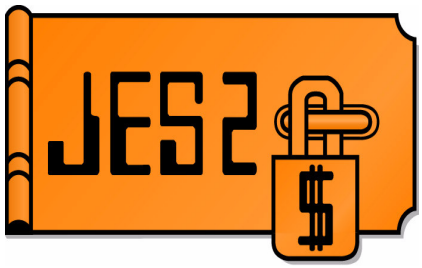


JES2 z/OS 1.2 Migration

SHARE 99, Session 2656

Wednesday, August 21, 2002

Chip Wood
JES2 Design/Development/Service
Poughkeepsie, NY

chipwood@us.ibm.com

JES2 z/OS 1.2 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** or **R5**
  - ➡ Note that **R5** and earlier releases are not supported on **z/OS 1.2** BCP (*enforced!!!*)
  - **$ACTIVATE** required to avoid **COLD** start
  - No MAS coexistence (all-member-warm start)
- MAS coexistence from **OS/390 R7-R10**
  - APAR **OW47328** needed on downlevel member
  - **$ACTIVATE** required on **R7-R8**

z/OS 1.2 Migration

- Most changes required only if $EXITs used
- Types of $EXIT changes required:
  1. Must change to migrate to z/OS 1.2 JES2
     - Renamed fields
     - New format JQA
  2. Must change to $ACTIVATE at z2 level
     - Checkpoint restructure
  3. Must change to use >64K jobs
     - JOBID format (affects more than exits)
- Services exist to make many of these changes transparent
  - Only helps if already used
  - Some are new in z/OS 1.2

>64K jobs challenges

- Increase maximum allowed job number
  - Halfword fields break at **65,536**
  - Jobid format **JOB***nnnnn* breaks at **100,000**
  - Job Index Table (JIX)
- Increase limits for JQEs, JOEs, TGs, BERTs
  - Chaining by 3-byte offset
    - JOEs break at **161,315** (current max + 1!)
    - JQEs break at **167,773** (or less)
  - Checkpoint "track 1" data
    - Max checkpoint CTLBs don't fit on track 1
- Migration via single-member warm start!

>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
  - Decrease fails if control block at or above new limit is in use
  - Code in compatibility APAR (**OW47328**) allows decrease to be recognized on down-level member

$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)

$D ACTIVATE

- **$D ACTIVATE**
  - Displays current checkpoint level

```
          $d activate
$HASP895 JES2 CHECKPOINT LEVEL IS NOW z/OS 1.2
```

  - Displays whether $ACTIVATE,LEVEL=Z2 will work or not

```
          $d activate
$HASP895 JES2 CHECKPOINT LEVEL IS NOW OS/390 RELEASE 4
$HASP895 A TOTAL OF 151 4K RECORDS ARE REQUIRED FOR $ACTIVATE.
$HASP895 ALL INUSE=YES DATA SETS ARE AVAILABLE AND LARGE ENOUGH.
$HASP895 $ACTIVATE WILL SUCCEED IF ISSUED FROM THIS MEMBER
```

$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.)

$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

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)

JOE to JQE

- To find the JQE associated with a particular JOE:

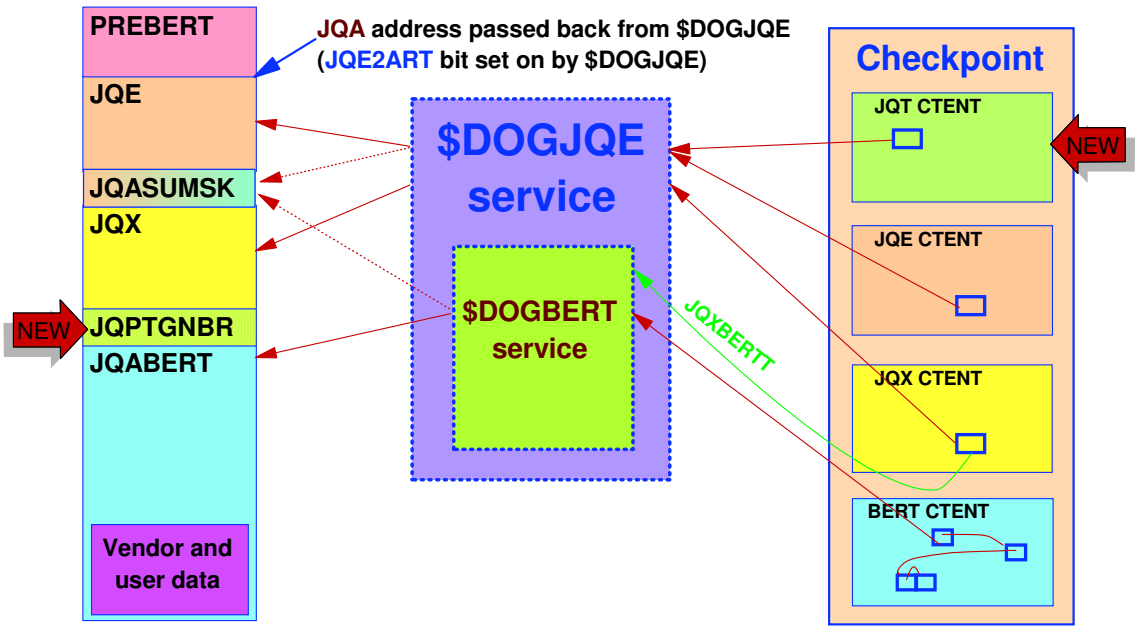
```
          ICM   R5,B'0111',JOEJQEI  Get JQE index/offset
          N     R5,$ZEROFFF         Clear hi-order byte
         $QINO  R=R5                Index to offset
+         CLI   $MSTRVER,$MSTRV12   Br if already
+         JL    INO2830               have an offset
+         MH    R5,$JQELEN  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

Job number fields

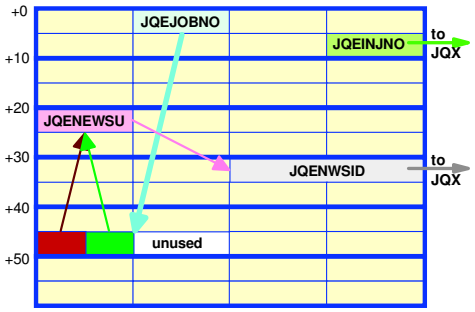
- SPOOL control blocks
  - 2 reserved bytes before ***xxx*JOBNO** reclaimed
  - New 4-byte field renamed to ***xxx*JBNUM**
- In-storage control blocks
  - Most are expanded and converted to 4-byte fields
- JQE
  - No room for 4-byte job number in R4 JQE
  - No room for 4-byte initial job number in R4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

JQA built by $DOGJQE

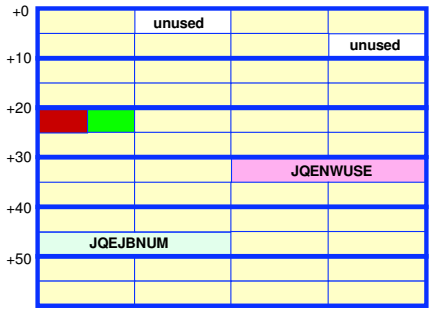


JQE field remapping

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 know 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

Counting track groups

- Code that relies on the count of the number of track groups must deal with both R4 and z2 mode
  - New field in JQA: **JQPTGNBR**
    - Composite of JQXTGWRP and value from JQT/JQE/master record
    - Always filled in, regardless of mode
  - **JQE6TGAE** (overflow bit) is set when
    - Job has allocated >**8,388,607** track groups
    - Job has allocated >**32767** track groups prior to $ACTIVATE,LEVEL=Z2
    - Job has allocated >**32767** track groups prior to $ACTIVATE,LEVEL=R4

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

Migration summary

- Lots of field names have changed
  - Field size or usage has also changed
    - 4-byte job numbers
    - JQE remapping
    - Offsets changed to indices
  - Just changing to use new names is not sufficient
  - **JQAs ALWAYS use new z2 mapping**
  - Services are provided (**$QJQE**, **$#JOE**, **$JQEJNUM**, etc.) to make most tasks easier
- Job id format changed from **JOB***nnnnn* to **J***nnnnnnn*
  - Check automation and SMF records
  - Both flavors of job id used in transition periods

Migration summary *(continued)*

- **$ACTIVATE,LEVEL=R4** makes return to R4 mode easier if there is a problem
  - Can fail if new limits are in use
  - JQE, JOE, BERT limits can be decreased via $T
  - **JOBDEF RASSIGN=NO** can be used to force job numbers within specific limits (1-64K or 1-99999, for example)
- IBM recommends running in z2 mode for a while before implementing new limits
- ***Read the migration manual carefully!!!***

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