either permitted or disallowed by specifications that only the system programmer can control. This structure thereby puts the system control in the hands of the system programmer, not the individual user who is submitting the job. JES2 becomes the base for input and output specifications that can then be overridden, as allowed by your installation, through the JCL and job submitter.

JES2 compared to JES3

z/OS provides two job entry subsystems from which to choose: JES2 and JES3. For a system that has only a single processor, JES2 and JES3 perform similar functions: they read jobs into the system, convert jobs to internal machine-readable form, select jobs for processing, process output, and purge jobs from the system. However, for a system that has multiple processors, there are noticeable differences in how JES2 exercises **independent control** over its job processing functions; each JES2 processor controls its own job input, job scheduling, and job output processing. In contrast, JES3 exercises **centralized control** over its processing functions through a single **global** JES3 processor. This global processor provides all job selection, scheduling, and device allocation functions for all other JES3 systems. The centralized control that JES3 exercises provides increased job scheduling control, deadline scheduling capabilities, and increased control by providing its

own device allocation. To gain a more complete understanding of the functional differences between JES2 and JES3, see *z/OS JES2 Initialization and Tuning Guide* and *z/OS JES3 Initialization and Tuning Guide*.

Overview of JES2

The remainder of this document describes JES2 processing configurations, functional capabilities, customization, and messaging and diagnostics. Each topic is introduced briefly here.

JES2 configurations

You can run JES2 in a variety of processing configurations that range from a single base control program with a single JES2 that is completely isolated from other processing systems, to one that is a part of a worldwide processing network. The choice of configuration complexity is yours and generally is a dynamic one that grows as your business needs grow. The following options are basic JES2 configurations:

- Single-processor
- Multiple-system (multi-access spool)
- Poly-JES
- Remote job entry (RJE)
- Network job entry (NJE)

These JES2 configuration options are discussed in detail in Chapter 2, “Scope of control and configurations,” on page 7.

JES2 functions

To manage job input/output responsibilities for z/OS, JES2 controls a number of functional areas, all of which you can customize. Some of the major functional areas and processing capabilities are:

- Getting work into and out of MVS (input/output control)
- Maximizing efficiency through job selection and scheduling
- Offloading work and backing up system workload
- Supporting advanced function printers (AFPs)
- Running multiple copies of JES2
- System security.

Each of these is discussed in Chapter 3, “JES2 job processing and functions,” on page 15.

Customizing JES2

JES2 has the flexibility to fulfill an installation's unique processing needs. An installation can virtually control every JES2 function. You can perform many customization tasks when JES2 is installed, and you can dynamically customize JES2 whenever your processing needs change. JES2 provides initialization statements, commands, IBM-supplied exit points, the ability to add installation-defined exit points that require minimal source code modification, and the ability to change many commands and system messages, all without the need to modify the IBM-supplied code. See Chapter 4, “Customizing your JES2 system,” on page 23 for a more complete description of these customization facilities.

Interacting with JES2

No large data processing system or subsystem can continually operate independently of system programmer or operator intervention. A two-way communication mechanism between the operator and JES2 is required. Based on system workload and device requirements, JES2 must communicate its status to the operator, system programmer, or both. JES2 issues messages to communicate job and device status, problem situations, system resource constraint situations, and performance status. Through commands, you can request current status and through the use of various tools you can obtain further information to diagnose and correct problem and system failure situations. Chapter 5, "Interacting with JES2," on page 27 provides an overview of these topics.

Chapter 2. Scope of control and configurations

This information answers the following questions:

- How does JES2 manage its work?
- Where does JES2 store its data, and how does JES2 provide data integrity?
  - What is spooling?
  - What is checkpointing?
- What processing configurations does JES2 support?
  - What is a multi-access spool complex?
  - What is poly-JES?
  - What is remote job entry?
  - What is network job entry?

The control that JES2 has over certain data sets and devices allows z/OS to offload work to JES2. The means by which JES2 maintains these data sets is unique to JES2. Furthermore, the configurations in which JES2 operates range from simple to rather complex. This chapter provides a basic understanding of the JES2 operating configurations.

JES2 data set control

JES2 maintains copies of its data sets that contain job and output queues on direct access storage devices (DASD). Future work is added to these queues and JES2 selects work for processing from them. These data sets and queues must remain current and accurate to maintain system integrity and performance.

Spool data sets and spooling

Simultaneous peripheral operations online (spool) has several meanings as used in this book and throughout JES2 documentation. **Spooling** is a process by which the system manipulates its work. This includes:

- Using storage on direct access storage devices (DASD) as a buffer storage to reduce processing delays when transferring data between peripheral equipment and a program to be run.
- Reading and writing input and output streams on an intermediate device for later processing or output.
- Performing an operation such as printing while the computer is busy with other work.

Spool also refers to the direct access device that contains the spool data sets. This definition is generally apparent from the context of its use and should not cause any misunderstanding in the JES2 documentation.

Spooling is critical to maintaining performance in a z/OS-JES2 environment. JES2 attempts to maximize the performance of spooling operations, which ultimately benefits the throughput of work through the JES2 installation. As noted previously, spooling provides simultaneous processing and a temporary storage area for work that is not yet completed. When JES2 reads a job into the system, JES2 writes the job, its JCL, its control statements, and its data to a spool data set until further processing can occur.

Maintaining integrity

Errors can occur while processing jobs and data. Some of these errors can result in the halting of all system activity. Other errors can result in the loss of jobs or the invalidation of jobs and data. If such errors occur, it is preferable to stop JES2 in such a way that allows processing to be restarted with minimal loss of jobs and data. The **checkpoint data set**, **checkpointing**, and the **checkpoint reconfiguration dialog** all help to make this possible.

Checkpoint data set is the general term used to describe the checkpoint data set that JES2 maintains on either DASD or a coupling facility. The checkpoint data set (regardless of whether it resides on a coupling facility structure or a DASD volume) contains a backup copy of the job and output queues, which contain information about what work is yet to be processed and how far along that work has progressed. Similar to the spool data sets, the checkpoint data set is commonly accessible by all members of the multiple-processor (multi-access) spool complex, but only one member will have control (access) of the checkpoint data set at any one time. Furthermore, the checkpoint data set provides communication among all members of the configuration about jobs and the output from those jobs. JES2 periodically updates the checkpoint data set by copying the changed information from the in-storage copy to the checkpoint data set copy that resides on either a coupling facility structure or on DASD. Information in the checkpoint data set is critical to JES2 for normal processing and in the event that a JES2 or system failure occurs.

Checkpointing is the concept of keeping a copy of the checkpoint data set that contains essential workload information on either a coupling facility structure or a DASD volume. This copy is updated from the in-storage copy of the checkpoint as each JES2 member of a multi-access spool updates the in-storage checkpoint data set. **Checkpointing** allows JES2 to be stopped and then restarted with little or no loss of job or data integrity.

The **checkpoint reconfiguration dialog** is a dynamic means by which the current checkpoint configuration can be changed (for example, adding a checkpoint device or moving the checkpoint data set to a different device). It is possible to enter the checkpoint reconfiguration dialog to continue processing without necessitating a JES2 restart. This process increases system availability.

JES2 configurations

The size and complexity of your data processing configuration is based on your business needs. Many factors contribute to your configuration requirements, including the following variables:

- Number of concurrent interactive users
- Size of your data bases
- Number of customers
- Geographic location of users and resources
- Number of users that run jobs, access data bases, and use programs
- Total number of jobs.

JES2 provides two distinct facilities for the expansion of your JES2 processing configuration: remote job entry and network job entry. **Remote job entry** allows for the extension of your local processing configuration by defining remote locations that can consist of job input terminals and output devices at a different physical site connected to the z/OS-JES2 processor through telecommunication links such as

telephone lines and satellites. In this manner, a z/OS-JES2 system can provide input and output support across various geographic locations. But all the remote sites and their attached devices are defined to a single z/OS-JES2 configuration. **Network job entry** takes this concept further by allowing individual z/OS–JES2 processors, that are geographically dispersed, to be connected in a network of JES2 and non-JES2 systems that can communicate, pass jobs, and route output to any attached output devices.

JES2 allows you to take advantage of the power of z/OS in:

- A single-system complex (one processor and one JES2)
- A multiple-system complex (up to 32 processors, each with its own JES2)
- A poly-JES configuration (more than one JES2 operating concurrently within a single z/OS)
- Remote job entry (RJE) workstations (remotely attached to a processor in the configuration)
- A network job entry complex (the linking of two or more single-system or multi-system configurations).

Single-system configuration

A JES2 **configuration** can contain as few as one processor (one z/OS and JES2 system) or as many as 32 processors linked together. A single processor is referred to as a **single-system configuration**, which is suitable for a configuration with a small work load, or one that requires a single processor to be isolated from the remainder of the data processing complex.

Multiple-system configuration

Many configurations take advantage of JES2's ability to link processors together to form a multiple-processor complex, which is generally referred to as a multi-access spool (MAS) configuration. A **multi-access spool configuration** consists of two or more JES2 (MAS) processors at the same physical location, all sharing the same spool and checkpoint data sets. There is no direct connection between the processors; the shared direct access data sets provide the communication link. Each JES2 processor can read jobs from local and remote card readers, select jobs for processing, print and punch results on local and remote output devices, and communicate with the operator. Each JES2 processor in a multiple processor complex operates independently of the other JES2 processors in the configuration.

The JES2 processors share a common job queue and a common output queue, which reside on the checkpoint data sets. These common queues enable each JES2 processor to share in processing the installation's workload; jobs can run on whatever processor is available and print or punch output on whatever processor has an available device with the proper requirements. Users can also specifically request jobs to run on a particular processor and output to print or punch on a specific device. If one processor in the configuration fails, the others can continue processing by selecting work from the shared queues and optionally take over for the failed processor. Only work in process on the failed processor is interrupted; the other JES2 systems continue processing.

Running multiple copies of JES2 (poly-JES)

z/OS supports multiple JES2 subsystems operating concurrently, with one subsystem designated as the primary subsystem and all others defined as secondary subsystems. A secondary JES2 does not have the same capabilities as the primary JES2 and some restrictions apply to its use. Most notably, TSO/E users can

only access the primary subsystem. The restrictions are necessary to maintain the isolation from the z/OS-JES2 production system, but the convenience for testing is a valuable function. Operation of multiple copies of JES2 is referred to as **poly-JES**. Secondary JES2s can be useful in testing a new release or installation-written exit routines. This isolation prevents potential disruption to the primary JES2 and base control program necessary for normal installation production work.

JES2 remote job entry (RJE)

The **remote job entry** (RJE) facility allows JES2 to define and use RJE workstations. An **RJE workstation** is a workstation that is connected to a member by means of data transmission facilities. The workstation can be a single I/O device or group of I/O devices or can include a processor such as a System/36, or System/390®. Generally, RJE workstations either include a programmable workstation (such as a personal computer) or a communication terminal (such as a 3770, 2780, or S/360) connected to the z/OS system through a telecommunication link. Such a link utilizes synchronous data link control (SDLC), or binary synchronous communication (BSC) for communicating between JES2 and remote devices. The remote device will be either a system network architecture (SNA) remote, that uses SDLC, or a BSC remote, that uses BSC. Figure 2 shows a simple RJE configuration.

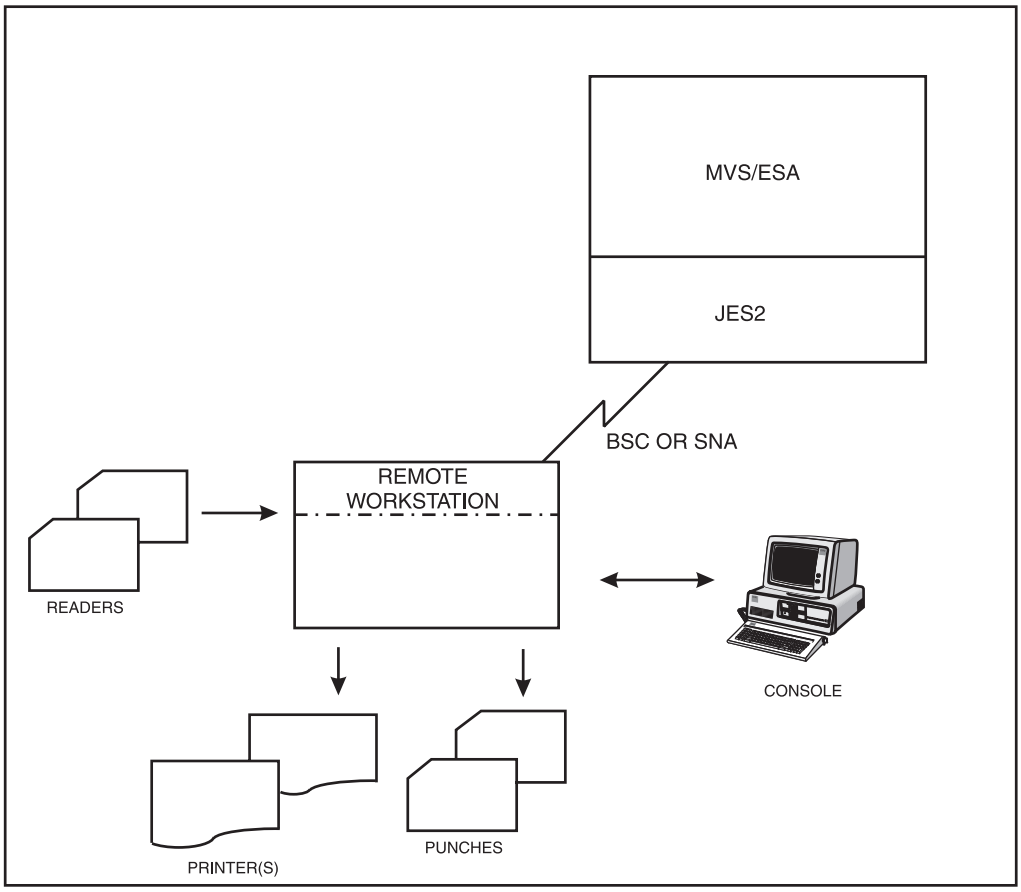

Figure 2. Remote Job Entry Configuration

An RJE workstation is an extension of the local processing facility, except that work is sent across teleprocessing lines. Sending work across teleprocessing lines is convenient for an installation that needs to provide many data entry points at

remote locations or route output to many diverse locations. The following illustrates the use of RJE in two familiar examples of daily business:

- In a large department store, many sales clerks need to print output on printers located at the many customer check-out desks throughout the store. Because the location of the building prevents direct connection to the z/OS system located in the central office located several hundred miles away, each printer can be defined to JES2 as a remote printer.
- Consider a clothing store chain in which the store managers at seven different stores all need to place orders, access inventory, and provide billing information, all the information for which is maintained on storage devices located in the computer center at the main office. An individual store's workstation is defined to JES2 as an SNA-attached remote to which the individual terminals are defined and as such become an extension of the JES2 configuration located at the main office.

Figure 3 presents a diagrammatic view of the RJE configuration as described in the preceding clothing store example.

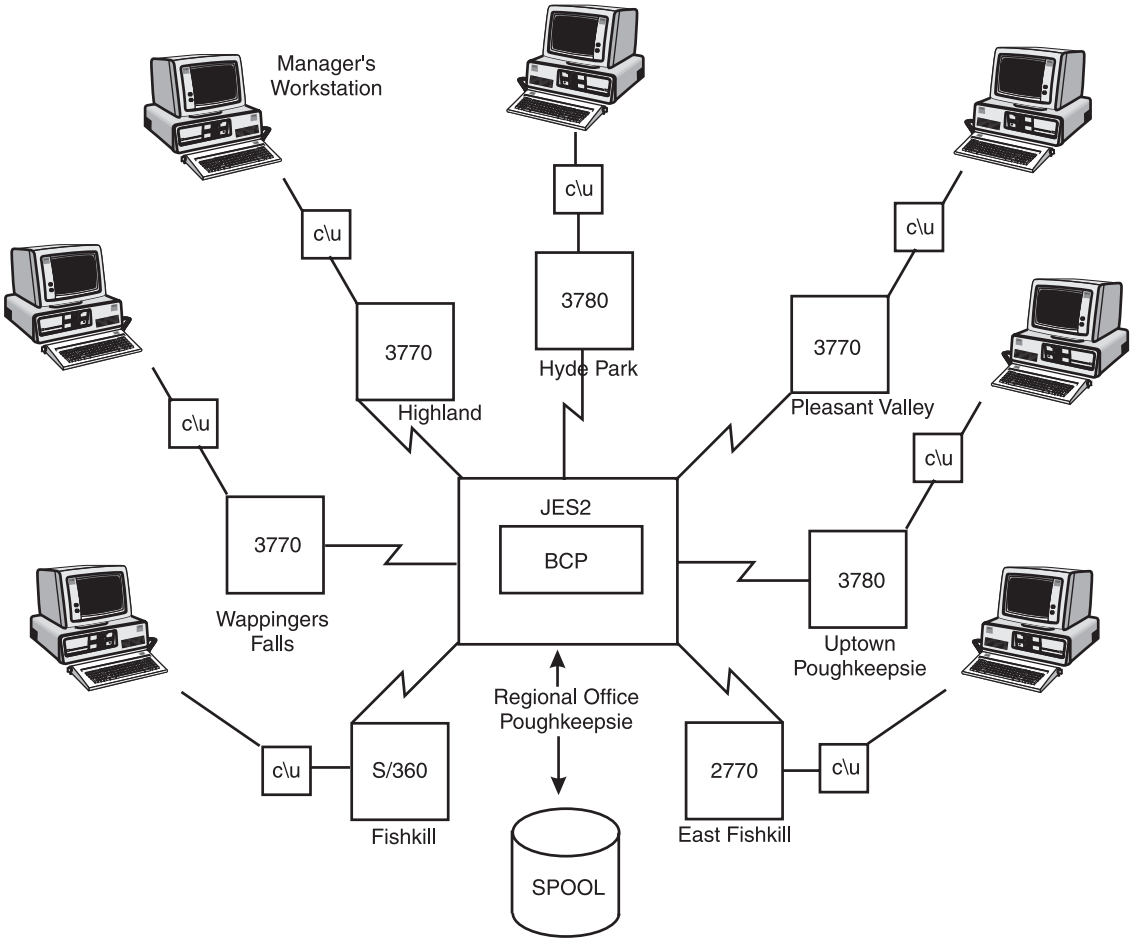

Figure 3. Example of a Remote Job Entry (RJE) Configuration

In Figure 3 each of the clothing stores within a localized region of New York, the Hudson Valley (Highland, Hyde Park, Pleasant Valley, and so forth), is connected to the single processor located at the regional office in Poughkeepsie. One z/OS–JES2 system conducts the business of inventory control, shipping, and billing for all of the stores in the region.

JES2 processes remote jobs in the same manner as those received from a local reader. (**Local devices** are printers, punches, card readers, and lines directly attached to the system without the need for transmission facilities.) The terminals and printers located in the Poughkeepsie Regional Office are **locally attached**; all other I/O devices (terminals and printers) in all the other branch stores are defined to JES2 as **remote** terminals and printers.

To provide RJE processing, the RJE workstation must be defined to the local processor. There are two protocols available by which JES2 can communicate with the RJE workstations: **synchronous data link control** (SDLC), and **binary synchronous communication** (BSC).

An RJE workstation can have a processor, like the System/370, that runs a JES2-generated program. The JES2-generated program allows the processor to send jobs to, and receive data from, JES2. Such RJE workstations have generally been replaced by either a programmable workstation, such as a personal computer or a network job entry configuration, and they are rarely used in today's processing environment. Some RJE workstations do not have a processor. These workstations use a remote terminal, for example, a 2780 or 2770, to enter jobs into, and receive data from, JES2.

See *z/OS JES2 Initialization and Tuning Guide* for a more complete discussion of RJE concepts.

JES2 network job entry (NJE)

The JES2 **network job entry** (NJE) facility is similar to remote job entry (RJE) in that they both provide extensions to a computer system. In its simplest terms, NJE is "networking" between systems that interact as peers, whereas RJE is networking between JES2 and workstations. The main difference between them is one of overall compute power and processor location. Remember, RJE is an extension of a *single* computer system (that is, either a single-processor or multi-access spool complex) that allows jobs to be submitted from, and output routed back to, sites that are remote to the location of that system. NJE provides a capability to link many such single-processor systems or multi-access spool complexes into a processing network. Each system can be located on the same physical processor, side-by-side in a single room, or across the world in a network of thousands of nodes. The important difference is that a processor and its local and remote devices make up a **node**. Three or more attached nodes make up an NJE network.

Furthermore, as discussed in “Multiple-system configuration” on page 9, a node can include as few as one processor with z/OS or as many as 32, all sharing the same spool and checkpoint data sets. NJE network nodes communicate (send and receive data) using various teleprocessing facilities. Nodes on the same physical processor use the Virtual Telecommunications Access Method (ACF/VTAM) program product to communicate (using SNA or TCP/IP);... ; no hardware is required. Nodes located in the same room or building can use channel-to-channel adapters (CTCA) or telecommunication links. Nodes that are geographically dispersed use SNA, BSC, or TCP/IP telecommunication links. The following example further explains this concept.

If we return to the previous example of the clothing store chain, the Poughkeepsie regional office is the location of the processor (z/OS and JES2); each of the sites (Highland, Hyde Park, Pleasant Valley, Fishkill, and so forth) are attached to the Poughkeepsie office as remote sites. Together, all the locations in Figure 3 on page 11 comprise the Poughkeepsie node. For a locally owned, relatively small clothing

store chain such as the one depicted, this complex may suffice. However, for a national clothier or one dealing in the import/export business, one processor would likely prove to be inadequate. Such a company would likely elect to establish many nodes throughout the world, each connected to all others in an NJE complex. See Figure 4 for a diagrammatic view of such a network. Note that the different groups of the ordering/billing department and the business administration/payroll department can be separate members of a MAS making up the New York and London nodes.

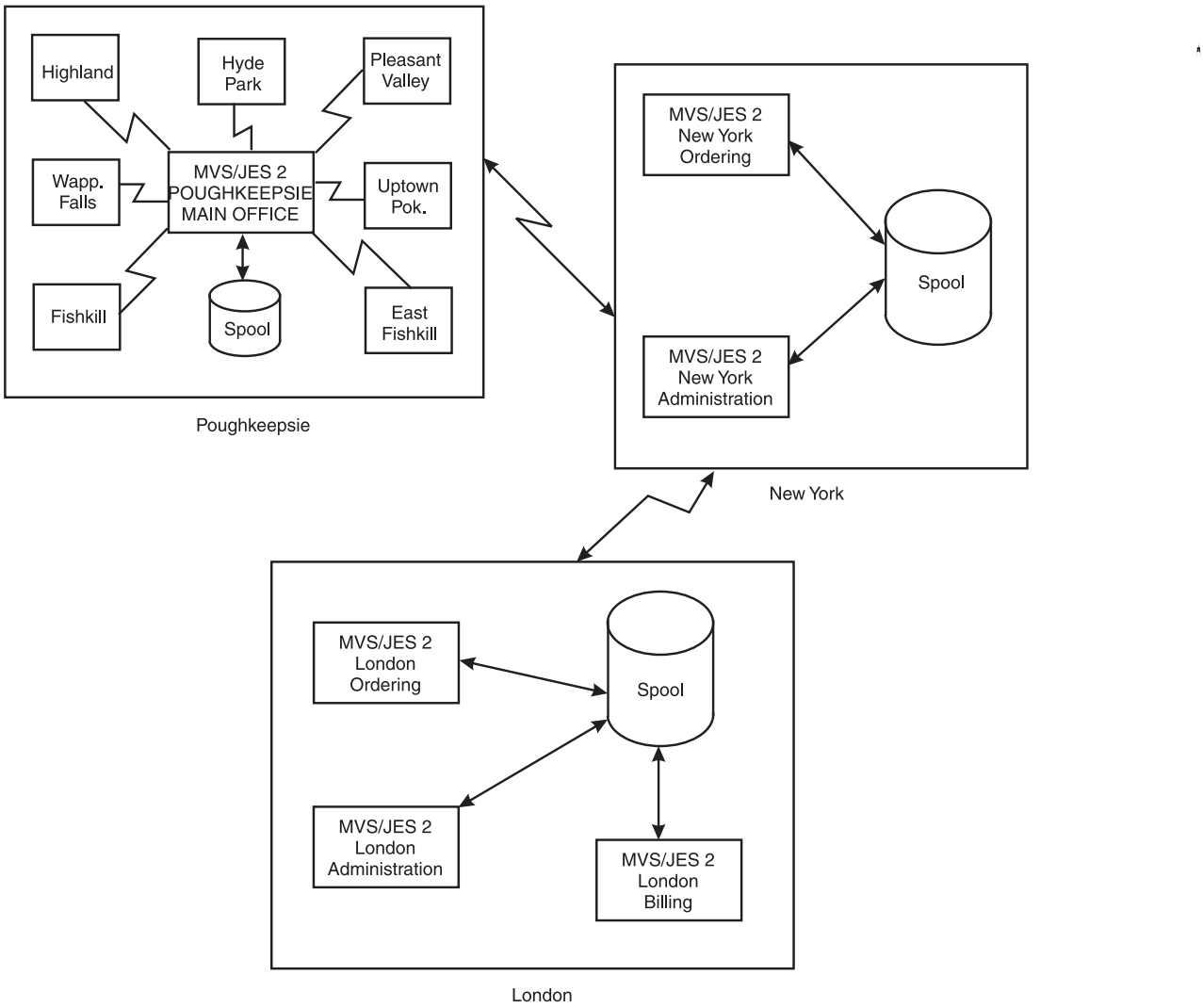

Figure 4. Example of a Network Job Entry (NJE) Configuration

Note that the New York and London nodes are multi-access spool configurations, Poughkeepsie is the configuration as described in Figure 3 on page 11, and all the other nodes located around the world (which are not depicted) are simple single-system configurations. In the network, a store manager in Pleasant Valley can order an item, the Poughkeepsie office (the main office for the Hudson Valley region) tallies the orders from this store and its other six stores and submits its order to the clothier chain headquarters located in New York City. The order is then forwarded to the London export office, where the item is procured and shipped. The request can be routed through Poughkeepsie and New York to London for accounting purposes or it could have been sent directly to London if the business is organized in that manner. The request and subsequent confirmation

of the order is not instantaneous because of the distance and the **traffic** in the system but is faster and a more efficient method of doing business than a telephone conversation, particularly if you further consider time zone differences. Traffic refers to the number of users, requests, jobs, and data currently being routed across available teleprocessing lines.

A JES2 network can include individual node types other than z/OS-JES2 nodes. As Figure 5 illustrates, each node can contain different processing systems; nodes with various levels of z/OS-JES2, z/OS-JES3, VM/RSCS, and VSE/POWER can all be linked in a network. This is true because the NJE formats and protocols used by each of these systems are, by design, release and product independent.

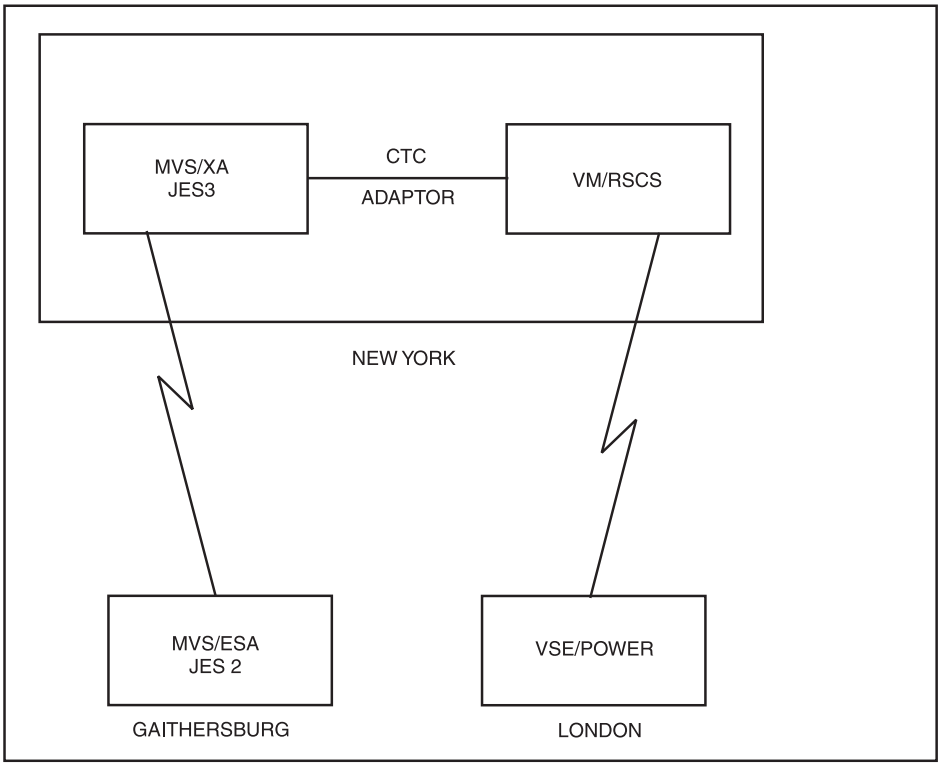

Figure 5. A Network of Various Processing Systems

Chapter 3. JES2 job processing and functions

This chapter answers the following questions:

- During the life of a job, what processing does JES2 do for z/OS?
- What are some of the major JES2 functions?

JES2 is responsible for all phases of job processing and monitors the processing phase. This chapter outlines the six phases and presents an overview of some of the major JES2 functions.

Phases of job processing

The z/OS base control program and JES2 share responsibility in a System z® system. JES2 is responsible for job entry (input), the base control program for device allocation and actual job running, and finally JES2 for job exit (output).

Figure 6 presents a view of the six phases, followed by a description of each.

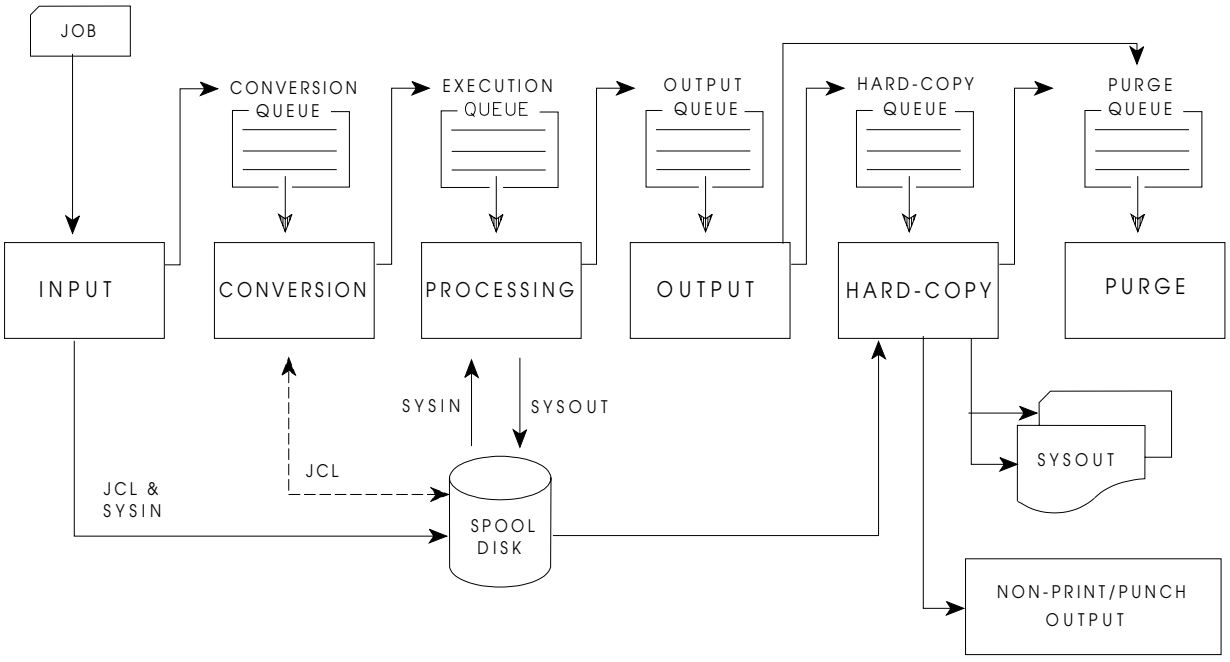

Figure 6. Job Processing Phases

The job queues contain jobs:

- Waiting to run - conversion queue
- Currently running - execution queue
- Waiting for their output to be produced - output queue
- Having their output produced - hard-copy (print/punch) queue
- Waiting to be purged from the system (following completion of all processing) - purge queue.

Input phase

JES2 accepts jobs (in the form of an input stream) from input devices such as card readers, remote terminals, or other programs. Input streams can also come from other nodes in a job entry network and from internal readers. An **internal reader** is a program that other programs can use to submit jobs, control statements, and send commands to JES2. Any job running in z/OS can use an internal reader to pass an input stream to JES2, and JES2 can receive multiple jobs simultaneously through multiple internal readers.

z/OS uses internal readers, allocated during system initialization, to pass to JES2 the job control language (JCL) for started tasks, START and MOUNT commands, and TSO LOGON requests.

The system programmer defines internal readers used to process all batch jobs other than STCs and TSO requests. JES2 initialization statements define these internal readers which JES2 also allocates during its initialization processing. The internal readers for batch jobs can be used for STCs and TSO requests, if not processing jobs.

As JES2 reads the input stream, it assigns a job identifier to each job and places each job's JCL, optional JES2 control statements, and SYSIN data onto DASD data sets called spool data sets. JES2 then selects jobs from the spool data sets for processing and subsequent running. See Chapter 2, “Scope of control and configurations,” on page 7 for an explanation of spool data sets and the concept of spooling.

Conversion phase

JES2 uses a converter program to analyze each job's JCL statements. The converter takes the job's JCL, merges it with JCL from a procedure library (such as SYS1.PROCLIB), and converts the composite JCL into converter/interpreter text that both JES2 and the job scheduler functions of z/OS can recognize. JES2 then stores the converter/interpreter text on the spool data set. If JES2 detects any JCL errors, JES2 issues messages, and the job is queued for output processing rather than run. If there are no errors, JES2 queues the job for execution. JES2 supports multiple converters; therefore, jobs may not always be processed in a first-in-first-out (FIFO) order. When work load manager (WLM) batch management is in use, JES2 queues the job according to its arrival time.

Processing phase

In the processing phase, JES2 responds to requests for jobs from the z/OS initiators. JES2 selects from a job queue, jobs that are waiting to run and sends them to z/OS.

By recognizing the current processing phase of all jobs on the job queue, JES2 can manage the flow of jobs through the system.

JES2 Job Scheduling: To process the jobs on the job queue, JES2 communicates with an initiator. An **initiator** is a system program that starts a job to allow it to compete for system resources with other jobs that are already running.

Initiators are controlled by JES2 or by z/OS workload management (WLM).

- JES2 initiators are started by the operator or by JES2 automatically when the system initializes. The initiators select jobs based on the job class(es) that are assigned to the initiator and the priority of the queued jobs. The system

associates each initiator with one or more job classes in a way to encourage the efficient use of available system resources.

- WLM initiators are started by the system automatically based on the performance goals, relative importance of the batch workload, and the capacity of the system to do more work. The initiators select jobs based on their service class and the order in which they were made available for execution.

After JES2 selects a job and passes it to the initiator, the initiator invokes the interpreter to build control blocks from the converter/interpreter text that the converter created for the job.

The initiator then allocates the resources specified in the JCL for the first step of the job. This allocation ensures that the devices are available before the job step starts running. The initiator then starts the program requested in the JCL EXEC statement.

Priority Aging: When all initiators are busy, throughput of certain jobs might fall below normal expectations. To help in these situations, JES2 uses the additional scheduling function of priority aging. **Priority aging** can help ensure that jobs that have been waiting to run have a chance of being selected to run before those jobs that just entered the system. By using priority aging, a system can increase the priority of a waiting job. The longer the job waits, the higher its priority becomes, up to a limit, and the greater its chances of being selected to run.

JES2-Base Control Program Interaction: JES2 and the base control program communicate constantly to control system processing. The communication mechanism, known as the subsystem interface, allows z/OS to request services of JES2. For example, a requester can ask JES2 to find a job, do message or command processing, or open (access) a SYSIN or SYSOUT data set. Further, the base control program notifies JES2 of events such as messages, operator commands, the end of a job, or the end of a task.

Output phase

JES2 controls all SYSOUT processing. **SYSOUT** is system-produced output; that is, all output produced by, or for, a job. This output includes system messages that must be printed and data sets requested by the user that must be printed or punched. After a job finishes, JES2 analyzes the characteristics of the job's output in terms of its output class and device setup requirements; then JES2 groups data sets with similar characteristics. JES2 queues the output for print or punch processing.

Hard-copy phase

JES2 selects output for processing from the output queues by output class, route code, priority, and other criteria. The output queue can have output that is to be processed locally or output to be processed at a remote location (either an RJE workstation or another node). JES2 handles each of these situations in different ways:

- Local Output:

  When output is to be processed at a local or remotely-attached output device, JES2 uses these local and remotely-attached output devices to produce a job's output. JES2 queues a job's print and punch data sets on the output queue for the local and remote output devices. The active devices, that are attached locally or through RJE connections, select the output data sets with characteristics that best match their selection criteria.

- Network Job Entry Output:

Job output passing through to another JES2 node resides on the network output queue. JES2 selects a job's output from the network output queue for transmission to another node based upon the priority and the desirability of reaching the output-processing node over the available transmission line. After the receiving node signals that it has accepted total responsibility for the output, the transmitting JES2 node releases the resources used to represent the output.

After processing all the output for a particular job, JES2 puts the job on the purge queue.

Purge phase

When all processing for a job completes, JES2 releases the spool space assigned to the job, making the space available for allocation to subsequent jobs. JES2 then issues a message to the operator indicating that the job has been purged from the system.

JES2 capabilities and functions

JES2 (in conjunction with VTAM®) is the link between the Time Sharing Options/Extensions (TSO/E) user and z/OS. As such, it is very externally oriented–it is visible to the data processing personnel and provides the ability to specify and tailor many installation-specific functions through JES2 initialization statements and JES2 commands.

This section provides an overview of the following major functions JES2 provides to manage its job input/output responsibilities for z/OS; all of which are under system programmer control:

- Getting work out of z/OS
- Selecting work to maximize efficiency
- Offloading work and backing up the system
- Supporting advanced function printers
- Providing security.

All are discussed in greater depth in *z/OS JES2 Initialization and Tuning Guide*.

Getting work out of z/OS

As the central point of control over the job output (or exit) function, JES2 controls output devices: local and remote printers, punches, and card readers. You can use JES2 initialization statements to define each device. All are directly under JES2 control, with the exception of printers that operate under the Print Services Facility™ (PSF). PSF printers provide all-points-addressable capabilities, and although defined by JES2, are driven by PSF. PSF thereby assumes the processing overhead that JES2 typically performs to support printer operation. (See “Supporting advanced function presentation (AFP) printers” on page 20for an introduction to PSF.)

Printed and punched output can be routed to a variety of devices in multiple locations. The control JES2 exercises over its printers ranges from the job output classes and job names from which the printer can select work to such specifications such as the forms on which the output is printed. This control allows the system programmer to establish the job output environment most efficiently without causing unnecessary printer backlog or operator intervention.

Through JES2, the system defines the job input classes, reader specifications, and output device specifications. As a result, JES2 is the central point of control over both the *job entry* and *job exit* phases of data processing.

Selecting work to maximize efficiency

To minimize contention for output devices, JES2 allows the installation to define work selection criteria that can be specific for each output device (local and remote printers and punches and offload devices). The work selection criteria on the respective device initialization statements create the following definitions:

- Specification of job and output characteristics that JES2 considers when selecting work for an output device
- Order of importance (priority) of the selection characteristics
- Characteristics of the printer and job that must match exactly.

JES2 initializes the setup characteristics of a device based on the specifications supplied on a device (printer/punch) initialization statement.

A job's output is grouped based on the data sets' output requirements. These requirements are defined by the job submitter with the job's job control language (JCL) or by JES2-supplied defaults.

When selecting work to be processed on a device, JES2 compares the device's characteristics to the output requirements associated with the pieces of work awaiting processing. In addition, JES2 compares each piece of work against all others to determine the best match for the device. If work output requirements are found that match a device's characteristics, JES2 sends the output associated with that piece of work to that output device. If a piece of work and an output device cannot be paired, the work will not be selected for output until the operator changes the output device specifications or the output requirements for the piece of work.

Work selection control provides efficient use of output devices by allowing them to print or punch specified output classes without requiring continual operator intervention of setup characteristics such as forms or print trains (the piece of hardware that carries the print type). Also, when too much work is adversely affecting system performance, the system programmer or operator can specify work to be **offloaded** (that is, moved off the work queues and temporarily out of contention for system resources. See “Offloading work and backing up the system” for an explanation of offloading.)

Offloading work and backing up the system

All input jobs and system output are stored on spool. JES2 supports offloading data from and later reloading data to the spool, which is useful for the following tasks:

- Preserving jobs and SYSOUT across a cold start, which entails the total rebuilding of work and output queues.
- Migrating your installation to another release of JES2. (You can use spool offload to reload the spool to the new or previous release.)
- Converting to another DASD type for spool.
- Archiving system jobs and SYSOUT.
- Relieving a full-spool condition during high-use periods. (You can reload at a later time.)
- Providing a backup for spool data sets.

- Backing up network connections.

Many selection criteria can be used to limit the scope of the spool offload operation. For example, work can be selected based on the system the job will run on, job number, job disposition (for example, only held data sets), destination, class, or many other criteria. These selection criteria can be changed by operator command following your initial specification with JES2 initialization statements.

Supporting advanced function presentation (AFP) printers

Advanced function presentation (AFPs) printers, such as the 3800-3 and 3820 provide all-points addressability. All-points addressability allows every point on the page to be available for reference. These points on the page include graphics and a range of font types and sizes unavailable to impact printers. AFP printers are controlled by print service facility (PSF) and used by JES2. JES2 and PSF work together by communicating through the functional subsystem interface, which is an extension of JES2 device control that removes the need for JES2 to be dependent on specific characteristics of the printing device. This independence allows JES2 to use AFP printers in **full-function mode** and **page mode**.

Full function mode uses printer functions that produce page mode output. The concept of **page mode** permits printed pages to contain both text data and graphic presentations. The user can define and request attributes such as **segments** (predefined portions of a page), **overlays** (predefined page templates), **images** (pictures and graphics), and **type fonts** (collections of unique or stylized characters). For example, the graphical material can include a wide range of print font types and sizes for use in text headings, logos, and imbedded artwork; shading of textural and user-produced graphics; and graph plotting, some of which you can see in this book. [2]

Providing security

Security in a data processing environment involves controlling and auditing access to resources that are important to your installation. In the JES2 environment, these resources include:

- JES2-owned data sets
- Input (from nodes, remote workstations, readers, offload devices, and commands)
- Job names
- System input/output residing on spool (SYSIN/SYSOUT)
- Output (to nodes, printers, punches, remote workstations, and offload devices).

JES2 provides a basic level of security for some of these resources through initialization statements. For example, each node in a network can be defined as having a certain level of control over work at each of the other nodes in the system, which can give one operator limited control over each of the other nodes.

The control available through initialization statements can be broadened by implementing several JES2 exits available for this purpose. You can implement a more complete security policy by using the system authorization facility (SAF) component of the base control program and a security product such as Resource

2. All the illustrations and headings in this book are imbedded within the text files, and all print as a single image on IBM's AFP printers.

Access Control Facility (RACF®). SAF provides a link to the security product to define any additional security controls your installation may require.

JES2 passes information to SAF to perform password validation, to request authority to access a resource, and to determine security information in various environments. When SAF and the security product indicate a decision on a security request, JES2 bypasses its own security processing.

Supporting APPC

JES2 processes SYSOUT data sets for **APPC transaction programs**, which are application programs that communicate with other APPC transaction programs through the Advanced Program-to-Program Communication/MVS (APPC/MVS). For details about APPC/MVS, see *z/OS MVS Planning: APPC/MVS Management*. SYSOUT processing is the only function JES2 provides for APPC transaction programs.

Chapter 4. Customizing your JES2 system

This chapter answers the following questions:

- How can JES2 be customized at initialization to meet my processing requirements?
- Does customizing JES2 require writing my own operating system code?
- What other means can I use to enhance JES2?
- What are JES2 table pairs?
- What is the difference between an IBM-defined exit and an installation-defined exit?

JES2 is designed to be customized to meet your system's job processing requirements. You can complete basic customization when you create the JES2 initialization data set, and then use JES2 exits and table pairs to change additional JES2 functions and processes.

To modify JES2 processing beyond the capability provided by initialization statements, you can provide installation-written code and isolate it from IBM source code. Changes to JES2 processing implemented through direct source code modification are error prone, counter-productive during migration to future releases, and can prove to be very time consuming when debugging, diagnosing, and applying IBM-written program temporary fixes (PTFs) and authorized program analysis report (APARs) fixes to code. Furthermore, alteration of JES2 processing in this manner complicates IBM® service assistance. Therefore, JES2 provides the means to customize processing without direct source code modification.

Recommended methods for customizing JES2 processing include JES2 table pairs, IBM-defined exits, and installation-defined exits. A general discussion of each is provided in this section. See *z/OS JES2 Installation Exits* for a complete description of each IBM-defined exit.

JES2 initialization data set

Because every system that uses JES2 to manage its work input and output is unique, so too are the requirements that each system has for JES2. To meet these varying requirements, most JES2 functions can be customized by changing and using the **JES2 initialization data set** that is provided with the product, in the HASIPARM member of the SYS1.SHASSAMP data set. Although this data set will not run as shipped without some installation additions, it is a valuable model that can save hours of system programmer input time.

With a set of approximately 70 initialization statements, you can control all JES2 functions. The JES2 initialization data set provides all the specifications that are required to define output devices (printers and punches), job classes, the JES2 spool environment, the checkpoint data sets, the trace facility, and virtually every other JES2 facility and function.

Each initialization statement groups **initialization parameters** that define a function. The use of most JES2 initialization statements is optional; you only need to define them to implement or customize a particular function. Furthermore,

many of the parameters provide default specifications that allow your system to perform basic JES2 processing with no explicit definition on your part. JES2 requires only a minimal set of initialization statements (or parameters) that you define when first installing JES2.

Minimum required statements

To begin using JES2, it is not required to configure any of the more sophisticated processing environments, such as a multi-access spool complex, nodes or remote workstations. You can begin by simply building a base upon which your system can grow. To assist with JES2 initialization, a sample initialization data set is shipped with the product. This sample data set, shipped in SYS1.PARMLIB, requires only the addition of installation-defined devices and installation-specific values. It contains all of the JES2 initialization statements and the defaults for all parameters.

A customized initialization data set is provided by IBM for each Custom Built Product Delivery Offering (CBPDO). [3]

JES2 table pairs

Table pairs provide a facility to change, delete, or add to JES2 processing, function, or both. Changes made to JES2 processing using table pairs are generally less prone to error than are changes made through installation exits. This is true because JES2 macros generate the tables and generally require you to write less code to be run.

A number of JES2 functions (such as initialization statement processing, command processing, and message building) use tables. You can customize these JES2 functions, and others, by extending their associated tables. JES2 examines two tables, known as a **table pair**. The first table (the **JES2 table**) provides the default processing specifications; the second table (the **user table**) is used to extend, change, or delete the default processing specifications. For example, you can add your own JES2 commands and messages, add a new initialization statement or parameter, abbreviate the length of a JES2 command, or delete an unnecessary command to prevent its accidental misuse.

To simplify the use of this facility, you can use the JES2 default tables as templates for constructing your own tables. Depending on the tables that you choose to start with, using table pairs generally takes less detailed knowledge of JES2 internal structures than does writing an exit.

Table pairs do not replace the need for exits. Table pairs and exit points can provide added capability either independently or in conjunction with one another.

JES2 IBM-defined exits

JES2 exits provide a clean, convenient, relatively stable interface between JES2 and your installation-written code. Installation-written exit routines are invoked from standard JES2 processing at various strategic locations in JES2 source code. These strategic locations in JES2 source code are called **exit points**. A JES2 exit is established by one or more exit points.

3. CBPDO is an IBM offering that builds, customizes, and delivers a full MVS operating system; only the installation is required. CBPDO is useful for all new MVS installations to ease the migration effort from the previous operating system, and is particularly useful for brand new MVS installations.

JES2 supports up to 256 exits; IBM provides some exits to allow customization of the most commonly modified functions. In addition, IBM often provides new exits as new function is added to the product. These new exits provide a facility for you to alter the new processing as appropriate to meet your needs.

JES2 exits allow a wide range of customization. For example, you can add your own code to achieve the following goals:

- Design your own print job separator page
- Verify or change jobs submitted by TSO/E users
- Change or disallow selected commands
- Define alternate processing for a job that uses too many resources
- Provide increased security and password checking for remote terminals and system data sets.

Using **IBM-defined exits** only requires that you write exit routines and incorporate the exits by using two initialization statements. IBM provides these JES2 exit points in the code. To ensure a proper implementation, however, you must thoroughly understand the IBM-defined exit and its JES2 operating environment. A comprehensive description of each exit is presented in *z/OS JES2 Installation Exits*.

If you find that none of these exits meet your installation's needs, you can establish your own exit point and provide your own exit routine. This requires a more thorough knowledge of JES2 processing than using an IBM-defined exit point does. However, the use of exits is still far superior to adding in-line source code modification that may require the addition of many lines of code–code that cannot be dynamically disabled in the way that an exit can.

JES2 installation-defined exits

The JES2 exit facility allows you to define and use your own exits, should the IBM-defined exits not suffice. Exits created by users as modifications to JES2 and are called **installation-defined** exits. You can define exits by placing an exit point at appropriate points in the JES2 code, or in your own exit routine code. IBM-defined exit points are established in the same manner, but the exit point is placed appropriately for the specific function for which it is intended. Note, however, that implementing your own exit can be considerably more difficult than writing an exit routine for an IBM-defined exit. See *z/OS JES2 Installation Exits* for explanations of JES2 coding conventions, restrictions and how JES2 exits are packaged. See *z/OS JES2 Macros* for the JES2 macros that are available and intended for your use.

Chapter 5. Interacting with JES2

This chapter answers the following questions:

- How does the data processing staff communicate with JES2?
- How much control do operators have over JES2 when it is running?
- Are there ways to automate some of the routine processing needs?
- If JES2 experiences problems, how will it inform the operator?
- Are there any tools available to assist diagnosis and recovery?

JES2 operations

To help you maintain your overall work environment, JES2 provides an interactive means to control much of its function and the devices under its control. Although the JES2 environment is initially determined through initialization statements, you can alter many of those definitions as the system's workload changes or your installation adds new devices or needs to redefine its overall configuration. JES2 provides commands to request current status of devices and functions and JES2 responds with informational messages. Based on those messages, the operator, system programmer or automated operations program, such as NetView® for z/OS or System Automation for z/OS (SA z/OS), can issue further commands to change processing (such as, implement a newly written exit routine), start or stop a printer, or start diagnostic functions (such as the JES2 trace facility). Much of the processing can be changed without disrupting the remainder of the system. The following sections provide a brief overview of the control you have over JES2. (See *z/OS JES2 Commands* for an explanation of JES2 commands, and *z/OS JES2 Messages* for the text and explanation of JES2 messages.)

Operator control

Almost all JES2 initialization statement definitions can be changed by operator commands. These commands are available to both operators and system programmers to allow them to change current definitions. The system programmer can implement certain security features or customization techniques to limit the degree of control an individual or group can have over the operating system.

As your JES2 complex becomes more sophisticated, you might connect your system to others to form a network of systems. You can use operator commands to control the lines that connect the separate systems and define the separate systems to yours. This is typically a very dynamic environment, as different systems are added or deleted from the network because of maintenance, hardware reconfiguration requirements, workload balancing, or to access a database at a different location. JES2 permits you to use commands to alter most of your original network definition, as required.

If this dynamic control were not available, the operator or system programmer would need to change the initialization data set definition, stop JES2 processing, and then restart the system so those changes can take effect. This is, of course, required when redefining some areas of processing, but it denies system users valuable time.

Stopping and restarting JES2

There are instances when JES2 must be stopped and restarted either by a warm or cold start. For example, redefining the number of systems in a network job environment requires a warm start. A **warm start** is a restart of JES2 that does not cause the current work and output queues to be rebuilt, but does provide the required changed information to be propagated and used by all necessary components. A warm start is far superior to a cold start.

The definition (or redefinition) of some JES2 facilities and resources require that the JES2 system be totally shut down. JES2 must be restarted with a cold start to allow all component systems to be aware of the changed facilities and resources. A **cold start** is a JES2 restart that causes all current work and output queues to be lost and rebuilt with new data. The time to restart JES2 in this manner is based on the work in the system and, if not scheduled, causes a disruption in data processing services.

JES2 commands

JES2 processes its initialization statements and commands in a way that allows most initialization statements to be changed by operator command. The following is a partial list of the control that JES2 commands have over JES2 processing. Operator commands can be used to:

- Add function and functional subsystems
- Modify previously defined processing, such as: output definition, the dynamic alteration of the checkpoint definition, enabling installation-defined exits, offload devices, printer and punch characteristics, and job characteristics
- Delete function, and network systems, exits, and diagnostic traces
- Start, stop, and halt devices under JES2 control
- Assign units to local printers, punches, card readers, and lines or reassign units to these devices
- Display current facility and device definition.

All JES2 commands are entered using standard MVS command interfaces (such as through an MVS console or within the JES2 initialization data set). The command prefix character, which defaults to a dollar sign character ($), distinguishes JES2 commands and messages from other components of the operating system. For commands, the prefix character defines the scope of the command as being JES2 only; for messages, the prefix character is informational in that it designates that the message was issued by the JES2 component.

Automatic commands

The operator can specify that certain commands or strings of commands take effect automatically at specific times or at regular intervals. 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 a system normally does one specific kind of work at 8:00 AM and another at 9:00 AM every day, automatic command processing can issue the operator commands that would ordinarily be required at those times. The set of commands can be entered directly into the initialization data set, and the command processor will then issue them daily. Such commands can be a single command, a rather complicated set of commands, or commands that are error prone when entered manually.

The automatic command facility is optional, and provided to ease the operator workload. A second method of automating the operating system is to prompt system response based on JES2 messages through the use of automation products such as System Automation for z/OS (SA z/OS).

Automating JES2 operations

JES2 messages (similar to all MVS messages) contain various elements. All messages contain a unique identifying number, some messages contain non-variable text, others contain variable text (the text is based on the specific error condition or status), others contain specific reason codes (a numeric value - the explanation of which is documented in *z/OS JES2 Messages*) and some contain combinations of the preceding elements. Based on the message number, reason code value or variable text, the system can interpret the status and have a programmable console automatically issue the required commands to respond to the situation. Additional IBM products, such as NetView and System Automation for z/OS (SA z/OS), can be installed to automate your operation based on your policies regarding the handling of particular message or error conditions. ,

For example, System Automation for z/OS can be used to issue the appropriate JES2 commands to relieve a spool shortage condition or issue the appropriate MVS command(s) to complete the shutdown of JES2 after a serious error condition.

JES2 communication mechanisms

JES2 communication mechanisms include the JES2 tracing facility and JES2-IPCS formatting.

JES2 Tracing Facility

To record register contents and data at specific points in JES2 processing, you can use the JES2 tracing facility. JES2 has defined a number of the 255 trace types that can be defined; new ones are occasionally added to new product releases as their need arises. Here, again, is another JES2 facility that your installation can augment. (Your installation can add installation-specific trace points if the available trace functions do not meet your requirements.)

The use of the facility is optional. Tracing is initially activated under system programmer control (by specifying initialization statements) and subsequently controlled through several operator commands. Generally, trace points are only activated as a diagnostic tool for short periods of time.

Traced data can be accessed either through a dump of unformatted trace tables or in formatted system output. (See *z/OS JES2 Initialization and Tuning Guide* and *z/OS JES2 Diagnosis* for further information regarding the initialization and use of the JES2 trace facility.)

JES2-IPCS formatting

To facilitate diagnosis of errors that require the viewing of storage dumps or control blocks, both the base control program and JES2 components exploit the interactive problem control system (IPCS). When diagnosing a problem, IPCS panels provide you the opportunity to continue the diagnostic process through the base control program and into selected JES2 data areas.

IPCS is a menu-driven facility that lets you interactively select control blocks that you need to examine. Selected control blocks are formatted, displayed, and available for printing. (See *z/OS JES2 Diagnosis* for a discussion of IPCS-JES2 support.)

Appendix A. JES2 Publications

The following list of JES2 publications includes the basic set of those publications provided with the product.

z/OS JES2 Initialization and Tuning Guide
: Contains descriptions of JES2 initialization process and the JES2 facilities and task descriptions for the system programmer to use the JES2 initialization statements and their parameters to use and tune those facilities.

z/OS JES2 Initialization and Tuning Reference
: Contains complete descriptions of the JES2 initialization statements and their parameters to include syntax and defaults. References tables provide lists of hardware supported by JES2, JES2 values for optimizing hardware usage, and a copy of the sample initialization data set shipped in SYS1.VnRnMn.SHASSAMP.

z/OS JES2 Commands
: Describes how to operate JES2. The book explains the use of each JES2 operator command in detail.

z/OS JES2 Messages
: Describes all JES2 messages and suggests appropriate operator and system programmer responses.

z/OS JES2 Installation Exits
: Provides information to customize JES2 through JES2 installation exits.and describes how to use JES2 programmer macro instructions.

z/OS JES2 Macros
: Describes how to use JES2 programmer macro instructions.

z/OS JES2 Diagnosis
: Describes procedures to use and interpret the output of diagnostic aids/tools and their output. The book assists the system programmer to: identify a problem, collect information about the problem, report the problem to IBM, and fix the problem.

Appendix B. 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

This publication documents information that is NOT intended to be used as programming Interfaces of JES2.

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

O

P

R

S

T

U

V

W

Product Number: 5650-ZOS

Printed in USA

SA32-0994-00