#SHAREorg

SHARE

Understanding The Other JES

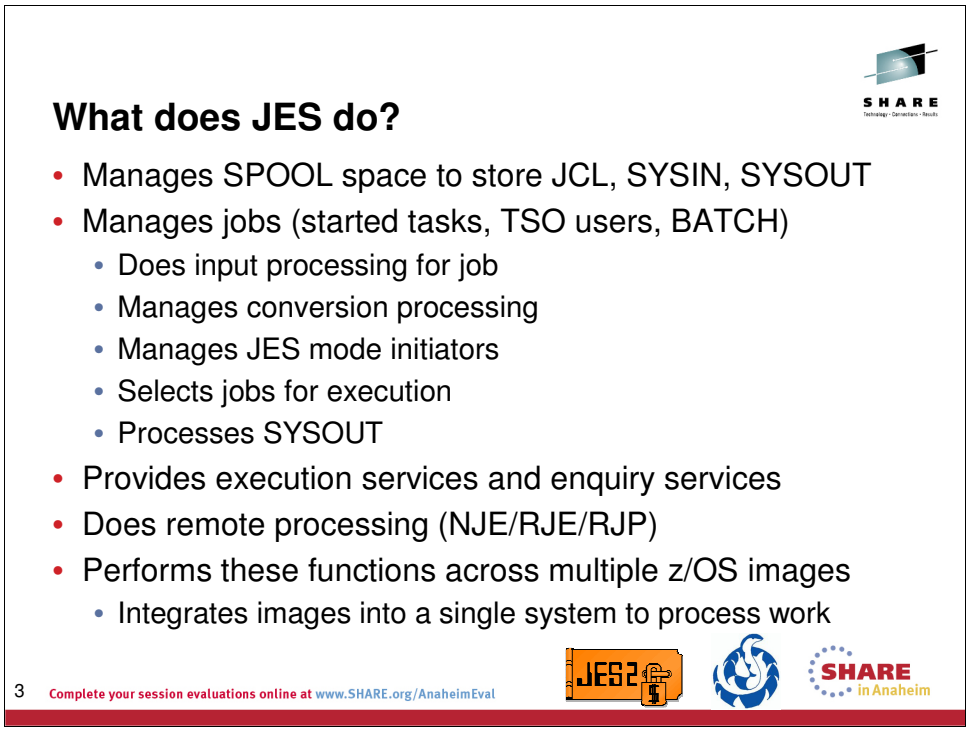
Greg Thompson
IBM System Test

March 13, 2014
Session Number 15094

SHARE in Anaheim

Trademarks

- **The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.**
- **IBM®**
- **MVS**
- **JES2**
- **JES3**
- **RACF®**
- **z/OS®**
- **zSeries®**
- **The following are trademarks or registered trademarks of other companies.**
- * Registered trademarks of IBM Corporation
- * All other products may be trademarks or registered trademarks of their respective companies.
- Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
- **Notes**:
- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will
- vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can
- be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual
- environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice.
- Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the 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.



The IBM z/OS operating systems use a job entry subsystem (JES) to receive jobs into the operating system, schedule them for processing by MVS or z/OS, and to control their output processing.

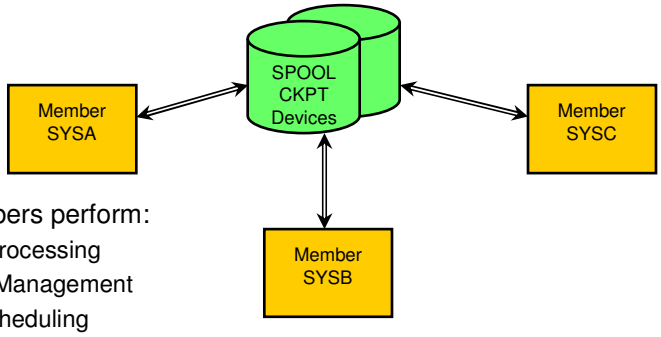
There are two versions of the job entry subsystem concept, JES3 and JES2. Some principle differences between the two JES systems include:

- JES3 provides resource management, dependent job control, and deadline scheduling for users of the system, while JES2 in the same system would require its users to manage these activities through other means.
- In cases where multiple z/OS systems are clustered (a sysplex), JES3 exercises centralized control over its processing functions through a single global JES3 processor. The global processor provides all job selection, scheduling, and device allocation functions for all of the other JES3 systems in the sysplex. JES2 is that component of MVS that only provides the necessary functions to get jobs into, and output out of, the MVS system. JES2 is designed to provide spooling, scheduling, and management facilities for the z/OS systems in a sysplex.

JES2

- Common set of work queues stored in its checkpoint
  - Member adds to or selects work from this common queue
  - Checkpoint is time sliced among members
- Simple mechanisms for managing work
  - Resource management done by MVS
  - Depend on MVS (scheduling environment, etc) to determine eligibility to select jobs for execution
  - Jobs sit in initiators waiting for resources
- Peer to Peer relationship between members
  - Members select work that it can process
    - Little regard to other members
  - No single point of control
  - No critical member
- Primary communication via JES2 checkpoint data set

JES2 MAS



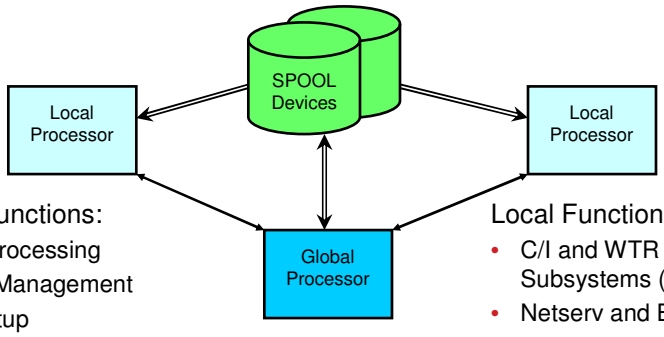
All members perform:

- Input processing
- Spool Management
- Job Scheduling
- SYSOUT scheduling
- SSI processing
- NJE/RJE

JES3

- JES3 does resource and workflow management before and after job execution.
- MVS does resource and workflow management during job execution.
- JES3 exercises centralized control through the global processor.
  - Global performs job selection, scheduling, and device allocation functions for the local processors.
  - Provides increased job scheduling control, deadline scheduling capabilities, and increased control with JES3 device allocation.

JES3 Global and Local Function



Global Functions:

- Input processing
- Spool Management
- Job setup
- Job scheduling
- SYSOUT scheduling
- SSI processing
- C/I and WTR Functional Subsystems (FSS)
- Netserv and BDT

Local Functions:

- C/I and WTR Functional Subsystems (FSS)
- Netserv and BDT
- Some SSI processing

JES3 Global

- Introduces all jobs into the system
- Handles scheduling of JCL Conversion/Interpretation (C/I)
- Pre-execution setup of JES3 managed devices
- Schedules jobs to all main processors (locals)
- Has awareness of all jobs in execution
- Handles scheduling of SYSOUT data sets for writers
- Manages space on the shared-spool devices

- Any Local can become a Global with a Dynamic System Interchange (DSI)

JES Structure

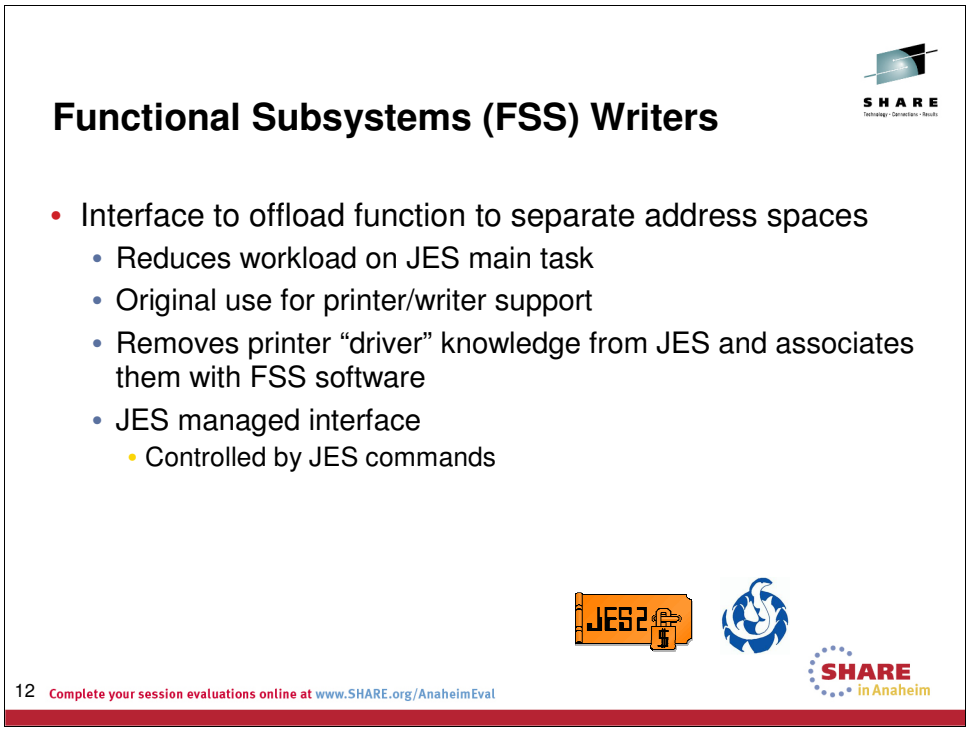
- JES address space
  - Main task
    - Sub-dispatched (shared) with multiple processes
  - Subtasks to perform tasks that must MVS wait
- Auxiliary address space to own processes/objects
  - Allows JES address space to fail yet JES functions continue
- JESXCF services
  - Provide XCF communications between address spaces
    - Primary communication between JES3 global and locals
    - Mainly used for enquiry/posting function for JES2
  - Manages status of JES address spaces
    - Notifies other members of a JES failure

JES2 Conversion Processing

- Converts JCL into internal format needed to run job
- JES2 calls MVS converter in the JES2 address space
  - PCE selects job and sets up environment
  - Subtask used to call z/OS converter
- JCL interpretation occurs when job executes in target address space
  - Certain JCL error not detected until job is in execution
- Converter parms (defaults) based on JOBCLASS
  - Journal, BLP, SMF exits, PROCLIB, region, SWA ABOVE, etc
- TYPERUN=SCAN just performs conversion processing
  - Errors discovered during interpretation not detected

JES3 Converter/Interpreter (C/I)

- C/I takes place in the JES3 global address space (C/I DSP) or can be offloaded to C/I FSSes.
  - JES3 C/I invokes the MVS C/I function to process the JCL.
- JCL interpretation is done immediately after conversion.
  - In the C/I DSP or FSS rather than after job selection in an initiator.
  - Can provide quicker feedback of JCL errors.
- Converter parms (defaults) set from CIPARM statement based upon the input device.
  - Journal, BLP, SMF exits, PROCLIB, region, SWA ABOVE, etc
- TYPRUN=SCAN will also invoke the MVS interpreter.
  - Parameter and specification errors can be detected and corrected before a job is submitted for execution.



A functional subsystem (FSS) is a collection of programs residing in an address space separate from JES that communicates with JES to provide a JES-related function, such as print processing and converter/interpreter (C/I) processing for JES3. An FSS extends the scope of JES processing. Because an FSS operates in its own address space, it functions independently of JES in several areas.

An FSS is responsible for:

• The management of storage resources that it needs during data set processing including print buffers.

• Its own recovery and serviceability.

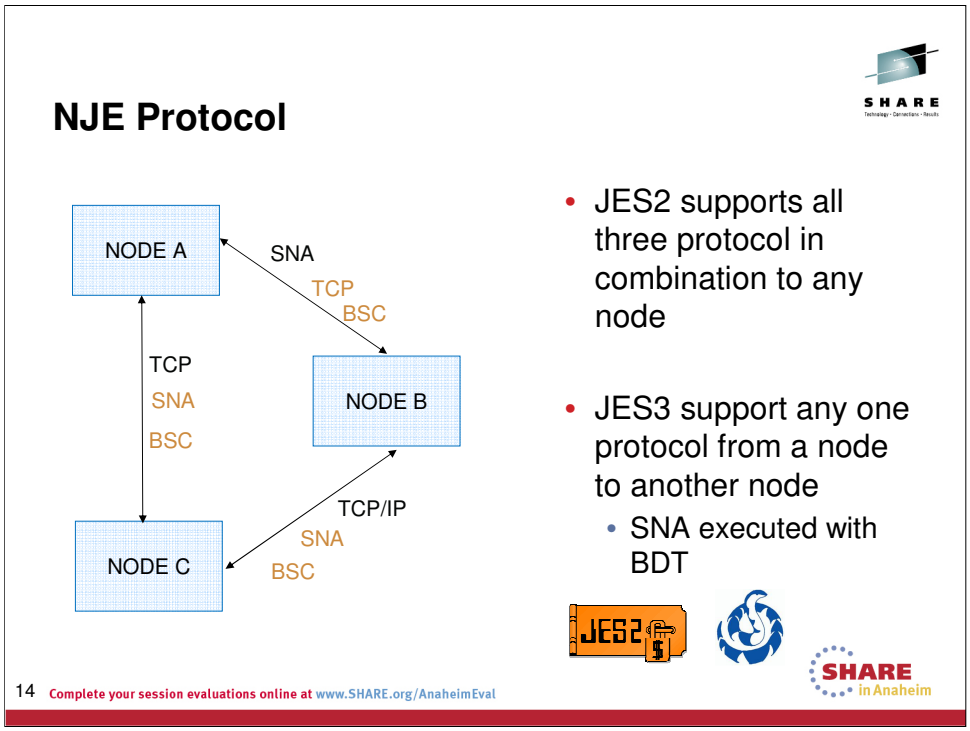
• Its performance and accounting measurements.

• The security of its own resources.

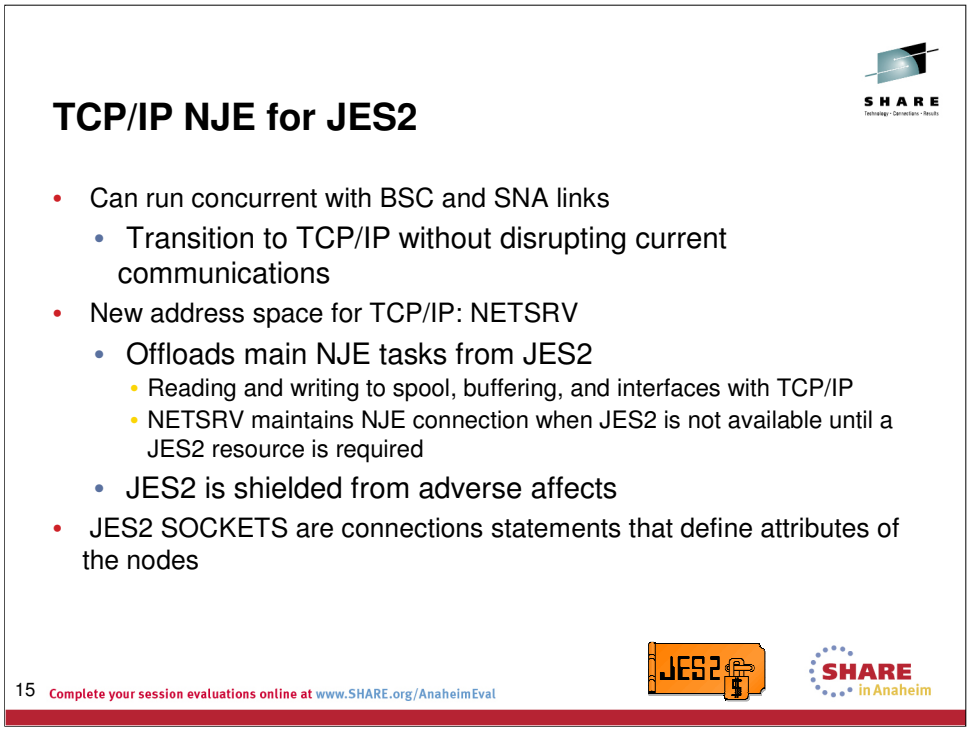
JES and the FSS communicate through the functional subsystem interface (FSI) and SSI. The FSI is a one level interface which provides two way communication. The FSI consists of a set of macro-invoked service routines provided by both JES and the FSS/FSA.

Converter Interpreters

- JES2
  - Runs under the main task
  - In V2.1, can run as a separate address space
- JES3 extended FSS to support Converter/Interpreter service
  - Allows conversion to be processed outside JES3 address space
    - Runs on both the Global and Local processors

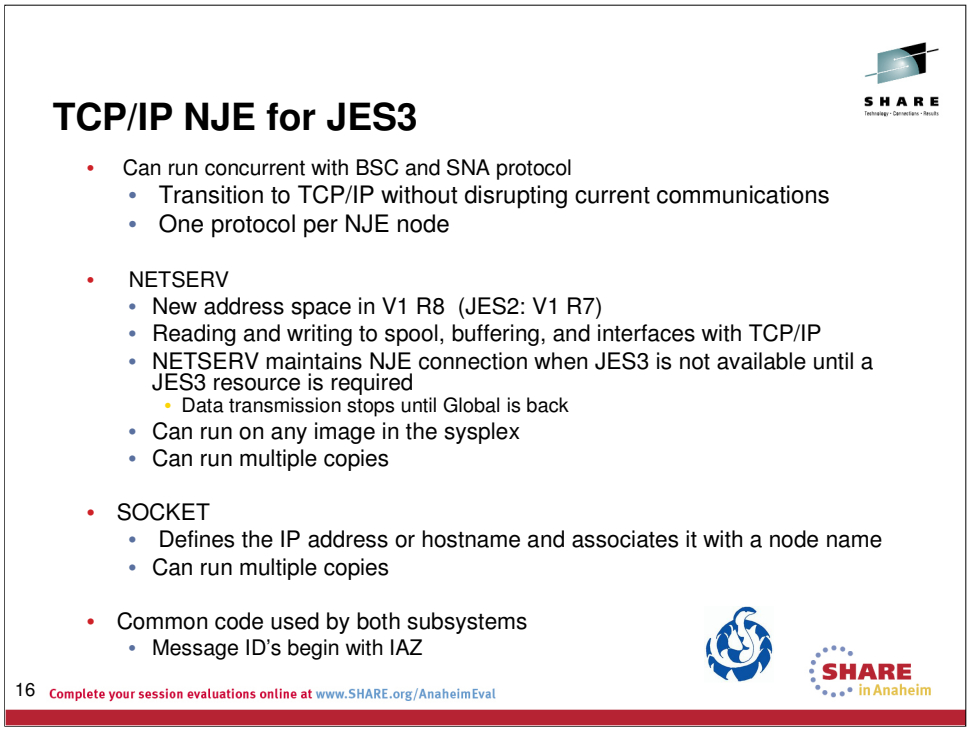


JES2 supports executing all three NJE protocols from any node to any other node. But JES3 only supports one protocol from one node to another. So, one can have SNA between NODE A and NODE B but not also include TCP/IP NJE between the same nodes. However, NODE A does support TCP from NODE A to NODE C. And it would not support SNA from NODE A to NODE C.



This support is being provided as a result of customer requests and also to relieve any restrictive hardware requirements needed to support BSC or SNA. Since the new support can run concurrently with existing BSC or SNA links, customers can transition to TCP/IP without disrupting current communications. Although new procedures/commands are required to establish the new connections, the underlying framework of NJE has been maintained, making it transparent to the end user.

Implementation of this new feature has resulted in a new address space called NETSRV. NETSRV will offload from the JES2 address space the main NJE tasks, such as reading/writing to spool, buffering and interfacing with TCP/IP. JES2 only needs to get involved when NETSRV requires such functions as obtaining a job number, allocating a new spool track allocation, purge processing, etc. A major benefit is additional stability in both NJE and JES2, such that when JES2 is unavailable NETSRV can maintain NJE connections (and traffic until a JES2 resource is required), should a networking problem arise, JES2 is shielded from any adverse affects.



This support is being provided as a result of customer requests and also to relieve any restrictive hardware requirements needed to support BSC or SNA. Since the new support can run concurrently with existing BSC or SNA protocol, customers can transition to TCP/IP without disrupting current communications. Although new procedures/commands are required to establish the new connections, the underlying framework of NJE has been maintained, making it transparent to the end user.

Implementation of this new feature has resulted in a new address space called NETSERV. Another new term, SOCKET, is used to define the IP address or hostname o the remote node.

What Is In A Name?

- NETSRV
  - JES2
    - NETSRV*n* – to JES2
    - JES2S00*n* to MVS
    - 'n' is based on number provided in the installation definition
      - *Via NETSRV initialization statement*
      - *Via $addsocket*(socket_name*),netsrv*=n *and $addnetsrv*n
- NETSERV
  - JES3
    - installation provided name known to JES3 and MVS
      - *Via NETSERV initialization statement*
      - *Via * F NETSERV,ADD*=netserv_name

The JES development of TCP/IP NJE was done by different organizations. So it is natural that there are some differences. Automation will need to be aware of the difference in spelling. If an installation is running both JESes, one difference will be the spelling of the TCP/IP server address space. It does not stop there though. JES2 has a different name for the address space than the name known to MVS. The MVS name starts with the 'JES2' nomenclature so that one can do a display of 'JES2*' and see all address spaces related to JES2.

NETSRV / NETSERV

JES2	JES3
Cannot delete definitions during life of MVS	**Can delete inactive definitions**
Can drain a NETSRV but cannot cancel it via JES2 (Can MVS cancel)	**Can cancel a NETSERV but cannot drain it (Can cancel via JES3 or MVS)**
LOCAL socket may need to be defined before starting	**Includes LOCAL socket**

There are differences in how JES2 and JES3 developed the TCP/IP server function. They are defined in the above chart.

A NETSRV can be defined in the JES2 init deck to start with JES2. There is no equivalent function in JES3 because the TCP DSP will auto start under most circumstances.

Two LOCAL sockets are created by JES2 during initialization; one is named LOCAL and one is named LOCALTLS. One or both of these sockets may be associated with NETSRV definitions. If additional NETSRV's are created, then the LOCAL SOCKET must be added and associated with the NETSRV. JES3 includes the LOCAL SOCKET in the NETSERV. The PORT and HOSTNAME parameters on the NETSERV statement provide the information required for the LOCAL SOCKET.

SOCKET

JES2	JES3
Cannot delete definition during life of MVS	**Can delete inactive definition**
A LOCAL SOCKET is required to start the NETSRV, separate definition	**LOCAL SOCKET built into NETSERV, no definition required**
IPADDR parameters defines remote node	**HOST parameter defines remote node**
Start sockets individually	• **Can start all sockets to a node with one command** • **Can also be done individually**
Cannot cancel a socket; must drain the line	• **Can cancel a socket** • Includes immediate option

- JES3 to JES2
  - Must have a socket available on remote JES2 system for JES3 connection
- JES2 to JES3
  - JES3 remote server will create connecting socket
    - Naming convention - @000000n

19 Complete your session evaluations online at www.SHARE.org/AnaheimEval

It becomes important to understand these differences with sockets when it comes to using JES2 and JES3 in a TCP/IP NJE network. If you are starting a socket on JES3 to a JES2 system, be sure to have the socket already defined on JES2. Otherwise the start of the socket will fail. If you are starting a socket from JES2 to JES3, you do not create the socket on the receiving side in advance because JES3 will automatically create the remote socket (known as the server socket). In other words, think of how the JES on the remote node behaves and act accordingly.

Transmitters and Receivers

- JES2
  - Exists in BSC and SNA NJE
  - Continues to exist in TCP/IP NJE
- JES3
  - Exists in BSC NJE
  - Does not exist in SNA (BDT)
  - Exists in TCP/IP NJE
- Possible configuration difference in JES2 / JES3 TCP/IP
  - Number greater than 1 may already be defined in JES2
  - Default in JES3 is 1
  - Unmatched JES2 transmitters/receivers are drained

    $HASP097 Ln.JT2 IS DRAINED
    $HASP097 Ln.JT3 IS DRAINED

Installations that use JES2 are very aware of NJE transmitters and receivers. But JES3 installations may not be as aware of them. When connecting TCP/IP NJE lines and sockets, one needs to be aware of their environment.

We had a test where normally the two systems used JES2 and had three transmitters and receivers defined. But in this test, one system was brought up as JES3. Two of the transmitters and receivers drained on the JES2 system because the default of one transmitter and receiver was used in the JES3 definitions. It initially caused some panic because we thought we had discovered a bug in JES2 GA code.

TCP/IP NJE Trace

- JES2
  - TRACEIO=(JES=YES,COMMON=YES,VERBOSE=YES) on NETSRV or LINE statement
    - Default is NO
  - Uses trace identifiers 33-39
  - Tracing done to spool
- JES3
  - SOCKET and NETSERV statements
    - JTRACE=YES – JES trace
    - ITRACE=YES – internal tracing of the NETSERV address space
    - VTRACE=YES – verbose trace
    - Default is NO
  - Must start GTF trace

Tracing has always been done differently on the two JES subsystems and nothing has changed. JES2 writes to spool so be aware of spool utilization. Additional parameters were added to TRACEIO to trace TCP/IP activity on the NETSRV address space.

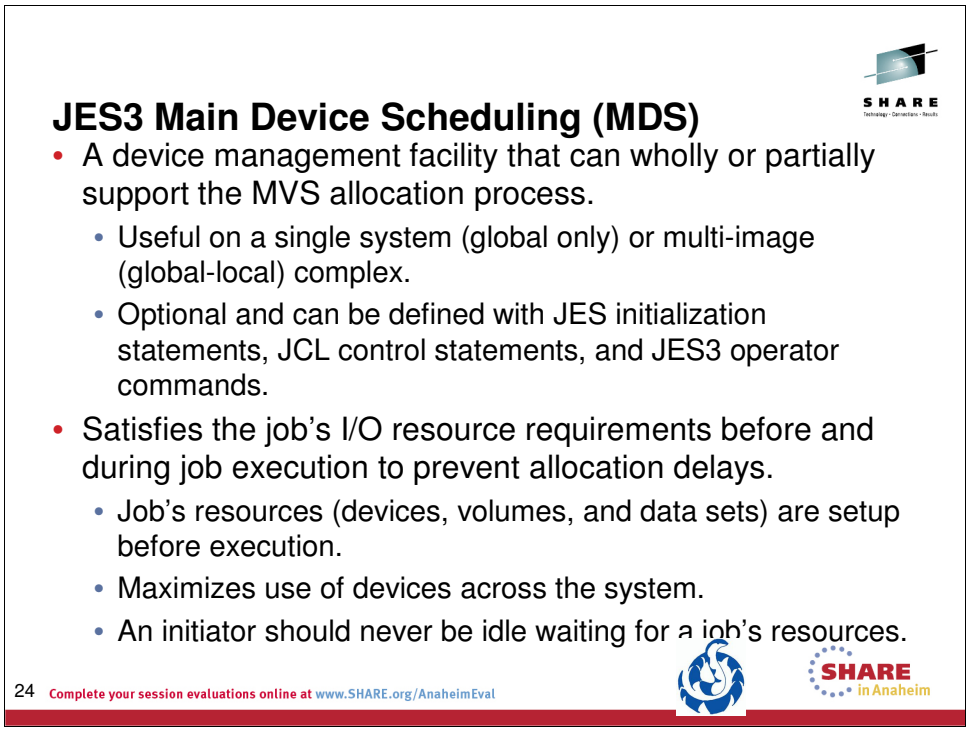
Tracing in JES3 can be done on the NETSERV address space and the SOCKET. JES3 also writes to GTF so be sure to have GTF trace running. (Yes, this notice is brought to you by our experiences.)

MVS Allocation

- A job's resource requirements are not known until a system initiator begins the step allocation process.
  - The job's requirements are satisfied one step at a time.
- Attempts to satisfy requirements are in contention with every other job step currently executing in the SYSPLEX.
- Jobs waiting for resources hold other available resources (an initiator, an address space, data sets, devices).
- Waiting jobs complicate the task of determining how many JES initiators are needed to keep the system full.

JES2 Job Scheduling

- Device/Data set scheduling managed by MVS
  - Job starts in initiator and waits for needed resources
  - GRS ENQ at allocation performs serialization and reports contention
- Allocations managed at a STEP level
  - No issues if one step creates data sets used by later steps
  - Steps that are skipped (due to conditional JCL) do not reserve resources
- System affinity and Scheduling environment used
  - Controls what jobs can be selected for execution
- Some balancing done for WLM initiators
  - Keep same percent busy on all members

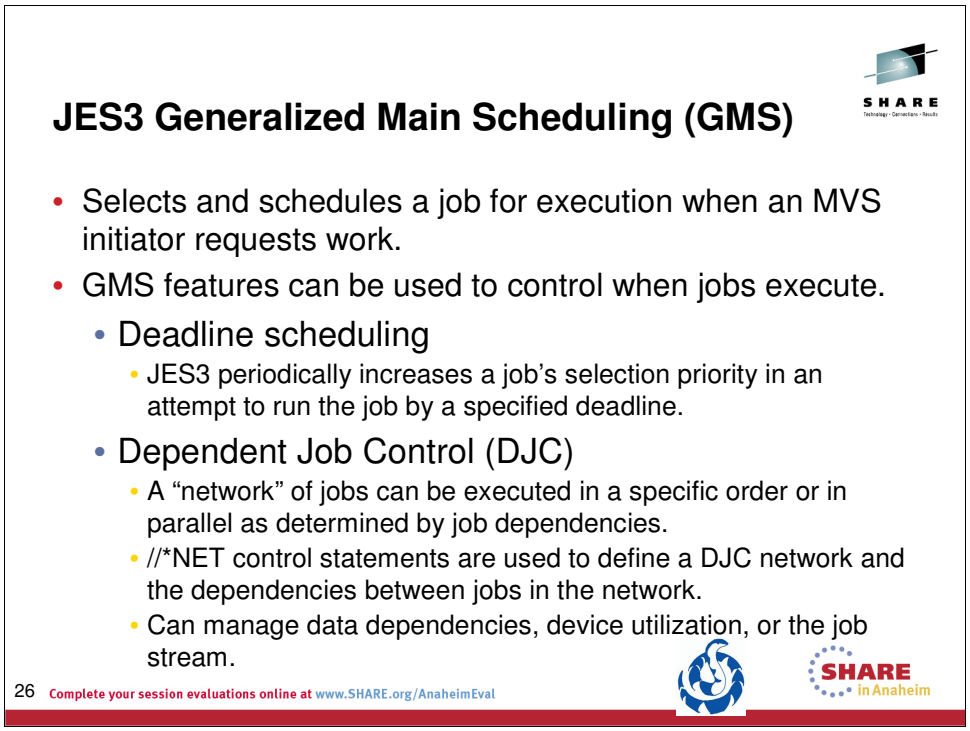


•*z/OS V1R12.0 JES3 Initialization and Tuning Guide* – Chapter 4.

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

JES3 MDS (continued)

- Considers the resources for all steps of a job and across all mains in the JESplex.
  - Has more information than is available to MVS allocation.
  - Can schedule combinations of jobs that will execute on a main without contention of resources attached to the main.
- Cooperates with Workload Management (WLM) to ensure scheduling environments for jobs are honored.
- With setup before job execution, dependencies between jobs or steps in a job cannot be determined.
  - Important for cataloging and passing data sets.
  - **JES3 assumes all steps will execute** and cannot determine if conditional job steps are skipped.



•*z/OS V1R12.0 JES3 Initialization and Tuning Guide* – Chapter 4.

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

Deadline scheduling and dependent job control (DJC), additional GMS functions, enable you to control when jobs execute.

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

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

WLM-Managed Initiators

- Initiators started and managed by WLM
- Initiators associated with WLM service class
  - Only select work for a specific service class
  - Job class can influence service class assignment
- WLM starts initiators based on
  - System capacity (WLM tries to balance work across SYSPLEX)
  - Whether service class is meeting goals
  - Relative importance of the service class
- JES tells WLM on each system how many jobs are waiting
  - Based on service class, resource availability, where jobs can run
- JES decides what job to start in each initiator
- Specified by MODE=WLM on
  - JOBCLASS for JES2
  - Job class group for JES3


JES2 – Managed Initiators

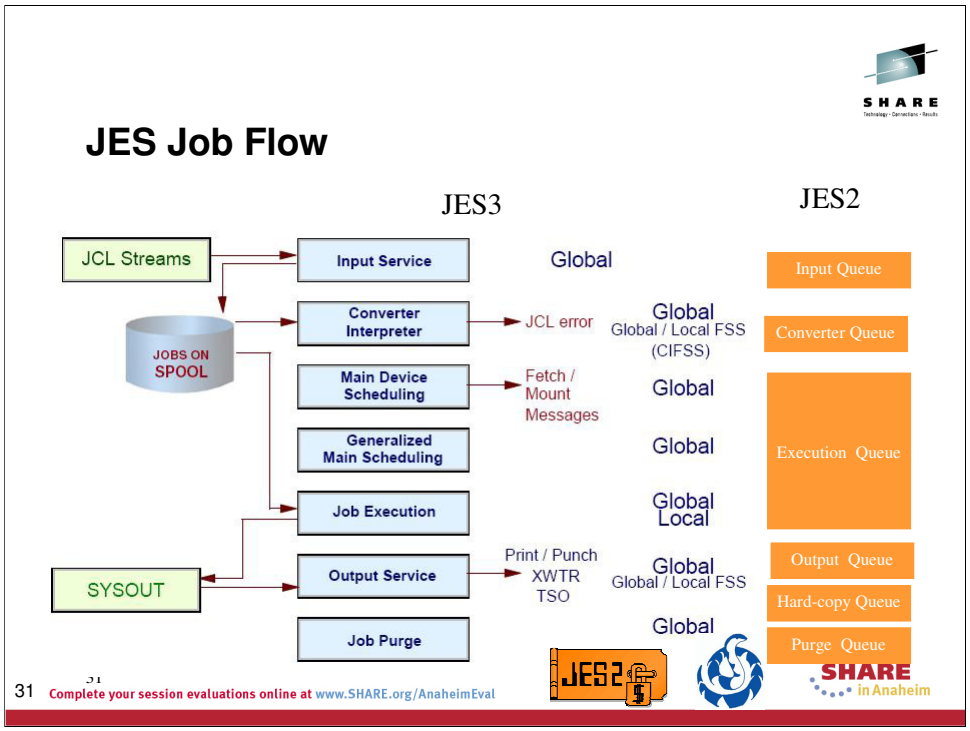
- Type of initiator used based on a JOBCLASS MODE=
  - Applies to all members of the MAS
  - MODE=WLM cannot be selected by JES2 initiators
- MODE=JES JOBCLASS uses JES2 initiators
  - Initiators started and managed by operator commands
  - Number of initiators defined at initialization
  - Initiators select jobs based on a ordered list of job classes.
- Start job command, $S J(nnn), causes WLM to start an initiator to run a specific job
  - Job can be in a WLM or JES mode class

JES3-managed Initiators

- Associated with a job class group.
  - GROUP initialization statement.
- Expressed in terms of resources, not goals.
  - How many to start, what systems, when to start/stop defined with EXRESC parameter of GROUP initialization statement.
- Started/stopped by JES3 based on definitions and the backlog of jobs.
  - Also with *MODIFY,G command.
- Jobs selected within the job class group.
  - Selected based upon priority within the group.
- Workload balancing is difficult to define and static in nature.

Why WLM-managed over JES-managed?

- Fewer and simpler externals are needed to control WLM-managed initiators and to perform workload balancing.
- Managed according to the service classes and performance goals specified in the WLM policy.
- Externals reflect customer expectations typically in terms that are found in service level agreements.
- Workload balancing is automatic as the number of initiators running is based on performance goals and the importance of batch work with respect to other work.
- Dynamic, goal oriented initiator management allows the system to adapt to changing conditions and how well the work is meeting its performance goals.



The term “job” has been traditionally defined as unit of work signified by the presence of a JOB JCL statement in the input stream. Within each MVS job, the control statements are grouped into job steps. A job step consists of all the control statements needed to run one program. If a job needs to run more than one program, the job would contain a different job

step for each of those programs.

In general, statements in the MVS Job Control Language (JCL) and in the Job Entry Control Language (JECL) for JES2 and JES3 subsystems, are used by the users to define a job’s execution steps and resource requirements. The MVS system requires an internal format a job's JCL statements before it can process it. MVS converter/interpreter transforms the external (JCL) format job description into the internal format called the scheduler work area (SWA).

Every MVS job must contain a minimum of the following two types of control statements:

•A JOB statement, to mark the beginning of a job and assign a name to the job. The JOB statement is also used to provide certain administrative information, including security, accounting, and identification information. Every job has one and only one JOB statement.

• An EXEC (execute) statement, to mark the beginning of a job step, to assign a name to the step, and to identify the program or procedure to be executed in the step. You can add various parameters to the EXEC statement to customize the way the program executes. Every job has at least one EXEC statement.

In JES3 context, a job is the work to be performed by JES3 DSPs to prepare a job submitted in the input stream for MVS execution. That work can be related to a specific MVS job from the input stream, or it can be related to the work JES3 must do in its role as resource manager when preparing the MVS jobs for execution.

When JES3 is signaled to read an input stream of MVS jobs, a chain of events begin. These include:

•Creation and scheduling of a card reader DSP - a JES3 job

•Reading of the input stream by a DSP

•Building of the JES3 control blocks describing the MVS job to JES3. JCT entries for each job in the input stream are added to the JES3JCT data set.

•Execution of DSPs represented by scheduler elements in the JCT entries for each job

The DSPs that provide the JES3 input service control the processing of a typical MVS job at the beginning. Input service routines create scheduler elements that represent a job’s flow through the various JES3 processing phases Input service, active on the global processor, accepts and queues all jobs entering the JES3 complex. The global processor accepts MVS jobs into the system from:

•A TSO SUBMIT command

•A local card reader (CR DSP)

•A local tape reader (TR DSP)

•A disk reader (DR DSP)

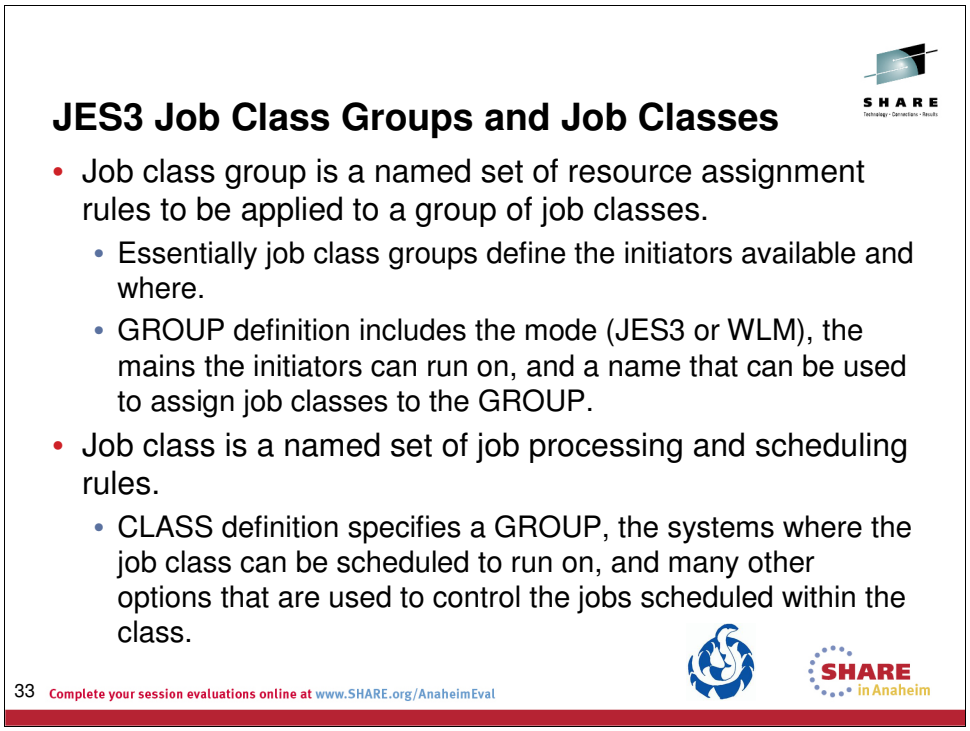
•A remote work station (RJP/SNARJP DSPs)

•Another node in a job entry network (NJE DSPs)

•The internal reader (INTRDR DSP)

JES2 Job Limits and Affinities

- JOBCLASS limits exist on a JESPLEX and member level
  - Number of concurrent jobs that can be active in JOBCLASS
  - Applies to JES and WLM mode JOBCLASSes (JOBs)
  - Limits affect number of available jobs reported to WLM
    - Impacts number of initiators WLM starts
- JOBCLASS affinity controls member where class is active
  - Lists systems that can select from the job class
  - Holding class same as null affinity list
  - Applies to JES and WLM mode JOBCLASSes
  - Affect number of available jobs reported to WLM
- Service class affinity limits where service class is active
  - Service class only registered if member in affinity list
  - WLM only starts initiators if service class is active



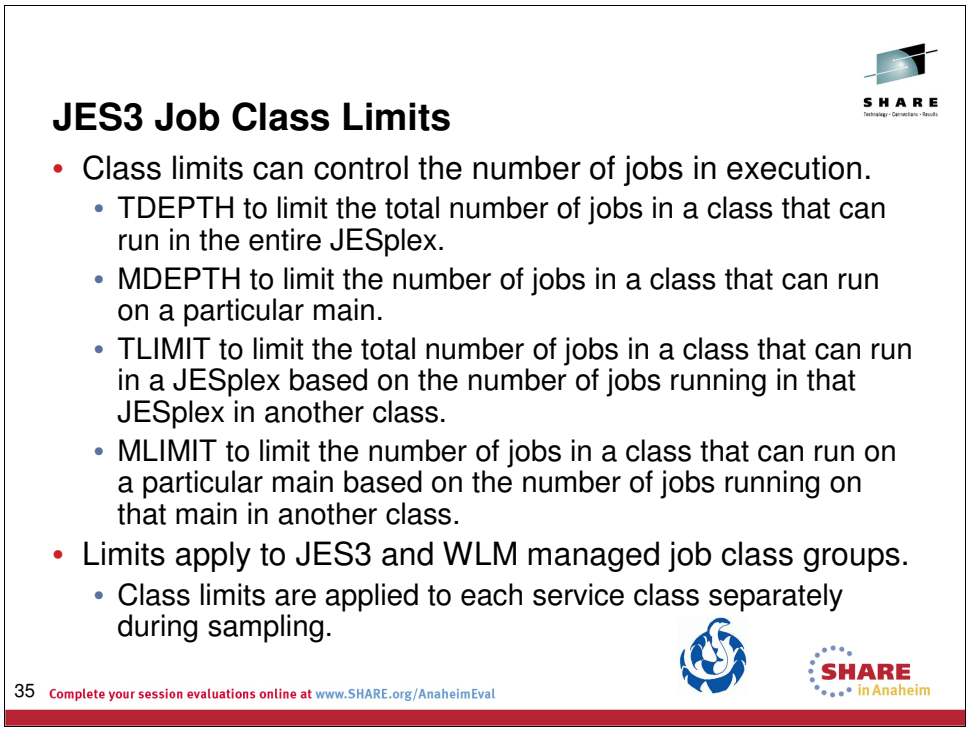
One way to think of JES3 GROUP is that it is used to define the initiators available and where. The definition includes the mode (JES3 or WLM), the mains the initiators can run on, and a name that can be used to assign job classes to the GROUP.

CLASS defines the job classes that can be used for jobs. The class specifies the GROUP, or set of initiators , than the job can be scheduled to, but the also the systems where the job class can be scheduled to run on. In addition there are many other JES3 unique options, not covered here, that are used to control the number of jobs scheduled within the class.

See *z/OS V1R12.0 JES3 Initialization and Tuning Reference* for more details.

JES3 Job Resources

- Device Fencing/Pooling
  - Can reserve devices for use only by jobs within a job class group or DJC network.
- Spool Partitioning
  - Can specify the spool partition to be used for jobs in a job class.
  - More on spool partitioning in a moment.



z/OS V1R11.0 JES3 Initialization and Tuning Guide

TDEPTH: Sets the maximum number of jobs of this class (0 to 255) that can execute in the total JES3 complex at one time.

MDEPTH: Sets the maximum number of jobs of this class (0 to 255) that can execute on a given processor at one time.

TLIMIT: Specifies the maximum number of jobs of other job classes that can execute in the total JES3 complex and still allow jobs in this class to be scheduled. If any class limit is exceeded, no more jobs in this class are scheduled; that is, jobs in this class are scheduled only when the number of jobs running from other classes is equal to or less than the assigned limit.

MLIMIT: Specifies the maximum number of jobs of other job classes that can execute on a given processor and still allow jobs in this class to be scheduled. If any class limit is exceeded, no more jobs in this class are scheduled on the given processor; that is, jobs in this class are scheduled only when the number of jobs running from other classes is equal to or less than the assigned limit.

TDEPTH, TLIMIT, MDEPTH, and MLIMIT are used for WLM-managed job class groups and JES-managed job class groups. However, keep in mind that JES3 applies the class limits to each service class separately during sampling. If you define class limits for a job class, and the jobs in the class are assigned to more than one service class, more initiators may be started than can actually run jobs. When class limits are used, it is suggested that all jobs in the class be assigned the same service class.

JES Monitor Tools

- JES2
  - $JD STATUS
    - Displays all current alerts and any notices that are outstanding
- JES3
  - F JES3,CHK
    - MVS modify command
    - Categories of exception conditions checked are:
      - *CI and MDS*
      - *FCTs and DSPs*
      - *General*
      - *JSS and Job queue*
      - *Spool*

Spool Offload

- JES2
  - Uses transmitters and receivers
  - Offload to either DASD or to TAPE
- JES3
  - Uses the Dump Job DSP (DJ)
  - Offload to TAPE only

Command Prefixes

- JES3 command prefix can be up to seven characters
  - Sysplex scope prefix which is to JES3 Global
    - Default is the asterisk (‘*’)
  - System scope prefix which is to local JES
    - Default is eight (‘8’)
  - Multiple JES3 plex capability
    - CONSTD…. SYN=(*,8) removes sysplex awareness of Global
    - Documented in V2.1 initialization guide though been allowed for years
- JES2 command prefix is one character in length
  - Command is to local JES
    - Default is dollar sign (‘$’)

The JES3 sysplex scope prefix sends the command from any system to the global and the response is received on the system that issued the command. If there are more than one JES3 global in the sysplex and the default prefix is used for both global systems, the first global becomes the global for the sysplex. In this configuration, it might be a good idea to define unique sysplex scope identifiers for each global or remove the sysplex scope prefix all together and use the system scope prefix and the route command. The route command is used in JES2 when communicating with each member in the sysplex.

Replies

- Three reply formats
  1. Standard reply (R n,reply-text)
  2. Short format (n,reply-text)
  3. Abbreviated short format (nreply-text)
- JES3 supports only the first two
  - Third format conflicts with ‘8’ command
- Third format should never be used with common automation for both JES

JECL Statements

JES2	JES3	Recommendation
/*JOBPARM SYSAFF=	//*MAIN SYSTEM=	Both can be placed into common JCL. V2.1 supports SYSAFF= or SYSTEM= on the JOB card, overrides JECL statement.
/*ROUTE XEQ	//*ROUTE XEQ	Use MVS XMIT card.
/*XEQ		
/*OUTPUT	//*FORMAT	Use the MVS OUTPUT card.

JCL and JES3 Locate

```
//STEP0 EXEC PGM=IEFBR14,COND=ONLY
//DD1  DD DSN=MYDSN.PROD,
//        UNIT=SYSALLDA,DISP=(,CATLG)
//*-------------------------------------------------------
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIND DD *
DEF CLUSTER NAME(MYDSN.PROD),
 (additional parameters)
//*-------------------------------------------------------
//STEP2 EXEC PGM=mypgm
//SYSPRINT DD SYSOUT=*
//INPUT DD DISP=SHR,DSN=MYDSN.PROD
```

- Objective of JCL
  - Create data set in step 1
  - Use data set in step 2
- Without STEP0, this job would fail in JES3 LOCATE
  - "DATASET NOT FOUND" for MYDSN.PROD in STEP2
- STEP0 does not execute
  - COND=ONLY
    - Only if abend
    - Impossible in first step
- This JCL logic will work on either JES

System Logs

- JES3 DLOG
  - When DLOG=YES specified
  - JESplex wide log
  - Written to global SYSLOG
  - When DLOG=NO specified, writes to each system SYSLOG
- JES2 SYSLOG
  - Written to its own SYSLOG
- Each JES has a unique format
- Suggest using OPERLOG for post processing routines

The 'Other' JES Commands for Jobs

Description	JES2	JES3
Display job, started task, TSO user	$D J|S|T(n)	*I J=n
Cancel executing job	$C J|S|T(n)	*C J=n
Cancel job and purge output	$P J|S|T(n)	*F J,=n,C
Cancel job with print	$CJ n	*F J=n,CP
Modify job	$T J|S|T(n)	*F J=n
Display all active jobs on all systems	$D A,J|S|T|X,ALL	*I A

The above chart is a starter for finding information related to jobs in either JES.

The 'Other' JES Commands for Output

Description	JES2	JES3
Hold job output	$TO J|S|T ODISP=HOLD	*F U J=n,Q=q NH=Y
Release job output	$O J|S|T(n)	*F U J=n,Q=q,NH=N
Display output characteristics	$D O J|S|T|JOBQ(n)	*I U J=n,Q=q
Find large output on spool	$DJOBQ,SPOOL=(PERCENT>n)	*I Q SP=ALL,U

The 'Other' JES Miscellaneous Commands

Description	JES2	JES3
Displays jobs on a queue and percent of spool utilization	$DQ	*I Q (queue) *I Q S (spool)
Display spool volume information	$DSPL,L	*I Q DD=ALL
Display information about a JESplex member	$DMEMB(name)	*I MAIN=name
Display a netserv	$D NETSRV1| NETSERV1| NSV1	*I NETSERV=name
Display a socket	$D SOCKET(name)	*I SOCKET=name

Command Automation

- JES2 support
  - $T A command
  - JES2 INIT deck
  - /*command verb in JCL for JES2 operator command
- JES3 support
  - //*command verb in JCL for JES3 operator command
- MVS supports
  - COMMAND statement
  - IEACMDxx parmlib member
- Automation packages

V2.1 Updates to Make JES More Alike

- Eight character job classes
  - New to JES2
  - JES2 and JES3 allow eight character CLASS= on JOB card
    - //*MAIN CLASS= overrides CLASS on JOB card
- System affinity on both JES2 and JES3 updated
  - SYSAFF= and SYSTEM= allowed on the JOB card
  - Old JECL statements still honored
    - JOB card parameters override these JECL parameters
  - Other JES JECL statements still ignored

In-stream data sets - Making JES More Alike

- JES3 supports in-stream data in JCL PROCs and INCLUDEs in V2.1
  - Works for batch jobs and started tasks.
  - Works like in-stream data in job JCL.
- Support is based on where the job converts – z/OS V2R1.
  - Job can then run on a down-level system.
- Additional SYSIN data sets are created which contain the in-stream data.
  - They are included in extended status DSLIST function.
  - They are visible via SDSF Data Set Display after selecting "Change include SYSIN to ON" under "Options."
  - They are NOT included with SPOOL Data Set Browse of JCLIN.
    - Not part of original JOB JCL submitted.

See the JES3 Product Update for more information about in-stream data sets in JES3 V2.1.

Questions?