# What's New in z/OS 2.2 JES2: Other Changes

SHARE Orlando, Session 17827

Tuesday, August 11, 2015

Tom Wasik

JES2 Development

Rochester, MN

wasik@us.ibm.com



#SHAREorg

SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

# 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.

# Agenda

- Growth line items
  - New $ACTIVATE level (z22)
  - Increase limits for Jobs and JOEs
  - Checkpoint processing enhancements
  - Automatically set HOLD and DORMANCY
  - Additional SYSOUT work selection performance enhancements
- Step completion code data set
  - Information on each step execution
  - Select SMF records for the job
- Job scheduling controls
  - Hold until
  - Start by
  - Run with
- JES3 JECL toleration
- PROCLIB SSI

# Growth Line Items Overview

- Problem Statement / Need Addressed
  - Ability to support larger numbers of jobs resident on spool at one time
  - Deal with performance problems related to larger number of jobs
- Solution
  - Update limits on job, JOE, and BERT numbers
  - SYSOUT work selection improvements (beyond what 2.1 did)
  - Better management of duplicate jobs
- Benefit / Value
  - Support larger workloads with improved performance

# Externals updated

- New $ACTIVATE level (z22) to support new functions
  - Level z2 is not supported in this release
- SYSOUT work selection performance extended to local devices
  - New option to activate support
- Updated limits for major checkpoint items
  - JOBNUM limit now 1,000,000
  - JOENUM limit now 2,500,000
  - BERTNUM limit now 2,500,000
- Increased BLOB size
  - Cache of available track groups increased from 256 to 1024
- New command to display duplicate job families
  - $D DUPJOB

# New $ACTIVATE level

- `$ACTIVATE,LEVEL=z22`
  - Requires SPOOLDEF CYL_MANAGED=ALLOWED be set
  - Enables a number of z22 function
    - Larger limits
    - Job execution zones (dependent job control)
    - Larger SPOOL BLOB size (256 updated to 1024)
    - Checkpoint extend/alter processing
- Same management options same as when z11 mode was introduced
  - $ACTIVATE supports `LEVEL=z11` or `LEVEL=z22`
  - `OPTSDEF COLD_START_MODE= Z11` or `Z22`
  - `S JES2,PARM=(UNACT)` warm starts in z11 mode
  - `$D ACTIVATE` displays current mode/what is needed for other mode
- Checkpoint size increases for BLOB size and new ZJC data area
  - About 6 tracks depending on settings

# Work Selection Performance

- JES2 SYSOUT work selection process for the SAPI applications extended to other devices (local printers and punches and offload)
  - SYSOUT work selection is the process of selecting SYSOUT output groups ($#GET)
- These enhancements address both:
  - active work selection – assigning work to a device (GET)
  - passive work selection – waking up the device when work with appropriate characteristics becomes available (POST)
- Active SYSOUT work selection uses JOE index added in V2R1
- In V2R2, improvements have been made to the JOE index management. In particular, JES2 will now periodically remove unused JOE index nodes.

# Work Selection Performance

- To enable active work selection optimization (GET):

```
$T OUTDEF,WS_OPT=YES
```

  - Activates both SAPI and local device support
  - This is the same option introduced in V2R1.
  - This type of optimization has MAS scope and persists until it is explicitly disabled.
- To enable passive work selection optimization (POST) for local devices:

```
$T OUTDEF,LDEV_OPT=YES
```

  - This type of optimization has a member scope and is only active until this member is restarted.
- All three optimization types are independent and can be enabled in any combination.
  - `LDEV_OPT=, SAPI_OPT=, WS_OPT=`

# Work Selection Performance - Cautionary statement

- The implemented algorithms are sensitive to the SYSOUT selection criteria configured for a device (keyword WS=), so “mileage can vary”
- If for some reason optimization causes undesirable results, it can be turned off at any time:

```
$T OUTDEF,LDEV_OPT=NO
```

  - to disable passive work selection optimization (POST)

```
$T OUTDEF,WS_OPT=NO
```

  - to disable active work selection optimization (GET)

# Work Selection Performance - Cautionary statement (cont.)

- Optimized path for active SYSOUT work selection (GET) uses index structure over JOEs
- This JOE index requires maintenance, which has its own CPU overhead
- It is unlikely but possible that in environment where majority of job SYSOUT is never processed (never printed or selected by SAPI applications), the overhead of index maintenance may not be offset by the performance improvements of work selection.
  In this case, using old path (WS_OPT=NO) may be preferable.

# Duplicate Job Processing Performance

- By default, only one job with a given name can be active at a time
  - DUPL_JOB= on JOBDEF and JOBCLASS can alter this
  - Does not apply to jobs in a job group
- Additional overhead is associated with tracking these jobs
  - Impacts statistics reported to WLM for how many jobs available to run
- Internal changes made to reduce overhead of processing
- New command added to display duplicate job information

  `$D DUPJOB(`*`jobname`*`)`

  - Output displays
    - Count of jobs with specified name
    - Indicator if job is active that will limit other jobs

# Duplicate Job Processing Performance

- Example command

```
$ddupjob
$HASP734 DUPJOB(IBMUSERG)   NUMBER=3,ACTIVE=YES
$HASP734 DUPJOB(IBMUSERQ)   NUMBER=2,ACTIVE=NO
```

- In this display
  - 3 jobs are in the execution phase named IBMUSERG
    - At least one is executing in a class with DUPL_JOB=DELAY
  - 2 jobs are in the execution phase names IBMUSERQ
    - Either no jobs are executing or
    - Jobs are executing in a class with DUPL_JOB=NODELAY
- If JOBDEF DUPL_JOB=NODELAY then you will never see ACTIVE=YES
  - The tables are still maintained and duplicates can be displayed
  - Little overhead in this case
- Can also be used to detect runaway program submitting multiple job with the same name

# 64 bit CKPT Processing

- To support increased limits, checkpoint processing re-worked
  - No impact to instorage control blocks (JQE, JOE, etc)
  - I/O area moved to 64 bit storage
  - Most DASD checkpoint I/O processing moved to new subtask
    - CF processing already in a subtask
  - Numerous performance and RAS enhancements
- DASD (as with existing CF) I/O sensitive to being “ripped out”
  - More so since multiple physical I/Os used to do one logical I/O
    - Takes advantage of multiple paths to CKPT DASD
  - If possible, at least ABEND JES2 before crashing a system
- Larger checkpoint size implies
  - Larger checkpoint versions
    - More data space than previous releases
  - More virtual storage usage
    - Even with I/O area and other CBs moved to 64 bit

# CKPT Reconfiguration Changes

- Problem Statement / Need Addressed
  - Reconfiguration dialog overly complex for simple changes
  - Cannot alter the size of a checkpoint data set without deleting it
- Solution
  - New fast path checkpoint reconfiguration process
  - Support dynamic changing of checkpoint size
    - Support ALTER processing on CF
    - Support extending a DASD data set into adjacent free space
- Benefit / Value
  - Simplified operations
  - Continuous operations

# CKPT Reconfiguration Commands

- New options on the CKPTDEF command

  `$T CKPTDEF,CKPTn=(DSNAME|VOLSER|STRNAME,INUSE)`

  - Runs fast path reconfiguration if needed
    - Not needed for DSNAME/VOLSER/STRNAME if INUSE=NO
  - Only one change at a time allowed
    - CKPT1 or CKPT2
    - DSN/VOL or STRNAME or INUSE
- New information on $D CKPTSPACE command
  - Number of tracks in a DASD data set (TRACKS=*xxxx*)
  - Current and maximum size of a CF structure (SIZE=(*cur*,*max*))
    - Size is in 1K blocks
- New options to increase the size of the checkpoint
  - SPACE=(TRK|CYL|MAX,*nnnn*) for CKPT on DASD
  - SIZE=*nnnn* for CKPT on CF
- CF structure deleted when changing INUSE to NO

# Reconfiguring the Checkpoint Data Sets

- Use $T CKPTDEF command to update CKPT specification
  - Uses the standard reconfiguration process to make changes
    - Works with down level members in the MAS
  - Will make the changes requested with no WTORs
    - Creates data set if it does not exist
  - Note, change is NOT reflected in the command response
    - Processing occurs asynchronously
- Sample command:

```
$T CKPTDEF,CKPT1=(VOL=CKPTPK)
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(DSNAME=SYS1.JESCKPT1,VOLSER=SPOOL1,
$HASP829          INUSE=YES,VOLATILE=NO),
   :
$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
$HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS OPERATOR
         REQUESTED SET COMMAND
$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY
         MEMBER IBM1
$HASP280 JES2 CKPT1 DATA SET (SYS1.JESCKPT1 ON CKPTPK) IS NOW IN USE
$HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE
```

# Reconfiguring the Checkpoint Data Sets

- If there is a problem, the reconfiguration fails
  - No WTORs asking for alternate actions
- Sample command:

```
$TCKPTDEF,CKPT1=VOL=UNKNOW
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(DSNAME=SYS1.JESCKPT1,VOLSER=CKPTPK,
$HASP829          INUSE=YES,VOLATILE=NO),
   :
$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
$HASP233 REASON FOR JES2 CHECKPOINT RECONFIGURATION IS OPERATOR
         REQUESTED SET COMMAND
$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTED - DRIVEN BY
         MEMBER IBM1
$HASP424 MEMBER IBM1 -- UNKNOW IS NOT MOUNTED
$HASP255 JES2 CHECKPOINT RECONFIGURATION FAILED
```

# Reconfiguring the Checkpoint Data Sets

- If CKPTn is INUSE=NO, then change can occur without reconfiguration
  - Changes appear in the command output
  - No actual allocation occurs, the checkpoint is not in use
- Sample command:

```
$DCKPTDEF
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(DSNAME=SYS1.JESCKPT1,VOLSER=CKPTPK,
$HASP829          INUSE=NO),CKPT2=(DSNAME=SYS1.JESCKPT2,
   :
$TCKPTDEF,CKPT1=VOL=SPOOL1
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(DSNAME=SYS1.JESCKPT1,VOLSER=SPOOL1,
$HASP829          INUSE=NO),CKPT2=(DSNAME=SYS1.JESCKPT2,
   :
```

# Reconfiguring the Checkpoint Data Sets

- This is NOT a replacement for the reconfiguration dialog
  - I/O errors still use the traditional format
    - Still need to specify NEWCKPTn and OPVERIFY=NO
  - Does not interrupt hung I/O processing
    - CKPT PCE must be running to notice a change is needed
  - Presumes you know what you are doing
    - A typo in the data set name or volume will create the data set
- Traditional dialog still exists for all the reasons described above
- Examples used DASD data sets but also works for CF structures

# Altering the Size of a CKPT

- New operand on CKPTn for DASD data set
  - SPACE=(TRK|CYL|MAX,*nnnn*)
  - Similar in function to SPACE= on SPOOL(*vvvvvv*)
  - MAX is the largest CKPT or all the free space, whichever is smaller
  - Data set must be INUSE=YES and allocated
    - Not CKPT2 in MODE=DUPLEX, DUPLEX=NO
  - Must be adjacent space AFTER the current data set
  - Must be $ACTIVATEd at z22 level
- Sample commands:

```
$DCKPTSPACE,CKPT1
$HASP852 CKPTSPACE  CKPT1=(CAPACITY=1788,UNUSED=1566,TRACKS=150)
$TCKPTDEF,CKPT1=SPACE=(TRK,160)
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(DSNAME=SYS1.JESCKPT1,VOLSER=CKPTPK,
$HASP829          INUSE=YES,VOLATILE=NO),
   :
$HASP740 Volume CKPTPK Data set SYS1.JESCKPT1 Extend successful.
$DCKPTSPACE,CKPT1
$HASP852 CKPTSPACE  CKPT1=(CAPACITY=1908,UNUSED=1686,TRACKS=160)
```

# Altering the Size of a CKPT

- New operand on CKPTn for CF structure
  - SIZE=*nnnn*|MAX
  - Units is 1K blocks (same a specification in policy)
  - MAX is the largest CKPT or all the policy limit, whichever is smaller
  - Only valid if current size is less than maximum size
    - Specify both INITSIZE and SIZE when defining structure
  - ALTER processing does round up the size specified
  - Must be $ACTIVATEd at z22 level
- Sample commands:

```
$DCKPTSPACE,CKPT1
$HASP852 CKPTSPACE  CKPT1=(CAPACITY=617,UNUSED=391,SIZE=(4096,10240))
$TCKPTDEF,CKPT1=SIZE=5000
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(STRNAME=SPOOLCKPT1,INUSE=YES,
$HASP829          VOLATILE=YES),CKPT2=(DSNAME=SYS1.JESCKPT2,                :
$HASP739 CKPT1 structure SPOOLCKPT1 size altered to 5120
$DCKPTSPACE,CKPT1
$HASP852 CKPTSPACE  CKPT1=(CAPACITY=857,UNUSED=631,SIZE=(5120,10240))
```

# Altering the Size of a CKPT

- JES2 CKPT structures are considered persistent
  - They are not deleted when JES2 goes down
  - Only way to delete them is to “force” them
- A new policy with a new size cannot fully active when structure exists
  - Policy change is “pending”
- CKPT reconfiguration altered to force structure when setting INUSE=NO
  - Both new fast path and traditional processing
  - Allows simple way to get a new structure with desired size
  - Does NOT work when forwarding the checkpoint
- Alternative is to do a system managed rebuild of structure

```
SETXCF START,REBUILD,STRNAME=ckptstr
```

# Altering the Size of a CKPT

- Sample commands:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=CTTEST2
IXC511I START ADMINISTRATIVE POLICY CTTEST2 FOR CFRM ACCEPTED
IXC512I POLICY CHANGE IN PROGRESS FOR CFRM
TO MAKE CTTEST2 POLICY ACTIVE.
1 POLICY CHANGE(S) PENDING.
$DCKPTSPACE,CKPT1
$HASP852 CKPTSPACE  CKPT1=(CAPACITY=857,UNUSED=631,SIZE=(5120,10240))
$TCKPTDEF,CKPT1=INUSE=NO
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(STRNAME=SPOOLCKPT1,INUSE=YES,
$HASP829          VOLATILE=YES),CKPT2=(DSNAME=SYS1.JESCKPT2,
   :
$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
   :
$HASP280 JES2 CKPT1 DATA SET (STRNAME SPOOLCKPT1) IS NO LONGER IN USE
IXC513I COMPLETED POLICY CHANGE FOR CFRM.
CTTEST2 POLICY IS ACTIVE.
$HASP255 JES2 CHECKPOINT RECONFIGURATION COMPLETE
$TCKPTDEF,CKPT1=INUSE=YES
$HASP829 CKPTDEF
$HASP829 CKPTDEF  CKPT1=(STRNAME=SPOOLCKPT1,INUSE=NO),
   :
$HASP285 JES2 CHECKPOINT RECONFIGURATION STARTING
   :
$HASP280 JES2 CKPT1 DATA SET (STRNAME SPOOLCKPT1) IS NOW IN USE
$DCKPTSPACE,CKPT1
$HASP852 CKPTSPACE  CKPT1=(CAPACITY=617,UNUSED=391,SIZE=(4096,12288))
```

# Self Tuning CKPT Overview

- Problem Statement / Need Addressed
  - JES2 checkpoint tuning key to good JES2 throughput
    - MASDEF HOLD= and DORMANCY= main parameters
  - Properly managing these settings depends on many factors
    - Workload, overhead, size of MAS, etc
  - Often not updated until performance is actually a problem
- Solution
  - Let JES2 manage the setting of the values
  - Adjusted based on workload, MAS size, etc
  - Set and forget
- Benefit / Value
  - Simplified operations
  - Better overall performance

# Self Tuning CKPT Commands

- New function – automatic checkpoint cycle management
  - Adjust HOLD= and DORMANCY= based on workload
  - Can display, but not set, current values
    - SDSF MAS display shows what members are using
  - Adjusts as member join and leaves MAS
- To enable automatic checkpoint cycle management:

```
$T MASDEF,CYCLEMGT=AUTO
```

  - This setting has MAS scope and persists until it is explicitly disabled.
- The function can also be enabled via MASDEF initialization statement
- The function can be turned off at any time via:

```
$TMASDEF,CYCLEMGT=MANUAL
```

  - All members will revert to HOLD and DORMANCY settings they had before automatic management had been turned on.

# Step Completion Code Overview

- Problem Statement / Need Addressed
  - A single completion code is provided for each job (influenced by JOBRC). Customers require more details concerning the execution of a job. Specifically, they would like information on the results of the execution of each step of a job.
  - Information should be in control block format vs human readable.
- Solution
  - JES2 was updated to create a new spool data set, named EVENTLOG, that will contain information about each step in a job and the completion code for that step.
- Benefit / Value
  - Provides customers with the granular step information they need to fully evaluate the execution of their jobs.

# Step Completion Code Changes

- The new JES2 spool data set EVENTLOG is allocated automatically for batch jobs, started tasks and TSO users

- EVENTLOG contains machine readable records
  - Not intended to be human readable
  - EVENTLOG is non-printable, non-spinnable
  - Not SYSOUT, cannot be accessed by SAPI/PSO/FSS
  - Viewed via spool data set browse

# Step Completion Code Details

- EVENTLOG will contain key SMF Type 30 subtype records
  - Subtype 1. Job start or start of other work unit.
  - Subtype 4. Step Total.
  - Subtype 5. Job termination or termination of other work unit.

- The user controls writing SMF records to EVENTLOG by using the SUP_EVENTLOG_SMF keyword on the JOBDEF command
  - Default is SUP_EVENTLOG_SMF=NO, indicating SMF records are not suppressed.
  - $T JOBDEF,SUP_EVENTLOG_SMF=YES will keep SMF records from being written to EVENTLOG
  - The value of SUP_EVENTLOG_SMF is captured at the time the job enters the system and effects the job's EVENTLOG for the life of the job

# Step Completion Code Details

- SMF Type 30 subtype 4 records will cause a STEPDATA record to be written to EVENTLOG
  - Contains key data values from the SMF Type 30 subtype 4 record
  - Contains the eyecatcher “STEPDATA”
  - STEPDATA records are ALWAYS written
    - Even if SUP_EVENTLOG_SMF=YES
    - Even if SMF is not active or SMF Type 30 records are suppressed

- STEPDATA records are mapped by the IAZLGSTP macro
  - Similar to data presented to IEFACTRT and sample exit IEEACTRT
- STEPDATA records are used by SDSF to implement the new JS panel

# Step Completion Code Details

- Records are written to denote the job has restarted
  - Contains the eyecatcher “RESTART”
  - RESTART records are ALWAYS written
- RESTART records indicate when a job ends and is requeued for execution with an additional eyecatcher
  - “JOB TERMINATED/RE-ENQUEUED”
- RESTART records indicate when a job restarts execution with a different additional eyecatcher
  - “JOB RESTARTED”
- RESTART records are mapped by the IAZLGSRST macro

# Step Completion Code Browsing Data Sets

- EVENTLOG records can be read using SPOOL Data Set Browse
- Read all EVENTLOG records using the fully qualified data set name
  - *userid.jobname.jobID*.D0000008.EVENTLOG
- Or, use the logical data set name
  - *userid.jobname.jobID*.EVENTLOG
- Use other logical data set names to see specific record types
  - *userid.jobname.jobID*.EVENTLOG.STEPDATA reads only STEPDATA records
  - *userid.jobname.jobID*.EVENTLOG.SMF reads only SMF records
  - *userid.jobname.jobID*.EVENTLOG.SMFSTEP reads only SMF Type 30 subtype 4 records
  - *userid.jobname.jobID*.EVENTLOG.RESTART reads only RESTART records

# Step Completion Code and NJE

- EVENTLOG data set is eligible for transmitting/receiving via NJE
- Sender and receiver must support non-print SYSOUT
  - New feature bit NCCINOS in the NCCIFEAT bytes exchanged in the initial signon record
    - NCCINOS indicates this NJE node supports transmitting and receiving non-printable SYSOUT data sets ... EVENTLOG is one of these
    - If both ends of the NJE connection support the NCCINOS feature, then EVENTLOG data sets can be transmitted/received on the connection
    - If NCCINOS is not a feature of the connection and the job is sent/received on that connection the EVENTLOG data set will be lost
- EVENTLOG data set sent with JESMSGLG data set

# Job Scheduling Enhancements

- Problem Statement / Need Addressed
  - Missing basic job scheduling controls in JES2
  - Desire to keep JESes in synch with basic batch functions
- Solution
  - New JCL keywords on SCHEDULE JCL statement:
    - keep a job in a held state until a specified time (HOLDUNTL=)
    - specify a desired time for a job to start (STARTBY=)
    - specify that a job should run on the same system where a reference job is currently executing (WITH=)
  - HOLDUNTL and STARTBY are sometimes collectively referred to as “deadline scheduling”
- Benefit / Value
  - JCL external to help schedule job execution

# Job scheduling enhancements – HOLDUNTL

- To hold a job until a specified time:
  `// SCHEDULE HOLDUNTL=<time>`
- <time> can be specified in several ways:
  - `HOLDUNTL='+hh:mm'`
    - A delta time from when the job entered the system.
    - This time is not subject to time offset changes.
  - `HOLDUNTL=('hh:mm',mm/dd/yyyy)`
    or
    `HOLDUNTL=('hh:mm',yyyy/ddd)`
    - A specific time in a future, when job should be released.
    - Date is optional.
    - This is local system time and is subject to time offset changes.

# Job Scheduling Enhancements – HOLDUNTL (cont.)

- When optional date part of a specific time is omitted, then if the target time has passed on the current day, the time is considered to refer to the next day
- If target time is in the past, job is not held
- The target time for a job can be displayed via JES2 command or retrieved via Extended Status SSI, e.g.:

```
$dj19,holduntl
$HASP890 JOB(SCHTEST)HOLDUNTL=(2015.029,13:55:00)
```

- Of course, the job can be manually released at any time

# Job scheduling enhancements – STARTBY

- To specify a target time for job to start:
  `// SCHEDULE STARTBY=<time>`
- <time> can be specified in several ways:
  - `STARTBY='+hh:mm'`
    - A delta time from when the job entered the system.
    - This time is not subject to time offset changes.
  - `STARTBY=('hh:mm',mm/dd/yyyy)`
    or
    `STARTBY=('hh:mm',yyyy/ddd)`
    - A specific time in a future for a job to start.
    - Date is optional.
    - This is local system time and is subject to time offset changes.

# Job Scheduling Enhancements – STARTBY (cont.)

- When optional date part of a specific time is omitted, then if the target time has passed on the current day, the time is considered to refer to the next day
- The target time for a job can be displayed via JES2 command or retrieved via Extended Status SSI, e.g.:

```
$dj19,startby
$HASP890 JOB(SCHTEST) STARTBY=(2015.029,13:55:00)
```

# Job Scheduling Enhancements – STARTBY (cont.)

- STARTBY specification does not mean that JES2 must start the job by the STARTBY time.
  - JES2 will do its best effort to gradually move a job to the top of the execution queue to give the job a better chance to be selected for the execution.
  - Actual selection for execution is still controlled by all the usual considerations – system affinity, availability of initiators etc.
- STARTBY function can be viewed as a more intelligent flavor of priority aging

# Job Scheduling Enhancements – HOLDUNTL vs STARTBY

- Both HOLDUNTL and STARTBY can be set for the same job to indicate the job execution window. e.g.

```
// SCHEDULE HOLDUNTL='+01:00',STARTBY='+02:00'
```

  indicates that the job will be executed between one and two hours from the job submission time (if resources are available during that time to run the job)
- If both HOLDUNTL and STARTBY are specified, they must use compatible time formats – either both use delta time specification or both use point in time specification

# Job Scheduling Enhancements – PROMO_RATE

- STARTBY function is controlled on a job class level by a new job class attribute PROMO_RATE (job promotion rate)
- PROMO_RATE controls how much a job can be moved up the execution queue in one STARTBY aging cycle (1 minute)
- Default value PROMO_RATE=0 means that STARTBY function is disabled for the job class
- PROMO_RATE can be changed at any time, e.g.:

```
$TJOBCLASS,PROMO_RATE=3
```

# Job Scheduling Enhancements – WITH

- To specify that job must be executed on the same system where another reference job is currently active:

```
//    SCHEDULE   WITH=jobname
```

- WITH specification is an additional limitation on where a job can run.
  - If WITH is specified, the job will not be eligible for execution until the reference job is active.
  - In addition, the job can only be executed on the same system where the reference job is active.
- Job having a WITH specification can be submitted before or after the reference job becomes active or even submitted.
  - It is recommended to submit a job after the reference job becomes active.
    - Additional processing if job with WITH submitted first.

# Job Scheduling Enhancements – Restrictions

- Do not create circular WITH relationships
  - Job A specifies WITH=B and Job B specifies WITH=A
  - The job on WITH must be running **before** specifying job
    - Attempt to detect after conversion and fail second job
- STARTBY specification is mutually exclusive with JOBGROUP keyword
  - Cannot combine STARTBY with dependent job control
  - WITH and HOLDUNTL can be combined with dependent jobs

# JES3 JECL Toleration Overview

- Problem Statement / Need Addressed
  - More customer are finding themselves running both JES2 and JES3
  - While migrating to a single JES may be beneficial, it may not be practical
  - Application (JCL) developers desire a consistent processing of JCL/JECL regardless of the JES being used
- Solution
  - Unify to the extent possible the JCL/JECL language to be independent of the JES
  - Provide the ability to turn off JECL statements to provide away to prevent JECL from creeping back in to an installation that has undergone a effort to remove JECL
- Benefit / Value
  - Simplified management of JCL/JECL
  - Support installations running/supporting both JES2 and JES3

# JES3 JECL Toleration – INPUTDEF

- JES3JECL on INPUTDEF controls how JES2 treats JES3 JECL

  `$T INPUTDEF,JES3JECL=PROCESS|IGNORE`

  - This setting has a member scope and is only active until this member is restarted.
  - JES3JECL=IGNORE ignores JES3 JECL statements
    - This is the default
  - JES3JECL=PROCESS enables the processing of JES3 JECL
    - Syntactically, JES3 JECL is not treated as a comment
    - Impacts what is passed to exits 4 and 54
- Note: Processing of individual JES3 JECL statements is controlled via JECLDEF (on the next chart)

# JES3 JECL Toleration – JECLDEF for JES3

- New parameters to control JES3 JECL processing

```
JECLDEF   JES3=(
    MAIN          =     PROCESS | IGNORE | WARN | FAIL
    FORMAT        =     IGNORE | WARN | FAIL
    ROUTE         =     IGNORE | WARN | FAIL
    OPERATOR      =     IGNORE | WARN | FAIL
    DATASET       =     IGNORE | WARN | FAIL
    ENDDATASET    =     IGNORE | WARN | FAIL
    PROCESS       =     IGNORE | WARN | FAIL
    ENDPROCESS    =     IGNORE | WARN | FAIL
    NET           =     IGNORE | WARN | FAIL
    NETACCT       =     IGNORE | WARN | FAIL
    PAUSE         =     IGNORE | WARN | FAIL
                  )
```

# JES3 JECL Toleration – JECLDEF for JES2

- Similar new parameters to control JES2 JECL processing

```
JECLDEF   JES2=(
    JOBPARM     =     PROCESS | IGNORE | WARN | FAIL
    MESSAGE     =     PROCESS | IGNORE | WARN | FAIL
    NETACCT     =     PROCESS | IGNORE | WARN | FAIL
    NOTIFY      =     PROCESS | IGNORE | WARN | FAIL
    OUTPUT      =     PROCESS | IGNORE | WARN | FAIL
    PRIORITY    =     PROCESS | IGNORE | WARN | FAIL
    ROUTE       =     PROCESS | IGNORE | WARN | FAIL
    SETUP       =     PROCESS | IGNORE | WARN | FAIL
    XEQ         =     PROCESS | IGNORE | WARN | FAIL
    XMIT        =     PROCESS | IGNORE | WARN | FAIL
                  )
```

# JES3 JECL Toleration – JECLDEF Options

- Options for how JECL statements are handled

| | |
|---|---|
| PROCESS | statement is processed |
| IGNORE | statement is ignored (treated as a comment) |
| WARN | statement is processed. However, a warning message is issued to record the occurrence of a JECL statement. |
| FAIL | an error message is issued and job is failed with a JCL error |

- Currently only supports processing keywords on JES3 MAIN JECL
  - Intend to add other statements over time
- IGNORE tells JES2 to treat statement as a comment
  - Does not affect what is passed to exits 4 and 54
  - Easier to develop exits to react to ignored JECL
- FAIL prevents JECL from creeping back into jobs after it has been eliminated

# JES3 JECL Toleration – Example

- Simple program with a //*MAIN card

```
//WASIKDG   JOB 'ACCT','IBM USER',MSGLEVEL=(1,1),NOTIFY=&SYSUID,CLASS=A
//*MAIN SYSTEM=SY1,LINES=(5,C),FAILURE=RESTART,DEADLINE=(0800,A,3)
//STEP1     EXEC  PGM=IEFBR14
```

- Output for job

```
12.27.11 JOB00019 ---- THURSDAY,  12 MAR 2015 ----
12.27.11 JOB00019  IRR010I  USERID IBMUSER  IS ASSIGNED TO THIS JOB.
        1 //WASIKDG   JOB 'ACCT','IBM USER',MSGLEVEL=(1,1),CLASS=A
        2 //*MAIN SYSTEM=SY1,LINES=(5,C),CLASS=TEST,DEADLINE=(0800,A,3)
        3 //STEP1     EXEC  PGM=IEFBR14
 STMT NO. MESSAGE
        2 HASP1133 Unsupported keyword DEADLINE used
```

- Display of the job

```
$HASP890 JOB(WASIKDG)    STATUS=(AWAITING EXECUTION),CLASS=TEST,
$HASP890                 PRIORITY=9,SYSAFF=(IBM1),HOLD=(NONE)
```

# JES3 JECL Toleration – JES2 exits

- Installation can also exercise a job level control of the JECL processing via JES2 exits
- Override default processing of JECL syntax in $XPL for
  - JOB JCL statement scan exit (exits 2 and 52)
  - JOB statement accounting field scan exit (exits 3 and 53)
- Exits can indicate whether JES2 or JES3 JECL statements is syntactically recognized (or both or neither, in any combination)
  - Similar to affect of INPUTDEF JES3JECL but for both JES2 and JES3 JECL
- Exits overrides syntax parsing of statements
  - Control on a JECL statement level is still determined by the options on the JECLDEF initialization statement.

# JES3 JECL Toleration – Null JCL Treatment

- JES3 ignores any records following a null JCL statement
  - "//" followed by all blanks
- JES2 ignores null JCL statement
  - Processing ends with /*EOF, //JOB, etc
  - Converter stops at "//" but JES2 JECL statements after "//" honored
- NULLJCL on INPUTDEF controls how JES2 treats a null JCL statement:

  `$TINPUTDEF,NULLJCL=IGNORE|EOF`
  - This setting has a member scope and is only active until this member is restarted.
  - NULLJCL=IGNORE maintains traditional JES2 behavior
    - Default value
  - NULLJCL=EOF treats null JCL statement as a logical end of file
    - This matches what JES3 does.

# PROCLIB SSI

- Also as part of this line item, a new option was added to JES property SSI
  - Get PROCLIB information (SSJPPROD, SSJPPRRS)
  - New mapping macro IAZJPROC
- Gets data set concatenations for JES2 PROCLIBs
- Options on what PROCLIBs to get
- Returns data set names, unit info, VOLSERs
- Similar to data returned via $D PROCLIB command

# RAS Enhancements

- JES2 internal CTRACEs moved to 64 bit buffers
  - Increased size of buffers and thus depth of trace
- New internal CTRACE for checkpoint operations
- New DEBUG option to drive regular job queue verifications

```
DEBUG QVERIFY=YES
```

  - High overhead but timely detection of queue errors
  - Very useful for debugging job/output/bert queue errors in test environment
  - NOT activated by DEBUG=YES
- Updated data in $D PERFDATA(CKPTSTAT)
- More granularity in WAIT data for $D PERFDATA(PCESTAT)
- Better selective $TRACE and filtering for processing that uses subtasks
- Many IPCS updates, new LOGRECs, etc.

# Migration & Coexistence Considerations

- Migrating from JES2 z/OS 1.11 and 1.12 is via an all member warm start
  - No coexistence support
  - No fallback (other than spool offload and cold start)
  - Must $ACTIVATE to z11 mode prior to warm start
- Migrating from JES2 z/OS 1.13 or z/OS 2.1
  - Must $ACTIVATE to z11 mode prior to starting JES2 z/OS 2.2
  - APAR OA41740 needed on z/OS 1.13, or z/OS 2.1 member to coexist in MAS with z/OS 2.2
    - UA76769 – HJE7780
    - UA76770 – HJE7790
  - APAR is required for fall back as well
    - Some new data structures created by z/OS 2.2 JES2 will result in problems for prior releases if OA41740 is not installed.

# Session Summary

- Support growing the number of jobs and output JES2 can support
  - Includes performance enhancements
  - Improved system management
  - Improved RAS
- Changes include
  - SYSOUT work selection enhancements
  - Updates for duplicate job processing
  - Reworking checkpoint on I/O processing
    - Includes using 64 bit storage
  - Simplify managing the JES2 checkpoint data sets
    - Size and configuration
  - Tuning MASDEF HOLD and DORMANCY automatically

# Session Summary

- JES2 now allows users to know more detailed information concerning the execution of their job steps thru the records recorded in the EVENTLOG data set
  - JES2 spool data set intended for program access
  - Optionally includes SMF records for a job
  - Can be sent over NJE with the job's output
- New job scheduling functions implemented in JES2 V2R2:
  - HOLDUNTL=
  - STARTBY=
  - WITH=
- JES3 JECL toleration
  - Support for JES3 Job Entry Control Language (JECL) statements
  - Control what JECL is supported in a customer environment
  - Provide options to eliminate subtle JCL differences
- PROCLIB SSI
- Numerous RAS enhancements

# Questions?

SHARE
Educate · Network · Influence

# Questions?

# Appendix

- Publications
  - z/OS V2R2.0 JES Application Programming – SA32-0987-01
  - z/OS V2R2.0 JES2 Commands – SA32-0990-01
  - z/OS V2R2.0 JES2 Initialization and Tuning Guide – SA32-0991-01
  - z/OS V2R2.0 JES2 Initialization and Tuning Reference – SA32-0992-01
  - z/OS V2R2.0 JES2 Installation Exits – SA32-0995-01
  - z/OS V2R2.0 JES2 Macros – SA32-0996-01
  - z/OS V2R2.0 JES2 Messages – SA32-0989-02
  - z/OS V2R2.0 MVS JCL Reference - SA23-1385-02
  - z/OS V2R2.0 MVS Using the Subsystem Interface – SA38-0679-02