JES2 Debugging

Adam Nadel
anadel@us.ibm.com
IBM - Poughkeepsie, NY

Thursday, March 3, 2015
Session Number 16632

#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.

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632

Get To Know Your Error With $HASP088s

```
$HASP088 JES2 ABEND ANALYSIS
$HASP088 ------------------------------------------------------
$HASP088 FMID = HJE7780 LOAD MODULE = HASJES20
$HASP088 SUBSYS = JES2 z/OS1.13
$HASP088 DATE = 2013.343 TIME = 10.23.45
$HASP088 DESC = DISASTROUS ERROR AT LABEL KBLOBERR
$HASP088 MODULE     MODULE    OFFSET    SERVICE   ROUTINE        EXIT
$HASP088 NAME       BASE      + OF CALL LEVEL     CALLED         ##
$HASP088 --------   --------  -------   -------   ----------     ---
$HASP088 HASPRAS    0003B480 + 0005E4   OA41318   *ERROR $DIS
$HASP088 HASPCKPT   1D8C44D0 + 0058F8   OA36155   $DISTERR
$HASP088 HASPCKPT   1D8C44D0 + 004BAE   OA36155   KBLEMPTY
$HASP088 HASPCKPT   1D8C44D0 + 0005CE   OA36155   KBLOB
$HASP088 PCE = CKPT (1DA4E3B8)
$HASP088 R0 =  E0000264 1DA4AFD4 00000000 1DBF18C8
$HASP088 R2 =  00000000 00000000 00000000 00000000
$HASP088 R4 =  00000000 1D8C9DC8 00000000 00000000
$HASP088 R6 =  00000F00 003A1700 00000000 1DA4AE70
$HASP088 R8 =  00000F00 00000000 00000000 00000000
$HASP088 R10 = 00000000 00000000 00000000 00007000
$HASP088 R12 = 00000000 0003B660 00000000 1DA4E3B8
$HASP088 R14 = 00000000 8003B9C0 00000000 0003C390
$HASP088 ------------------------------------------------------
```

Exit Code in Control

Save area calling sequence

PCE = maintask

DTE = subtask

Registers contents

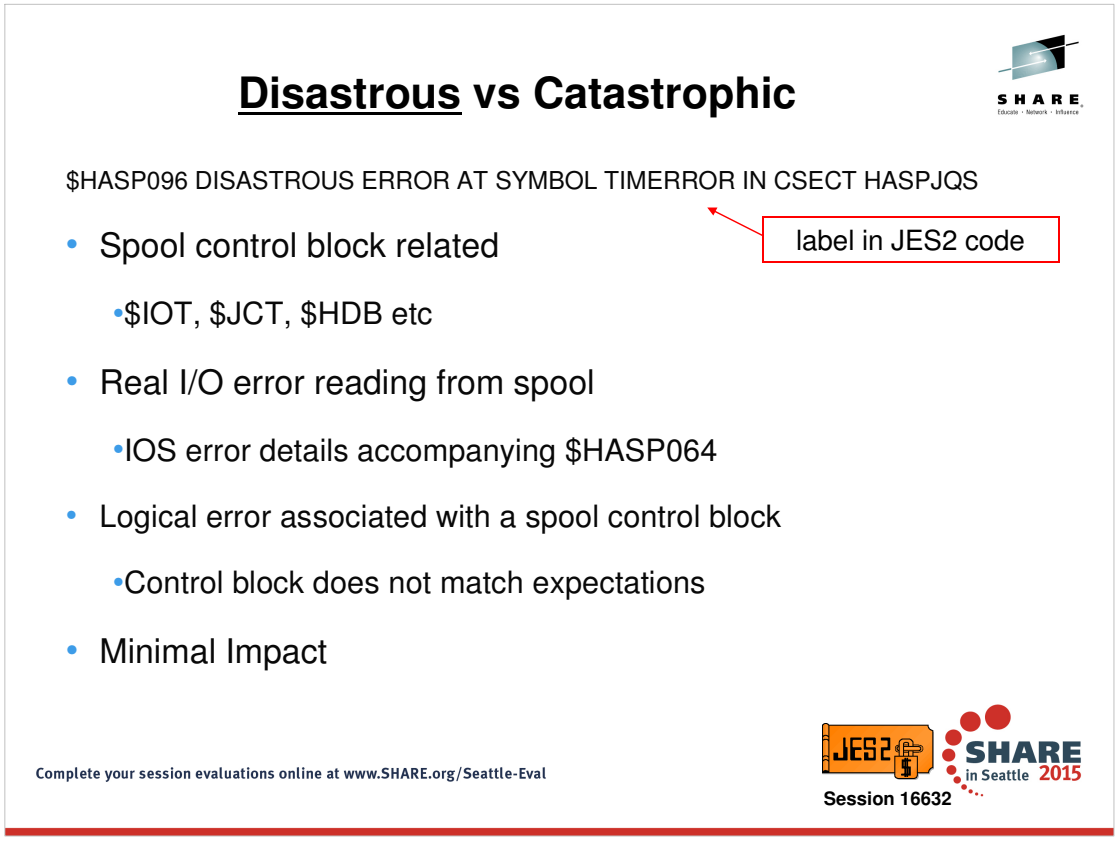
Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632

Most JES2 abends will be accompanied by diagnostic $HASP088 messages. These will be preceded by the JES2 message indicating whether the dump is catastrophic ($HASP095) or disastrous ($HASP096) and followed by message indicating what level of recovery has occurred otherwise termination options.

Depending on the type of error produced, the $HASP088 messages may also contain other useful information such as the jobname (and jobid) being processed at time of error, the home/primary/secondary ASID at time of error (not guaranteed that JES2 is the primary ASID) etc.

The NETRV address space (JES2Snnn) has an equivalent version via $HASP5088



Which is worse – Disastrous or Catastrophic errors? Since most forms of disastrous errors are logical errors in which part of control block does not match expectations, they are typically far less severe than catastrophic in terms of impact. In logical error cases, there is typically no loss of JES2 function and the impact of error confined to the JOB in-hand. Real I/O errors are far less common, but under those circumstances the impact would not necessarily be confined to a single JOB etc.

While less severe in terms of impact, disastrous errors can often be more difficult to debug – because it often entails reviewing the entire lifespan of the JOB(s) affected to understand what may have caused the spooled block to not match expectations. Did something prevent an IO from completing successfully such as an error/abend within the job itself? Or was there an disruption to JES2 overall (not a clean shutdown etc)?

What Is a CBIMPL4?

```
$HASP096 DISASTROUS ERROR AT SYMBOL CBIMPL4 IN CSECT HASPNUC,
MQTR=040000A1B90C,UNIT=A056,VOLSER=JES11

$HASP088 JES2 ABEND ANALYSIS
$HASP088 ----------------------------------------------------
$HASP088 FMID = HJE7790 LOAD MODULE = HASJES20
$HASP088 SUBSYS = JES2 z/OS 2.1
$HASP088 DATE = 2014.005 TIME = 09.20.19
$HASP088 DESC = DISASTROUS ERROR AT LABEL CBIMPL4
$HASP088 MODULE      MODULE      OFFSET      SERVICE  ROUTINE      EXIT
$HASP088 NAME        BASE        + OF CALL   LEVEL    CALLED       ##
$HASP088 --------    --------    -------     -------  ----------   ----
$HASP088 HASPRAS     00022E30    + 0005E4    OA37847  *ERROR $DIS
$HASP088 HASPNUC     00007000    + 0095A4    OA37654  $DISTERR
$HASP088 HASPTRAK    1A630EE0    + 000DC6    OA37847  $CBIOM
$HASP088 HASPTRAK    1A630EE0    + 0002DA    OA37847  PURSAF
$HASP088 HASPTRAK    1A630EE0    + 003432    OA37847  $PURGER
$HASP088 HASPTRAK    1A630EE0    + 002C54    OA37847  VIOTPRG
$HASP088 PCE = PURGE (1A84F0A0)  JOB12345 ADAM1
...
```


CBIMPL4 is the most common of JES2 errors. It is a logical error (disastrous) in which JES2 is attempting to access a control block for job A (in above case JOB12345 – ADAM1) and instead reads in a block for job B. The buffer identifying job B can be found within the respective PCE save area chain (which we will cover shortly) or in the respective SYMREC indicating SPOOL TRACKGROUP RECOVERY.

If this is a single/isolated instance, then there is no cause for high alarm and we are looking for some kind of disruption within the lifespan of the job(s) in question that could have prevented an IO from completing. If this is a one of MANY errors of the same/similar nature, then would be greater concern as it could be a reflection of adverse impact to spool and/or checkpoint – such as accidentally starting JES2 with wrong spool or checkpoint volume(s) etc.

What Is a CBIMPL4?

- Recovery is confined to job identified
  - Job is purged
  - Track recovered by spool trackgroup reclamation (SNIFFER)
    - SYMREC produced
    - Can be controlled/expedited via SPOOLDEF,GCRATE=NORMAL/FAST

```
COMPONENT ID: 5752SC1BH
COMPONENT RELEASE LEVEL: Z113
SERVICE RELEASE LEVEL: OA38671
DESCRIPTION OF FUNCTION: SPOOL TRACKGROUP RECOVERY
PROBLEM ID: SYMTABB SUBSYSTEM ID: JES2Z113
...
...
FREE FORMAT COMPONENT INFORMATION:
     KEY = 010D LENGTH = 000003 (0003)
     +000 0304A4
     KEY = 010E LENGTH = 000008 (0008)
     +000 00000000 00000000
     KEY = 010F LENGTH = 000008 (0008)
     +000 80004F2A B6B46F7F
     KEY = 0110 LENGTH = 000256 (0100)
     +000 C8C4C240 ... ... ...
```

SYMREC type

MTTR/MQTR

Control block contents (HDB, IOT, etc)

The impact is that the affected job will be purged and the trackgroup in question (that contained residual data for a different job) will be temporarily be marked as not owned. Thereafter, the JES2 trackgroup reclamation PCE (aka SNIFFER) will run and clean up the track group, restoring it to the track group map for future reuse. SNIFFER defaults to NORMAL setting – it will cycle through all tracks within ~7 days. It can be increased to $TSPOOLDEF,GCRATE=FAST which causes SNIFFER to interrogate all tracks immediately (and after that it automatically returns to NORMAL rate).

Disastrous vs Catastrophic

$HASP095 JES2 CATASTROPHIC ABEND. CODE = S0C4 (RC = 00000004)

JES2 error condition $nnn

-or- MVS ABEND

- CODE=ERROR
  - JES2 detected error condition
    - $Knn – CKPT read/write errors – module HASPCKPT
    - $Qnn – problem with job (JQE) – module HASPJQS
    - $Jxx – problem with output (JOE) – module HASPJOS
  - Error regs found in $ERROR save area
  - JES2 internal Ctraces useful in diagnosis
- CODE=ABEND
  - MVS detected error (0C4, 878, B00, etc)
  - JES2 maintask ESTAE gets control for recovery
  - RTM2WA generated
  - System trace table

Session 16632

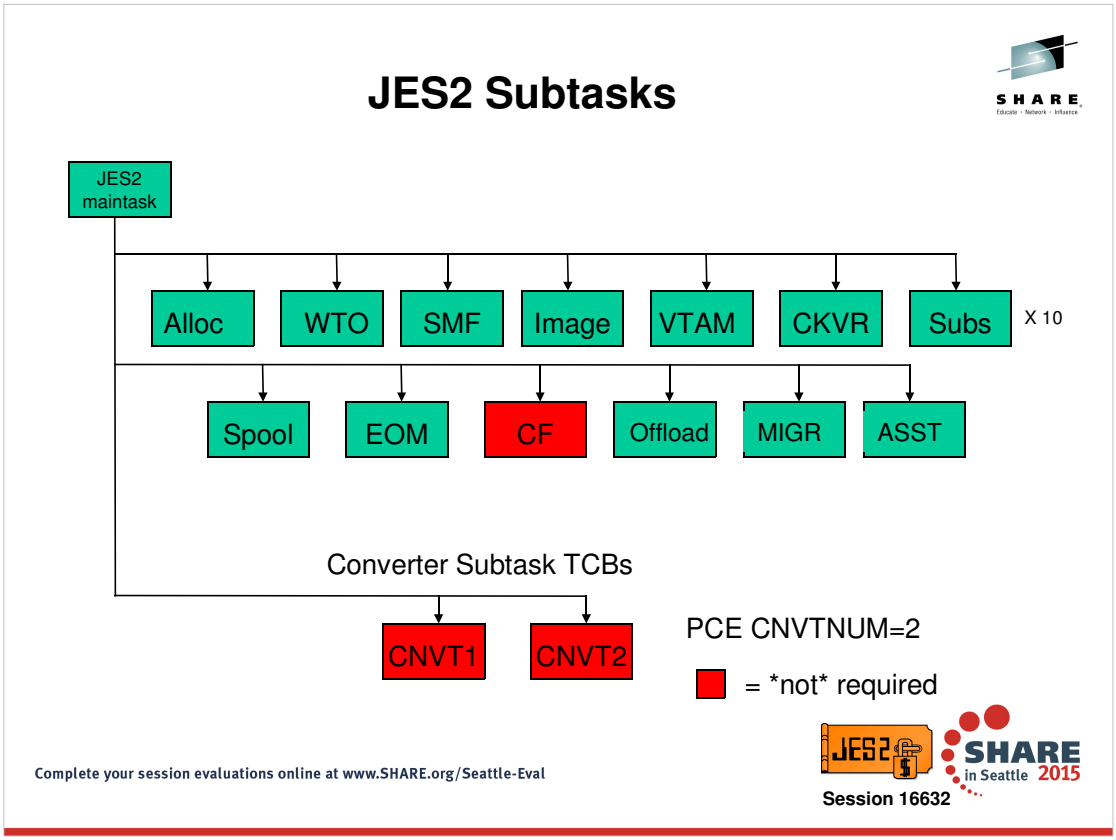
Catastrophic errors are unexpected, logically detected errors. They encompass both JES2 detected errors as well as general MVS abends encountered under JES2. For MVS abends, it is appropriate to approach their diagnosis as you would any other MVS type abend – using RTM2WA, systrace, SUMM FORMAT, etc.

For JES2 detected errors, there are diagnostics available within JES2 such as $ERROR save area calling sequence and internal Ctraces.

PCE Recovery (or not)

- $HASP098 Enter Termination Option – worst
  - Required PCE failed and could not be recovered
- $HASP073 Recovery Successful – best
  - Normal processing resumes
  - May be confined to job in hand
- $HASP068 Partial Recovery Successful – good enough?
  - PCE has terminated and will not run again
  - Processing continues without that PCE
    - How many PCEs remain of that type
    - Is function impacted
    - How can I recover PCE

Depending on the severity of the error, there are varying degrees of JES2 recovery. Partial recovery is intended to keep JES2 operating and stable and allow time to schedule hostart/IPL at your nearest convenience to recover lost PCE. The type of impact may vary based on the specific type of PCE affected. A device PCE (PRT1) means the device will not function (may be critical). Other PCEs such as Sysout API (SAPI) interface may have far smaller impact depending on the number of PCEs defined. When JES2 terminates a PCE, it produces a message indicating the PCE has terminated and also how many of that type remain. An ended PCE will prevent a clean shutdown of JES2 and can be identified via $DPCE(*),ENDED.



JES2 maintask does not like to MVS wait. JES2 creates separate subtask TCB's to invoke services that may result in an MVS wait. There are 14 different JES2 subtask types of which one is for conversion.

The number of converter subtasks corresponds to PCEDEF CVNTNUM parameter. The default is 2. The MVS converter is linked to in order to converter the JCL images.

The MVS converter also performs the PROC expansions.

Brief Summary of Subtask functions:

ALLOC- used to perform dynamic allocations

WTO – issues MVS WTO to put out JES2 messages

SMF – writes SMF records to SMF dataset

IMAGE – allocates and opens SYS1.IMAGELIB (only done during JES2 startup)

VTAM – used to open or close VTAM ACB

CKVR – checkpoint versions and WLM sampling

SUBS – general purpose subtasks most often used for performing SAF calls (there are 10 of these TCB's)

SPOOL – handles spool volume allocations etc

EOM – z4 and up. Processes $SJB placed on the EOM queue for end of memory SSI processing

CF – used when CKPT is on coupling facility to interface with the CF to read/write CKPT data

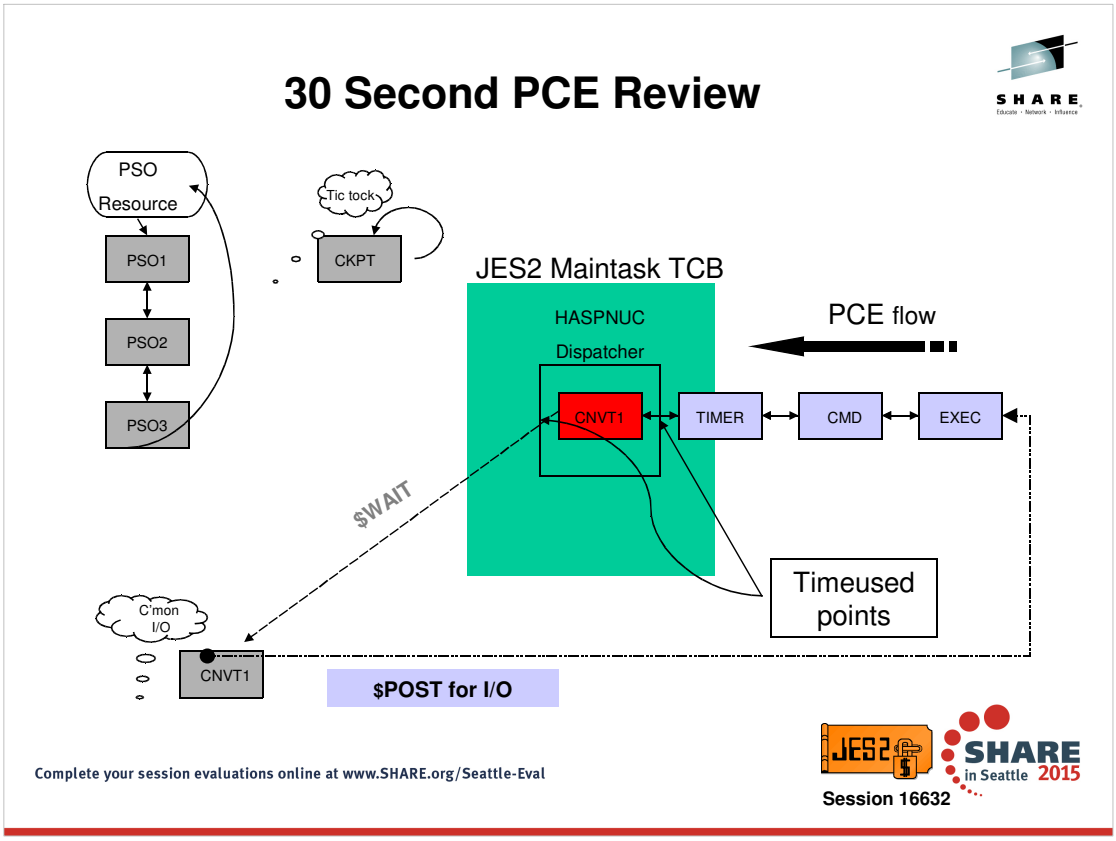
OFFLOAD – used to perform I/O etc to offload datasets

MIGR and ASST– involved in spool migration processing

JES2 Subtask Recovery (or not)

- $HASP078 Subtask failed
  - Indicates the failing JES2 subtask
  - Always MVS abend code
- $HASP095 error $Z03 issued if a required subtask cannot be recovered
- Potential function loss when subtask terminates

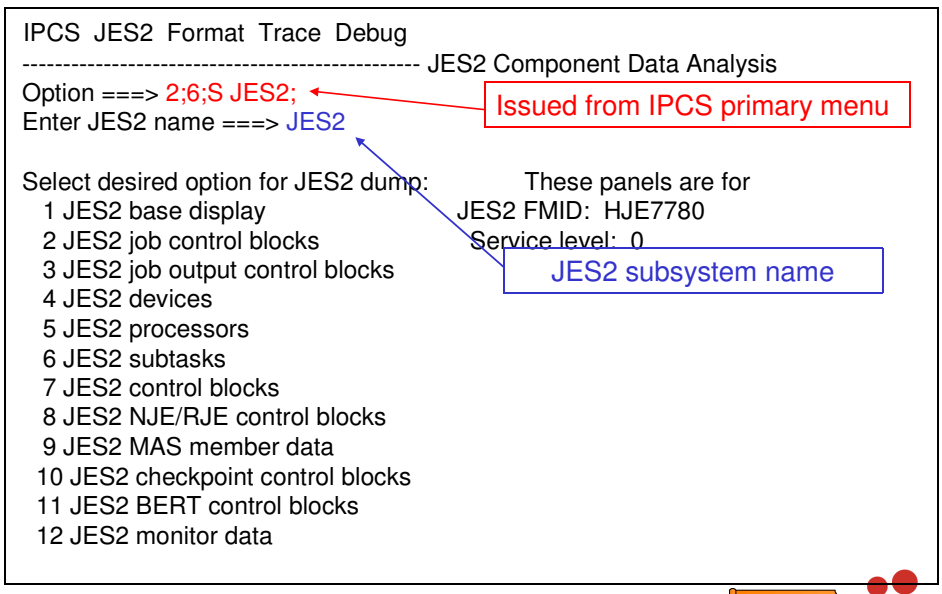
Impact may vary depending on the type of subtask that was impacted. Barring the *not* required subtasks, the overall health JES2 is typically going to be in trouble if it loses a subtask. For instance, losing a CNVT subtask may not be critical if you have 10 defined. However, the loss of the CKPT version subtask may prevent the updating of checkpoint versions (copies) – which could affect respective exploiters like SDSF. The loss of the VTAM subtask would impact SNA communications etc.



The Processor Control Element (PCE) represents an instance of a “process” running under the control of JES2 main task – each PCE is a dispatchable unit of work controlled by the JES2 dispatcher. “Process” is synonymous for JES2 service – such as EXEC, CMD, SAPI, PSO, CNVT, etc. There are one or more PCEs for each process, some dictated by PCEDEF statement definitions.

Above illustrates basic flow of PCE’s being dispatched by JES2 maintask: When a PCE has work to do, it is moved into the ready queue (awaiting their turn to be dispatched). When the PCE’s runs through the dispatcher its entry and exit into and out of it is framed with TIMEUSED macros. This allows JES2 to capture CPU time information that shows up in internal traces and PERFDATA.

JES2 Component Panels



The following slides assume that the JES2 IPCS Support modules (SHASPARM, SHASMIG, SHASPNL0) have been loaded into the requisite concatenations on your system. For specific information on JES2 IPCS Support modules, please refer to z/OS JES2 Diagnosis manual (chapter: Using IPCS for Diagnosis).

From this panel, you have various formatting options based on what you are attempting to debug; however, option “1 JES2 base display” is often the best place to begin diagnosis as it is surfaces an abundance of pertinent information. Most of the options place you in another panel with prompting fields for additional information. The panels do have help screens to assist in navigation and data entry.

The subsystem name defaults to JES2, but is an overtype field for alternative JES2 subsystem names (JESA etc)

Also on this panel (but not illustrated above), is option “101 – Select JES2 control blocks for non-JES2 address space”. These panels may be useful for JES2-related abends that occur within a user address space – allowing formatting of JES2 control blocks that reside in common storage

JES2 Base Display

*** JES2 Base Display ***

Subsystem "JES2" is in address space ASID(X'002D')
Dump for JES2 release="z/OS 2.1", Product level=43, Service level=0 (pointed to by SSCTSUSE); CVTPRODI=HBB7790
Maximum extended region size for "JES2" is 1,395M (per LDAELIM)
*** **WARNING**: ASCBDSP1=80
System set non-dispatchable and this ASCB is not exempt (per ASCBSSND bit)
*** **WARNING**: DEBUG BERT=NO specified (per $DBGBERT bit off in $DEBGOPS in $HCT)
*** **WARNING**: $EVENT(s) exist (PCBEVNTF¬=0 in $PERFCB)

*** **NOTICE**: $QSUSE is NOT in effect (per $QSONDA bit in $STATUS in $HCT)
*** **NOTICE**: SPOOLDEF FENCE=ACTIVE=YES in effect (per CCTSMVFN bit in CCTSTUS in $HCCT)

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632

This is the top portion of the Base Display. It includes the JES2 product information along with WARNING, NOTICE, and ERROR alert messages. You will always find alert messages in a dump, so their presence alone is not indicative of any particular problem. However, WARNING and NOTICE messages draw attention to key pieces of information that will assist the debugger in understanding the state of JES2 at time of dump. Some examples are:

-JES2 is abending/abended

-JES2 is quiesced via $P or $PXEQ

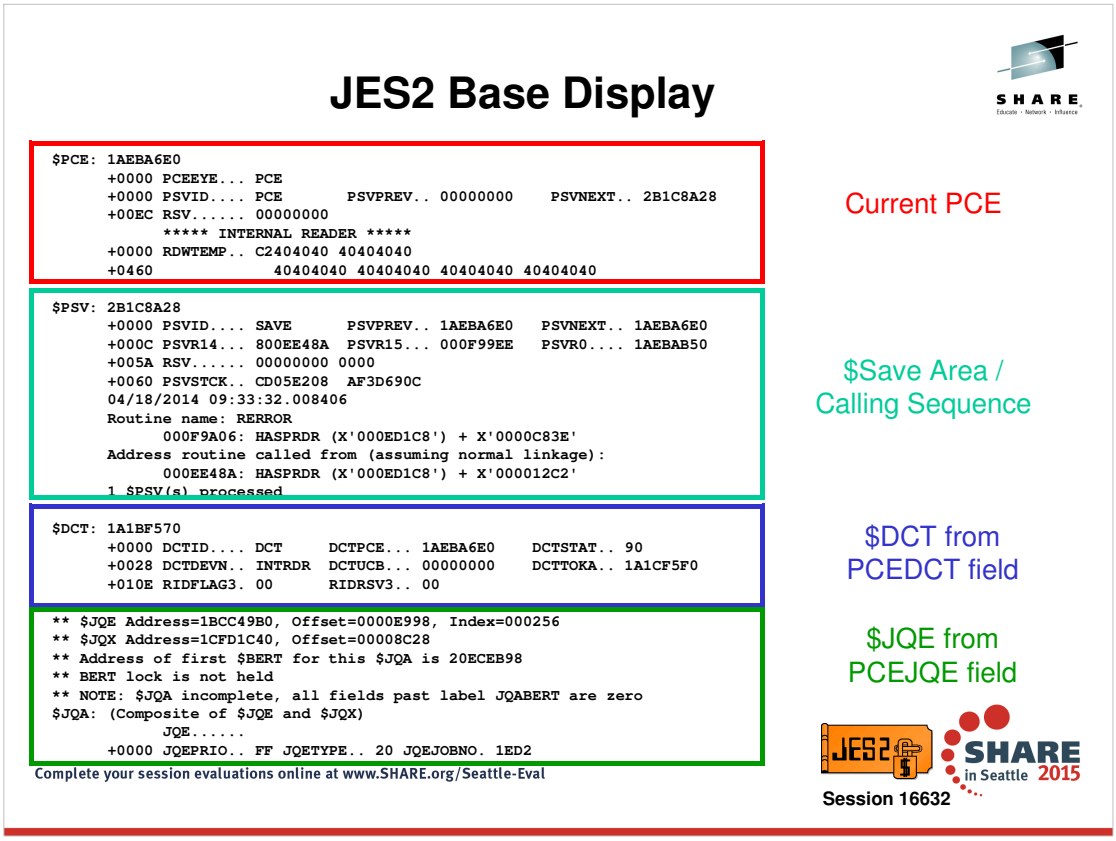
-$ZAPJOB has been issued

-$EVENTS exist (produced by JES2MON)

-JES2 ASCB is not dispatchable

etc

ERROR alerts often indicate that certain areas are not able to be formatted. These may be rather innocuous and simply reflect that some storage area was not dumped, or can shed insight into control block overlay scenarios etc.



This section is towards the bottom of the Base Display panels. I have omitted the middle section which also displays the $HCT and $HCCT control blocks. All of this information can also be formatted via other JES2 panels (such as PCE panels, job display panels, subtask panels, etc).

The PCE Save Area ($PSV) can be thought of as the JES2 version of a linkage stack – one entry produced per PCE to represent the state of processing as it issued a $SAVE (but not yet issued the $RETURN). Once the $RETURN is issued, the PSV is dechained and available for reuse. It provides a lot of insight into the path leading up to the error (including register contents) and will match up to the calling sequence identified in the $HASP088 messages.

It will also format and display other control blocks that are active/in-hand at time of error such as device blocks (device control table $DCT), job blocks (job queue element $JQE, output blocks (job output element $JOE), etc.

Useful Commands & Module Background

```
Command ===> IP CBF 000091A0 STR($MODLOC)
******************************** TOP OF DATA ********************************
000091A0: HASPNUC  (X'00007000') + X'000021A0'    OA36155/UA68055
```

- HAS**C**nnnn → JES2 module in **C**ommon storage
  - Maintenance hitting module typically requires WARMstart (IPL)
- HAS**P**nnnn → JES2 module in **P**rivate storage
  - Maintenance hitting module typically requires HOTstart

```
Command ===> IP CBF 072E3050 STR($PCE)
******************************** TOP OF DATA ********************************
$PCE: 072E3050
  +0000 PCEEYE... PCE
  +0000 PSVID.... PCE          PSVPREV.. 00000000    PSVNEXT.. 072E3050
  +0018 PSVR1.... 069CC230 PSVR2....   069CC138    PSVR3.... 00003000
```


Browsing raw storage, JES2 module eyecatcher information is at the beginning of each module; however, the maintenance level information is at the end of the module. For this reason, is often very helpful to use the $MODLOC formatter to verify if/where an address is in JES2 code. The formatters will also work within the user address space for common modules.

JES2 common modules HASCnnnn are primarily responsible for:

-SSI calls

-Extended status

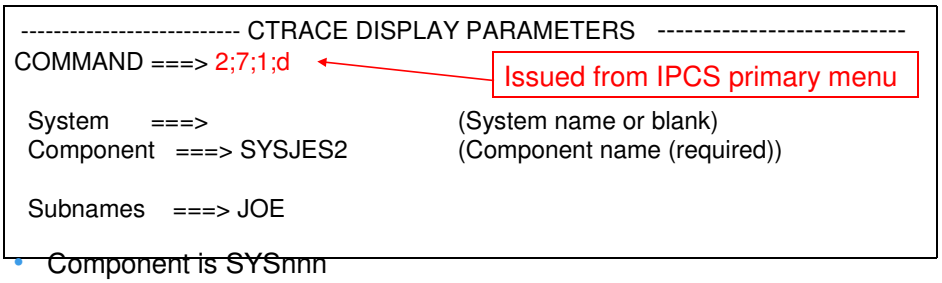
-Sysout allocation / open / close / PUT / GET / POINT

-SAPI / PSO

The mainline recovery for common modules is HASCLINK

The second example shows formatting an address as a $PCE. JES2 has formatters for many control blocks ($JQE, $DTE, $JOE, etc), so it may be worthwhile to attempt a CBF against that respective block to assist in formatting (rather than dealing with raw storage).

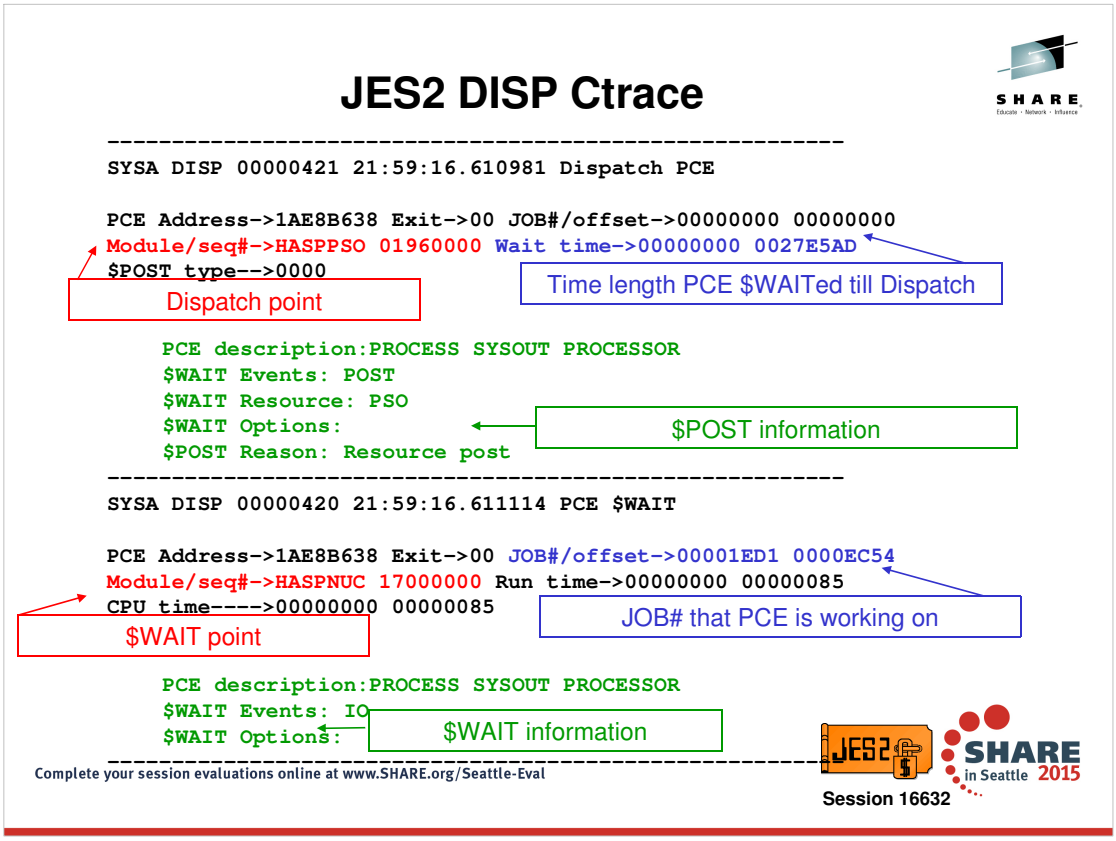
JES2 Ctraces



- Component is SYSnnn
  - nnn = JES2 subsystem name (JES2, JESA, etc)
- Subnames
  - DISP
  - JQE
  - JOE
  - SAPI **new** (delivered via APAR OA43882)

The JES2 Ctraces are component traces that are always running internally. They are in-storage only and cannot be put out to external writer etc. The installation does not control the size of the trace, and they are rolling traces. These traces can be displayed via the IPCS component trace facility as displayed above. Alternatively, you can use the “TRACE” drop down menu from the JES2 primary panel (shown on slide 11).

There are four types of traces/subames: DISP, JOE, JQE, SAPI



The JES2 dispatcher rolling ctrace shows information on each PCE as it respectively enters/exits the JES2 dispatcher. It also will show when JES2 encounters an MVS WAIT.

Things to consider while reviewing DISP ctrace:

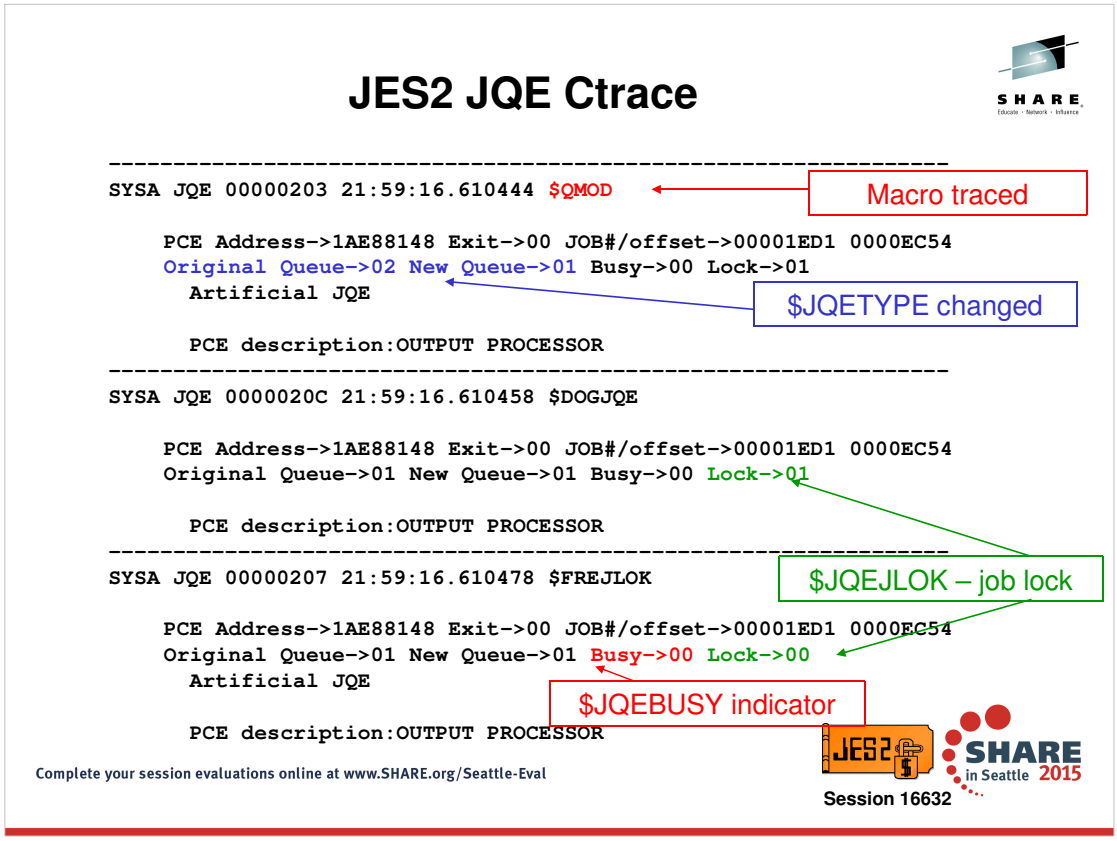
-Are there abnormally large time gaps between entries or large MVS waits?

-Are one (or more) specific PCE unexpectedly monopolizing the dispatching?

-Any PCEs appear to be looping?

-Any unusual $WAIT conditions?

-Is an exit in control (related to any of the above)?



The JQE rolling ctrace The above shows information about job state a job state changes – particularly the (un)busying of the job block, (un)locking of the job, and transitioning of job from queue-to-queue. The above case illustrates a job moving from the output queue (being serviced by a OUTPUT PCE), to the hardcopy queue. As part of this process we can observe the joblock is obtained and then freed.

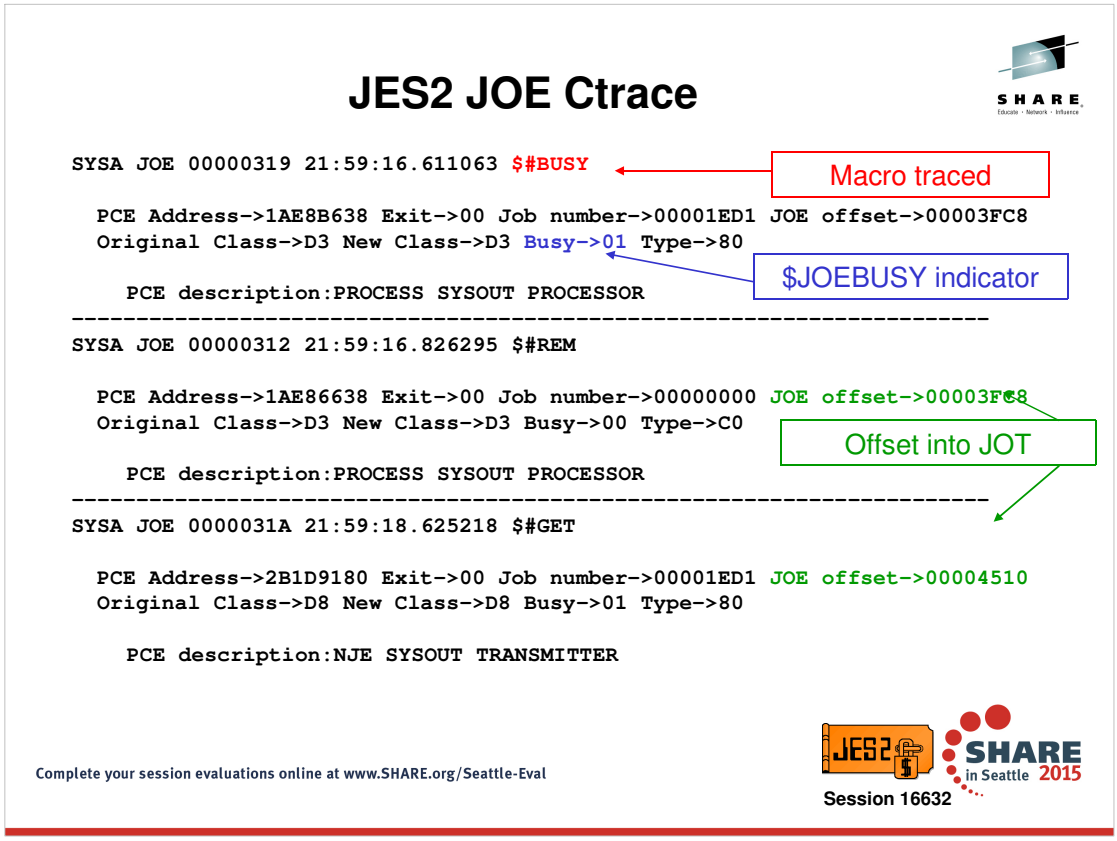
Things to consider while reviewing JQE ctrace:

-Are there any large gaps in processing?

-Are you looking for a specific job?

-Are you looking to see that a particular queue/phase is being serviced (backlog?)?

--Is an exit in control (related to any of the above)?



The JOE rolling ctrace The above shows information about job output state a job state changes – particularly the (un)busying of the output block, (un)locking of the output, and transitioning of output from queue-to-queue. The above case illustrates two pieces of output within the same job being processed. This is evident by the two different offsets into the JOT along with each JOE being within different classes. The first piece of output is processed by PSO and purged (noted by JOETYPE=C0). The second piece of output is then selected by a NJE sysout transmitter Lnn.STn.

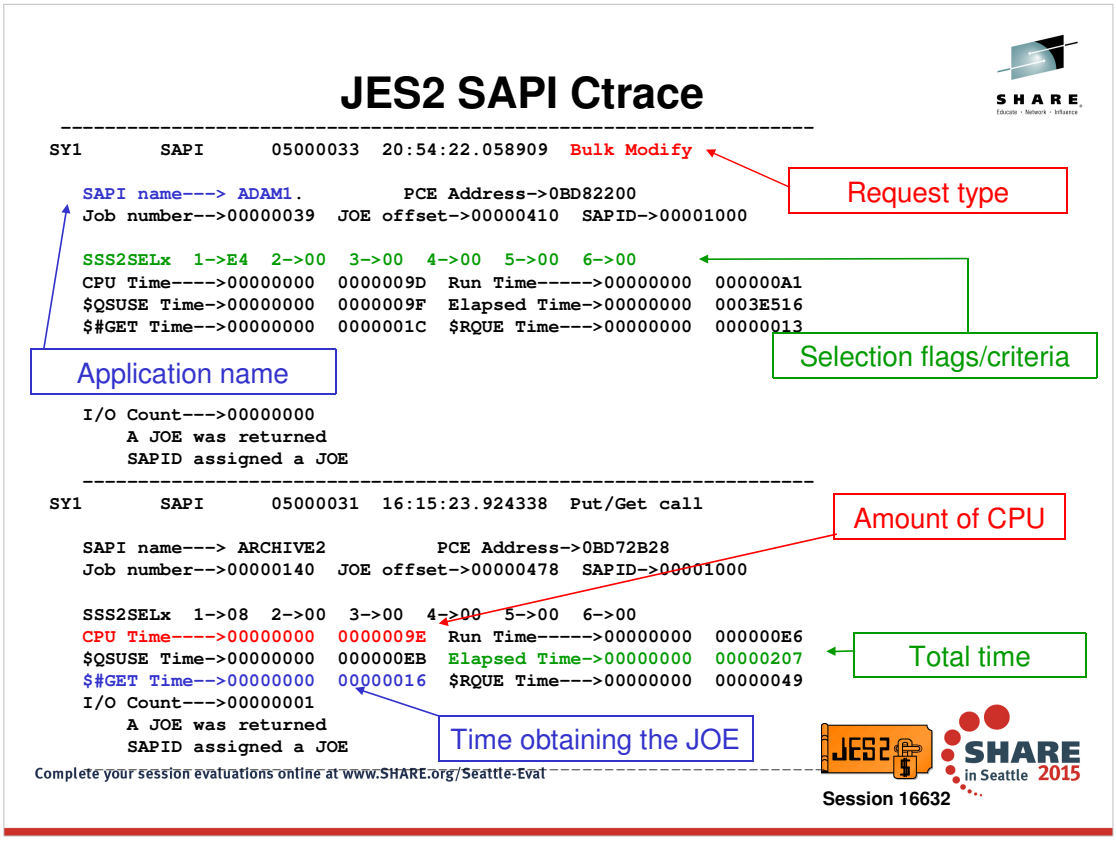
Things to consider while reviewing JOE ctrace:

-Are there any large gaps in processing?

-Are you looking for a specific output?

-Are you looking to see that a particular queue/phase is being serviced (backlog?)?

--Is an exit in control (related to any of the above)?



The SAPI rolling ctrace captures the last 2000 SAPI requests at time of dump. It identifies the requestor, the type of request and breakdown of the overhead of the request. Much of the same information is captured in the JES2 id traces (ID=28,29 for SAPI, ID=20 for $#GET) – depending on the duration of the SAPI problem and timeliness of dump, the internal Ctrace may be sufficient. JES2 id traces are more appropriate for capturing data across a wider timeframe.

Things to consider while reviewing SAPI ctrace:

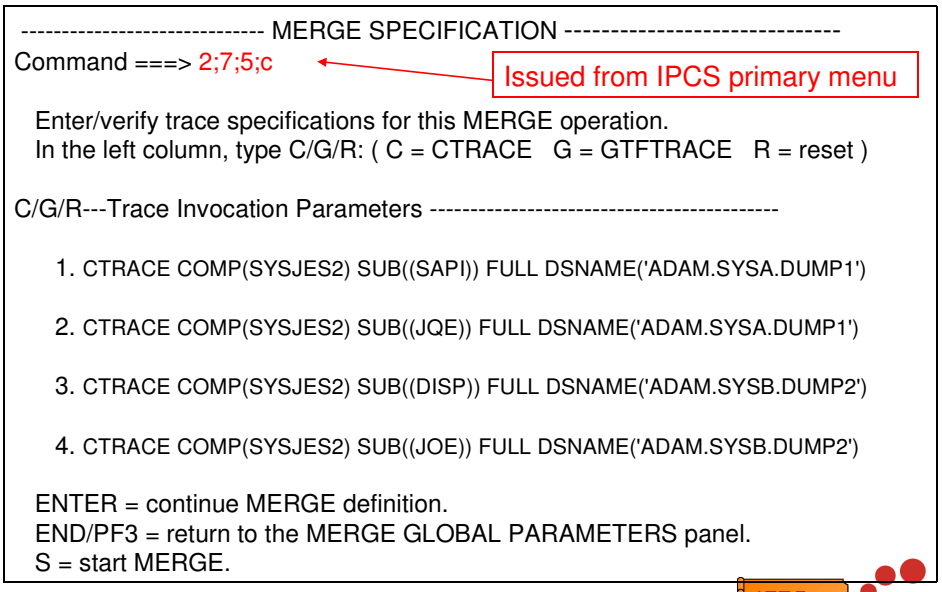
-Are there any unexpected SAPI applications involved in the processing of job output?

-Any SAPI application appear to be looping/processing same output?

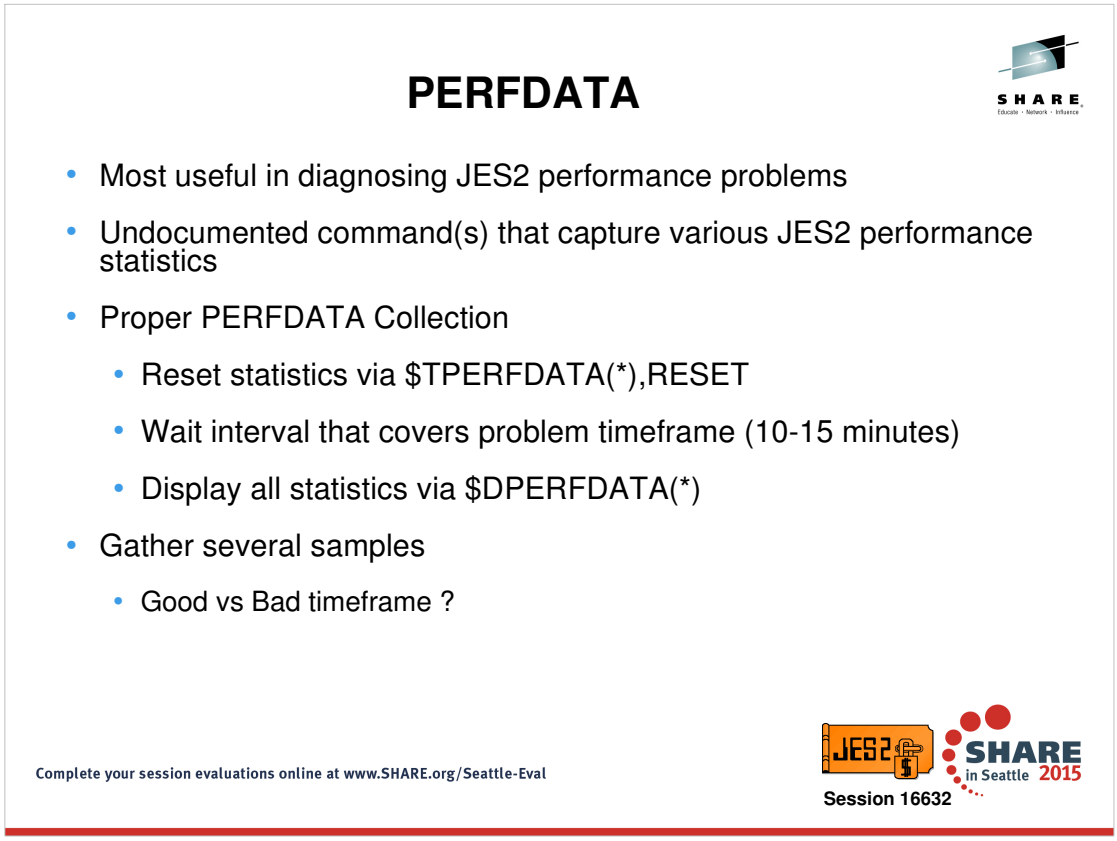
-Any requests taking long in duration (wall clock or CPU time)?

-What is the specific request type and criteria (related to any of the above)?

Merging Ctraces



Sometimes it may be beneficial to merge ctraces in order to gain a better understanding of the processing flow. This can be achieved via the MERGE ctrace facility. Note that you can specify multiple datasets which can be handy if processing for a job spanned different JES2 MAS members.



When diagnosing JES2 performance concerns, it is of paramount importance that PERFDATA is collected correctly. PERFDATA statistics are always running/accumulating, so resetting the statistics is always the first step to collecting an accurate sample. Without the reset, you may be investigating a problem in which JES2 CPU utilization drastically increased over a 15 minute timeframe using statistics covering an interval of 20+ days! In those cases, the data is considered oversaturated in that there is no way to discern what actually happened in those specific 15 minutes. Typically Level 2 recommends gathering samples in 10-15 minute increments (and you can always gather multiple/back-to-back samples).

It is also worth consideration to occasionally capture sample(s) when processing is good/normal. These can become handy to compare and contrast if JES2 performance drastically changes for the worse. A debugger can view the samples side by side to observe differences in PCE processing, job throughput, checkpoint cycling, etc.

PERFDATA

- $T PERFDATA(*),RESET – resets performance data
- $D PERFDATA(INITSTAT) – JES2 initialization stats
- $D PERFDATA(QSUSE) – PCE $QSUSE summary
- $D PERFDATA(PCESTAT) – detailed PCE stats
- $D PERFDATA(SAMPDATA) – WLM Sampling data
- $D PERFDATA(CPUSTAT) – PCE CPU usage
- $D PERFDATA(CKPTSTAT) – CKPT read/write stats
- $D PERFDATA(SUBTSTAT) – JES2 subtask
- $D PERFDATA(EVENT) – $EVENTS captured
- $D PERFDATA(WSSTAT) – work selection ($#GET & $#POST) stats
- $D PERFDATA(*) –all of the above

The $TPERFDATA(*),RESET is absolutely essential to ensuring a healthy sample is gathered. Thereafter, the type of display requested may be dictated by the specific problem being investigated; although Level 2 most commonly asks for all data $DPERFDATA(*). The WSSTAT option is a newly added section delivered In APAR OA43882. Disclaimer – the output of these displays can be rather abundant!

$D PERFDATA(CKPTSTAT)

$HASP660 **CKPT PERFORMANCE STATISTICS** SYS1-INTERVAL=11:10:12.320961,

$HASP660 **AVGHOLD=0.318337,AVGDORM=45.305289**,**TOT$CKPT=3284**,

$HASP660 WRITE-4K=0,WRITE-CB=788,OPT$CKPT=2496,OPT4K=0,

$HASP660 IO=R1,COUNT=875,AVGTIME=0.010943,

$HASP660 IO=R2,COUNT=0,AVGTIME=0.000000,TOTAL4K=0,TOTALCB=118,

$HASP660 IO=PW,COUNT=876,**AVGTIME=0.004066**,TOTAL4K=39,TOTALCB=0,

$HASP660 IO=IW,COUNT=878,**AVGTIME=0.003776**,TOTAL4K=0,TOTALCB=670,

$HASP660 IO=FW,COUNT=876,**AVGTIME=0.003888**,TOTAL4K=0,TOTALCB=118

AVGHOLD & AVGDORM = average HOLD and DORMANCY values

TOT$CKPT = total number of $CKPTs

AVGWAIT = average I/O times to CKPT

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632

The CKPT statistics section will illustrate whether your CKPT cycling is what you would expect based on HOLD and DORMANCY settings as well as provide information about the relative health of CKPT I/Os. On each member in the MAS you can compare AVGHOLD and AVGDORM versus the HOLD and DORMANCY coded values to verify if JES2 is cycling the CKPT as expected. Large differences in these values may suggest additional tuning is needed based on workload distribution etc (eg should some members be favored more because it does most of the job submit? … or archiving?)

Comparing TOT$CKPT on each member of the MAS is a quick way to assess which members have the most checkpoint activity.

The AVGWAIT values associated with primary/intermediate/final write can be used to assess relative health of the I/Os. These times can vary (particularly depending on checkpoint placement on DASD vs coupling facility); however, the numbers should be consistent for I/O on the device.

$D PERFDATA(CPUSTAT)

$HASP660 **CPU PERFORMANCE STATISTICS** SYS1 - **INTERVAL**=14:49.926816,

$HASP660 **CPU**=3.067221,

INTERVAL = length of time data has been accumulating

CPU = CPU time used by all of JES2 over that interval

$HASP660 **PCENAME=SPI**, **CPU%=6.16** ,**CPU=0.120145**,**TIME=0.151743**,

$HASP660 QSUSE_TIME=0.098023,IOCOUNT=599,CKPT_COUNT=28280,

PCENAME = name of the group of PCE's captured (see $DPCE for details)

CPU% = percentage of total JES2 maintask time this subset of PCE's used

CPU = total CPU time used by this subset of PCE's

TIME = Wall clock time this subset of PCE's was disp

This will show a breakdown of CPU utilization by PCE type. These numbers may change greatly/frequently based on configuration and workload. For instance, a member that is primarily used for archiving may demonstrate SAPI utilization far higher than a member used for communications/NJE (in which that member may show higher utilization for NET.SR etc). It is typically very reasonable to see CKPT PCE towards the top of this list.

Additional fields:

QSUSE_TIME = Wall clock time subset of PCE's ran when they acquired queue ($QSUSE)

IOCOUNT = Total number of I/O's issued by this subset of PCE's

CKPT_COUNT = Number of $CKPT's issued by this subset of PCE's

$D PERFDATA(PCESTAT)


```
$HASP660 PCENAME=NET.SR,TIME=43.011997,CPU=40.892083,CPU%=7.63,
$HASP660 QSUSE_TIME=0.277515,IOCOUNT=278356,CKPT_COUNT=7631,
$HASP660 WAIT=IO,MOD=HASPNUC,SEQ=17000000
$HASP660    COUNT=4066,AVGWAIT=0.001505,
$HASP660      POST=IO,COUNT=4066,AVGWAIT=0.001505,
$HASP660 WAIT=BUF,INHIBIT=NO,MOD=HASPNSR,SEQ=70272000
$HASP660    COUNT=4982,AVGWAIT=0.001099,
$HASP660      POST=RESOURCE,COUNT=4970,AVGWAIT=0.000878,
$HASP660      POST=IO,COUNT=12,AVGWAIT=0.092677,
$HASP660 WAIT=CKPT,MOD=HASPNUC,SEQ=28410000
$HASP660   COUNT=221,AVGWAIT=0.342762,CMOD=HASPJQS,CSEQ=03330000,
$HASP660     POST=RESOURCE,COUNT=221,AVGWAIT=0.342762
```


This section starts with the CPU statistics (CPUSTAT), and then follows with a breakdown of activity of that PCE type. Specifically, it will breakdown the PCE $WAITs and $POSTs by type and count. The AVGWAIT for WAIT=CKPT help give insight into CKPT access time. All wait times include queue time for the PCE type – the time it is on the ready queue awaiting dispatch.

Fields:

WAIT = wait type(s) passed on the $WAIT macro

MOD/SEQ = module and sequence number where $WAIT was issued

COUNT = the number of $WAITs (or $POSTs) issued from this location

AVGWAIT = Average time the PCE spent at this location waiting

POST = Post type that woke the PCE up from this $WAIT

PERFDATA - CPU Increase Example

Problem: JES2 CPU spike! $DPERDATA for a 30 minute interval provided:

$HASP660 **PCENAME=STAC**,**TIME=9:05.678529**,**CPU=5:58.996610**,**CPU%=95.67**,

$HASP660 QSUSE_TIME=9:01.582558,IOCOUNT=0,CKPT_COUNT=0,

$HASP660 WAIT=CKPT,MOD=HASPNUC,SEQ=28410000

$HASP660 COUNT=1653,AVGWAIT=0.756200,CMOD=HASPSTAC,CSEQ=13100000,

$HASP660 POST=RESOURCE,COUNT=1653,AVGWAIT=0.756200,

$HASP660 **WAIT=STAC**,INHIBIT=NO,MOD=HASPSTAC,SEQ=09900000

$HASP660 **COUNT=1646917**,**AVGWAIT=0.008434**,

CPU% & PCENAME identify STAC (Status/Cancel) as likely culprit.
WAIT=STAC is the STAC PCE wait for work.
COUNT with STACNUM=2 on PCEDEF indicates 823,458 SSI requests made relative rate of **AVGWAIT**.

The above demonstrates a scenario in which STAC PCE is monopolizing the JES2 activity – using 95% of the total CPU consumed by JES2. It is interesting to observe TIME vs CPU divergence. In the above case CPU is ~2/3 of the wall clock TIME – indicating JES2 is ready to run but is not getting CPU cycles. The CPU cycles it is getting are clearly funneling into STAC. It is also helpful to review the AWGWAIT associated with CKPT to understand checkpoint access is healthy. From here we would begin focusing on whether there was a loop within STAC processing or whether someone was continuously driving STAC requests – possibly requiring separate traces and/or dumps.

PERFDATA - Throughput Analysis

PCENAME=**JQRP**,TIME=0.903077,CPU=0.700405,CPU%=3.13,

QSUSE_TIME=0.278596,**IOCOUNT=2196**,CKPT_COUNT=35317,

WAIT=WORK,INHIBIT=NO,MOD=HASPJQS,SEQ=70910000

COUNT=2061,AVGWAIT=0.329332,

POST=IO,COUNT=936,AVGWAIT=0.000606,

POST=$$POST,COUNT=1125,AVGWAIT=0.602831,

WAIT=CKPT,INHIBIT=NO,MOD=HASPJQS,SEQ=70923300

COUNT=3542,**AVGWAIT=0.095666**,

POST=RESOURCE,COUNT=1528,AVGWAIT=0.188931,

POST=IO,COUNT=947,AVGWAIT=0.000848,

IOCOUNT / 2 = number of jobs created during this interval

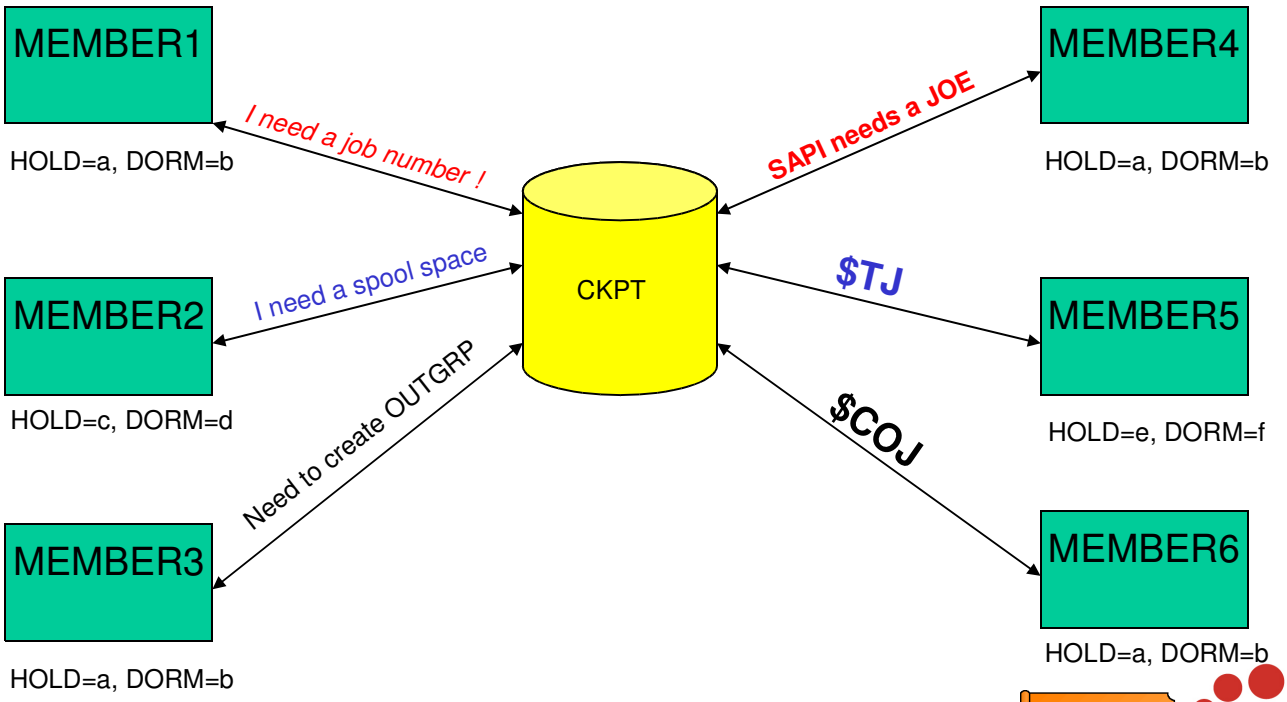
AVGWAIT = average time PCE waiting for CKPT

JQRP PCE shows the total I/O count for the interval. Since JES2 performs 2 I/Os for each job created, the IOCOUNT divided by 2 yields the count of how many jobs were created during the PERFDATA interval. In the above example 2196 / 2 = 1098. If the interval were 6 minutes, then that would suggest approx ~3 jobs were created per second.

The AVGWAIT time gives insight in any problems surrounding CKPT access. Generally, we view this health based on order-of-magnitude where anything larger than 0.10 seconds *could* indicate contention for checkpoint.

30 Second Checkpoint Review

Checkpoint access in a MAS looks like…



How equitably the checkpoint is shared amongst MAS members is controlled by the following MASDEF parameter (which have a scope of member):

HOLD= The minimum length of time a member will hold the checkpoint before it will try to release it

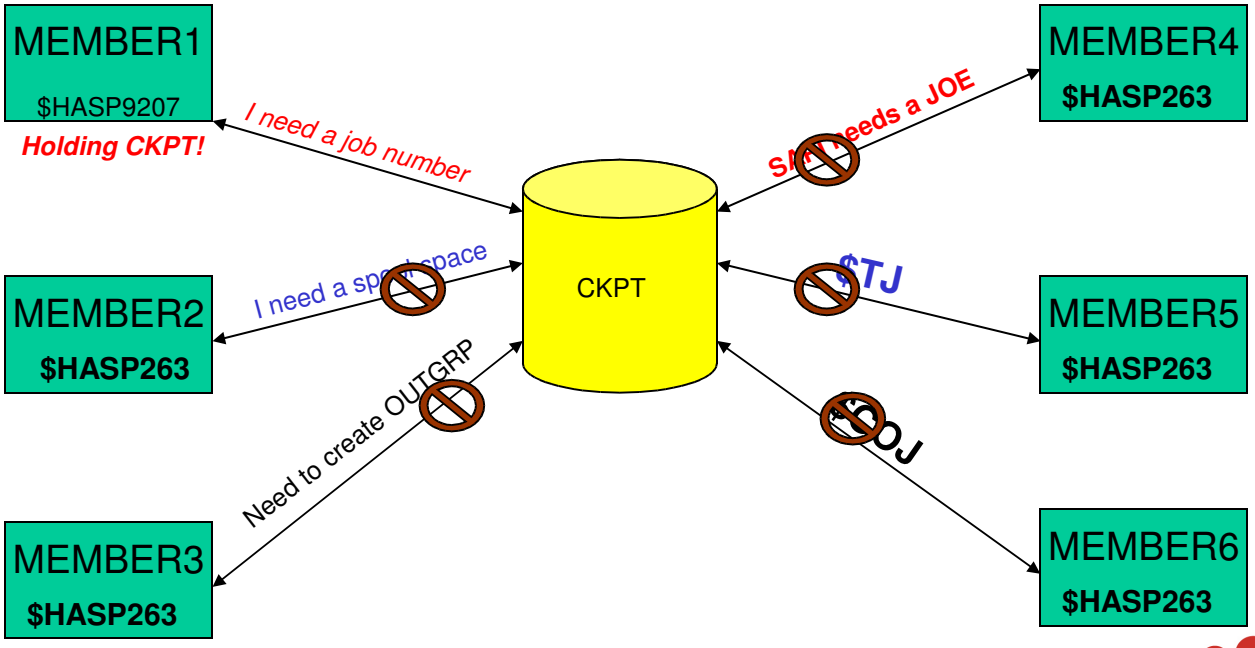
DORMANCY= The length of time a member will wait before attempting to reacquire the CKPT

Notification of a checkpoint lockout condition is based on the MASDEF parameter:

LOCKOUT=The length of time a member needing the CKPT will wait before issuing $HASP263

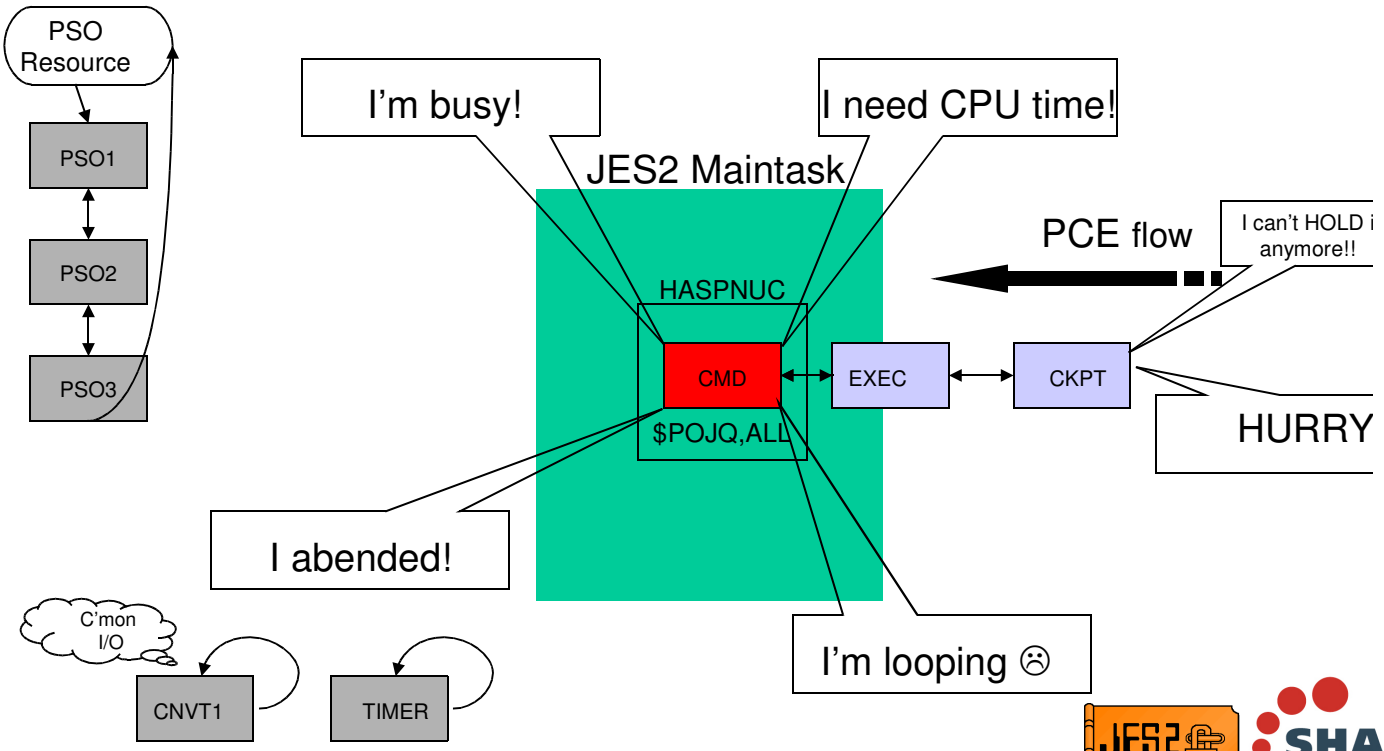
Another 30 Seconds about Checkpoint…

Checkpoint lockout in a MAS looks like…



Just 30 More Seconds about Checkpoint (don't lockout on me…)

Checkpoint lockout on the holding member could look like…



The first predicament that the CKPT PCE can find itself in, is being unable to get dispatched under the maintask. There are several reasons this could occur.

1. The PCE currently processing under maintask is busy doing valid work. It may be the nature of the work that is resulting in the excessive processing time. Any command that requires the scanning and/or filtering of a large number of jobs or outgrps can take some time to complete. The $POJQ with a filter command is an example if it needs to process tens of thousands of jobs. It is cpu intensive and could result in $HASP263's on other members depending on the coded LOCKOUT value.
2. JES2 is currently CPU restricted. That is the maintask TCB is not getting any or enough cycles to get through the chain of PCE's in a timely fashion to allow the CKPT PCE to run.
3. A PCE abended and has issued $HASP098 for a termination option. If this abend occured while the CKPT was held and the WTOR is not replied to in a timely fashion then the CKPT will not be released
4. A PCE is in a loop in which no $WAIT is issued so it will never give up control of the maintask TCB

The first two conditions can be transient in nature the $HASP263's will be issued but then stop as either the PCE completes its work or JES2 gets the needed CPU cycles. For second two, the $HASP263's will be issued until the causing condition is resolved.

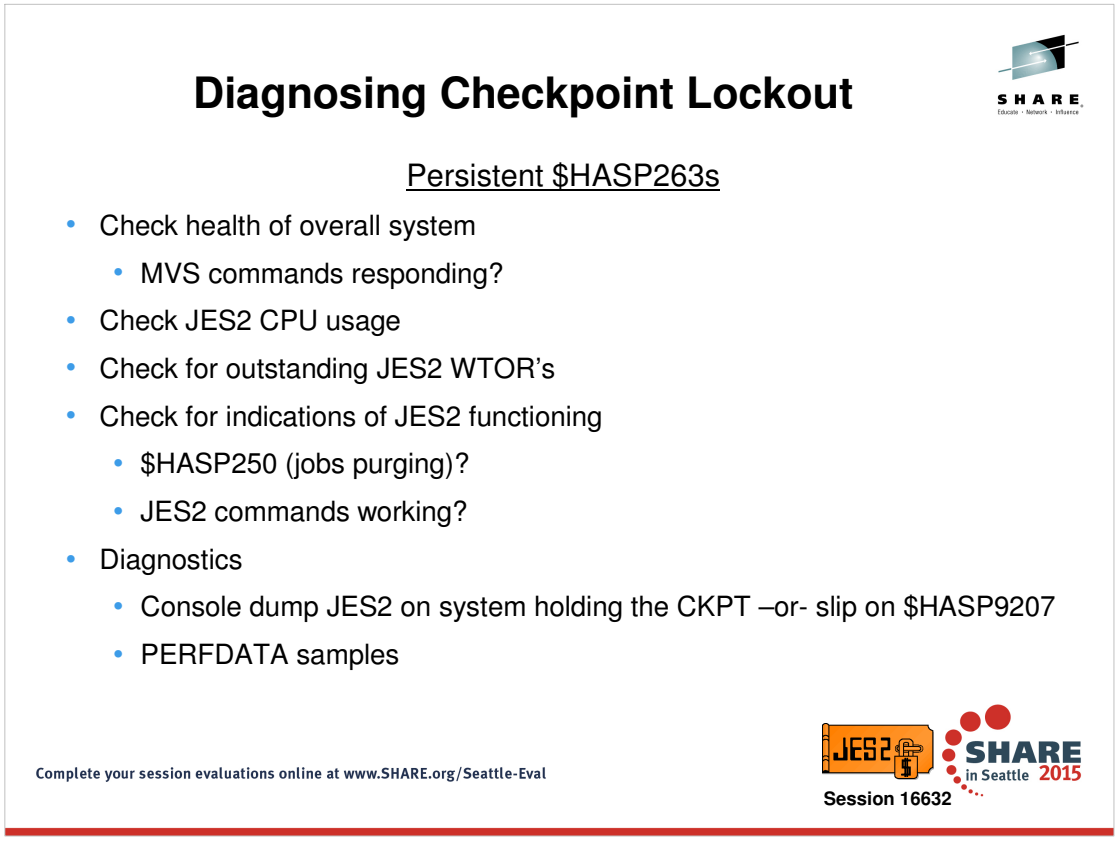
Diagnosing Checkpoint Lockout

- Diagnosis must occur on the system that is HOLDing the checkpoint
  - The system that is HOLDing the checkpoint….
    - Will not issue $HASP263
    - Will not issue IOS071I 016E,**,*MASTER*, START PENDING
    - Will issue $HASP9207 JES CHECKPOINT LOCK HELD DURATION xyz
      - Possibly other JES2 monitor $HASP92xx messages too
  - The system that is a victim not HOLDing the checkpoint…
    - Will issue $HASP263
      - LOCK HELD BY MEMBER abc (if CKPT on CF)

Once the $HASP263's have started the first step is determining which MAS member is holding the checkpoint.

The HASP263's and IOS071I's really indicate who is NOT holding it. Absence of these messages on a member would suggest that it is the one holding the CKPT. The $HASP9207 message issued by the JES2 monitor identifies the system holding the CKPT.

$HASP263 WAITING FOR ACCESS TO JES2 CHECKPOINT. LOCK HELD BY **SYSTEM** is a special case. The **SYSTEM** referred to in the message means that XES has indicated to JES2 that no member holds the lock but it is currently in the hands of XES. If this message persists and no JES2 member is showing signs of getting any access then a system with XCF/XES errors occurring is likely the problem. A Vary out of the plex of the system should force XES to release the lock.



The first step is to determine whether or not the problem is at the system or JES2 level. If MVS commands are not responding then JES2 is likely not releasing the CKPT due to problems outside of JES2 so terminating or taking other actions on the JES2 asid are likely not to resolve the lockout condition.

If MVS appears healthy then the focus can shift to the JES2 asid: Has JES2 abended? Are there an outstanding WTOR's for JES2? Is JES2 using a lot or any CPU? Resource monitors and SDSF can assist with this or a D A,JES2 followed by another D A,JES2 will show how much CPU was used between commands. If no, CPU is used then JES2 is simply not getting a chance to run so other higher priority tasks may need to be examined. Is JES2 responding to commands? Yes, then this would indicate that JES2 PCE's are running which is predicament number two.

If JES2 is not responsive to commands and there are no other messages being issued such as $HASP100, $HASP250 ($HASP395's do not count) and there is high CPU then JES2 is likely looping. Scanning the syslog looking for the last commands or messages issued by JES2 may give an indication of whether it is related to a CPU intensive command being issued.

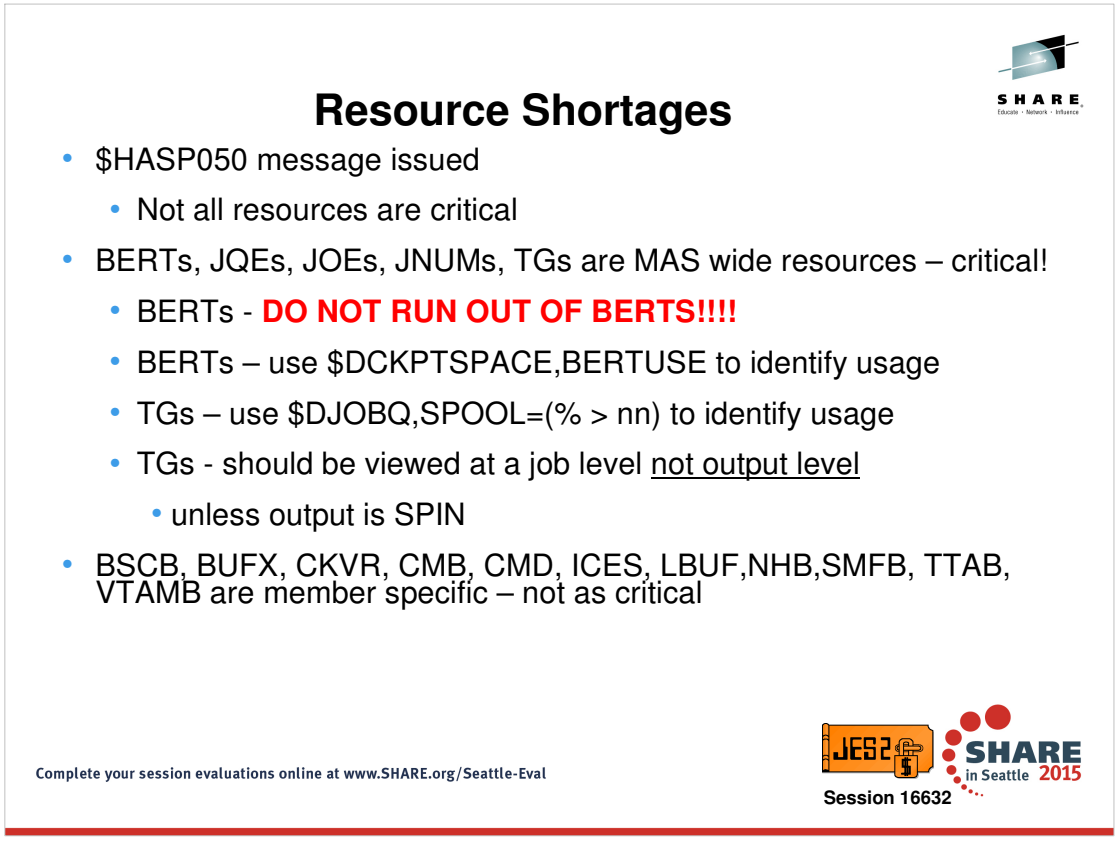
Diagnosing Checkpoint Lockout

Transient $HASP263s

- Messages appear on one or more systems but do not persist
- Normally caused by JES2 being temporarily busy with work/commands or short on CPU
- Diagnostics
  - Console dump JES2 on system holding the CKPT –or- slip on $HASP9207
  - PERFDATA samples

Session 16632

These transient $HASP263's come in two flavors: the first is "every once in while". The second is of a more "roaming" nature. The first type is usually the result of a temporary condition that either resulted in JES2 being busy or not being dispatched. The second is a little more troublesome. The messages appear consistently however the system identified as holding the lock changes and may cycle through all of the members of the MAS. This is much more difficult to isolate to any specific type of problem or system and could be require a small amount of tuning of HOLD/DORMANCY



JES2 will produce $HASP050 message indicating resource shortage – message will repeat until condition is relieved.

BERTs are one of the most critical JES2 resources. They represent non-contiguous pieces of storage on checkpoint that back/comprise other JES2 blocks such as $CAT, $DJB, $JQA, etc. There are some processes in JES2 that cannot wait for BERTs to become available, thus it is imperative to avoid complete exhaustion. For the BERT resource, in addition to $HASP050 you may also encounter:

$HASP051 EXTREME BERT SHORTAGE detected …

$HASP052 JES2 BERT resource shortage is critical -- IMMEDIATE action required…

Ideally you want to have enough BERTs defined such that you would exhaust any resource it is backing (such as JOEs, JQEs, $CATs) first rather than exhausting BERTs themselves. When in a BERT shortage condition, you want to identify and address any offending job/output –and/or- increase BERTNUM definition.

For TG shortages, you again want to identify and address any offending job/output –and-or- add additional spool space. It is important to approach TG usage at a job level because trackgroups are not restored to the TG map until the entire job is purged (exception being SPIN output, in which TGs restored when output is processed). Consider a job ADAM1 that has two pieces of non-SPIN output: JOE1 using 1 TG and the other JOE2 using 9K TGs. If you purge JOE2, it will not restore the 9K TGs because JOE1 still exists with 1 TG!

Spare spool volumes can be formatted in advance and then volume simply started $S if needed –or- can be formatted dynamically on the $SSPL command.

Resource Shortages

- $JDHISTORY command will show historical usage
  - Since JES2 warmstart/IPL
  - Hourly time slices of usage (interval at the top of each hour)
    - Limit/current/low/high/average usage
  - Same resources as $HASP050 message
- SDSF equivalent
  - RM panel
  - JH command on the MAS panel
- MSGID slip or console dump JES2 *if* needed

When approaching resource shortages one of the key pieces of information is whether the resource utilization is a sudden unexpected spike vs slow creep vs simply running at a fairly constant number too close to the warning limit etc. The above displays provide the answers, broken down in 1hr intervals. It also helps illustrate any relationships between resource trends - eg. Is job output growing at a 3x rate while jobnums are not growing at all? It may prompt the debugger to scrutinize a particular time interval in syslog/operlog– does it correspond with a peak or change in workload?

Questions?

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632

JES2 Service Information

yy/mm/dd	APAR	COMMENTS
14/06/10	OA45349	IMPROVEMENTS TO JES2 JOB SUBMISSION PROCESS
14/06/18	OA45436	$HASP875 RC02 USING HASIBLD
14/07/11	OA45560	ABEND0C4 IN HASPXEQ
14/07/11	OA45532	ABEND878 IN ANY ADDRESS SPACE DUE TO CSA/ESCA OR SQA SHORTAGE
14/07/11	OA45483	JES2 SSI RETURNS INCORRECT SRVCLASS STSCQNUM AND STSCQACT
14/08/12	OA45760	ABEND0C4 IN HASCPHAM AFTER FSS APPLICATION ENCOUNTERS ABEND
14/08/12	OA45713	$Q12 WHILE PROCESSING $DA COMMAND
14/08/13	OA45752	MISLEADING $HASP375 EXCESSION MESSAGE FOR TRANSACTION OUTPUT
14/09/04	OA45923	ABEND0C4-4 IN JES2 MONITOR ADDRESS SPACE DURING INITIALIZATION
14/09/04	OA45859	ABEND0C4 IN HASCDSAL WHEN SYSLOG BROWSE AND PURGE COINCIDE
14/09/04	OA45845	INCORRECT SCHENV ASSIGNED
14/09/26	OA46115	SERIALIZATION ERROR WHEN MULTIPLE TASKS CREATING SPIN OUTPUT
14/10/01	OA46125	SECURITY VIOLATION FOR JOBS WITH INSTREAM DATA SET
14/10/06	OA46199	JOB CARD WITH INVALID NOTIFY=&SYSUID DOES NOT FAIL

LEGEND:

HiPer APARs (Hi Impact, or Pervasive) Security/Integrity

PE APAR

JES2 Service Information

yy/mm/dd	APAR	COMMENTS
14/10/06	OA46200	SAPI APPLICATION CAN WAIT IN HASCRQUE
14/10/20	OA46266	ENHANCEMENT TO $DOGCAT AND $CAT CACHING
14/11/03	OA46342	IN MIXED MAS, JOB CARD SCHENV CLEARED IN EXIT6 (or 60) ACTIVE
14/11/10	OA46455	INACCURATE SOCKET DATA FOR LINE VIA SSI83
14/11/17	OA46554	COMPACTION TABLE NOT GENERATED IF NO RJE DEFINED
14/11/17	OA46503	SYSOUT TRANSMITTED FROM JES2 TO JES3 ARRIVES WITH JOB PRIORITY 0
14/11/17	OA46510	POINT MACRO MAY POSITION TO INCORRECT RECORD ADDRESS (RBA)
14/11/21	OA46596	$SJnnn FAILS WITH $HASP003 RC=109 RSN=15
14/11/26	OA46621	INCONSISTENT SJB FLAG SETTINGS FOR EXIT 58
14/12/15	OA46706	$ADD SRVCLASS FAILS IF SERVICE CLASS CONTAINS AN UNDERSCORE

LEGEND:
HiPer APARs (Hi Impact, or Pervasive) Security/Integrity
PE APAR

OA45349 – IMPROVEMENTS TO JES2 JOB SUBMISSION PROCESS

Problem

Steady influx of job submission on members that do have a large slice of checkpoint (HOLD and DORM)

JES2 not dispatching enough JQRP PCEs; potentially leading to a slowdown/bottleneck in job submission

Algorithm improved to dispatch more JQRP PCEs

Bypass

None

Fix

JES2 z/OS 1.12, 1.13, and 2.1

Session 16632

OA45436 – $HASP875 RC02 WHEN USING HASIBLD

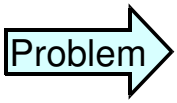
Problem HASIBLD, the sample JCL to build production/sample JES2 product and object load libraries does not correctly process a handful of new modules in z/OS JES2 2.1

Attempting to start a secondary subsystem built with HASIBLD will fail

Bypass None

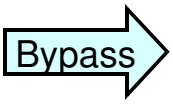
Fix JES2 z/OS 2.1

OA45560 – ABEND0C4 IN HASPXEQ



If JES2 Monitor address space fails to start, then at JES2 termination time ($PJES2) an abend0C4 can occur

Minimal impact since JES2 is already terminating



Resolve error that is preventing JES2 Monitor address space from starting

JES2 z/OS 1.13, and 2.1

OA45532 – ABEND878 IN ANY ADDRESS SPACE DUE TO CSA/ECSA OR SQA SHORTAGE

Problem JES2 produces ENF records for job (and job output) notification. Storage for parameter list is obtained in subpool 241 key 1 and expected to be freed by ENF.

A high rate of ENFs causes ENF buffer area to fill up, then ENF will not be able to receive the ENF (and free the area). Moreover, JES2 failed to detect this full condition and resent ENF records; thereby, exhausting storage over time.

Bypass None

Fix JES2 z/OS 1.12,1.13, and 2.1

OA45483 – JES2 SSI RETURNS INCORRECT SRVCLASS STSCQNUM AND STSCQACT

Problem JES2 SSI 80 returns incorrect SRCLASS statistics; specifically total number of jobs in queue and number of active jobs in queue.

Reflected in incorrect (higher than expected) totals in SDSF “Job Information” popup on the Input display panel

Bypass None

Fix JES2 z/OS 1.12, and 1.13

OA45760 – ABEND0C4 IN HASCPHAM AFTER FSS APPLICATION ENCOUNTERS ABEND

Problem Memory for FSS printer (in FSS address space) is obtained by the subtask driving printer. The FSS subtask encounters an abend and frees the storage; thereafter, the mother task attempts to free the same storage as part of FSA DISCONNECT and encounters abend0C4

FSS application cannot be restarted until all abends are processed

Bypass None

Fix JES2 z/OS 1.13, and 2.1

OA45713 – $Q12 WHILE PROCESSING $DA COMMAND

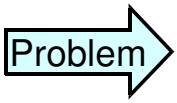
Problem In a MAS environment, one member begins processing $DA command. If a $WAIT occurs while JES2 is processing job queue elements, then it is possible when command processing resumes the element in hand will be invalid – resulting in $Q12

JES2 terminates and is hotstart'able

Bypass None

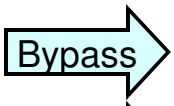
Fix JES2 z/OS 2.1

OA45752 – MISLEADING $HASP375 EXCESSION MESSAGE FOR TRANSACTION OUTPUT



For long running transaction servers, JES2 calculates whether (lines/kbytes/etc) excession occurs based on the limit set for the server (BPXAS init running the task).

However, the $HASP375 message identifies the transaction output being processed by the server – which may have not have the same name as the server.



None

JES2 z/OS 1.12, 1.13, and 2.1

OA45923 – ABEND0C4-4 IN JES2 MONITOR ADDRESS SPACE DURING INITIALIZATION

Problem At start time, JES2 Monitor address space attempting to reference storage that is not yet initialized by JES2 address space. This results in protection exception abend in Monitor address space

Bypass None - JES2 Monitor will restart following abend

Fix JES2 z/OS 1.12, 1.13, and 2.1

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632

OA45859 – ABEND0C4 IN HASCDSAL WHEN SYSLOG BROWSE AND PURGE COINCIDE

Problem Timing issue can cause abend0C4 if the data set for an active SYSLOG is purged at the same time SYSLOG browse is being entered (such as SDSF LOG panel)

Bypass Avoid overlap of SYSLOG browse and purge of active SYSLOG data set

Fix JES2 z/OS 1.13, and 2.1

Session 16632

OA45845 – INCORRECT SCHENV ASSIGNED

Problem If no SCHENV is coded on the JOB card, then the SCHENV associated with the *CLASS specified on the INTRDR* will be assigned to the job.

Ignores the SCHENV associated with the job CLASS on the JOB card.

Bypass Explicitly code SCHENV on the JOB card

Fix JES2 z/OS 1.13, and 2.1

OA46115 – SERIALIZATION ERROR WHEN MULTIPLE TASKS CREATING SPIN OUTPUT

Problem Timing issue if multiple subtasks within same ASID attempt spin output processing (SPIN=UNALLOC or SEGMENT=. One (or more) will be deferred since only one subtask can own $SJB and $SDB locks at a time.

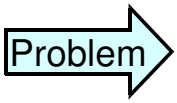
Serialization error exists managing those locks that can cause one (or more) subtasks to hang awaiting spin processing or abend0F7-7C.

Bypass Avoid multiple subtasks with same ASID creating spin SYSOUT at same time. If hang occurs, recycling the task that owns $SDB lock will clear the condition

Fix JES2 z/OS 1.13, and 2.1

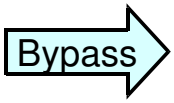
Session 16632

OA46125 – SECURITY VIOLATION FOR JOBS WITH INSTREAM DATA SET



For jobs that contain instream data sets (SYSIN), JES2 does not build the correct resource name for security check. Specifically, the node name qualifier is missing from the resource entity name.

Can result in security failure for job



None

JES2 z/OS 2.1

OA46199 – JOB CARD WITH INVALID NOTIFY=&SYSUID DOES NOT FAIL

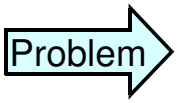
Problem If job card NOTIFY=&SYSUID refers to a *special or reserved* value based on DESTDEF setting (such as DESTDEF UDEST=SPLOCAL or NDEST=NODE), then the job should fail with:

$HASP100 nnnnnnn – Illegal Job card – value of NOTIFY= parameter is not valid

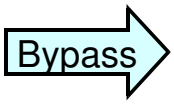
Bypass Correct DESTDEF settings or use non-reserved /non-special userids for NOTIFY

Fix JES2 z/OS 1.13, and 2.1

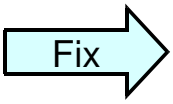
OA46200 – SAPI APPLICATION CAN WAIT IN HASCRQUE



If only one SAPI application running and that member is dominating the holding of JES2 checkpoint, it is possible the SAPI application may encounter a delay/wait in HASCRQUE

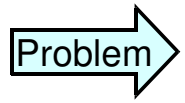


Issue a command to restart SAPI application work such as $TASAPI,I=300,'$S'



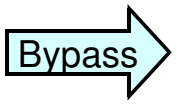
JES2 z/OS 1.12, 1.13, and 2.1

OA46266 – ENHANCEMENT TO $DOGCAT AND $CAT CACHING

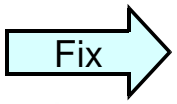


Logic added to caching algorithms to re-establish an appropriate resume point in cases where processing of operator commands is interrupted by a $WAIT

External symptom is truncated command output:
$HASP611 LIST INCOMPLETE



None



JES2 z/OS 2.1

OA46342 – IN MIXED MAS, JOB CARD SCHENV CLEARED IF EXIT6 (or 60) ACTIVE

Problem Job with JOB card SCHENV= is submitted on 1.13 member, but it converts on 2.1 member. If exit6 (or exit60) active and the exit does not alter SCHENV, then the original JOB card SCHENV is cleared.

Job may attempt execution on system not defined with the SCHENV of the JOB card

Bypass Explicitly update SCHENV in exit6 via X006SCHE (or exit60 via X060SCHE)

Fix JES2 z/OS 2.1

Session 16632

OA46455 – INACCURATE SOCKET DATA FOR LINE VIA SSI83

Problem More than one SOCKET defined to same NODE and one of those SOCKETs has a dedicated line which is inactive. SSI83 can return the incorrect (inactive) information instead of the active SOCKET information.

SDSF uses SSI83 for its LINE display panel

Bypass None

Fix JES2 z/OS 1.13, and 2.1

OA46554 – COMPACTION TABLE NOT GENERATED IF NO RJE DEFINED

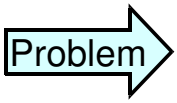
Problem COMPACT statements coded in JES2 Init deck, but no RJE devices defined (or TPDEF RMTNUM=0). The compaction table is not generated; thus, cannot be used for any NJE sessions.

Bypass Set TPDEF RMTNUM to a non-zero value

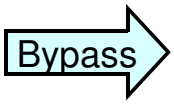
Fix JES2 z/OS 1.13, and 2.1

Session 16632

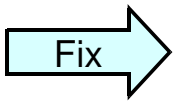
OA46503 – SYSOUT TRANSMITTED FROM JES2 TO JES3 ARRIVES WITH JOB PRIORITY 0



When spin sysout is sent via NJE (store and forward) from JES2 to JES3, the original priority is lost. Priority 0 may be defined as HOLD on JES3, thereby preventing processing of the sysout.



Manually set job priority on the receiving node



JES2 z/OS 1.13, and 2.1

OA46510 – POINT MACRO MAY POSITION TO INCORECT RECORD ADDRESS (RBA)

Problem Difference in RBA assignment for locally produced output versus output received from another node.

Record counting for NJE'd output may be off (usually by one) as compared to local output, causing unexpected POINT results

Bypass None

Fix JES2 z/OS 1.13, and 2.1

APAR marked PE and fixed by OA46919

OA46596 – $SJnnn FAILS WITH $HASP003 RC=109 RSN=15

Problem In a JES2 MAS, Job nnn is submitted with affinity to only member A. $SJnnn will succeed on member A. $SJnnn may fail on member B with:

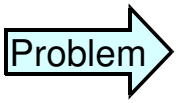
$HASP003 RC109 RSN15 NO JES2 CAN SELECT

It should succeed and start the job on member A. Timing related based on whether JES2 has registered with WLM

Bypass Issue $SJnnn on the member where job is eligible

Fix JES2 z/OS 1.13, and 2.1

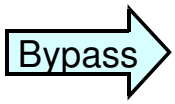
OA46621 – INCONSISTENT SJB FLAG SETTINGS FOR EXIT 58



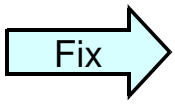
$Ejnnn,STEP issued to evict job which sets SJBEVICT. Exit58 explicitly indicates job should *not* be evicted. The exit is honored on first pass and job not evicted.

However, next invocation of exit58 for the same job will be represented with the SJBEVICT setting

No impact unless your exits are dependent upon the SJB evict flag settings

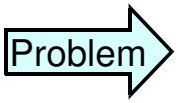


None



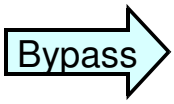
JES2 z/OS 1.13, and 2.1

OA46706 – $ADD SRVCLASS FAILS IF SERVICE CLASS CONTAINS AN UNDERSCORE

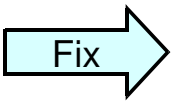


WLM has support for underscore character (_) in service class name, so JES2 should also accept it rather than fail with:

$HASP003 RC=04 ADD SRVCLASS(yyy_yyy) – CONTAINS AN INVALID SUBSCRIPT



None



JES2 z/OS 1.13, and 2.1

Questions?

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Session 16632