JES2 What's New in z/OS 1.13

SHARE Atlanta, 2012
Session 10639 – Tuesday, March 13

Tom Wasik
JES2 Design/Development/Service
Rochester, MN
wasik@us.ibm.com

This presentation includes technical details of the changes made in z/OS 1.13 JES2 including migration consideration. It is intended as a guide to installing the new release and exploiting the new functions.

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®

* Registered trademarks of IBM Corporation

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.

The required trademark page.....

Session Objectives

- **Batch Modernization**
  - Instream data in PROCs (cataloged and instream)
  - Controlling job return code
  - Spin and SPIN data set
  - Requeue job by command on a step boundary
- **SPOOL Enhancements**
  - Extend SPOOL data set
  - Greater flexibility on names and volumes
  - SPOOL Migration
- **Enhanced SSIs**
  - Completion of device SSI

z/OS 1.13 has made a number of updates to improve BATCH job processing. These items include support for instream data in PROCs and includes, a mechanism for controlling job return codes, the ability to spin any spin data set by time, size, or operator command, and a feature that allows jobs to be removed from execution and restarted on a step boundary.
A number of enhancements were made to SPOOL processing including the ability to extend an existing SPOOL data set into contiguous free space on the volume, greater flexibility on SPOOL data set names and volume serials, and support that allows a quicker way to move data off an existing SPOOL volume.
The device SSI that was introduced in z/OS 1.12 for printers now supports all JES managed devices. Also the node subfunction of the JES property SSI now supports getting information from other members of the MAS.

Batch Modernization

- **Instream data in PROCs and INCLUDEs**
  - Simplifies writing JCL PROCs
    - No need for separate control data set
  - Support DD * and DD DATA in full in PROCs and INCLUDEs
    - Works with instream PROCs
    - Does not automatically generate SYSIN DD * like JCL
  - Works for all users of PROC (batch and started tasks)
    - Job must run under JES2 (not MSTR subsystem)
  - Must convert on a z/OS 1.13 member
    - Can run on any level member

z/OS 1.13 added support for instream data sets in JCL PROCs and INCLUDEs. This allows JCL coders to combine the JCL and control data sets in one PROC member. The DD * and DD DATA JCL cards and all their operands can now be placed in a JCL PROC followed by the instream data. During conversion processing, this instream data will be stripped out and placed into a JES2 instream data set. When the job runs, it can then access this data set like it would an instream data set included in the JCL. The only difference between this support and standard instream support is JES2 would generate a //SYSIN DD * card whenever it encountered a non-JCL card in the job stream. With this support, if a non-JCL card is encountered in a PROC or INLCUDE outside a DD * or DD DATA, it will continue to be flagged as an error.

This support works for all users of PROCs started tasks and batch jobs. However it only works for jobs running under a JES2 subsystem (does not work if the job is run under the master subsystem). Once z/OS 1.13 is installed, all that is required is the job convert on a z/OS 1.13 member. It can later execute on any level member.

Batch Modernization

- **Instream data sets in PROCs…**
  - Are NOT included in SPOOL Data Set Browse of JCLIN
    - Were not part of original JCL submitted
  - Are NOT transmitted to other nodes or offloaded
    - Were not part of original JCL submitted
  - Are included in extended status DSLIST function
- **Works for batch jobs as well as started tasks**

This instream data sets are not part of the submitted JCL so they are not included when looking at the original JCL (eg via SDSF SJ command) and are not sent over NJE to other nodes. They do appear in the data set list (eg via SDSF ? – JDS command).

Instream data sets can be used with batch jobs and started tasks.

Batch Modernization

Instream data in PROC example

```
//HELLO   PROC
//STEP1   EXEC ASMHCLG
//C.SYSIN DD *
TEST     CSECT ,
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVAREA+4
         LA    13,SAVAREA
         SPACE 1
         WTO   'Hello world!'
         SPACE 1
         L     13,SAVAREA+4
         LM    14,12,12(13)
         SR    15,15
         BR    14
         SPACE 1
SAVAREA  DC    18F'0'
         END
//L.TEST  DD DUMMY
//L.SYSXX DD *
//        PEND
```

```
s hello
$HASP100 HELLO    ON STCINRDR
$HASP373 HELLO    STARTED
+Hello world!
$HASP395 HELLO    ENDED
```


Here is a simple example of a PROC that calls another nested PROC and has an instream input. This can be run in a batch job or by just doing a S HELLO.

```
s hello
$HASP100 HELLO    ON STCINRDR
$HASP373 HELLO    STARTED
+Hello world!
$HASP395 HELLO    ENDED
```

Batch Modernization

- **New job card operand to control job RC**
  - JOBRC= MAXRC | LASTRC | (STEP,*name.name*)
    - MAXRC is existing processing (default)
    - LASTRC is return code of last step
    - (STEP,*name.name*) is return code of identified step
      - If step not executed, defaults to MAXRC
- **Affects return code seen in**
  - Extended status (eg SDSF)
  - ENF 70
  - HASP165 message
  - $DJ,CC= command
- **JOBCLASS JOBRC= MAXRC|LASTRC to affect processing for all jobs in the job class**

The success or failure of a JCL stream is not always determined by the highest completion code of any step. Sometimes it is the completion code of the last step or sometimes it is the completion code of a specific step. The new JOBRC= operand on the JOB card now gives the JCL writer the ability to control what completion code will be presented for the job.

There are 3 values for JOBRC, MAXRC (current and default processing), LASTRC (use the return code of the last step) or (STEP,*name.name*) to specify a specific step. If a specific step is specified and that step does not run, then the processing goes back to the maximum return code.

The return code for the job is presented in extended status (SDSF), ENF 70, the $HASP165 message and the $DJ CC= command.

An installation can elect to change the default processing for JOBRC on a job class basis by using the new JOBRC= setting on the JOBCLASS JES2 parameter.

Batch Modernization

- **Updated HASP165 message text**
  - *Jobname* ENDED AT *node reason*
  - Examples of *reason*:
    - **MAXCC**=*code* - JOBRC was not specified
      - Code is now always 4 digits (MAXCC=0000)
    - JOBRC=*code* - JOBRC was specified and affected the return code
    - **MAXRC**=*code* - JOBRC was specified but MAXRC was returned
    - ABENDED S*xxx*,U*yyy*
    - ABENDED *abend_code*, JOBRC=code
      - JOBRC=(STEP,*stepname*), step executed, but later step ABENDed
- **Two additional error case return codes defined**
  - CONVERTER ERROR – Conversion processing ABENDed processing the job
  - SYSTEM FAILURE – System crashed while job was running and job could not be restarted.

The HASP165 message had to be updated to accommodate the new JOBRC processing. First an unrelated requirement was implemented that always displays the job completion code as a 4 digit value (with leading zeros). The second change was to indicate if the completion code displayed was in fact the maximum completion code for the job or was instead affected by JOBRC processing (last or specific step completion code). This helps identify if specific step was requested in JOBRC and that step did not run resulting in the maximum return code being returned.

Another case that can occur is JOBRC requests the return code from a specific step but the job also ABENDs. In this case, the ABEND code and the JOBRC are both presented in the HASP165. The completion code in the ENF and extended status only returns the completion code of the step specified on JOBRC.

Two new conditions for job failure were also added:

•CONVERTER ERROR – for when conversion processing for the job ABENDs

•SYSTEM FAILURE – for when the job was running when the system crashed and the job was not restartable.

Batch Modernization

- **Extended Status STTRMXRC updated with this support**
  - New bit indicates JOBRC affected completion code
  - Old maximum return code available in verbose area (STVBMXRC)
- **IAZSSS2 (SAPI) fields SSS2MXRC and SSS2LSAB are not affected**
  - Origin is JCTMAXRC and JCTLSTAB which are not changed
- **Can get new combinations of conditions:**
  - JOB ABENDed but has a condition code from
    - A step that was named in JOBRC=(STEP,xxxx) completed
    - A final step that specified COND= ONLY or EVEN and JOBRC=LASTRC
  - Job EOM similar to ABEND if JOBRC=(STEP,xxxx) and step completed
  - Ended by CC and completion code from JOBRC=(STEP,x)
    - Not CC from step that caused job to end
- **JES2 cancel, JCL error, System failure override any JOBRC specification**

Extended status max return code field was updated in this support to return the new JOBRC information. A new bit was added to define that the completion code is not a maximum completion code but was affected by the JOBRC specification. The old maximum completion code is available in the verbose job level area.

Processing for extended status is not affected by this support because it returns a different view of the job completion code. It did not indicate how the job ended, just the last ABEND code and a high return code.

One effect of the changes is that you can get new combinations of return codes that may not have been expected. This generally happens when the job ends abnormally but the step specified on JOBRC ran normally. The key to ensuring the correct data is reported is ensuring that the bits STTRXAB and STTRXCDE are used to interpret the information in the 3 byte code field STTRMXCC rather than using how the job completed to interpret the contents of STTRMXCC.

As always, a JES2 cancel command, a JCL error, or a system failure will override any JOBRC or maximum return code for a job.

Batch Modernization

- **Added function to spin any spin data set**
  - Similar to what was done for JESLOG
  - Applies to any data set allocated as SPIN
    - No application code/JCL change needed
  - Spin based on size, time, operator command
- **Update to SPIN= DD operand**
  - SPIN=(UNALLOC,*option*)
    - '*hh:mm*' - Spin at specific time
    - '+*hh:mm*' - Spin every hh:mm interval
    - *nnn*, *nnn*K, *nnn*M - Spin every nnn lines
    - NOCMND - Cannot be spun by command
    - CMNDONLY - Can be spun via operator command (default if no interval)
- **$TJ*n*,SPIN,DDNAME=*name* command added**

JESLOG SPIN processing was added in z/OS 1.2 to allow long running jobs to spin their job log and system messages data sets. However, may jobs have additional log data sets that could consume large amounts of SPOOL space. z/OS 1.13 extended the JESLOG support to any SPIN data set the job may allocate. The SPIN= operand on the DD statement was enhanced with an optional value indicating when a SPIN data set should be spun off. The spin option can be a time of day, and interval, or a size. In addition, the function can be disabled by specifying NOCMND. By default spin data sets can be spun by operator command. The DDNAME= operand was added to the $TJ.SPIN command to spin a specific data set.

Batch Modernization

- **Remove job on step boundary**
  - New STEP operand on $EJ command
    - Causes job to exit execution at end of current step
    - Optional HOLD operand makes job held
    - Job is requeued for execution
  - Job must be journaling (JOURNAL=YES on JOBCLASS)
  - Uses existing continue restart function of z/OS
    - Previously used to restart jobs after an IPL
  - Full syntax $EJxxx,STEP [,HOLD]
    - Full cross member support

Long running jobs can be an issue when an installation is trying to shutdown a system. If the job has multiple long running steps, this enhancement can help the situation. A new STEP operand was added to the $E J command (restart job). When specified, the job will be removed from execution at the next step boundary. The job must be restartable for the command to be accepted (must have a JES journal data set). This support utilizes the existing continue restart function on z/OS to restart the job from the next step.

Once removed from execution, the job is placed back in awaiting execution state and optionally held. If the job is not held, it is assumed that the appropriate classes or initiators were drained so the job will not resume execution on the member where it was executing.

Extend SPOOL

- **Command to extend SPOOL to adjacent free space**
  - $TSPOOL(*xxxxxx*),SPACE=
    - Syntax for SPACE= same as $S SPOOL
      - MAX, (TRK,xxxx), (CYL,xxxx)
  - SPOOL is still limited to a single data set extent on the volume
- **SPACE= is the NEW TOTAL size of the data set**
  - It is NOT the increment
- **Extend occurs without impacting running jobs**
  - New space is always formatted by JES2
- **New message $HASP740 indicates Extend is successful**
- **$DSPOOL displays the results of the extend**
  - $DSPOOL,TGNUM displays the number of track groups in the data set
  - $DSPL,UNITDATA displays the track range (TRKRANGE) of the data set

New in this release is the ability to extend (expand) a SPOOL data set to adjacent free space on the device after the current SPOOL data set. The space must be adjacent because JES2 only supports a single extent on a SPOOL volume. The EXPAND is by specifying the SPACE= parameter on the $TSPOOL command. The syntax of the SPACE parameter is same as the $S SPOOL command. SPACE= specifies the new TOTAL size of the SPOOL data set (not an increment). The extend can occur while the SPOOL volume is active and does not impact any running address spaces. JES2 will format the new space added to the volume.

The success of the extend is indicated by the $HASP740 message. Any errors are indicated by the $HASP741 message. To display the results of the extend, use the $DSPOOL command to display the TGNUM or UNITDATA for the SPOOL extent.

SPOOL DSNAME and VOLSER

- **Data set name for SPOOL can now be specified on $S SPOOL**
  - $SSPL(xxxxxx),DSN=SYS1.MYSPOOL
  - Data set names must conform to new data set name mask or match SPOOLDEF DSNAME= value
    - SPOOLDEF DSNMASK= can be $T'ed
    - Can have generics (eg DSNMASK=SYS1.*SPOOL)
    - Default is blank (only names matching SPOOLDEF DSNAME= OK)
- **Must be in z11 $ACTIVATE mode to specify non-default DSN**
  - Cannot go back to z2 mode if non-default DSN exists
- **Should not use until all members migrated to z/OS 1.13**

The SPOOL data set name and VOLSER specifications have been relaxed with this release. The data set name can now be specified on a $S SPOOL command. The name must match a pattern specified on the SPOOLDEF DSNMASK= statement or match the default data set name specified on SPOOLDEF DSNAME=. The DSNMASK can be set via a $TSPOOLDEF command (and can have the generic characters * and ?). The default for DSNMASK is null which implies the only valid data set name is the one specified on DSNAME.

You must be in z11 $ACTIVATE mode to specify an non-default data set name. Once you have specified a non-default data set name, you cannot $ACTIVATE to z2 mode. It is recommended that all members be migrated to z/OS 1.13 before using this support.

SPOOL DSNAME and VOLSER

- **Increased flexibility on SPOOL VOLSERs**
- **SPOOLDEF VOLUME= can now have generics**
  - Still limited to 5 characters
  - Can be set with a $T command
  - Only used when volume is started
- **If no generics, then prefix**
  - Acts like it does in prior releases
- **If generics in value, then starting SPOOL VOLSER must match pattern specified**
  - For example SPOOLDEF VOLUME=SPL*
- **Should not use until all members migrated to z/OS 1.13**
  - $T SPOOLDEF VOLUME= to non-generics to bring in down level member

Similar to the flexibility of the DSNAME, the VOLSER for a SPOOL volume can now be less restrictive. The SPOOLDEF VOLUME= keyword has been enhanced to support generics. Any SPOOL volume being started must either match the VOLUME= prefix (if there are no generics) or match the pattern specified on VOLUME=. The value for VOLUME= can be altered by a $TSPOOLDEF command. Though it is still limited to 5 characters you should be able to get the flexibility you need for your SPOOL VOLSERs.

If VOLUME= contains generics, you cannot start a pre z/OS 1.13 JES2 into the MAS. However, if you $T the value back to a prefix (without generics) down level members can again join the MAS. The existing SPOOL volumes will continue to be used (even though they do not match the prefix).

SPOOL DSNAME and VOLSER

- **New SPOOL initialization statement**
  - SPOOL(*volser*) {DSName=*name*} {,SYSAFF=(*list*)}
  - Must still match SPOOLDEF VOLUME= and DSNMASK= restrictions
- **Used to locate SPOOL volumes on a COLD start**
  - Always use volumes from last time JES2 was active on a warm start
- **Two schemes to locate SPOOL volumes on a COLD start**
- **Old scan UCB method**
  - Look for UCBs matching SPOOLDEF VOLUME= prefix
  - Try allocating DSNAME from SPOOLDEF on volume
    - If works it is a SPOOL volume
- **New SPOOL initialization statement**
  - Allocate SPOOLs using DSNAME with SPOOL volume specified
- **New method is used exclusively IF either**
  - Any SPOOL initialization statements found
  - If SPOOLDEF VOLUME= has generics

To define what SPOOL volumes are to be used on a COLD start (warm start uses the volumes that were active when JES2 came down), a new SPOOL initialization statement has been created. The SPOOL data set name and SYSAFF are optional parameters on the SPOOL statement. The VOLSER (subscript) must match the VOLUME= specification on SPOOLDEF. The DSNAME parameter must match the current DSNMASK= specification on SPOOLDEF.

When JES2 COLD starts, it used one of 2 methods to locate the SPOOL volumes to use. The traditional method scans the UCBs on the system looking for volumes that match the VOLUME= prefix. If it matches, then an attempt is made to allocate the data set specified on DSNAME=. If the allocate works, we use this as a SPOOL volume.

The new method only allocates the SPOOL volumes that have SPOOL initialization statements. This method is used if there are any SPOOL initialization statements or if the SPOOLDEF VOLUME= specification has any generics. This implies that if you have no SPOOL initialization statements and have generics in VOLUME=, then JES2 will start with no SPOOL volumes active.

SPOOL Migration

- **$M SPOOL command to move data off volume**
  - Faster than $P SPOOL (Minutes not days)
  - Function enabled with OA36158 (PTF UA64366)
- **Command works with active address spaces using volume**
  - Less activity is better/faster but no need to IPL to stop active jobs
- **Goal of SPOOL migration is to stop using SPOOL data set**
  - It is NOT to eliminate the internal representation of the volume
  - Old data set can be deleted and SPOOL volume taken offline
- **After a successful SPOOL migration**
  - $DSPOOL still shows old volume
  - $DJQ,SPOOL= still displays jobs that think they have space on the volume (data is actually on the target volume)
  - Status of the source volume is MAPPED

SPOOL migration allows an installation a way to quickly move data off of a SPOOL volume in a period of minutes instead of the days that a drain command would take. The processing can be done with active address spaces still accessing the volume. The goal of the command is to get the source data set moved to either a new volume or merged onto an existing SPOOL volume. The internal representation of the volume will remain after it is merged onto an existing volume and will persist until all jobs that were using the volume have been purged. This implies that the volume will still be displayed in $D SPOOL commands and in the volume list of a $DJQ,SPOOL command. The status of the “remnant” volume will be MAPPED.

SPOOL Migration

- **Two forms of SPOOL migration, MOVE and MERGE**
  - Move takes all data on an existing volume and moves it to a new one
    - Source must be INACTIVE ($Z SPOOL done)
      - No active jobs on the volume
    - Target cannot be currently an active SPOOL volume
    - Can specify space to use to create data set on target
    - At the end of move, old (source) volume does not exist
    - Target after a move is active
  - Merge takes all data on one volume and merges it onto free space on another volume
    - Most flexible migration option
    - Source can be in any state with active jobs/address spaces
      - Less activity is good
    - Results is a mapped volume that goes away when all jobs using it are deleted
      - Similar to $P SPOOL but device is no longer in use

There are 2 forms of SPOOL migration, MOVE and MERGE. In a move migration, you take one existing, INACTIVE SPOOL volume and move it to a new volume that is not currently part of the SPOOL configuration. If you have 3 SPOOL volumes before a move, then you will have 3 SPOOL volumes after the move. As stated, for a move, the source volume must be INACTIVE (HALTED). There are other restrictions listed later for a move,

A merge migration takes the data on an existing SPOOL volume (in any state) and merges in into contiguous space on a target volume. With a merge, if you start off with 3 volumes before the merge you end up with 2 volumes after the merge. The 3rd volume will display but it is not being used. It is considered mapped.

Merge is the least restrictive process. Any source volume can be merged to an appropriate target volume.

SPOOL Migration

- **Command syntaxes**
- **$M SPOOL command syntax (merge)**
  $M SPOOL(*volser*),TARGET=*target*
- **$M SPOOL command syntax (move)**
  $M SPOOL(*volser*),TARGET=*target*
  [,SPACE=(CYL|TRK|MAX,*size*)]
  [,DSNAME=*dsname*]
  [,RESERVED]
- **$M SPOOL cancel command**
  $M SPOOL(*volser*),CANCEL

- **Multi-source move command is also supported**
  $M SPOOL(*volsr1,volsr2,volsr3…*),TARGET=*volser*
- **1st volume can be a move or a merge, remainder are merges**
- **Migration happens 1 volume at a time (one per target)**

The command syntaxes are listed here. A merge supports a source and target volume. Move supports specifying the target size, data set name, and reserved status. The move can create the SPOOL data set on the target in the same way a $S SPOOL command can.

To cancel a SPOOL migration, specify the source volser whose migration you want to cancel

SPOOL Migration

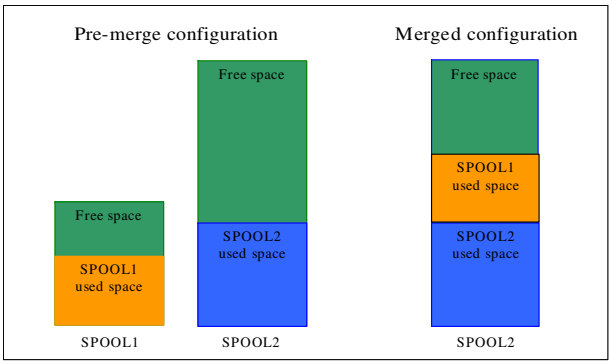
- **Merge migration moves a S*ource Volume* to an free space on an active *Target Volume***
- **Upon successful completion**
  - The *Source Volume* still exists but is STATUS=MAPPED
    - Still displays in $DSPOOL and in $DJQ,SPOOL lists
  - The *Target Volume* is a mapped on volume
- ***Source Volume* STATUS= values:**
  - INACTIVE ->MIGRATING ->MAPPED
- **Additional merge migration restrictions**
  - The *Target Volume* must be *Active* (can be *Reserved*).
  - The *Target Volume* cannot be stunted.
  - The *Target Volume* must use relative addressing.

A merge migration merges data from a source volume (in any state) to available free space on an existing target volume. At the end of the merge, the source is mapped and is selectable if the target is selectable. The source still appears in $D SPOOL and $DJQ,SPOOL command until all jobs that have space on the volume have purged. However, JES2 is no longer allocated to the source SPOOL data set and it can be deleted.

There are no additional restrictions on the source volume, but there are some restrictions on the target.

SPOOL Migration

- **MERGE Migration :**
  - Copies an existing *Source Volume* to free space on a *Target Volume* :



- **Upon completion, the *Source Volume* becomes a *Mapped Volume.***
  - Remains *MAPPED* until all jobs and SYSOUT that have space on the *Source Volume* are purged. It then goes away (no longer exists).

This is a pictorial view of a merge migration.

SPOOL Migration

- **$D SPOOL..,MIGDATA helps determine migration requirements**
  - SPACE_USED is high water mark of used space on volume
  - LARGEST_FREE is largest contiguous free space on the volume

```
$D SPOOL,MIGDATA
$HASP893 VOLUME(SPOL1X)  MIGDATA=(SPACE_USED=433410,
$HASP893                 LARGEST_FREE=16590)
$HASP893 VOLUME(SPOL1Y)  MIGDATA=(SPACE_USED=418215,
$HASP893                 LARGEST_FREE=31785)
```

  - Display all volumes having contiguous free space greater than 17000 tracks**:**

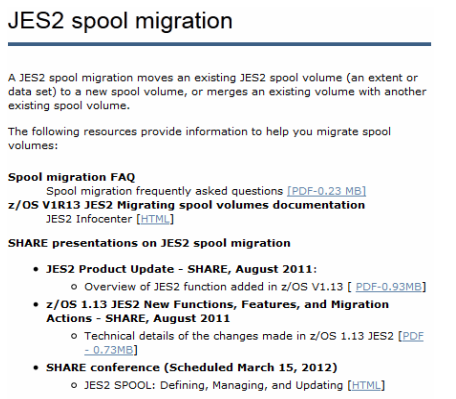
```
$D SPOOL,MIGDATA=LARGEST_FREE>17000,MIGDATA
$HASP893 VOLUME(SPOL1Y)  MIGDATA=(SPACE_USED=418215,
$HASP893                 LARGEST_FREE=31785)
```

- **Note:** Track groups in the BLOB are considered to be used (not free)

The $D SPOOL,MIGDATA command will provide information you need to determine what volumes can be merged or moved where. It list the current high water mark on each volume and the largest contiguous free space. This can be used to determine where to move a SPOOL volume to. One note, SPOOL space in the BLOB is considered used when listing used and available space. To get space out of the BLOB, set the volume to reserved (but do not reserve all volumes because then SPOOL will be considered full).

SPOOL Migration Enabled

- **JES2 SPOOL migration function has been enabled**
  - APAR OA36158 (PTF UA64366) closed February 24, 2012
- **New SPOOL migration page on the web**
  - http://www-03.ibm.com/systems/z/os/zos/jes2_spoolmigration.html
- **SHARE session**
  - **10844: JES2 SPOOL: Defining, Managing, and Updating**
    - Thursday 8:00AM



JES2 spool migration

A JES2 spool migration moves an existing JES2 spool volume (an extent or data set) to a new spool volume, or merges an existing volume with another existing spool volume.

The following resources provide information to help you migrate spool volumes:

Spool migration FAQ
Spool migration frequently asked questions [PDF-0.23 MB]
z/OS V1R13 JES2 Migrating spool volumes documentation
JES2 Infocenter [HTML]

SHARE presentations on JES2 spool migration

- **JES2 Product Update - SHARE, August 2011:**
  - Overview of JES2 function added in z/OS V1.13 [PDF-0.93MB]
- **z/OS 1.13 JES2 New Functions, Features, and Migration Actions - SHARE, August 2011**
  - Technical details of the changes made in z/OS 1.13 JES2 [PDF - 0.73MB]
- **SHARE conference (Scheduled March 15, 2012)**
  - JES2 SPOOL: Defining, Managing, and Updating [HTML]

The SPOOL migration function has been enabled with APAR OA36158. Due to the number and repetition of questions from various early support customers, a web page was created to try to help explain the function and provide a quick mechanism to get information to customers. The FAQs will be updated as we receive feedback from customers. This should be used to supplement the JES2 manuals to get a good understanding of what the SPOOL migration function does and how to use it.

SSI Enhancements

- **SSI 80 – Extended status**
  - Enhanced to support a jobid list as input
    - Only JES2 job ids (not transaction job ids)
    - Mutually exclusive with job name list and various other JOBID options
    - Can use live version for data if only job level information requested
  - New indicator that CKPT version vs live version used for request
  - New capability to limit the number of jobs (STATJQs) returned
    - SSOBRETN=0 with STATREAS=4
- **For more information, see publication MVS Using the Subsystem Interface**

A new JOBID list function was added to SSI 80 to get information for a list of JES JOBIDs. These requests will attempt to use a live version to get the most current data if only job information is requested (unless other input requires a CKPT version be used).

To help understand if your requests used a live version, a bit was added that indicates a copy of the checkpoint was used to obtain the data returned.

Also, the capability to limit the output of a single request was added. This limits the number of jobs (STATJQs) that are returned.

SSI Enhancements

- **SSI 82 – JES Property SSI**
- **Node information SSI – sub-function of JES properties SSI (SSI 82)**
  - Enhanced to provide information from all active members of JES2 MAS
  - Available from MAS members starting from z/OS 1.11
    - Requires APAR OA35942 (760 – UA90569, 770 – UA90570)
- **New function is exploited by SDSF**
- **For more information, see publication MVS Using the Subsystem Interface**

The node SSI was enhanced to provide the local view of the node (connection, tracing, etc) from other members of the MAS. This requires OA35942 on z/OS 1.11 and 1.12 members.

SSI Enhancements

- **SSI 83 – Device information SSI**
  - Enhanced to support all types of devices managed by JES2 (readers, punches, transmitters, receivers, lines, offload etc.)
  - Provides extensive filtering capabilities – e.g. by device state, device name, a variety of device attributes
  - Provides information about all devices managed by all active members of JES2 MAS
  - Device information is available from JES2 MAS members starting from z/OS 1.11 (requires coexistence APAR on z/OS 1.11 and 1.12)
- **New function is exploited by SDSF**
- **For more information, see publication MVS Using the Subsystem Interface**

The JES device SSI was completed in this release to support providing information for all JES managed “devices” over the SSI.

Other Enhancements

- **JES subsystem data set allocation support for XTIOT**
  - Option on DYNALLOC request
    - S99TIOEX bit for authorized callers
    - S99DXACU bit supports all callers (unauthorized)
  - Moves allocations control blocks from 24 to 31 bit storage
- **Relieves pressure on 24 bit storage**
- **Increases number of concurrent allocations**
  - Reduces pressure on 24 bit TIOT
- **Implications include not being able to find DD in TIOT**
  - Could break applications looking into TIOT
  - Very unlikely
- **Controlled by parmlib option**
  - NON_VSAM_XTIOT=NO|YES in DEVSUP*xx*
- **Good ideas for use include**
  - SPOOL data set browse, SPIN data set allocation

JES, allocation, and DFSMS have all added support to allow the use of the XTIOT option for subsystem data sets. This allows a number of control blocks used by allocation and DFSMS to move from 24 to 31 bit storage. It also does not place the subsystem allocations in the 24 bit TIOT of current allocations. This relieves pressure on the number of data sets a step can be allocated to.

To use this new feature, you must set a bit in the dynamic allocation RB structure. There are 2 bits for this, the traditional bit S99TIOEX requires the caller to be authorized. The new bit S99DXACU supports all callers.

This new function would be a good candidate for any JES subsystem allocation but especially for address spaces that allocate a large number of data sets or are 24 bit storage constrained. Applications that use SPOOL browse or a create large number of SPIN data sets would also be good candidates.

There may be some applications that look at the TIOT and expect to find all allocations. These will not find the XTIOT allocations. However, this is normally something done by the program that allocated the data set in the first place so it is unlikely that use of this function would cause any difficulties.

There is a PARMLIB option to activate this support. By default it is NOT active. To activate this support code NON_VSAM_XTIOT=YES in your DEVSUP*xx* member.

Migration/Coexistence

- **From JES2 z/OS 1.9 or 1.10**
  - Can all member warm to z/OS 1.13
  - No coexistence support
  - Fall back implications
    - Some new data structures created by z/OS 1.13 JES2 may result in problems in z/OS 1.10 and prior
    - Prior to z/OS 1.10 may not be able to use SPOOL volumes with non-standard data set names
- **From JES2 z/OS 1.11 or z/OS 1.12**
  - COMPAT APAR OA31806 is needed on a z/OS 1.11, or z/OS 1.12 member to coexist in a MAS with z/OS 1.13
    - HJE7760 UA59434
    - HJE7770 UA59435
  - APAR also highly recommended for fall back as well
    - Some new data structures created by z/OS 1.13 JES2 may result in problems if OA31806 is not installed.

The migration/coexistence PTFs are listed here. These should also be applied for fallback support.

Any questions???