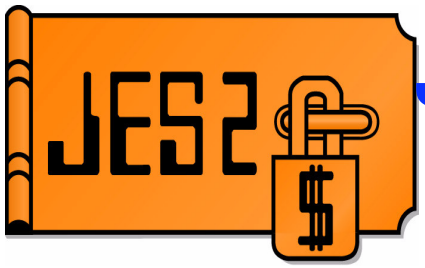
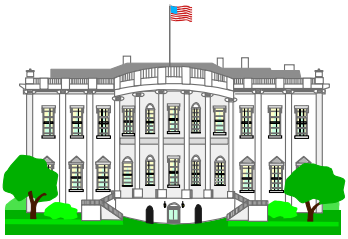


JES2 z/OS 1.2/1.4 Product Update



SHARE 101, Session 2655

Tuesday, August 12, 2003

Chip Wood
JES2 Design/Development/Service
Poughkeepsie, NY

chipwood@us.ibm.com

JES2 z/OS releases

- **JES2 z/OS 1.2**
  - Greater than 64K jobs support
  - Dynamic PROCLIB support
  - INCLUDE initialization statement
- **JES2 z/OS 1.4**
  - INCLUDE enhancements
  - JES2 Monitor
    - **Session 2657, Wed. 11:00**
  - HAM performance/restructure
  - End of Memory cleanup
  - // XMIT JCL
  - Miscellaneous enhancements

JES2 z/OS 1.4 installation

- From **JES2 OS/390 R3** or earlier
  - Migrate to more recent spool-compatible release first (preferably **R8)** to avoid **COLD** start
- From **JES2 OS/390 R4**, **R5**, or **R7**

  Note that **R7** and earlier releases are not supported on **z/OS 1.4** BCP (*enforced!!!*)
  - **$ACTIVATE** required to avoid **COLD** start (**R4**, **R5**)
  - No MAS coexistence (all-member-warm start)
- MAS coexistence from **OS/390 R8-z/OS 1.2**
  - APAR **OW52833** needed on downlevel member
  - **$ACTIVATE** required on **R8**

JES2
z/OS 1.2

>64K jobs support

- Internal limits increased
  - **JOBDEF JOBNUM** up to **200,000**
  - **JOBDEF RANGE** up to **999,999**
  - **OUTDEF JOENUM** up to **500,000**
  - **CKPTSPACE BERTNUM** up to **500,000**
  - **SPOOLDEF TGSPACE=MAX** up to **16,580,355**
- Limits can now be decreased by operator command
- Job id format changes if new limits exploited
  - **J***nnnnnnn*, **S***nnnnnnn*, **T***nnnnnnn*
- Lots of changes to JQE and JOE mappings

$ACTIVATE

- **$ACTIVATE,LEVEL=Z2**
  - LEVEL=Z2 is a required parameter
  - Allows new higher limits to be used
  - Rebuilds JIX
  - Changes job and output queue pointers in checkpoint from offsets to indices
  - Remaps JQEs
  - Builds additional checkpoint structures
  - Updates checkpoint level ($MSTRVER)

$ACTIVATE

- **$ACTIVATE,LEVEL=R4**
  - Reduces increased limits, if possible
  - Rebuilds JIX
  - Changes JQE and JOE indices in checkpoint back to offsets
  - Remaps JQEs
  - Updates checkpoint level ($MSTRVER)
- Also can all-member-warm or all-member-hot start with PARM=UNACT
- Fails if:
  - Job numbers above 65534 in use
  - JQEs/JOEs/BERTs/TGs in use above R4 limits
    - If not in use, limits are automatically reduced

Determining checkpoint level

- Some code must work differently based on whether $ACTIVATE level is z2 or R4
  - **$MSTRVER** in checkpointed HCT
    - **R4** - value is $MSTRVR4 (6)
    - **z2** - value is $MSTRV12 (7)

```
CLI    $MSTRVER,$MSTRV12     $ACTIVATE at z2 level
BL     NOTACT                Branch if not
BNL    ACT                   Branch if so
```

- Many macros/services are updated to make R4/z2 differences transparent
  - **$#JOE**, **$QJQE**, **$DOGJQE**, **$QLOC**
  - **$QINO**, **$QOIN**, **$#INO**, **$#OIN**

Field Changes

- New structures imply many field changes
  - JQE fields moved to create a 4 byte job number
  - JQE and JOE offsets converted to indices
  - Queue heads converted from offsets to index
    - SPOOL control blocks continue to use 4-byte offsets
    - High order byte is now significant
    - WLM service class queue heads remain indices
  - JQE fields are remapped

  ALL AFFECTED FIELDS RENAMED TO CAUSE ASSEMBLY ERRORS IF NOT UPDATED
  - Review migration manual for details

Examining job and output queues

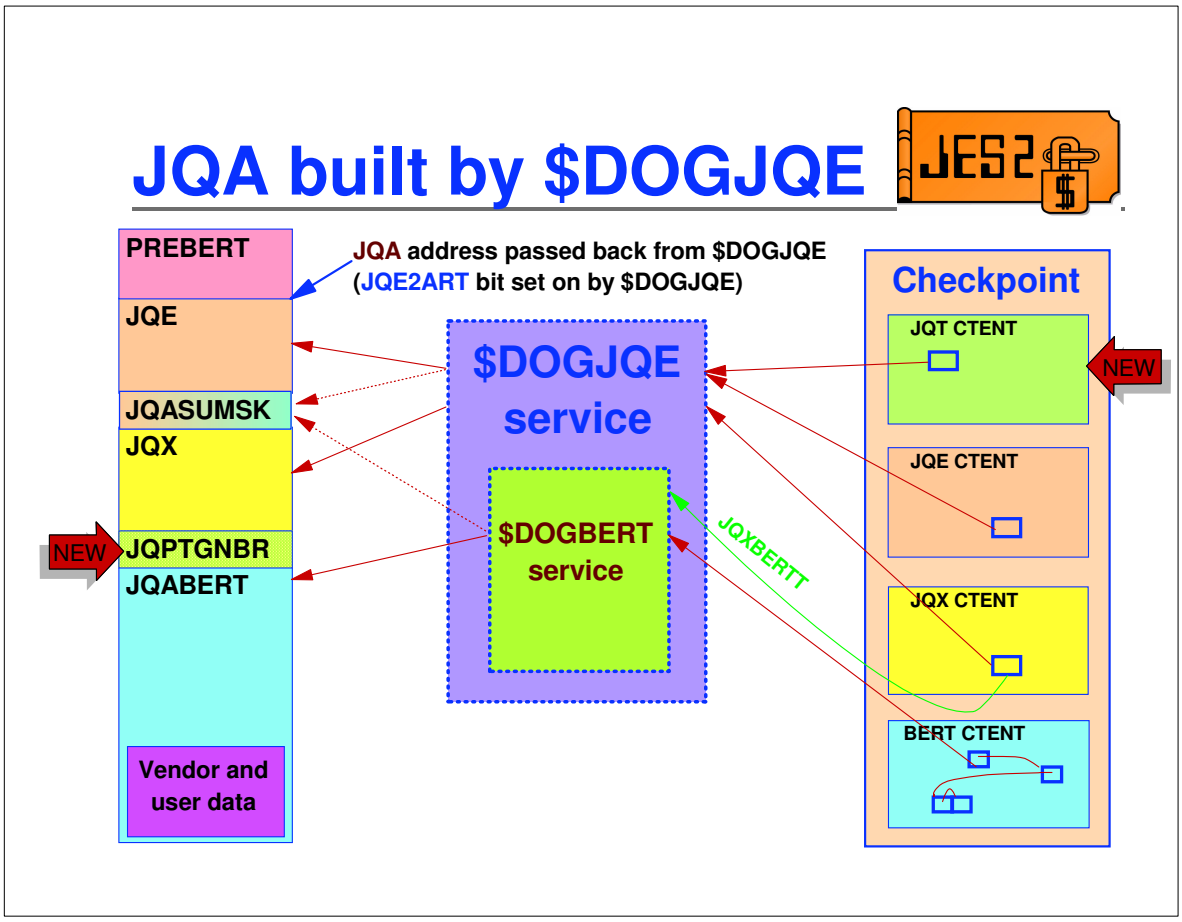
- Use macros to examine job/output queues in exits
  - Logic to determine whether offsets or indices are used is built in
  - **$QJQE** - runs job queues
    - WLM service class queues
    - Execution class queues
    - Special queues ($HARDCPY, CNVT, etc.)
  - **$#JOE** - runs output queues
    - SYSOUT class queues
    - Queues from JQE
    - Queues from Characteristics JOE
    - Special queues (Network, Hold, etc.)

Helper macros

- **$QINO** - JQE index to offset
  - Input: JQE index/offset (mode dependent)
  - Output: JQE offset
- **$QOIN** - JQE offset to index
  - Input: JQE offset
  - Output: JQE index/offset (mode dependent)
- **$#INO** - JOE index to offset
  - Input: JOE index/offset (mode dependent)
  - Output: JOE offset
- **$#OIN** - JOE offset to index
  - Input: JOE offset
  - Output: JOE index/offset (mode dependent)

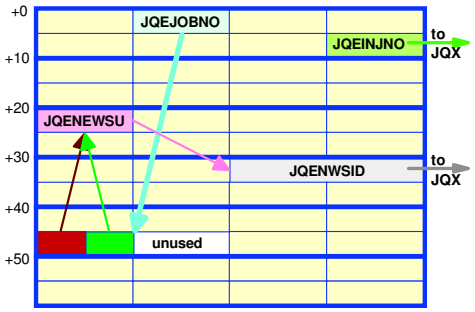
Job number fields

- SPOOL control blocks
  - 2 reserved bytes before ***xxx*JOBNO** reclaimed
  - New 4-byte field renamed to ***xxx*JBNUM**
- In-storage control blocks
  - Most are expanded and converted to 4-byte fields
- JQE
  - No room for 4-byte job number in R4 JQE
  - No room for 4-byte initial job number in R4 JQE
  - **$ACTIVATE** moves some fields around in JQE
    JQAs **ALWAYS** map using Z2 mode JQE mapping

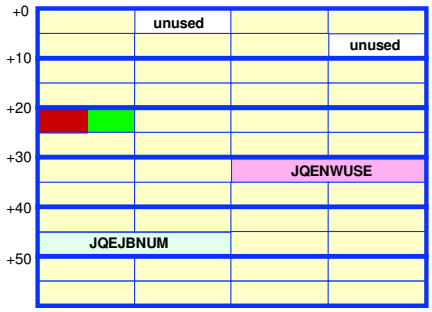

JQA built by $DOGJQE
PREBERT
JQE
JQASUMSK
JQX
NEW
JQPTGNBR
JQABERT
Vendor and user data
JQA address passed back from $DOGJQE
(JQE2ART bit set on by $DOGJQE)
$DOGJQE service
$DOGBERT service
JQXBERTT
Checkpoint
JQT CTENT
NEW
JQE CTENT
JQX CTENT
BERT CTENT

JQE field remapping

JQE mapping (R4 mode)



JQE mapping (z2 mode)



$ACTIVATE,LEVEL=Z2 - remaps R4 fields to z2 fields
$ACTIVATE,LEVEL=R4 - remaps z2 fields to R4 fields

$DOGJQE always copies data to z2 fields in artificial JQE (JQA), regardless of current mode

JQE Field remapping

- JQEJOBNO, JQEINJNO, and JQENEWSU all require increase from 2 to 4 bytes
  - **JQEJOBNO -> JQEJBNUM**
  - **JQEINJNO -> JQXIJNUM**
  - **JQENEWSU -> JQENWUSE**
- Moving these fields displaces other rarely referenced fields
  - **JQEARMID -> JQEARMMI**
  - **JQEWSLOK -> JQEWSLCK**
  - **JQENWSID -> JQXNWSID**
- Always use new field names in JQA
- Use new fields in real JQE in z2 mode
- Use old fields in real JQE in R4 mode
  - Reference by old field name + **_R4**

JQE field remapping

- Current job number is the only commonly referenced field
- Use new **$JQEJNUM** macro to find job number

```
$JQEJNUM  JQE=<jqe address>,
          REG=register
```

- **$JBIDBLD** macro allows new **JQE=** specification
  - Uses $JQEJNUM internally to get job number from JQE

JOBID format

- JOBID format changed based on upper limit of **JOBDEF RANGE=** high value
  - < 100,000 then format unchanged (**JOB***nnnnn*)
  - >= 100,000 then format is **J***nnnnnnn*
    - **STC***nnnnn* becomes **S***nnnnnnn*
    - **TSU***nnnnn* becomes **T***nnnnnnn*
- Note: transition period if job number range increased above 99,999 via $T command
  - SPOOL and running jobs will have old format
  - Operator commands new format
  - SMF records could contain either format
  - Transition period also exists when decreasing range

What could be affected by jobid changes?

- SMF records (6, 24, 26, 57, 90, and others)
- SMF exits
- JMR and JSAB
- MPF exits and automation scripts
  - Jobid in WQE (WQEOJBID)
- SAF profiles - entity names for spool data sets
- CANCEL and STATUS commands
- Applications using CANCEL, STATUS, PSO, Extended Status, or SAPI SSI calls
- CLISTs, EXECs, or panels that generate an interface with any of the above
- Who knows what else??????

Converting jobids

- **$JBIDBLD** macro
  - Converts binary job number to EBCDIC job id
  - Understands all rules for which form to use
  - Input:
    - **JQE=**, **JOBNUM=**, or **JNUMREG=**
    - **JOBTYPE=** - determines whether J, S, or T
    - Optional with **JQE=**
  - Output:
    - **JOBID=** - 8 byte field for job id
  - Example:

```
$JBIDBLD JQE=(R6),       JQE address in register 6
       JOBID=$DOUBLE     Return job id in field $DOUBLE
```

Converting jobids

- TSCNVJB service ($CALL)
  - Converts EBCDIC job id to binary job number
  - Understands all forms of job id
  - Input:
    - Register 0 - length of job id field
    - Register 1 - address of job id field
  - Output:
    - Register 0 - binary job number
  - Example:

```
$CALL  TSCNVJB,              Convert job id to binary
       PARM0=L'JCTJOBID,     Length of job id field
       PARM1=JCTJOBID        Address of job id field
```

JOBID format

- Jobs from NJE or spool offload can use new job id format regardless of **JOBDEF RANGE** if:
  - Original job number >**99,999**
  - **JOBDEF RASSIGN=YES**
  - Original job number available
- If this is a problem
  - Set **JOBDEF RANGE** to desired range
    - 1-99999 for JOBxxxxx job ids
    - 1-65535 if halfwords used for job numbers
    - 1-32767 if signed operations are used on fields (LH vs. ICM, etc.)
  - Set **JOBDEF RASSIGN=NO**

Data structures - JIX

- **JIX** - **J**ob **I**nde**x** Table
  - R4 mode - 32K or 64K 2-byte entries - 1 per possible job number
    - Index into table by job number
    - 2 byte index of job for that job number
  - Z2 mode - 64K 4-byte entries - hash table
    - Hash value is low 2-bytes of job number
    - Chained via **JQXNJIXI** (3-byte index)
- Change is transparent if **$QLOC** used

$QLOC performance

- Pre-z2 - Many processes used $QLOC in a loop to find all jobs in system
  - Much slower now that max job number is **999,999** (and could increase to **9,999,999** in the future)
- Solution 1:
  - Use **$QJQE** to run all queues (if job number is not important)
- Solution 2:
  - New **$QLOCNXT** macro to locate the job with the next higher job number (if order is important)
    - Input: **JOBNUM=** or **JQE=**
    - Output: Real JQE address of job with next highest job number

Data structures - JQT

- Prior to z2:
  - Master checkpoint record contained 2000 "JQE extensions"
  - Contains count of track groups used by executing job
    - Prevents checkpoint of JQE every time a track group is allocated
    - Reduces amount of checkpoint data written
  - **JQETGNUM** contains either:
    - Count of track groups (high bit off)
    - Offset of JQE extension (high bit on)
  - Overflow bit (**JQE6TGAE**) turned on at 32K Track groups
  - **Problem:** More extensions are needed

Data structures - JQT

- Prior to **$ACTIVATE,LEVEL=Z2,** same structure used
  - **JQETGNUM** renamed to **JQETGNBR**
- After **$ACTIVATE,LEVEL=Z2**
  - New JQT checkpoint CTENT with 20000 entries is used
    - JQT contains TG count modulo 32768
  - **JQETGNBR** contains
    - Index of JQT (high bit on)
    - Track group count modulo 32768 (high bit off)
  - **JQXTGRP** contains number of times count has wrapped in either case
  - **JQE6TGAE** set after **8,388,607** track groups

APPLCOPY

- Support for application copy of checkpoint (**APPLCOPY**) has been discontinued
  - Impractical for new large checkpoint sizes
  - **$HASP003** message if **CKPTDEF APPLCOPY=PRIVATE** or **APPLCOPY=COMMON** is specified
  - Use checkpoint versions (SSI 71) instead

HASX05C

- JES2 sample command translation exit **HASX05C** has been moved to SHASSAMP
  - Not automatically loaded or enabled

    Functionality of exit has not changed
  - Will continue to ship in SHASSAMP for the foreseeable future
- See DOC APAR OW51031 for more information

Dynamic PROCLIB

- **Problem:** PROCLIBs defined in the JES2 start proc require a JES2 restart to change
  - Change may require ALL MAS members to be restarted
  - Error in JES2 PROC may prevent restart
  - SHARE requirement ***SS-JES2-98.203***
- **Solution:** Allow dynamic allocation of PROCLIBs
  - **PROCLIB(*xxxx*)** initialization statement
  - **$ADD PROCLIB(*xxxxxxxx*)** command
  - **$T PROCLIB(*xxxxxxxx*)** command
  - **$DEL PROCLIB(*xxxxxxxx*)** command
  - **$D PROCLIB(*xxxxxxxx*)** command

INCLUDE statement

- **Problem:** Changing JES2 init deck concatenation requires changing JES2 PROC
  - If update is incorrect, JES2 will not start
  - May be difficult to fix when JES2 is down
- **Solution:** New INCLUDE initialization statement
  - Reduces need to update JES2 PROC

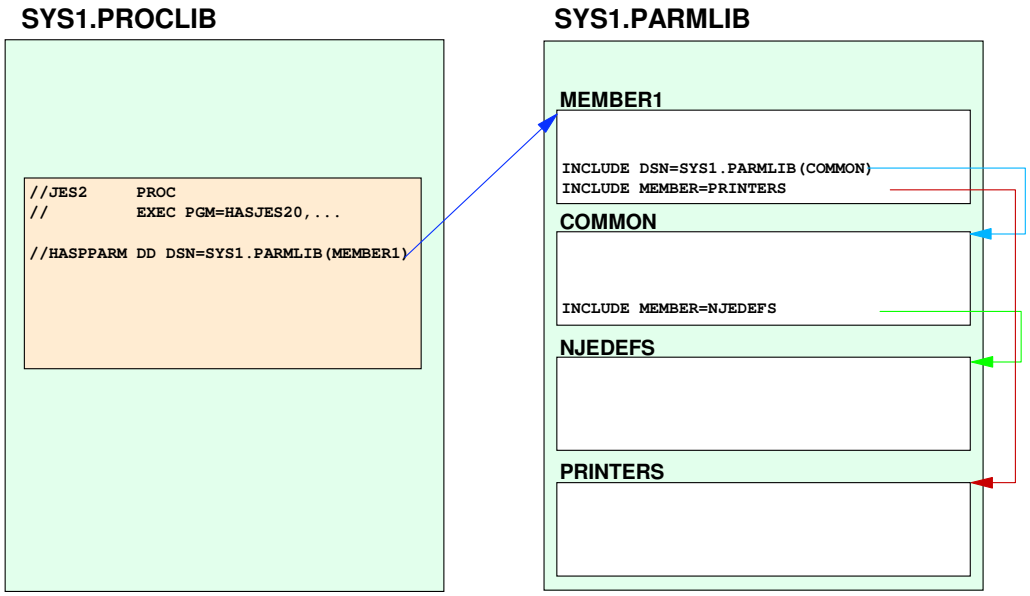
JES2
z/OS 1.4

Enhanced INCLUDE Externals

- Updates to INCLUDE initialization statement (new in z2) based on customer input
  - Current syntax of the INCLUDE init statement
    - **INCLUDE DSNAME=***dsn*,**VOLSER=***vol*,**UNIT=***unit*
  - New syntax added in z4
    - **INCLUDE MEMBER=***member*
      - *member* should be in current parmlib data set being processed
    - **INCLUDE PARMLIB_MEMBER=***member*
      - *member* should be in default logical parmlib
      - Specifying **MEMBER=** in a dataset included by **PARMLIB_MEMBER=** results in logical parmlib search, not just single dataset

Include statement example

- New way:



Default Parmlib Member

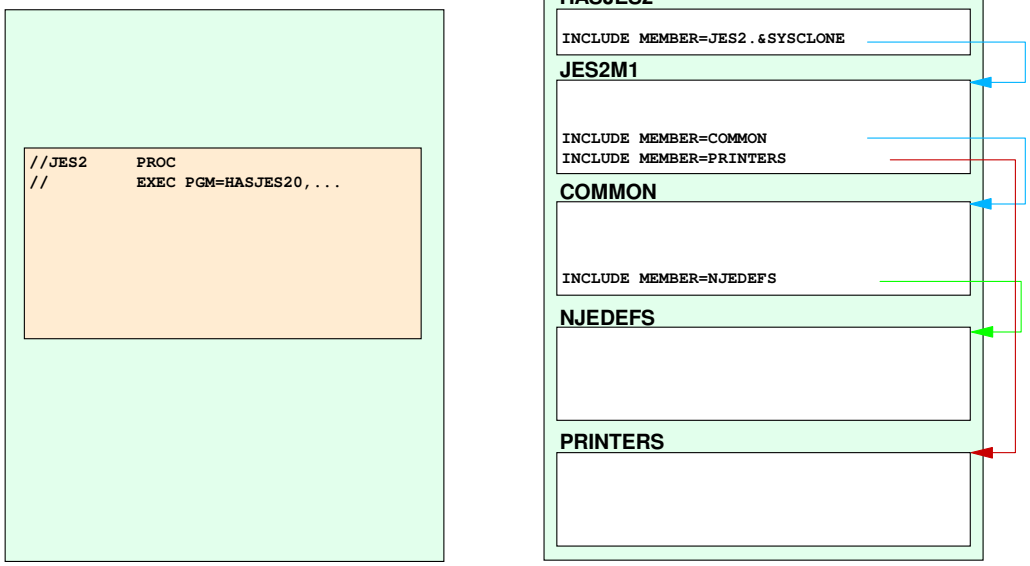
- New start option (to read from default PARMLIB)
  - **S JES2,PARM=('MEMBER=*member*')** or
  - **S JES2,PARM=('PARMLIB_MEMBER=*member*')**
  - Reply **MEMBER=***member* to **$HASP467**
  - *member* should be the member of logical parmlib in each of these cases
  - **HASPPARM=***ddname* and **MEMBER=** are mutually exclusive
  - If neither **HASPPARM=** nor **MEMBER=** is specified, then it will process from the default **HAS*jesx*** member of logical parmlib (***jesx*** is subsystem name)
    - IBM does not ship a default parmlib member

Search order

- If **HASPPARM=*ddname*** parameter is specified, use that DD
- If **MEMBER=/PARMLIB_MEMBER=** parameter is specifed, use that member from logical parmlib
- If neither is specified
  - Try to open DD with ddname of **HASPPARM**
  - If **HASPPARM** DD is not found, use **HAS*jesx*** member from logical parmlib

Include statement example

- New way:



Combining PROCLIB and INCLUDE

- Simplify JES2 PROC
  - PROC and EXEC statements only
  - Define PROCLIBS via PROCLIB statement
  - INCLUDE HASPPARM datasets
- In emergency, **start JES2 without a PROC!**

```
S IEESYSAS,PROG=HASJES20,JOBNAME=JES2
```

  - Assumes HASJES20 in LINKLIST (no STEPLIB)
  - When **HASPPARM** open fails, logical parmlib member **HASJES2** will be used
  - If **HASJES2** not found, **$HASP469** issued
  - Security via RACF **STARTED** class, profile **IEESYSAS.JES2** (procname.jobname)

JES2 Monitor

- JES2 "Health" Monitor
  - Not a "performance" monitor (yet)
  - Some basic performance information is tracked
  - Intent is to surface **VERY SEVERE** performance problems (JES2 not responding, etc.)
- Separate address space
  - Name is ***jesx*MON** (***jesx*** - subsystem name)
  - Samples JES2 address space to determine current status of JES2 main task and resource utilization
  - Anomalies surfaced as "alerts" if not resolved in a few seconds (MVS WAITs, long running PCEs, etc)
  - **$J***xxx* commands to display when JES2 is sick
- **Session 2657, Wednesday 8:00**

HAM I/O Improvements

- **Problem:**
  - HAM I/O is single record (except FSS)
  - HAM code is old with RAS problems
- **Solution:**
  - Use multi record I/O to read and write SYSOUT and SYSIN data from/to SPOOL
  - Use EXCPVR to reduce overhead
  - Improve overall RAS
  - (Almost) No external changes
  - Massive internal changes

HAM internal changes

- Internal reader processing was unchanged
- SVC 111 only used for internal reader
  - PC routines used in all other cases
- New SDB lock to serialize all PC services
- EXCPVR used instead of EXCP
- Format 1 CCWs used (31 bit CCW)
- I/O associated with job step TCB not current
  - Less code needed at EOT
- SDBs are above the line
  - DEB pointer to SDB is 31 bit address shifted 7 bits
  - SDBs are obtained on appropriate boundary
- PUT no longer uses "unprotected" buffer

$EXIT 9

- Records are counted for each PUT, not each full buffer
- Exit 9 gets control at exact excession count, not first full buffer after excession
- "Should" not affect most exits

Count on entry (pre-z4)	Count on entry (z4)
4039	4001
8067	8001
12095	12001
16017	16001
20045	20001
24073	24001
28101	28001
32023	32001
36051	36001

End of Memory (EOM)

- **Problem:**
  - The BCP began terminating EOM TCBs in z2 if they took too long to finish without making progress (abend 30D)
  - JES2's EOM activities entailed WAITing for the JES2 main task.
  - The JES2 main task could be down (JES2 abend) or awaiting access to the checkpoint
  - JES2 EOM could be abended resulting in lost (JES2) resources
- **Solution:**
  - (z2) Set STIMER in EOM to fake out 30D
  - (z4) Don't wait for JES2 main task from EOM processing

EOM Line item side effect

- JES2 no longer checks TSO logons for duplicate logons. The same userid can be logged on more than once in a JESplex.
- Installation controls this by making **SYSIKJUA** be a **SYSTEMS** ENQ
  - New routine in sample exit **HASX44A** can be enabled to restore checking to HASPCNVT
- Multiple instances of a single userid logged onto a JESplex is **not officially supported**.

EOM Line item side effect *(continued)*

- What will happen if multiple logons of same userid?
  - TSO GR (Generic Resource) will fail to determine what MVS should be used for reconnect.
  - An MCS **SEND** command to a given user will work if user is logged onto same MVS image as SEND command. The message will go only to the "local" instance
  - An MCS **SEND** command will not go to any of the multiple instances if none are logged onto the "local" MVS image
  - Notify messages will only be sent to first instance found
  - Other potential serialization issues

XMIT JCL card

- JES2 now supports the // **XMIT** JCL card to route job execution to another node.
  - Previously only supported by JES3
  - Requirement ***SO-JES2-93.006***
  - The **SUBCHARS=** keyword is not supported by JES2 (JCL error)
  - **/*XMIT**, **/*XEQ** and **/*ROUTE XEQ** still supported

Invalid Jobname Character

- Invalid JOBNAME character now specifiable
  - Requirement ***SO-JES2-97.203***
  - Was always a **?**
    - For displays, treated as generic
    - **$DJOBQ(*?*)** does not display all jobs with a bad jobname character
  - **JOBDEF BAD_JOBNAME_CHAR=**
    - Can now be **? / + : _ - !** or alpha/numeric/special
    - Only affects newly arriving jobs
  - Example
    - **JOBDEF BAD_JOBNAME_CHAR=!**
    - **$DJOBQ(*!*)** displays all jobs with an !

$TRACE 32

- **$TRACE 32** - $#REM
  - Traces every $#REM call
  - Use to determine why output is disappearing unexpectedly
  - Dumps JOE contents in hexadecimal and EBCDIC

```
14.47.51.42  ID =  32 $#REM    JOB00009   COMM      06B7B3C8  JOE REMOVAL
       0  80000000 C1000006 00000000 00000005   20000000 48000005 00000000 02000000  *....A..
      20  00000000 09000000 0C001005 0000001D   00000000 00000000 00000000 0B001004  *.......
      40  00000000 00010000 F1404040 40404040   00010001 B7F86252 4E4E4E4E 4E4E4E4E  *.......
      60  00000000 00000000                                                         *.......
```

$TRACE 33

- **$TRACE 33** - NJE Header/Trailer
  - Traces every NJE Header or Trailer or NMR
  - No need to decode $TRACE 4 or 5
  - Specify **TRACE=YES** on **OFFLOAD**, **LINE**, or **NODE**

```
14.58.14.14  ID =  33 NJEHDR   STC00011   L19.ST1   06B17180  TRANSMIT JOB HEADER
JOB HEADER PREFIX
           NJHLEN=0170 NJHFLAGS=00 NJHSEQ=00
JOB HEADER GENERAL SECTION
           JOBNAME=DEALLOC  JOB NUMBER=11  ORIGIN=NODE1    """""""" EXECUTION=NODE1
         0  00D40000 000BD0C1 400F0F01 00000000   00000000 00000000 C4C5C1D3 D3D6C340  *.M....
        20  00000000 00000000 00000000 00000000   00000000 00000000 B7F8646C A62E3245  *......
        40  D5D6C4C5 F1404040 00000000 00000000   D5D6C4C5 F1404040 40404040 40404040  *NODE1
        60  D5D6C4C5 F1404040 40404040 40404040   D5D6C4C5 F1404040 40404040 40404040  *NODE1
        80  E2E3C440 40404040 00000002 00015180   3B9AC618 000F423F 40404040 40404040  *STD
        A0  40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *
        C0  40404040 0000001D 0000000B 00000000   00000000                            *    ..
JES2 SECTION OF THE JOB HEADER
         0  00348400 01000000 00000000 00000000   00000000 00000000 00000000 00000000  *..d...
        20  00000000 00000000 00000000 00000000   00000000                            *......
JOB SCHEDULING SECTION OF THE JOB HEADER
         0  000C8A00 000F423F 7FFFFD78                                                *......
SECURITY SECTION OF THE JOB HEADER
         0  00588C00 00048000 50012006 C0010000   00000000 00000000 D5D6C4C5 F1404040  *......
        20  00000000 00000000 D5D6C4C5 F1404040   00000000 00000000 E2E3C3C9 D5D9C4D9  *......
        40  00000000 00000000 4E4E4E4E 4E4E4E4E   00000000 00000000                   *......
```

Miscellaneous changes

- **INITDEF PARTNUM=0** now valid
  - No JES initiators defined
- JES2 now passes a count of jobs that can run only on constrained systems in WLM sampling data.
  - WLM can better decide whether to start initiators on constrained systems
  - **$DPERFDATA(SAMPDATA)** displays count

```
$HASP660 $DPERFDATA(SAMPDATA)
$HASP660 SERVICE CLASSES KNOWN TO JES2:
$HASP660 SRVCLASS(WLMSHORT)=(TOKEN=16748000,REGISTERED,SYSTEMS=
$HASP660 (AQFT,AQTS))
$HASP660 SERVICE CLASS SAMPLING DATA:
$HASP660 SRVCLASS(22)=(SYS_QUEUE=0,SYS_INEL=9,SYS_LIMIT=0,
$HASP660 LOCAL_QUEUE=0,CONSTRAINT_AFFINITY=0,LOCAL_INEL=9)
$HASP660 REPORT CLASS SAMPLING DATA:
```

Appendix

Data transformations (JQE chaining)

- JQE offsets fields changed to JQE indexes

Old name	New name	Length
$JQFREE in $HCT	$JQFREEI	4 bytes
$JQHEADS in $HCT	$JQHEADI	47*4 bytes
$JQRBLD in $HCT	$JQRBLDI	4 bytes
JOEJQEB in $JOE	JOEJQEI	3 bytes
JQENEXTB in $JQE	JQENEXTI	3 bytes
CATQHEAD in $CAT	CATQHDI	4 bytes

Data transformations (JOE chaining)

- JOE offsets fields changed to JOE indexes

Old name	New name	Length
JOTFREQ in $JOT	JOTFREQI	4 bytes
JOTCHRQ in $JOT	JOTCHRQI	4 bytes
JOTPURGQ in $JOT	JOTPRGQI	4 bytes
JOTHOLDQ in $JOT	JOTHLDQI	4 bytes
JOTCLSQ in $JOT	JOTCLSQI	3*36 bytes
JOTNTWKQ in $JOT	JOTNTWQI	4 bytes
JOTRDYWQ in $JOT	JOTRDWQI	4*JOTNUMWQ
JOTRBLDQ in $JOT	JOTRBLQI	4 bytes
JQEJOEB in $JQE	JQEJOEI	3 bytes
JOENEXTB in $JOE	JOENEXTI	3 bytes
JOEPREVB in $JOE	JOEPREVI	3 bytes
JOEJQNXB in $JOE	JOENXJQI	3 bytes
JOECHARB in $JOE	JOECHARI	3 bytes
JOECHNXB in $JOE	JOECHNXI	3 bytes
JOEWKPTB in $JOE	JOEWKPTI	3 bytes
JOENETCH in $JOE	JOENETCI	4 bytes

Data transformations (Job numbers)

- Job number fields changed from 2 to 4 bytes

Old field name	New field name	Control block
CRXJOBNO	CRXJBNUM	$COMWORK
DCNVJBNO	DCNVJNUM	$DTECNV
FAXBCJP	FAXBJCJP	$FSAXB
GTWJQNUM	GTWJBNUM	$GTW
GTWJQEMX	GTWJQMAX	$GTW
GTWJQEFR	GTWJQFRE	$GTW
JIBJOBNO	JIBJBNUM	$JIB
JNEWJQE	JNEWJNUM	$JNEW
PSOJOBNO	PSOJBNUM	$PSO
ROTEJBNR	ROTEJNUM	$ROTT
SFRJBNO	SFRJBNUM	$SFRB
SSWJOBNO	SSWJBNUM	$SFSWORK
SJBJOBNO	SJBJBNUM	$SJB
TTEJOBNO	TTEJBNUM	$TTE

Data transformations (Job numbers)

- Job number equates changed from 2 to 4 bytes

Old name	New name	New value	Control block
DSIDJBNO	DSIDJNUM	EQU 0,4	$FSIEQU
$MAXJBNO	$MAXJNUM	EQU 999999	$HASPEQU
$MAXJQES	$MAXNJQE	EQU 200000	$HASPEQU
COFSEC	COFSEC	COFOPT2+1,4	$COMWORK

Data transformations (Job numbers)

- 2 byte job number fields in SPOOLed control blocks changed to use preceeding 2 byte reserved fields

Old field name	New field name	Control block
HDBJOBNO	HDBJBNUM	$BUFFER
CHKJOBNO	CHKJBNUM	$CHK
IOTJOBNO Note: also delete the IOTJBNMB equate	IOTJBNUM	$IOT
JCTJOBNO	JCTJBNUM	$JCT
NHSJOBNO	NHSJBNUM	$NHSB
OCTJOBNO	OCTJBNUM	$OCT
SWBJOBNO	SWBJBNUM	$SWBIT
&S.JOBNO	&S.JBNUM	$SPID

Data transformations (Job numbers)

- 2 bytes job number fields moved to create 4 byte fields

Old 2 byte field name	New 2 byte field name	4 byte field name	Control block
DASJOBNO	DASJOBNO_R4	DASJBNUM	$DAS
$MAXJOBS	$MAXJOBS_R4	$JQENUM	$HCT
$JQEFREC	$JQEFREC_R4	$JQEFRCN	$HCT
JCTINJNO	Field deleted	Field deleted	$JCT
JQEJOBNO	JQEJOBNO_R4	JQEJBNUM	$JQE
JQEINJNO	JQEINJNO_R4	JQXIJNUM	$JQE

- Fields moved to create 4 byte fields

Old field name	New field name (R4 mode)	New field name (z2 mode/JQA)	Control block
JQEARMID	JQEARMID_R4	JQEARMMI	$JQE
JQEWSLOK	JQEWSLOK_R4	JQEWSLCK	$JQE

Data transformations (JESNEWS)

- Fields updated to support JESNEWS

R4 field name	Change	Control block
JOEJNEWS	Name changed to JOEJNEWL. In R4 mode, job number. In z2 mode, JESNEWS level	$JOE
JQENEWSU	JQENEWSU_R4 in R4 mode (2 bytes) JQENWUSE in z2 mode (4 bytes)	$JQE
JQEJOEID (for JESNEWS JQE only)	JQENWSID_R4 in R4 mode JQXNWSID in z2 mode	$JQE
JNEWJQE	Name change to JNEWJNUM (4 bytes)	$JNEW
JNEWLEVL	Size changed to 4 bytes	$JNEW

$QJQE operands

- **LOOP=**
  - Label generated within macro expansion
  - Branch here to get next JQE
- **NOMORE=** - Where to go when end of queue is reached
- **RX=** - Register for JQE or JQA address
- **MODE=**
  - **READ** - Read mode JQA
  - **REAL** - Real JQEs (in performance paths only)
- One of the following
  - **TYPE=**, **CLASS=**, **SRVCLASS=**, **CAT=**, **WSCQ=**
- All these options exist prior to z2

$QJQE example

- To look at all the jobs on a specific execution class queue:

```
          $QJQE  CLASS=<8-byte-field>,
                 REG=Rx,           Register for JQE address
                 MODE=READ,
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=LABEL2     Where to go at end of queue

           USING JQA,Rx
.
.          <process JQE for job>
.
           B     LABEL1            Loop for more (LOOP=)

LABEL2     DS    0H                Done with all JQEs (NOMORE=)
```

$QJQE example

- To look at all the jobs on a specific WLM service class queue:

```
          $QJQE  SRVCLASS=<8-byte-field>,
                 REG=Rx,            Register for JQE address
                 MODE=READ,
                 LOOP=LABEL1,       Label to keep looping
                 NOMORE=LABEL2      Where to go at end of queue

           USING JQA,Rx
.
.          <process JQE for job>
.
           B     LABEL1             Loop for more (LOOP=)

LABEL2     DS    0H                 Done with all JQEs (NOMORE=)
```

$QJQE example

- To look at all the jobs on a specific non-execution queue:

```
          $QJQE  TYPE=CNVT,        Conversion queue
                 REG=Rx,           Register for JQE address
                 MODE=READ,
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=LABEL2     Where to go at end of queue

           USING JQA,Rx
.
.          <process JQE for job>
.
           B     LABEL1            Loop for more (LOOP=)

LABEL2     DS    0H                Done with all JQEs (NOMORE=)
```

$QJQE example

- To look at all the jobs on all execution class queues:

```
         SLR     Ry,Ry             No previous CAT
NEXTCAT  $DOGCAT ACTION=FETCHNEXT,
                 CAT=(Ry),         Previous CAT
                 ERRET=LABEL2      Where to go if no more CATs
         LR      Ry,R1             Copy CAT address
         $QJQE   CAT=(Ry),
                 REG=Rx,           Register for JQE address
                 MODE=READ,
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=NEXTCAT    Where to go at end of queue

          USING JQA,Rx
.
.         <process JQE for job>
.
          B      LABEL1           Loop for more (LOOP=)

LABEL2    DS     0H               Done with all JQEs
```

$QJQE example

- To look at all the jobs on all queues:

```
         SLR     Ry,Ry             No previous CAT
NEXTCAT  $DOGCAT ACTION=FETCHNEXT,
                 CAT=(Ry),         Previous CAT
                 ALLQUES=YES,      or... ALLQUES=(YES,REBLD)
                 ERRET=LABEL2      Where to go if no more CATs
         LR      Ry,R1
         $QJQE   CAT=(Ry),
                 REG=Rx,           Register for JQE address
                 MODE=READ,
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=NEXTCAT    Where to go at end of queue

          USING JQA,Rx
.
.         <process JQE for job>
.
          B      LABEL1           Loop for more (LOOP=)

LABEL2    DS     0H               Done with all JQEs
```

$#JOE operands

- **LOOP=**
  - Label generated within macro expansion
  - Branch here to get next JQE
- **NOMORE=**
  - Where to go when end of queue is reached
- **RX=**
  - Register for JOE address
- One of the following
  - **Q=**, **JQE=**, **CHAR=**
- Many of these operands are NEW in z2

$#JOE example

- To look at all the JOEs associated with a particular JQE:

```
          $#JOE  JQE=<jqe-address>,
                 REG=Rx,           Register for JOE address
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=LABEL2     Where to go at end of queue

           USING JOE,Rx
.
.          <process JOE>
.
           B     LABEL1            Loop for more (LOOP=)

LABEL2     DS    0H                Done with all JOEs (NOMORE=)
```

$#JOE example

- To look at all the Work-JOEs associated with a particular Characteristics-JOE:

```
          $#JOE  CHAR=<char-JOE-address>,
                 REG=Rx,           Register for JOE address
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=LABEL2     Where to go at end of queue

           USING JOE,Rx
.
.          <process JOE>
.
           B     LABEL1            Loop for more (LOOP=)

LABEL2     DS    0H                Done with all JOEs (NOMORE=)
```

$#JOE example

- To run a special JOE queue (hold, network):

```
          $#JOE  Q=JOTHOLDI,       Hold queue
                 REG=Rx,           Register for JOE address
                 LOOP=LABEL1,      Label to keep looping
                 NOMORE=LABEL2     Where to go at end of queue

           USING JOE,Rx
.
.          <process JOE>
.
           B     LABEL1            Loop for more (LOOP=)

LABEL2     DS    0H                Done with all JOEs (NOMORE=)
```

$#JOE example

- To look at Work JOEs on a specific SYSOUT class queue:

```
          $#JOE  Q=<address of SYSOUT class>,
                 DEST=LOC,          Local queue only
                 REG=Rx,            Register for JOE address
                 LOOP=LABEL1,       Label to keep looping
                 NOMORE=LABEL2      Where to go at end of queue

           USING JOE,Rx
.
.          <process JOE>
.
           B     LABEL1             Loop for more (LOOP=)

LABEL2     DS    0H                 Done with all JOEs (NOMORE=)
```

  - Requires multiple calls (DEST=LOC, DEST=RMT, DEST=USE) to run all queues for a class

JOE to JQE

- To find the JQE associated with a particular JOE:

```
          ICM   R5,B'0111',JOEJQEI  Get JQE index/offset
          N     R5,$ZEROFFF         Clear hi-order byte
         $QINO  R=R5                Index to offset
+         CLI   $MSTRVER,$MSTRV12   Br if already
+         JL    INO2830               have an offset
+         MH    R5,$JQELEN              else index->offset
+INO2830  DS    0H
          AL    R5,$JOBQPTR         Convert offset to address
```

  - **$QINO**:
    - Determines, based on mode, whether the value is an offset or an index
    - If an index, converts to an offset