#SHAREorg

SHARE

JES2 Bootcamp – Part 1 of 2
What is JES2 and what does it do

Tom Wasik
IBM Rochester, MN
wasik@us.ibm.com
Wednesday 9:30AM
Session Number 15325

Mainframe 50 April 7th 1964 - April 7th 2014
http://www.ibm.com/mainframe50/

SHARE in Anaheim

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- IBM®
- MVS™
- Redbooks®
- RETAIN®
- z/OS®
- zSeries®

The following are trademarks or registered trademarks of other 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.
- All other products may be trademarks or registered trademarks of their respective companies.

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.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Brief History of JES

- Original 2 versions
  - Attached Support Processor (ASP)
  - Houston Automatic Spooling Priority (HASP)
- Both affiliated with the US space program
- Both were field developed programs
- Both addressed the problem of slow peripheral I/O
- Spool was staging area for input (jobs) and output (SYSOUT)
- Made more efficient use of expensive CPU resource
- Provided functions the operating system (MVT) chose not to
- Eventually became JES2 (HASP) and JES3 (ASP)
  - JES - Job Entry Subsystem

Why do I need a JES? (What does it do?)

- JES implements SPOOL
  - Storage for job JCL, SYSOUT, SYSIN
- JES manages batch
  - Gets jobs from multiple sources and places them on SPOOL
    - Includes TSO logon, started tasks and Batch
  - Schedules jobs to execute (manages work queues)
    - WLM initiator started by WLM, but JES selects the work
  - Provides interfaces to influence job scheduling
    - SSI, commands, exits
- JES manages SYSOUT
  - Interfaces to JES printers
  - Provides output to “FSS” printers (PSF)
  - Has interface for generic “print” drivers (InfoPrint, etc)
    - SAPI – SYSOUT API
- JES connects multiple MVS images
  - MAS/Complex predates SYSPLEX
  - Allows multiple systems to process single job/output queue (1975)

Why do I need a JES? (What does it do?)

- JES implemented early “client server”
  - RJE/RJP allows jobs from client computers to be run on MVT and get output back (1967)
  - Protocol still used today (though greatly reduced)
- JES implemented multi-node connections
  - NJE – Network Job Entry
  - Allows peer systems to interchange jobs and output for processing (1976)
  - Early e-mail is TSO transmit over NJE
  - Supports TCP/IP as well as traditional BSC and SNA
  - Still actively used by many installations
- Interfaces with other components via the Subsystem Interface (SSI)
  - Gets control from the BCP to process events (command, wto, etc)
  - Invoked to perform functions for other components (Allocate, Open, TSO cancel/status, etc)
  - Provides basic JES2 functions (Extended status, SAPI, SWB read, etc)

What makes JES special (unique)?

- Originally field developed program
  - Not part of the operating system
  - Today, JES would have been considered a mods to the operating system
  - Before the SSI, JES front ended SVCs etc to intercept normal processing
  - Provided functions the operating system (MVT) chose not to
- Rich history of modification by customers
  - Originally as source mods
  - Later as exits
- Source distributed and maintained
  - Mostly assembler code (HLASM), some PL/X and common (IAZ) code in C
  - Customers/vendors look to JES for examples of how to do things
  - Lots of customer written exits to enhance functions
  - One customer has 35K lines of JES2 exit code
  - Customer “interface” to the operating system

How does JES work

- Mini operating system within an operating system
  - Sub dispatches single main task TCB
    - JES2 has hundreds of sub dispatchable units
    - Could not have a TCB for each
      - *Serialization and storage constraints*
  - Manages data sets on SPOOL
  - Supports rich set of operator commands
  - Implements network protocols
    - BSC is at a CCW level
  - Drives real devices
    - At CCW level
  - Manages work queues, does scheduling
  - Manages initiators that run work
    - Selects work to run in initiators (even WLM initiators)
    - Also manages JES mode initiators

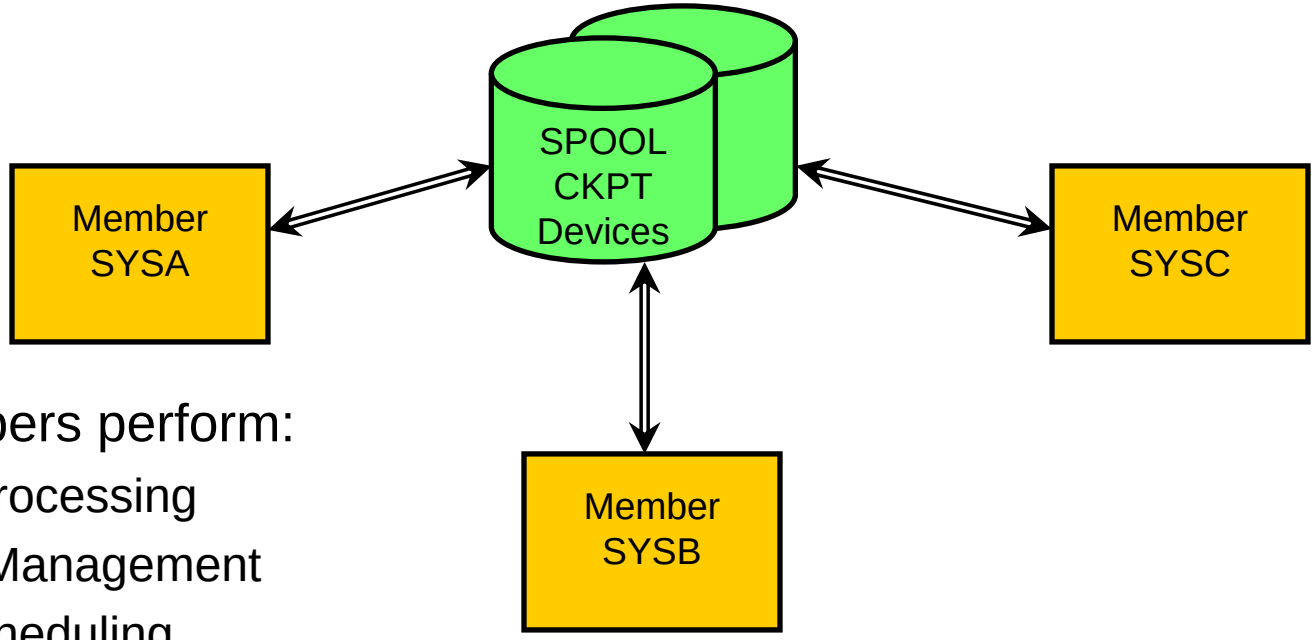
OK, but why 2 JESes?

- 2 philosophies of system management:
  - JES2
    - A collection of peer systems sharing a single job/output queue
    - Each system selects the work it can processes
  - JES3
    - A single job/output queue managed on one system with work being passed from main (global) system to worker (local) system for processing
    - Centralized control of where work is sent to be processed
- Functional differences
  - JES2
    - Simpler to use and set up
    - Changes can be made dynamically
    - No single control point to fail
  - JES3
    - More functions
      - *Dependent job control*
      - *Resource scheduling*
      - *Deadline scheduling*
    - Additional complexity to set up, less dynamic

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 (eg DSN ENQs)
- 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
  - XCF usage is increasing with time
  - Mostly special purpose requests
    - Device settings
    - Status updates
    - Managing processes (like checkpoint reconfiguration, and spool migration)
- ENF used for reporting job and SYSOUT status
  - Used to track device activity and job status
  - Allows application to listen to signaling sent between members

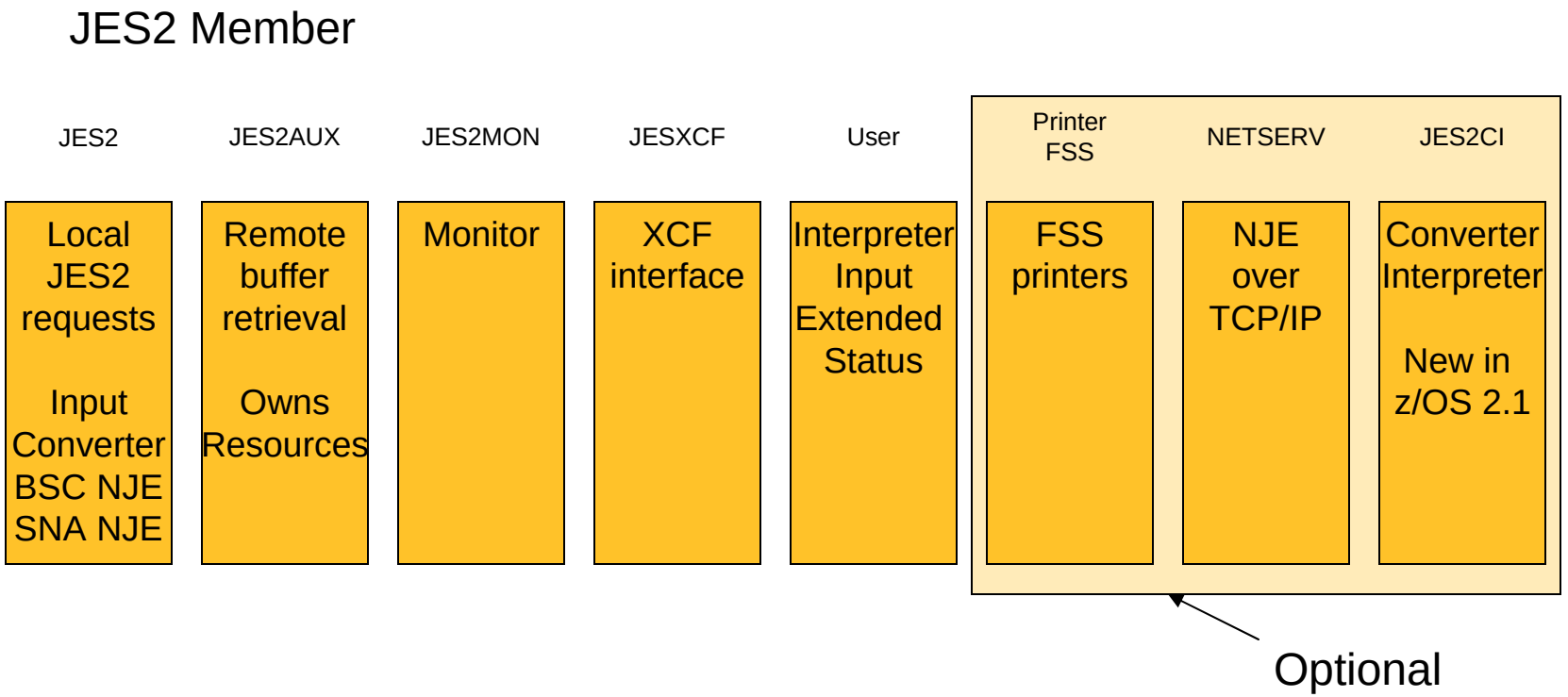
JES2 MAS



All members perform:

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

JES2 related address spaces



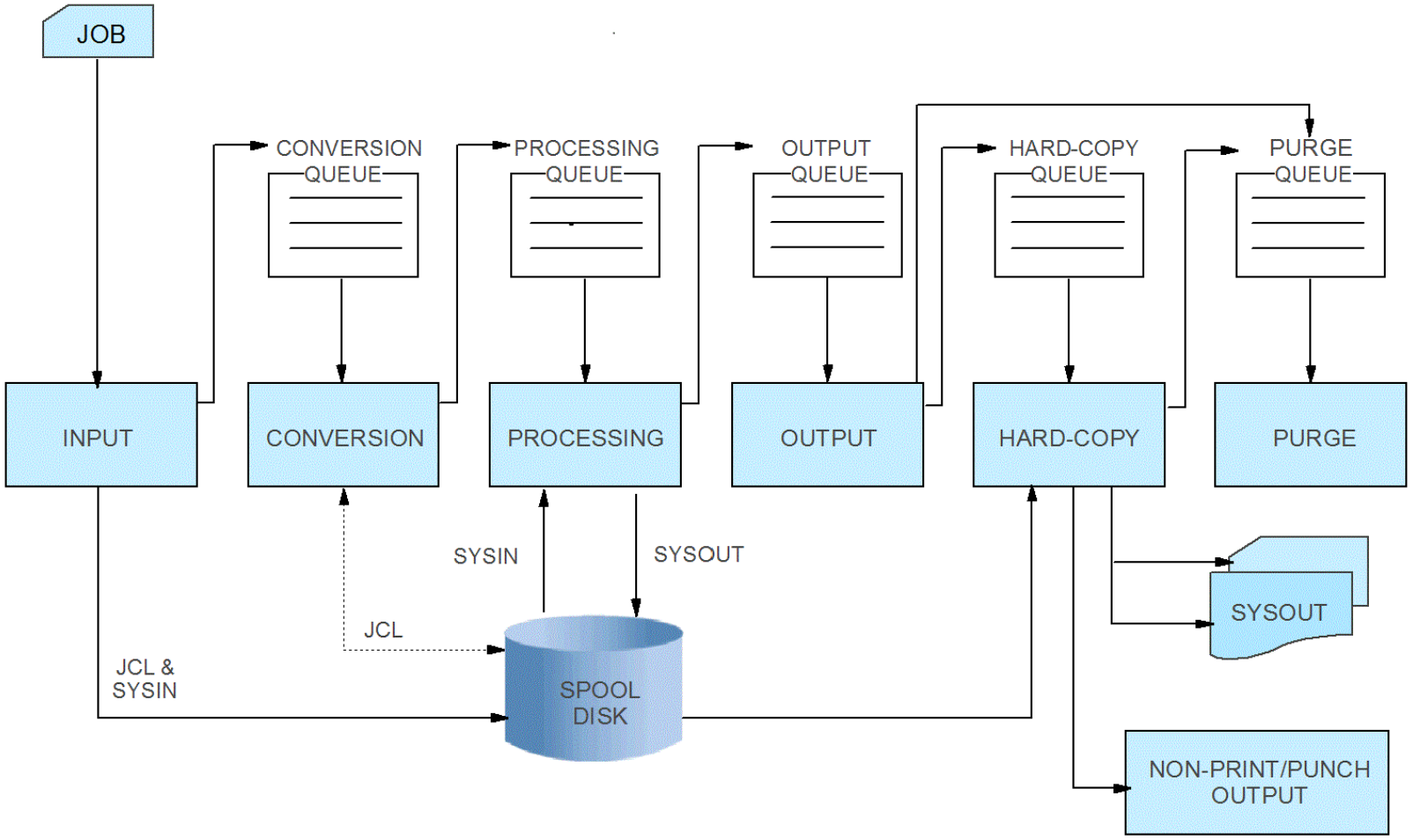
JES2 Structure

- JES2 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 JES2 address space to fail yet JES functions continue
- JESXCF services
  - Provide XCF communications between address spaces
    - Mainly used for enquiry/posting function for JES2
    - More messaging is done in recent releases
  - Manages status of JES2 address spaces
    - Notifies other members of a JES2 failure
- JES2 address space can terminate yet “JES2” is active
  - Address space is only part of JES2 function
  - Other address spaces and SSI survive when JES2 address space terminates
  - Can restart JES2 address space and resume full JES2 function
  - Can cleanly terminate JES2 when all address spaces using JES2 are terminated

JES2 PCEs

- JES2 Main task processing is performed by PCEs
  - Sub-dispatchable units of work running under main task
  - Cooperative dispatching (non-interruptible from a JES2 point of view)
    - Control given up by a $WAIT macro
    - SHOULD NEVER EVER MVS WAIT
    - Need to understand is a called service can MVS wait
- Most job phases are processed by a PCE type
  - Conversion, Execution, Output, Purge
- JES2 functions are also implemented by PCEs
  - Checkpoint, spool management, etc
- Table pairs can be used to define and created PCEs
  - Tables are defined in modules loaded via JES2 LOADMOD initialization statement
  - Can be dynamically created using $ADD LOADMOD command
- Table pairs can be used to extend other things in JES2
  - This is in addition to exit points
  - Installations can add entries to various processing tables dynamically
  - For example commands and initialization statements

Job Phases



JES2 Input Processing

- Internal reader processing occurs in submitter address space
  - Request sent to JES2 address space to get a job number, job structure (in checkpoint), and initial SPOOL space
  - JCL is parsed, SYSIN data sets created, and SPOOLed in submitter address space
  - Passed symbols are saved on SPOOL (JES2 2.1)
- Job completes input processing when
  - ENDREQ macro is issued
  - /*EOF card submitted
  - New job card is encountered
  - INTRDR “data set” is closed
- NJE/TCP processing is similar (occurs in NETSERV address space)
- Other input processing occurs in the JES2 address space (main task)
  - BSC and SNA NJE, RJE, SPOOL Reload, and card readers
- Single submission streams are limited by ability to access the JES2 checkpoint
  - Parallel streams can submit much faster (10 parallel streams are 10x one single stream)

JES2 Conversion Processing

- Converts JCL into internal format needed to run job
  - Two part process – Conversion and interpretation
- JCL interpretation can occur either
  - Before job starts execution in target address space
  - After conversion processing in the JES2CI address space
- Controlled by JOBDEF INTERPRET=JES|INIT
- In either case
  - PCE selects job and sets up environment
  - Subtask used to call z/OS converter and optionally interpreter
- Certain JCL error only detected by interpreter
- Converter parms (defaults) based on JOBCLASS settings
  - Journal, BLP, SMF exits, PROCLIB, region, SWA ABOVE, etc

Execution phase - 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

WLM and JES – Managed Initiators

- WLM 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 statement
- JES initiators
  - Type of initiator used based on a JOBCLASS MODE=
    - Applies to all members of the MAS
  - MODE=WLM JOBCLASS uses WLM initiators
    - MODE=WLM cannot be selected by JES2 initiators
    - Work selected by service class
  - MODE=JES JOBCLASS uses JES2 initiators
    - Initiators started and managed by operator commands ($SI)
    - Number of initiators defined at initialization
    - Work selected by an ordered list of job classes and/or job class groups
  - 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 job class

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.

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

Execution services

- JES is involved to provide services while a job executes
  - Creation of SYSOUT data sets
  - Job submission services
  - Job message logging
  - Other JES services (SSI functions)
- One special type of SYSOUT data set is a SPIN data set
  - Allocated separate from normal job output on spool
  - Queued to JES for output processing when closed/unallocated
  - Available to print while job is still running (separate JOE)
  - Purging data set (JOE) frees spool space

OUTPUT phase – SYSOUT Grouping

- The OUTPUT phase builds output groups from SYSOUT data sets
  - A SYSOUT group is defined as the set of data sets that prints between a set of job separator pages
  - Always for the same job and security information
  - For SAPI, between group begin and group end indicators
  - For SDSF, data sets in a row on the O or H panel
- Grouping based on various characteristics
  - SYSOUT class, forms, writer name, hold type, destination, security info, etc
- Print scheduling based on output group (JOE)
  - PSO conversational support is only exception
- SPIN data sets are never grouped with other data sets
- Can influence (prevent) grouping using JCL (GROUPID=)
- Re-grouping only done with SAPI
  - PSO for certain held data sets
- SYSOUT cloning can create multiple copies
  - /*JOBPARM COPIES=
  - Allows multiple copies of entire job output
    - ABC – ABC vs AABBCC
  - Cannot be re-grouped
  - Problem for SAPI/PSO

Hardcopy phase

- Parking place for job that have run and are waiting to print
  - No real actions taken on the job level
  - Processing is based on the output groups (JOEs)
- Once output for the job has been processed, job complete
  - Job queued to purge processing

Functional Subsystems (FSS)

- Interface to offload function to separate address spaces
  - Reduces workload on JES main task
  - Used in JES2 for printer support
  - Removes printer “driver” knowledge from JES2
    - Associates them with FSS software
  - JES managed interface
    - Controlled by JES commands

SYSOUT API (SAPI)

- A general purpose API to process output groups (JOEs)
  - Another way to implement a “printer”
  - Application manages the device
    - No JES2 definition or control for the device
  - Local/RJE destined output (no NJE data passed)
- Can also be used as a logical interface to manage output
  - Can chance certain characteristics at a data set level
  - Results in JOEs being re-grouped
    - Only SAPI can regroup any JOE’s data sets
    - PSO (an older API) can regroup certain held JOEs
- Very common SYSOUT interface (use growing)

Purge phase

- Frees spool space and deletes jobs and output groups
  - Both job and JOEs can be queued to purge
- Job purge processing
  - Jobs enter purge when all JOEs are processed (gone)
  - All track groups returned and control blocks frees
- Output group (JOE) purge processing
  - Used for SPIN data sets (JOEs)
  - Used to perform cleanup for data set purge
  - If appropriate, frees spool space
  - Control blocks are then freed

Is it JES or MVS?

- JCL processing:
  - JES processes
    - /* JES2 JECL cards
    - //XMIT cards
    - Instream (SYSIN) DD data
      - *Except in PROCs and INCLUDES*
    - Some keywords on JOB, DD, and JCLLIB cards
  - MVS converter process
    - All other JCL cards
  - PROCLIBs defined and OPENed by JES2
    - Read and processed by MVS converter
- SYSLOG
  - MVS writes to SYSLOG data set
  - JES stores data on spool
- JOBLOG data set (1st data set in output)
  - JES captures messages to place in JOBLOG
- System messages data set (3rd data set)
  - MVS writes messages to the data set

Is it JES or MVS?

- TSO TRANSMIT/RECEIVE
  - TSO formats data and writes it to spool
  - JES sends data to correct system
- RACF checking for new work
  - Started tasks and TSO logon is verified by MVS/TSO
  - Batch jobs are verified (VERIFYX) by JES
- TSO SUBMIT
  - Submit pre-processes JCL and writes it to JES
- DD SYSOUT=(A,INTRDR)
  - Each record written is processed by JES directly

Questions?

Questions?

Session 15325

Mainframe 50 April 7th 1964 - April 7th 2014
http://www.ibm.com/mainframe50/