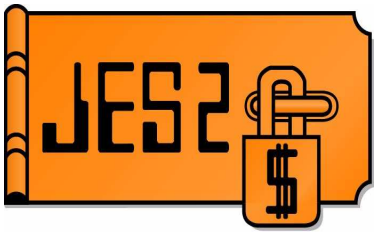


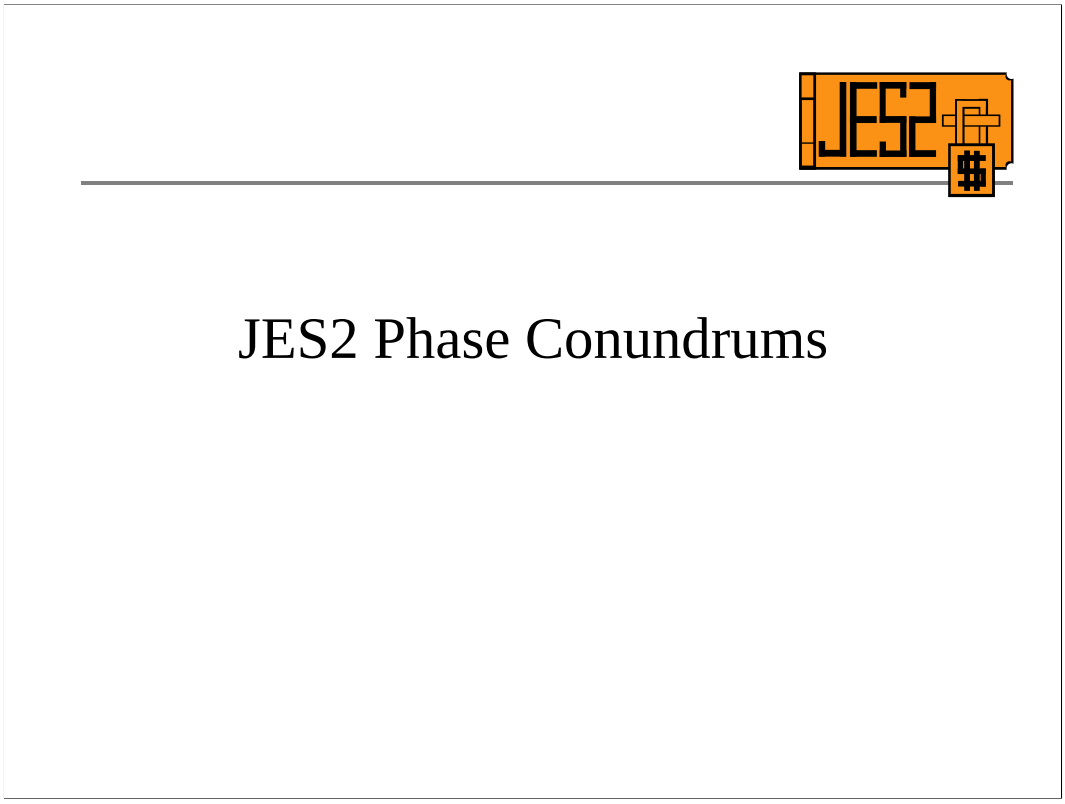
JES2 Project
Sessions 2663
Lorne Parks
lparks@ca.ibm.com

JES2 Level2 Bit Bucket

SHARE 104, Winter 2005
Anaheim, CA

Topics I will discuss

- JES2 Phase Conundrums
- MAS Surprises
- The Joy of Spooling
- Shutting down JES2
- Diagnostic Helpers
- Quickies



Phase by Phase

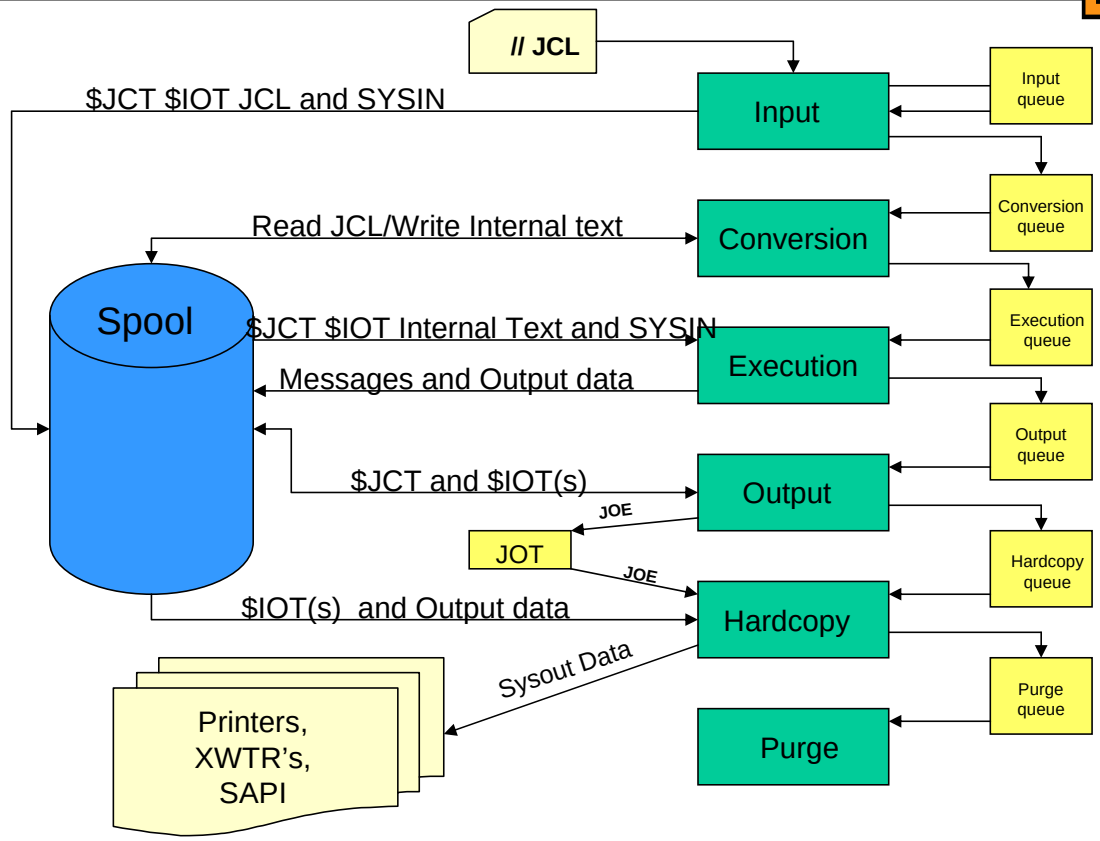
-brief overview of each phase

-what are the common issues phase by phase

-what are possible solutions

-what doc to collect if it cannot be resolved

JES2 Job Phases



Intro into JES2 processing

Input/Conversion – JCL prep work to get it ready for a MVS asid

Execution – getting it to the right asid and running

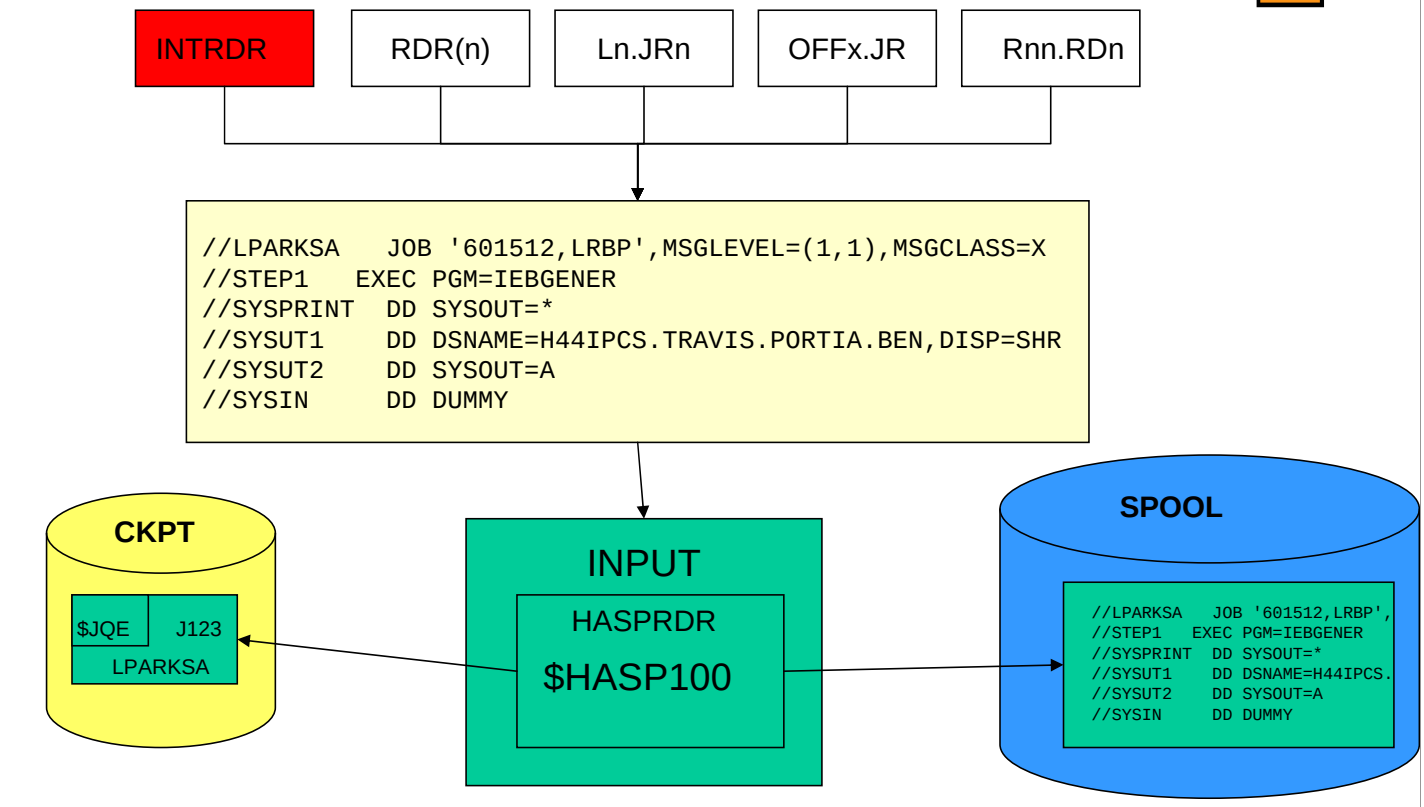
Output/Hardcopy/Purge – dealing with the sysout leftovers

INPUT

JCL: IN then PUT

back out

INPUT: Basic Function



Taking JCL from a variety of sources:

Card reader

Internal Reader

NJE job receiver

Spool offload job receiver

RJE reader

Examines and processes the JOB card and any JECL (/*).

Creates the basic JES2 job control block structure ($JQE, $JCT, $IOT's).

Spools the JCL images for the next phase.

HASPRDR is the JES2 module that controls this function for INPUT.

INTRDR's: Don't get caught short

- INTRDR RDINUM= defines number available
- JOB's, STC's and TSU's allocate them to use for JCL submission
- TSO Submit only allocates the INTRDR for the duration of the job submission
- CICS, IMS, JOB Schedulers allocate them for the life of their STC's
- An increase in the number of these "permanent allocators" can result in INTRDR shortages

Symptoms

- IKJ56241I SYSOUT DATASET NOT ALLOCATED - UNIT NOT AVAILABLE on a TSO Submit command or Dynalloc
- $HASP304 jobname WAITING FOR AN INTERNAL READER for a JOB with SYSOUT=(,INTRDR) coded

What now?

- $DRDI(*) to the rescue

```
$HASP603 RDI(INTRDR) 750
$HASP603 RDI(INTRDR)    STATUS=ALLOCATED,AUTH=(DEVICE=NO,JOB=NO,
$HASP603                SYSTEM=NO),CLASS=A,HOLD=NO,MSGCLASS=A,
$HASP603                OWNER=(STC00355,OPC2),PRTDEST=LOCAL,
$HASP603                PUNDEST=LOCAL,SYSAFF=(A090),TRACE=NO
$HASP603 RDI(INTRDR) 751
$HASP603 RDI(INTRDR)    STATUS=ALLOCATED,AUTH=(DEVICE=NO,JOB=NO,
$HASP603                SYSTEM=NO),CLASS= ,HOLD=NO,MSGCLASS=A,
$HASP603                OWNER=(JOB14443,CICSCICS),PRTDEST=LOCAL,
$HASP603                PUNDEST=LOCAL,SYSAFF=(A090),TRACE=NO
$HASP603 RDI(INTRDR) 752
$HASP603 RDI(INTRDR)    STATUS=ALLOCATED,AUTH=(DEVICE=NO,JOB=NO,
$HASP603                SYSTEM=NO),CLASS=A,HOLD=NO,MSGCLASS=A,
$HASP603                OWNER=(JOB14443,CICSCICS),PRTDEST=LOCAL,
$HASP603                PUNDEST=LOCAL,SYSAFF=(A090),TRACE=NO
```

OWNER field identifies the job that has allocated an INTRDR.

Examine command output for similar types of jobs that are using the INTRDR's.

What Now? continued

- RDINUM can only be increased on a single member warmstart
- Using OWNER information, INTRDR owning jobs will have to be stopped to free up INTRDR's
- Addition of CICS regions is almost always the cause of the sudden increase in INTRDR usage

When this issue is seen by Level 2 it has more often than not been a change in the number of CICS regions that has resulted in the shortage.

Conversion

(How should this be interpreted?)

Conversion: Basic Function

```
//LPARKSA   JOB '601512,LRBP', MSGLEVEL=(1,1),MSGCLASS=X
//STEP1   EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSNAME=H44IPCS.TRAVIS.PORTIA.BEN,DISP=SHR
//SYSUT2    DD SYSOUT=A
//SYSIN     DD DUMMY
```

HASPCNVT

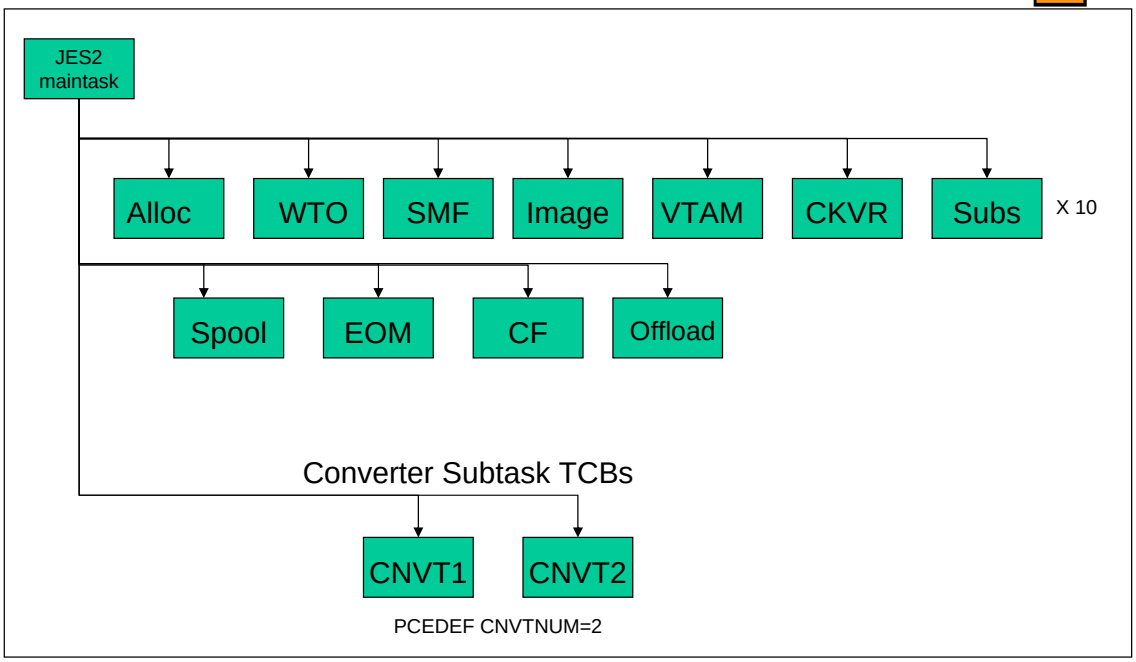
HASPCNVS

IEFVH1

```
 LPARKSA 601512,LRBP   1 1   Xs        00   0001M
40400B00DDCDDEC0FFFFFF6DDCDB00F0FB00EA000000B00FFA00FFFFDF00000000
310614273719221B601512B392732111121172140002012O0E1500014E21040001
   m  STEP1   IEBGENERl  0030 00
200900EECDF800CCCCCDCD900FFFF0FFF00000000
A2041523571A189527555932400302O0E21040002
.  >  SYSPRINT.  *¢  IBMUSER.LPARKSA.JOB00012.D0000101.?
402600EEEDDCDE4005402CCDEECD4DDCDDEC4DDCFFFFF4CFFFFFFF46444444444F00000000
B40E1828279953B11CA1C9244259B3719221B16200012B40000101BF000000000E21040003
   >  SYSUT1¢  H44IPCS.TRAVIS.PORTIA.BEN   SHR
300600EEEEEF401CFFCDCE4EDCECE4DDDECC4CCD400ECDF00000000
840E16282431A198449732B391592B769391B255613289E21040004
   >  SYSUT2.  A¢  IBMUSER.LPARKSA.JOB00012.D0000102.?
402600EEEEEF400C402CCDEECD4DDCDDEC4DDCFFFFF4CFFFFFFF46444444444F00000000
940E16282432B111A1C9244259B3719221B16200012B40000102BF000000000E21040005
   >  SYSIN
110600EEECD30F00000000
741E1528295E0E21040006
```

To take the JCL images spooled during INPUT and convert them into internal text to be used by the interpreter. HASPCNVT and HASPCNVS control most of this process. HASPCNVS loads and links to the MVS JCL converter loadmod IEFVH1. It is this MVS code that actually generates the internal text. JES2 spools it so that when the job is initiated it can be passed into the MVS Interpreter.

JES2 Subtasks



JES2 creates separate subtask TCB's to invoke services that may result in an MVS wait. JES2 maintask does not like to MVS wait. There are 12 different JES2 subtask types of which one is for conversion.

The number created is the same as the CVNTNUM on the PCEDEF statement. 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)

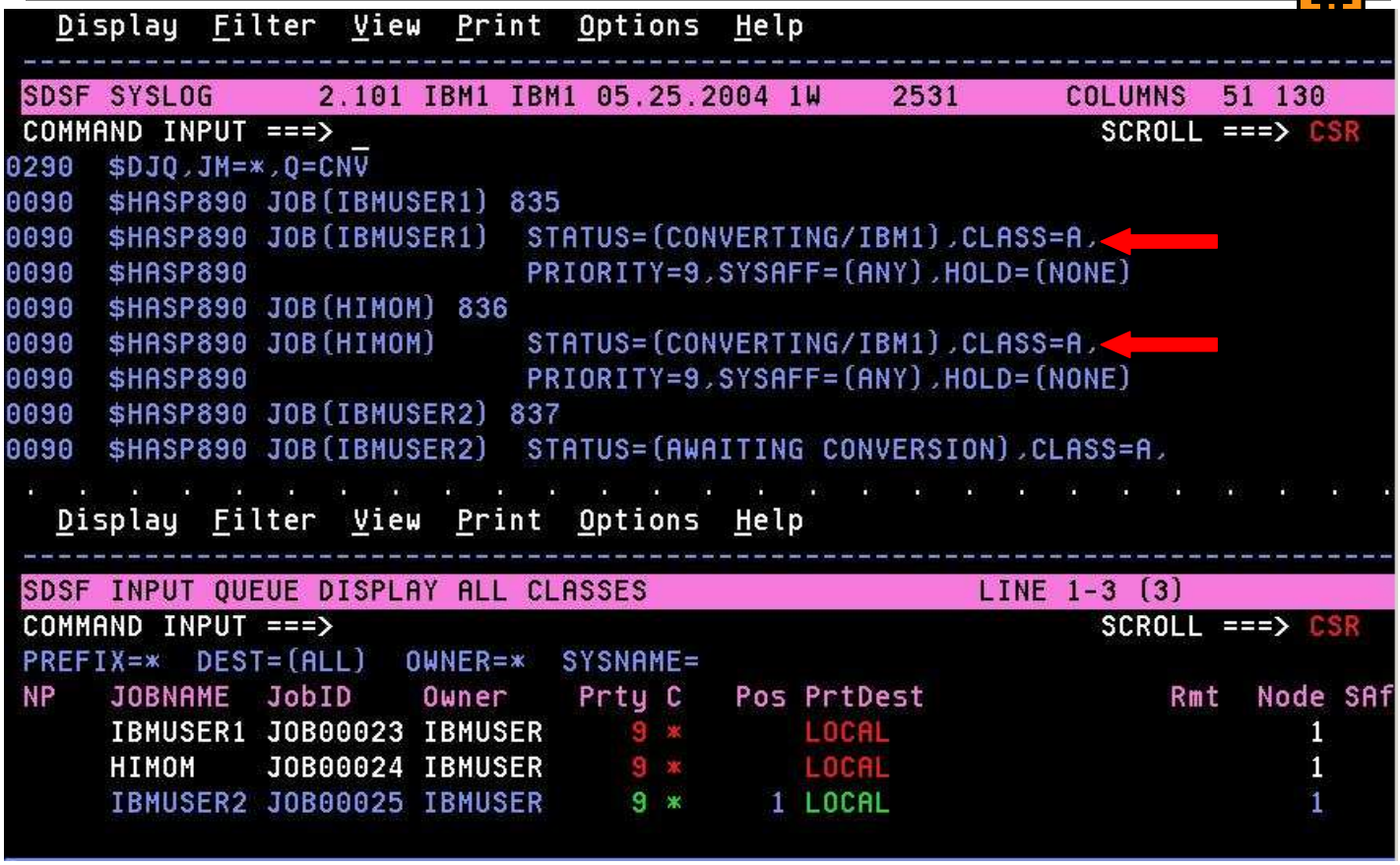
SPOOL – handles spool volume allocations etc

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

Converters not converting!



$DJQ,JM=*,Q=CNV shows jobs on the conversion queue and awaiting conversion.

SDSF Input panel shows jobs with Class of *.

These jobs appear stuck as newly submitted jobs queue up behind those currently in conversion.

Converter PCE Display

```
 $DPCE(CNVT),DETAILS
        $HASP653 PCE(CNVT)
$HASP653 PCE(CNVT)      NAME=CNVT,WAIT=CCAN,XECB,MOD=HASPCNVT,
$HASP653                SEQ=05961600,TIME=2004.147,10:07:57,
$HASP653                CURJOB=JOB00028,ACTIVE=1,I/O=0,
$HASP653                ADDR=06B6C550,
$HASP653                NAME=CNVT,WAIT=CCAN,XECB,MOD=HASPCNVT,
$HASP653                SEQ=05961600,TIME=2004.147,10:07:42,
$HASP653                CURJOB=JOB00027,ACTIVE=1,I/O=0,
$HASP653                ADDR=06B6C700
```

When the converter subtasks are hung, a display of the converter PCE's shows the WAIT for subtask post (XECB) or Cancel (CCAN).

The CURJOB shows the jobs that are actively converting.

D GRS,C

SYSZTIOT contention within the JES2 asid

Contention for the SYSZTIOT resource in the JES2 asid will stop conversion.

A SYSZTIOT exclusive ENQ is needed when the converter subtask processes a JCLLIB card.

A SYSZTIOT shared ENQ is needed if a PROCLIB (PROCxx) needs to be opened.

The number of TCB's listed in the ISG343I message can equal or exceed the number of converter PCE's defined.

What now?

The usual suspect: $SOFFLOAD

```
          $soffload1,type=transmit
          $HASP000 OK
          IEF244I JES2 JES2 - UNABLE TO ALLOCATE 1 UNIT(S)
        AT LEAST 1 OFFLINE UNIT(S) NEEDED.
          IEF877E JES2 NEEDS 1 UNIT(S)
FOR JES2 SYS00008
FOR VOLUME LORNE1
OFFLINE
101E
:
OFFLINE, NOT ACCESSIBLE
00C0-00C5 02E1 02E3 02E5 02E9 02ED 02EF-02F0 02F3 02F7-02F8 02FD-02FE
0340-034F 0370-037F 0984 0987 098B 098D 098F 0181-0182 018B-018C 018E
0330-0336 033B-033D 0350-035F 048E 1000-1004 100C-100E 1010 1013-101A
101F-108F B100-B1FF F180-F18F 0480-048D 048F
:
          IEF878I END OF IEF877E FOR JES2 JES2 SYS00008
         *09 IEF238D JES2 - REPLY DEVICE NAME OR 'CANCEL'.
```

And the outstanding MOUNT message

Old apar OY10335 describes this issue.

SYSZTIOT is held while the MOUNT is outstanding thus locking out converters

Matters can be made worse if a $SPRT is issued for a MODE=JES printer. JES2 will attempt to allocate the printer device and hang up behind the SYSZTIOT as well. This hangs the Command processor and no further JES2 commands will be processed until the mount is satisfied.

What Next?

- D GRS,C shows SYSZTIOT owning TCB
- $DJQ,JM=*,Q=CNV shows CONVERTING jobs
- No external way to link the two
- Option One: $C jobs that are CONVERTING. The abend422 that results may free up the ENQ
- Option Two: Hotstart JES2 ($PJES2,ABEND)
- Option Three: Dump the JES2 asid and call IBM

The GRS display shows which TCB is holding the ENQ and it is most likely one of the converter subtask TCBs if no mount message outstanding.

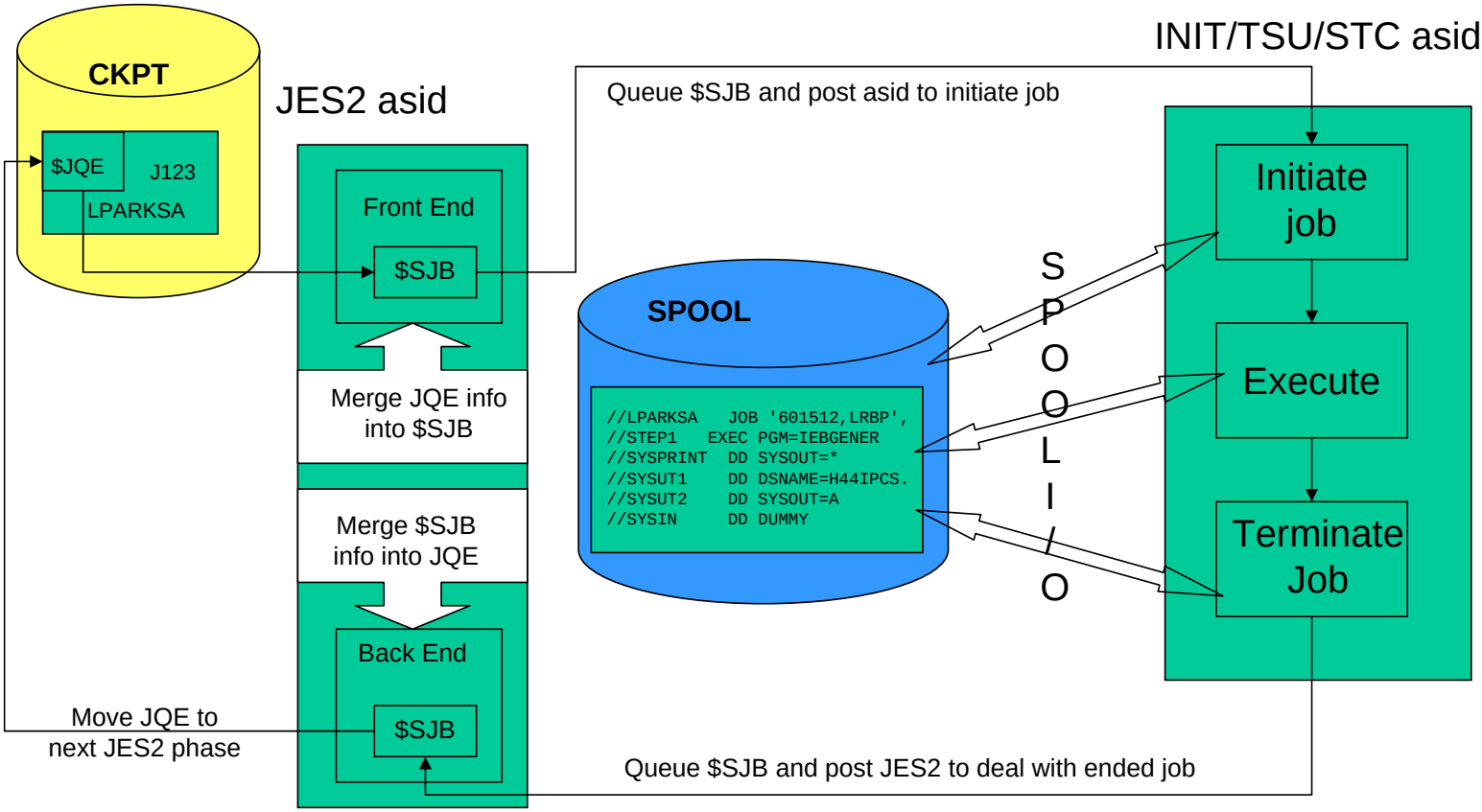
The $DJQ command and SDSF Input/ST panel shows what jobs are actively converting. Aside from dumping JES2 and looking at control blocks (Get owning TCB from display. Format Converter Subtask $DTE's look at DTETCB for match then take DTEPCE and match it to the address on the $DPCE(CNVT),DETAILS command and then cancel that specific job) one option is to cancel via $C the jobs that are converting as opposed to awaiting conversion. This command generates an abend422 against the subtask. This asynchronous abend may free up the enqueue. $PJES2,ABEND will definitely free up the enqueue. In both cases though this enqueue is usually a sign of an abnormal condition and documentation should be gathered.

Most recently reported problem was fixed by DFSMS ptf UA09832 pe'd and fixed by OA07438. This was a PDSE latch issue.

Execution

To MVS and Back Again

Execution: Basic Function



Execution has a front end and back end.

The front end deals with getting the converted JCL into the appropriate MVS asid. For STC's and TSU's the target asid was created via the START and LOGON commands. For JOBs, the target asid is an initiator either JES2 or WLM managed. In each case the JES2 asid takes information from the checkpoint and fills in the $SJB (resides in ECSA). The $SJB is the major anchor CB for jobs that are executing. The $SJB is placed on an execution queue and the target asid is posted.

The back end processing handles the JOB/STC/TSU as it leaves the MVS asid and goes back into JES2 control. The job termination code in the job's asid completes termination of the job, queues the $SJB to the termination queue and posts the JES2 execution phase processor to handle the terminated job. The job is then queued to the next appropriate JES2 phase.

Initiate Job Indications

- $HASP373 JOB STARTED
- IEF403I JOB - STARTED
- D A command recognizes the job
  - CSCB filled in

Terminate Job Indications

- IEF404I JOB – ENDED
- $HASP395 JOB ENDED
- D A command no longer recognizes the job

The CSCB is only invalidated once JES2 job termination code running in the INIT asid as given the job back to execution processing in the JES2 asid. It is important to note that these messages are issued from the asid that the job is executing in. They are not issued from the JES2 asid, so they are indicatiors of the job initiation/termination process outside of the JES2 asid.

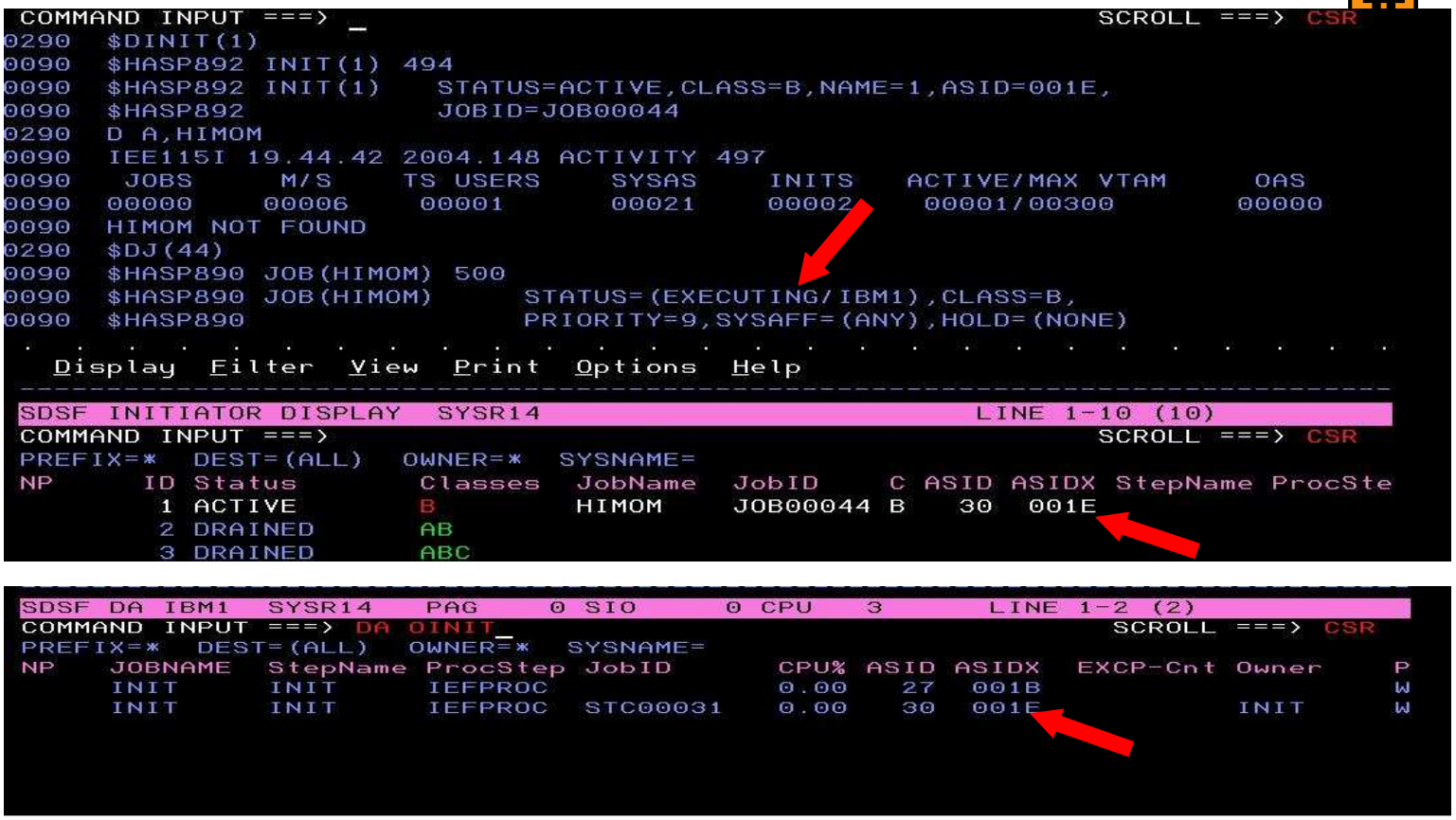
Execution: Mixed Messages


```
D A,HIMOM
IEE115I 15.43.56 2004.148 ACTIVITY 445
 JOBS    M/S   TS USERS   SYSAS   INITS  ACTIVE/MAX VTAM    OAS
00000   00006   00001     00021   00002   00001/00300      00000
HIMOM NOT FOUND

$DJ(43),LONG
$HASP890 JOB(HIMOM) 447
$HASP890 JOB(HIMOM)    STATUS=(EXECUTING/IBM1),CLASS=A,
$HASP890               PRIORITY=9,SYSAFF=(ANY),HOLD=(NONE),
$HASP890               CMDAUTH=(LOCAL),OFFS=(),SECLABEL=,
$HASP890               USERID=IBMUSER,SPOOL=(VOLUMES=(SPOOL1),
$HASP890               TGS=2,PERCENT=0.3809),ARM_ELEMENT=NO,
$HASP890               CARDS=11,REBUILD=NO,SRVCLASS=DISCRETN,
$HASP890               SCHENV=,SCHENV_AFF=(ANY),CC=(),NOTIFY=
```

JES2 indicates that the job is executing however the MVS D A command does not find the job executing. The MVS D A command uses the CSCB associated with an asid to locate the job number. The CSCB is filled in almost immediately after the posted ASID begins to initiate the job.

As seen by SDSF



SDSF INIT panel looks at the JES2 initiator control blocks to format out its information. This is the same control blocks that the $DI command uses.Both are useful since they provide the asid where the job is supposed to still occupy which can be handy when gathering documentation.

The DA OINIT will display INITs both JES2 and WLM managed that are currently idle. The WLM INIT does not have a JobID associated with it. When an INIT is processing a job then it will not be displayed by the DA OINIT panel.

Possible Reasons

- No $HASP373 JOB STARTED issued.
  - Init address space likely not getting CPU
- $HASP395 JOB ENDED issued.
  - OA07712 JES2 EOM hung in z/OS 1.4
  - INIT asid hung in End of Memory processing (prior to z4)
  - JES2 not processing the $SJB for ended job
  - HOT: PE APAR OA10668

The MVS D A command uses the CSCB to locate the jobname for an address space. The JES2 asid assigns to a job to an initiator and posts the initiators address space. When the initiator runs after this post the fills in the JSAB with the assigned job name immediately. The $HASP395 is issued shortly thereafter. Once the CSCB is updated the D A command will show the job on the command display. The lack of the job started message with JES2 $DJ, $DINIT (SDSF ST and INIT panels too) showing the job as executing likely means that the INIT asid is not getting CPU cycles to wake up from the POST made by JES2.

If the job ended message was issued then the problem is that the JES2 asid has not processed the $SJB associated with the terminating job. If the initiator abended and goes through EOM then there is the potential for a hang in EOM processing prior to z4. There is still a possibility that JES2's EOM code could abend and the SJB remains unprocessed. If there are no abends then the JES2 asid is simply not processing the $SJB for some reason and further diagnosis will be needed.

What to do?

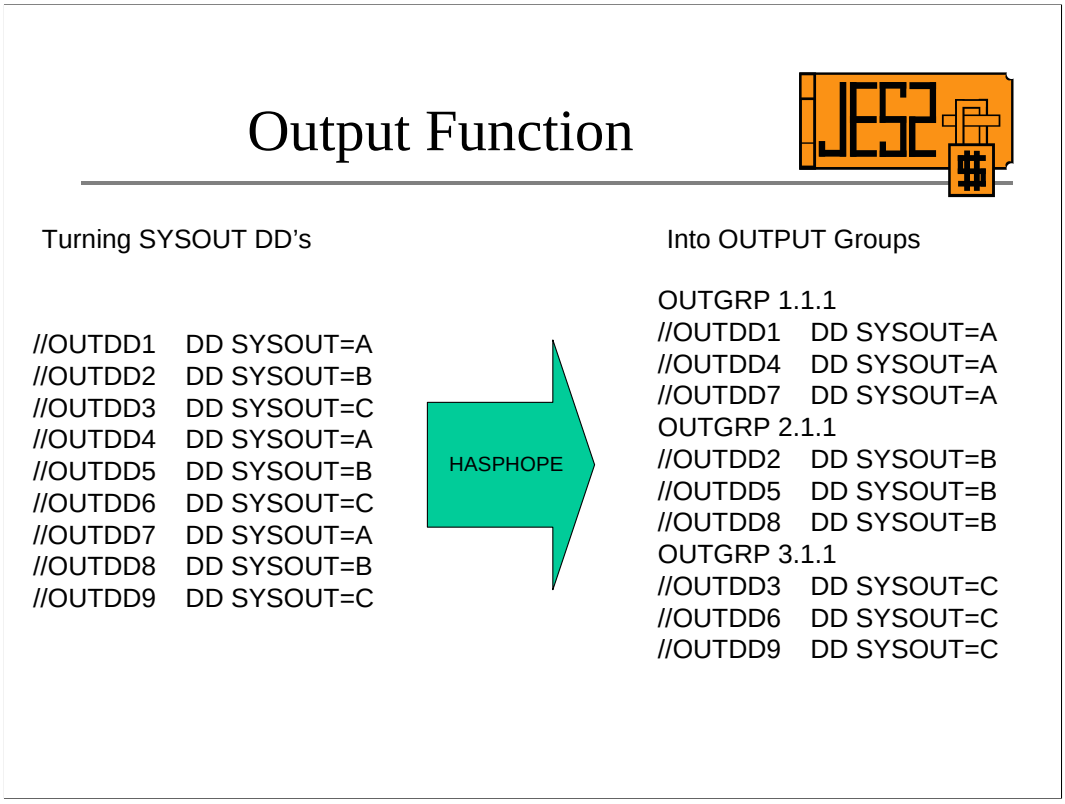
- No started message?
  - Check to insure the INIT is running at proper DP
  - ABTERM INIT via ISV “kill” command
    - CANCEL/FORCE not valid against idle INITs
- Ended message received?
  - $P involved INIT
  - Dump of JES2, MASTER and INIT asid will likely be needed.

Although there are no firm answers these are definitely situations we see reported frequently. For the situation where the job has ended there is usually some sort of error. Could be anything from a JES2 bug, to EOM abends to $SJB overlays. A dump of involved asids will be needed for further diagnosis.

Output

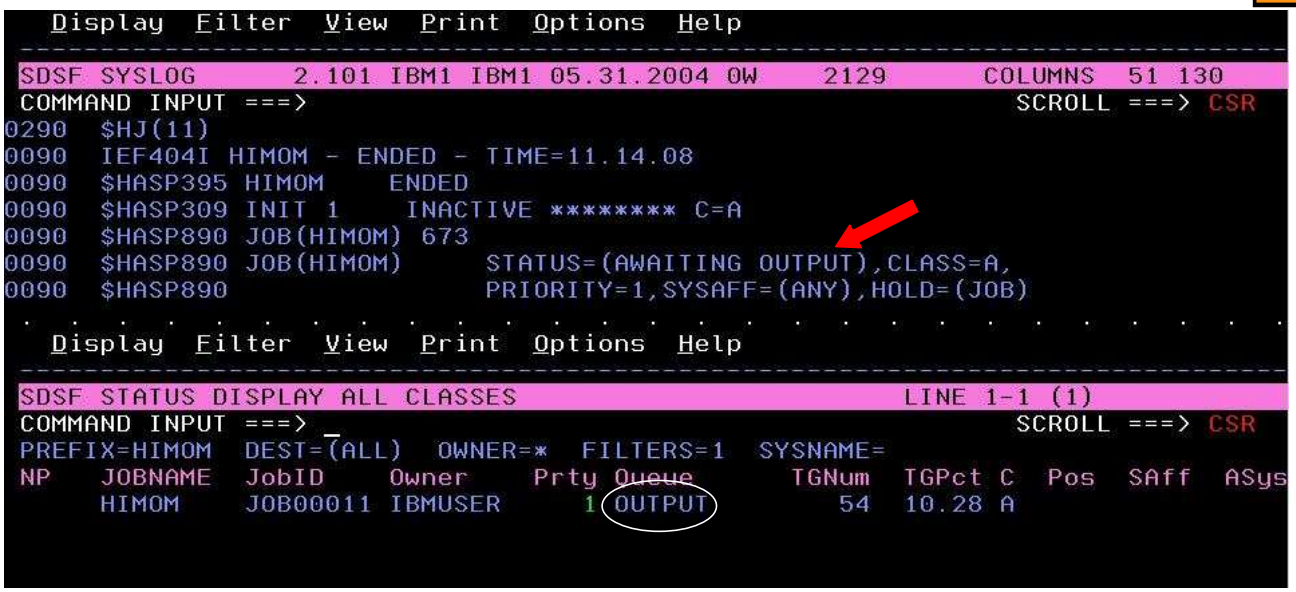
Organizing the leftovers

Output Function



Output is the phase of JES2 processing that deals with the execution phase sysout remnants. It takes the non-null sysout datasets and collects them into output groups or JOE's (Job Output elements) and then places them on the appropriate hardcopy queue.

$HJ Confusion



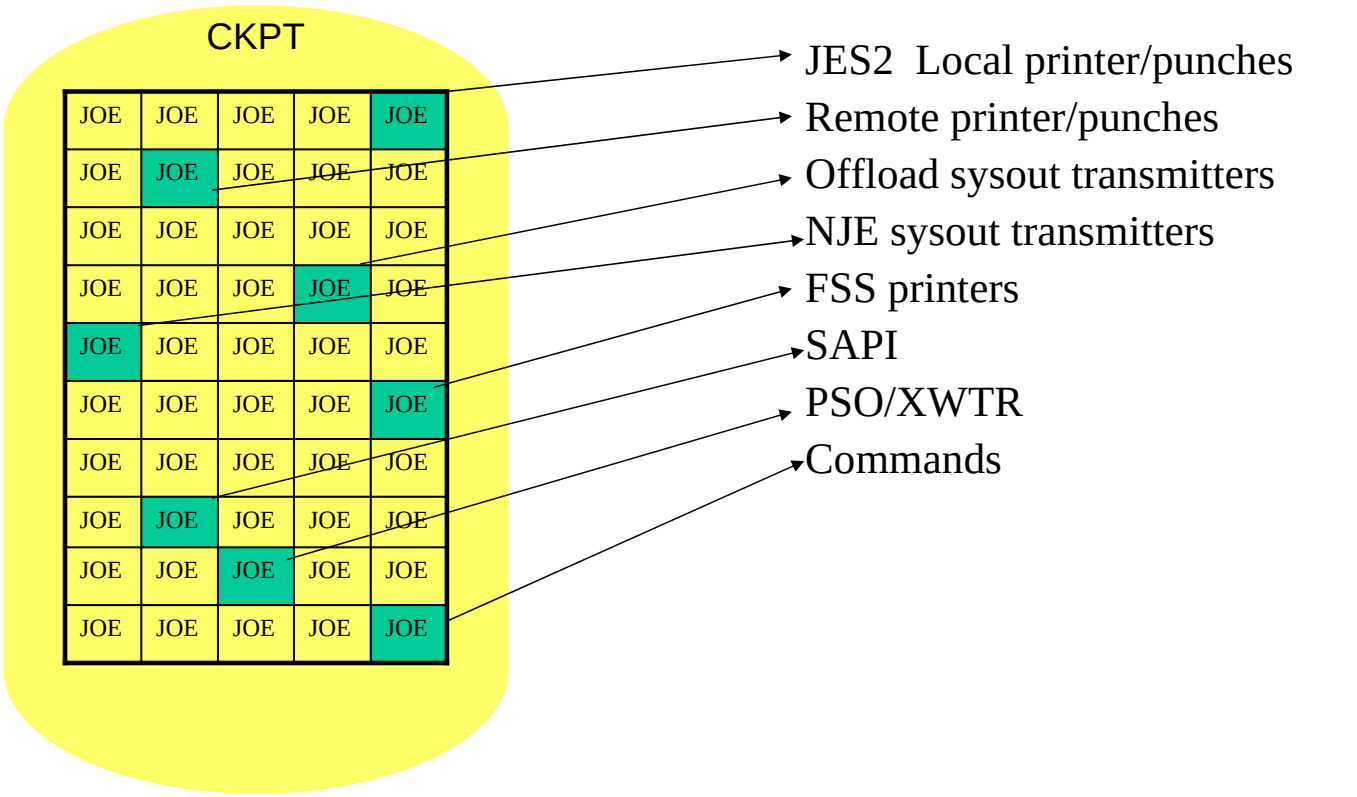
- JOB's in this status cannot be spool offloaded
- Prevents executing JOB's spun sysout from being processed
- $AJ to release the job

A $HJ issued against a job while it is executing will prevent it from going through output processing. This means no OUTGRP's will be produced for the job so it will only show up on the SDSF ST panel. JOB's in this state cannot be offloaded or processed by printers/sapi/pso since these interfaces work with JOE's.

Hardcopy Phase

Come get some!

Hardcopy phase: Pick me!



The hardcopy phase deals with the processing of JOE's by JES2 devices, commands and SSI interfaces. All of the interfaces can supply a filter (work selection criteria) to determine which OUTGRP's they want to select and process.

Stuck JOE's

IF FSS mode printer and:

```
 $DOS(12)
STC00012  $HASP686 OUTPUT(IRRDPTAB)
$HASP686 OUTPUT(IRRDPTAB)  OUTGRP=1.1.1,STATUS=(ON PRT1/IBM1),
$HASP686                BURST=NO,FCB=****,FLASH=****,
$HASP686                FORMS=STD,HOLD=(NONE),OUTDISP=WRITE,
$HASP686                PRIORITY=144,PRMODE=LINE,QUEUE=A,
$HASP686                RECORDS=(97 OF 97),ROUTECDE=LOCAL,
$HASP686                SECLABEL=,TSOAVAIL=NO,UCS=****,
$HASP686                USERID=++++++++,WRITER=
```

AND

```
 $DPRT1
       $HASP603 PRT1
$HASP603 PRT1     UNIT=0017,STATUS=INACTIVE,BURST=NO,
$HASP603         CKPTLINE=0,CKPTMODE=PAGE,CKPTPAGE=100,
```

Stuck JOE's continued

1. Attempt to drain the printer ($PPRT1) (see note)
2. If printer goes into DRAINING then force the printer if option available (PSF MODIFY FORCE)
3. Shutdown FSS asid and/or cancel it

Note: if after step1 printer does go drained and OUTGRP still shows busy then take dump of JES2 and FSS asid and call me.

Apar: OA10769 for JOE's stuck on RJE devices (z5 only)

1. Even though the FSS printer has gone inactive the FSS code driving the printer may not have released the OUTGRP back to JES2, Until this release is done the OUTGRP will appear busy.
2. By forcing the TCB that represents the printer this will drive printer recovery code as well as JES2's End of Task code. This code will force the release of the datasets when this printer TCB ends.
3. If the FSS product does not allow the force at the printer level or if the printer still shows as active/draining then a shutdown and/or force of the asid will drive FSS level end of task code and JES2 has end of Memory code to insure that printers and their work are cleaned up.
4. Apar OA07070 is a recent FSS interface apar that is worth reviewing.
5. As for my NOTE. Even with the various levels of code that try to insure that OUTGRP's are released from the printer there is still some cases when this does not occur. A dump of JES2 and the FSS asid taken early is important in trying to determine the nature of the problem.

Purge Phase

The Return of the Track Groups

Purge Phase Basics

- Spin JOE's and JOE-less jobs placed on purge queue
- Spool space returned at this time only
- I/O Intensive
- $HASP250 issued for job purge.
  - No message for a SPIN JOE purge

Job is purged ($HASP250) when all JOE's for the job have been processed or $C/$P was issued against the job. The purge process is the only time track groups (the units that spool space is allocated in) are returned. For non-spin datasets, the data associated with the job remains on spool until all of the JOE's are processed or cancelled. When the $HASP250 is issued for the job then all the spool space that could be returned has been.

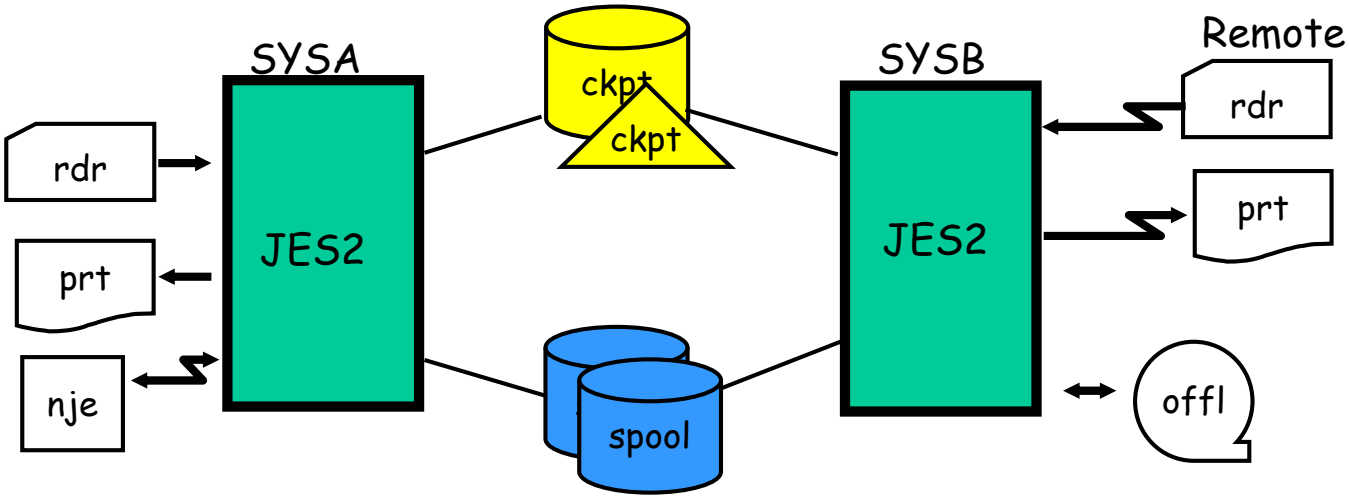
It’s still Purging!

- Purge of a large job can take a long time
- Minimum twice as many I/O’s as trackgroups
- Spool fencing means the I/O’s driven to a subset of spool volumes
- $DJ,LONG will show the TGS= count of the job decrease
- Spool space is being returned through the purge of the job

Purging is very I/O intensive. A large job that has 50,000 TG’s will need to do at least 100,000 I/O’s in the course of purging the job. This can take time. The good news is that TG’s are being returned while the job is purging. The TG count for the job shown on the $DJ,LONG or the SDSF ST panels will decrease as each secondary allocation $IOT is completely purged. This results in decrements of approximately 1000-1200 at a time.

MAS Surprises

MAS Structure



A basic MAS structure is diagrammed here. A MAS (multi-access spool) configuration allows for more than one JES2 image to put jobs through the phases discussed earlier. Oddly enough this shared work concept brings with it one or two surprises: some expected, others not.

Commonly noted Quirks

- Jobs can convert on any member of the MAS
- Jobs can execute on any member of the MAS
- With z/OS 1.4 duplicate TSO logons allowed

These are commonly reported as errors when JES2 members form a MAS.

JOB's unless restricted by SYSAFF via /*JOBPARM or Exit or INTRDR default ($TINTRDR,SYSAFF=) can go through the conversion phase on any member of the MAS. Usually it will convert on the member that submitted the job (INPUT PHASE) simply because that member has access to the checkpoint during the input phase. With access to the checkpoint the job can be queued immediately for conversion and if there is a free converter it will be picked up. However if all converters on the submitting member are busy then the job will stay on the AWAITING CONVERSION queue and if the ckpt is released, another member of the MAS can convert the JCL for the job.

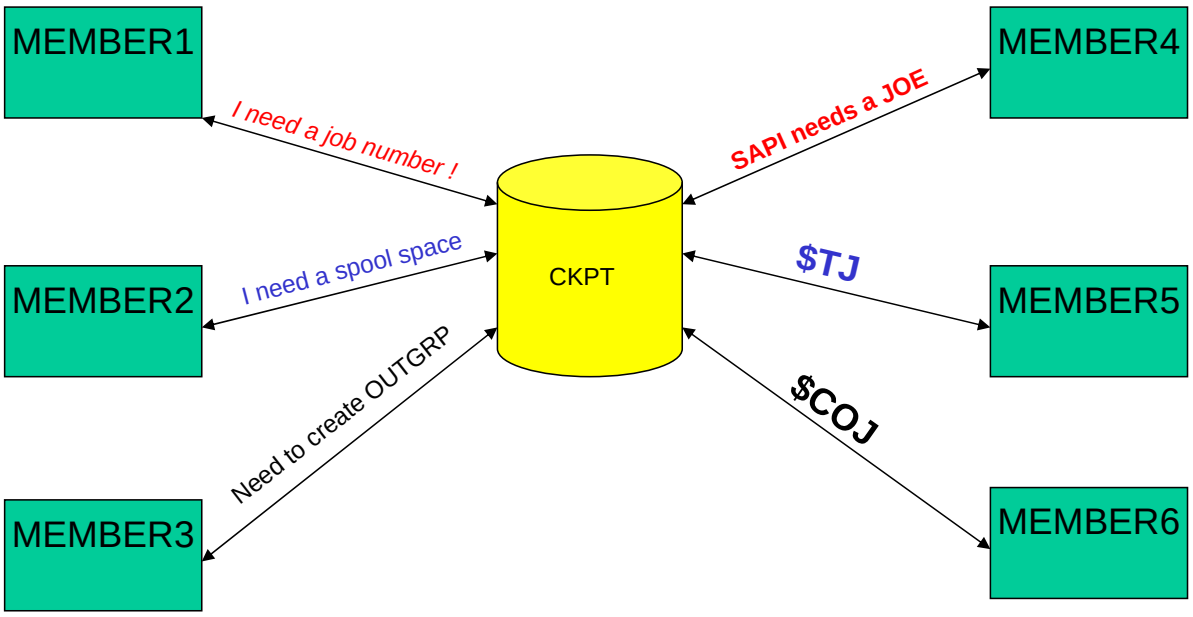
The same holds true for job execution. When using JES2 managed initiators if there was an available INIT on the member that submitted the job, the job would run on that member. However if there were no INIT's on the submitting member available it can run in any other available INIT managed by other MAS members.

Both of these typically show up as a job failing because it does not have access to proclibs or datasets on the system on which it converted or executed.

As per the Migration Notebook JES2 at z/OS 1.4 no longer polices duplicate TSO logons

The dreaded message for those running a MAS. What?, When?, Why?

Everyone, play nice and share!

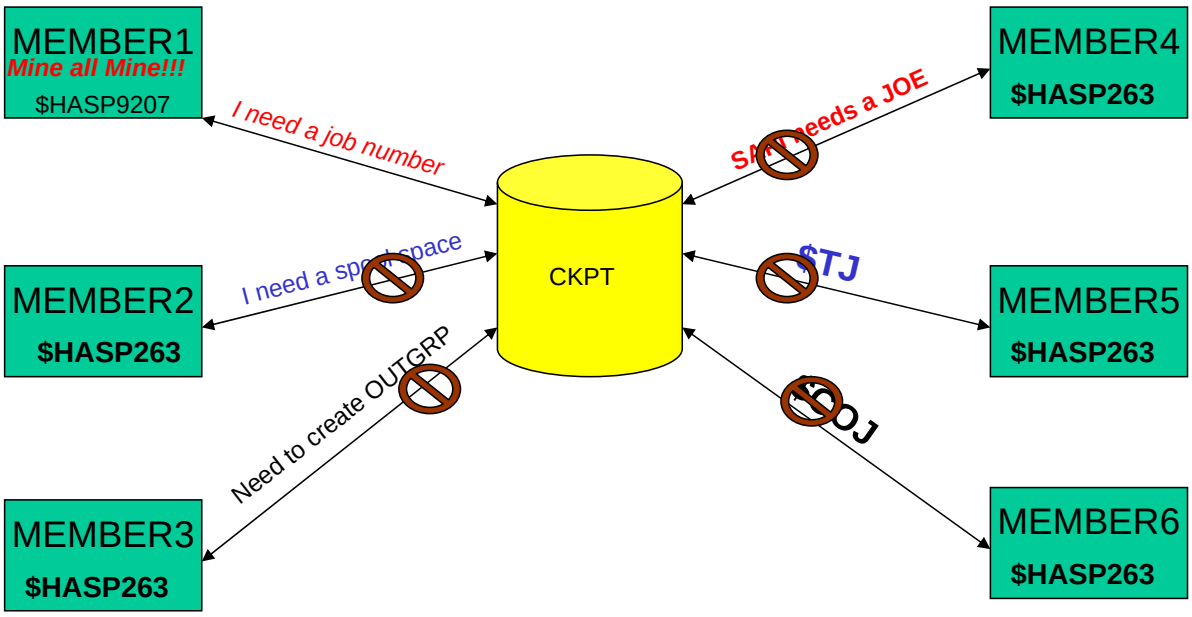


The checkpoint contains a variety of control blocks. $JQE's which represent a job as it moves through JES2 phases. $JOE's that represent the OUTGRP's created during SPIN and OUTPUT processing. JOB numbers, spool space, spool volume information and others are stored on the CKPT. When there is only one MEMBER then it can always get access to the CKPT. When there are multiple members these shared JES2 resources need to be serialized amongst the members. This is done via CKPT dasd reserve or coupling facility list structure lock. Now that there are multiple members competing for the checkpoint and access is exclusive, the potential exists for a member not to play nice and refuse to share the goodies.

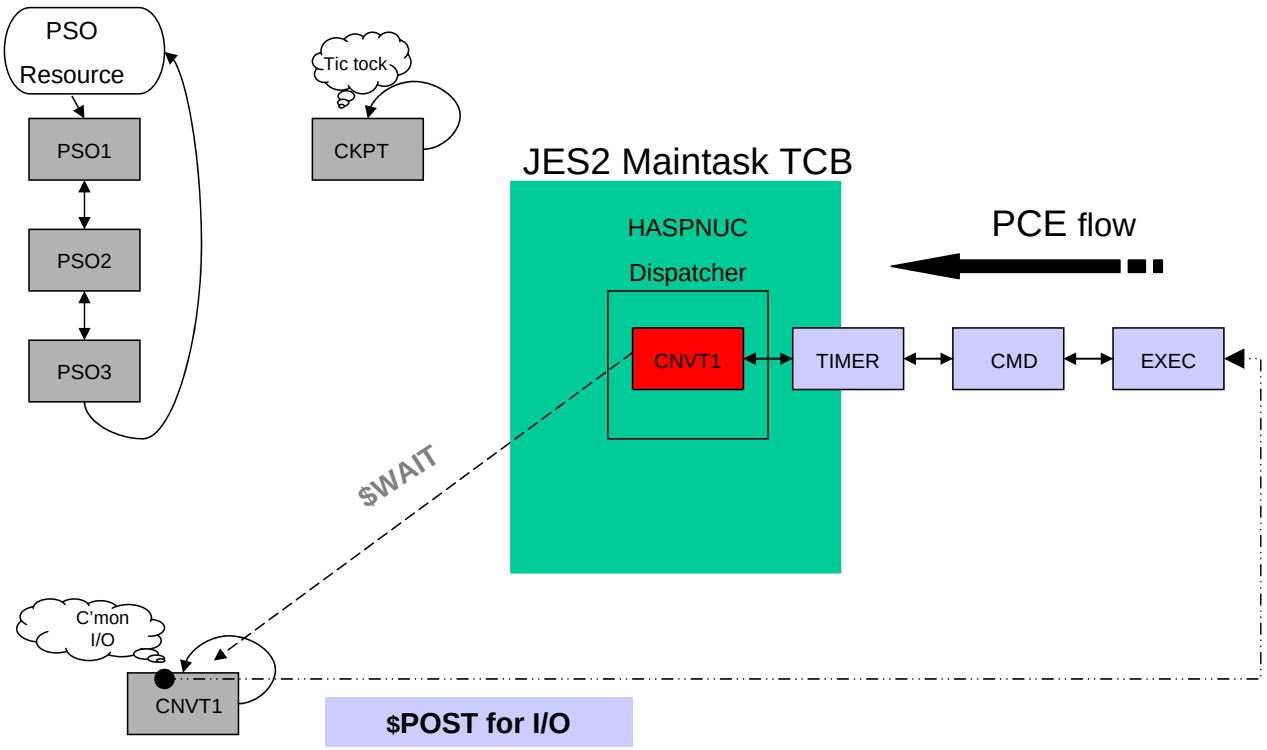
MASDEF Settings

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

If not then…


MEMBER1
Mine all Mine!!!
$HASP9207
I need a job number
MEMBER2
$HASP263
I need a spool space
MEMBER3
$HASP263
Need to create OUTGRP
CKPT
MEMBER4
$HASP263
$TJ
MEMBER5
$HASP263
$OJ
MEMBER6
$HASP263

One TCB MVS Wannabe

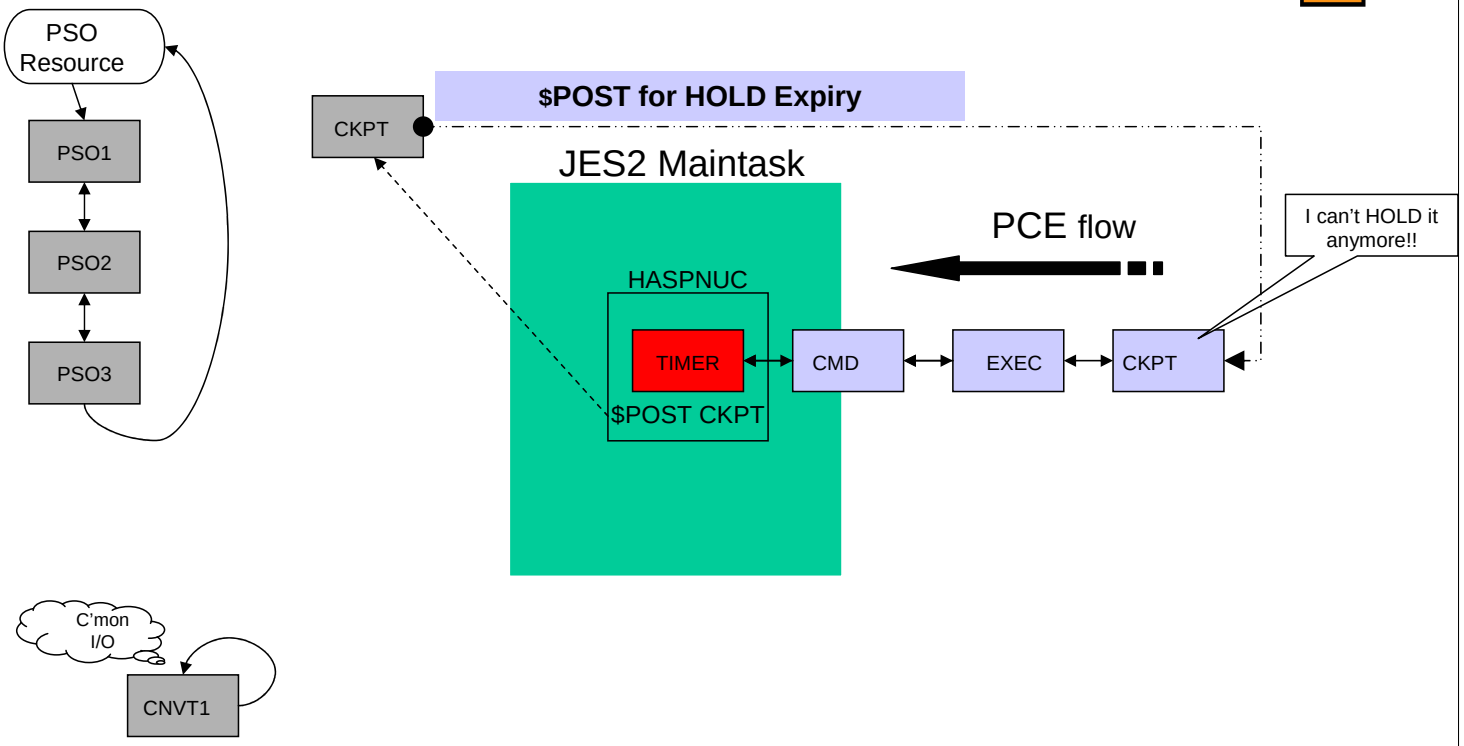


But how could one JES2 member get itself into this situation. In order to explain this we need to talk a little bit about how JES2 manages its work.

There are many TCB's in the JES2 asid but there is only one MAINTASK TCB and it does not like to MVS WAIT since this prevents it running other units of JES2 work. The other TCB's (see JES2 Subtasks slide) are essentially used to subtask routines that could end up in an MVS wait: conversion, saf calls, dynallocs. It is under the MAINTASK that the bulk of the JES2 work is done (JES2 address space level work). How does it do this? By using a JES2 dispatchable unit of work called a $PCE (processor control element) it is like a TCB JR. Instead of MVS wait and post, JES2 has a $WAIT and $POST mechanism.

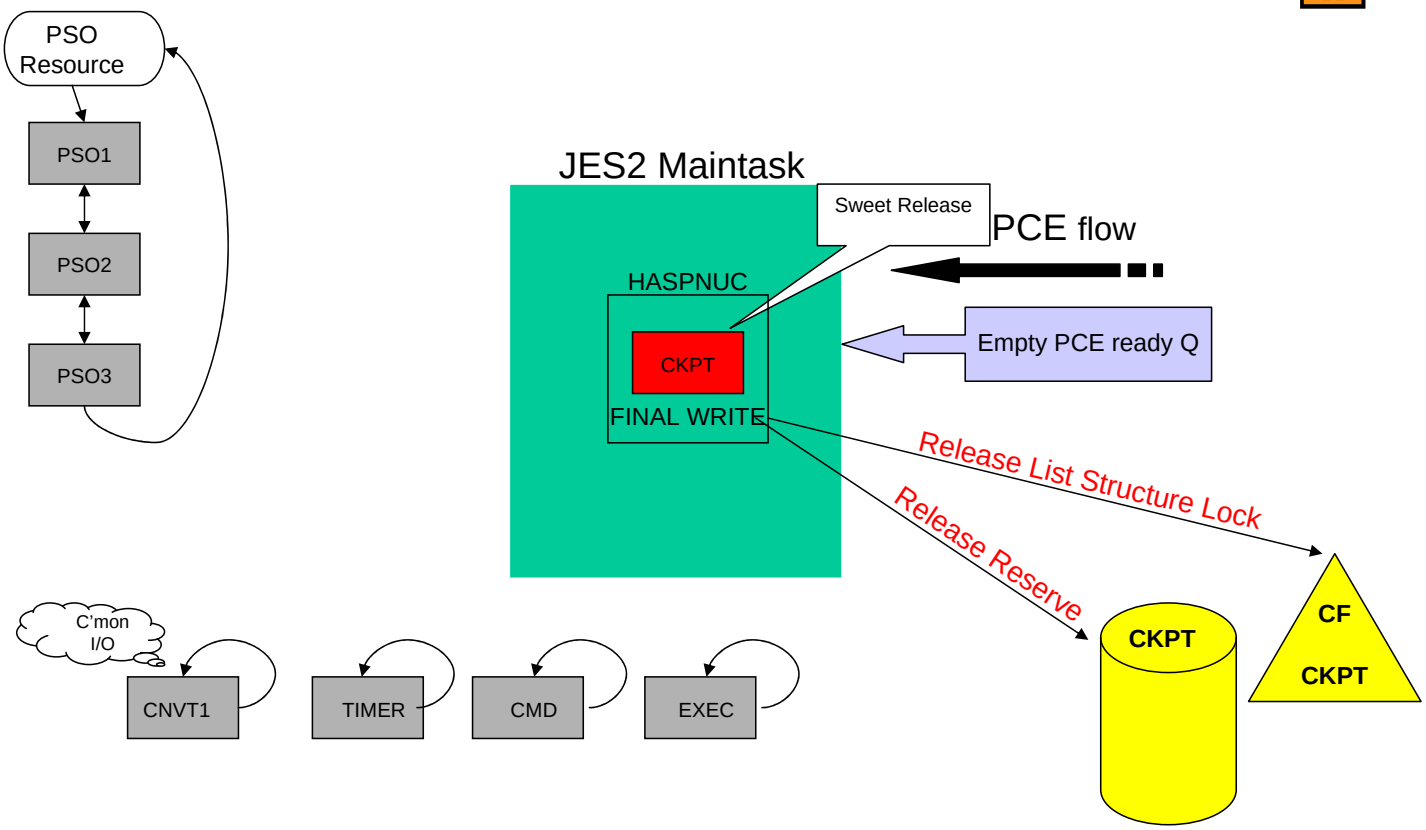
There are $PCE's for almost every JES2 function and they vary in number. INTRDRs, printers, punches, NJE devices, SAPI, PSO etc all have $PCE's. Basically if JES2 is doing something for you it is running as $PCE. So what does a $PCE $WAIT for? There are a variety of reasons that a PCE can wait but the most common are? Waiting for work, such as a INTRDR waiting for a job to be submitted to it. Waiting for spool I/O, most processes have to read or write something from the spool. Waiting for access to the CKPT, if a $PCE needs to update a control block that resides on the checkpoint and this member does not own the ckpt then it will wait ($QSUSE).

MASDEF HOLD= Expires



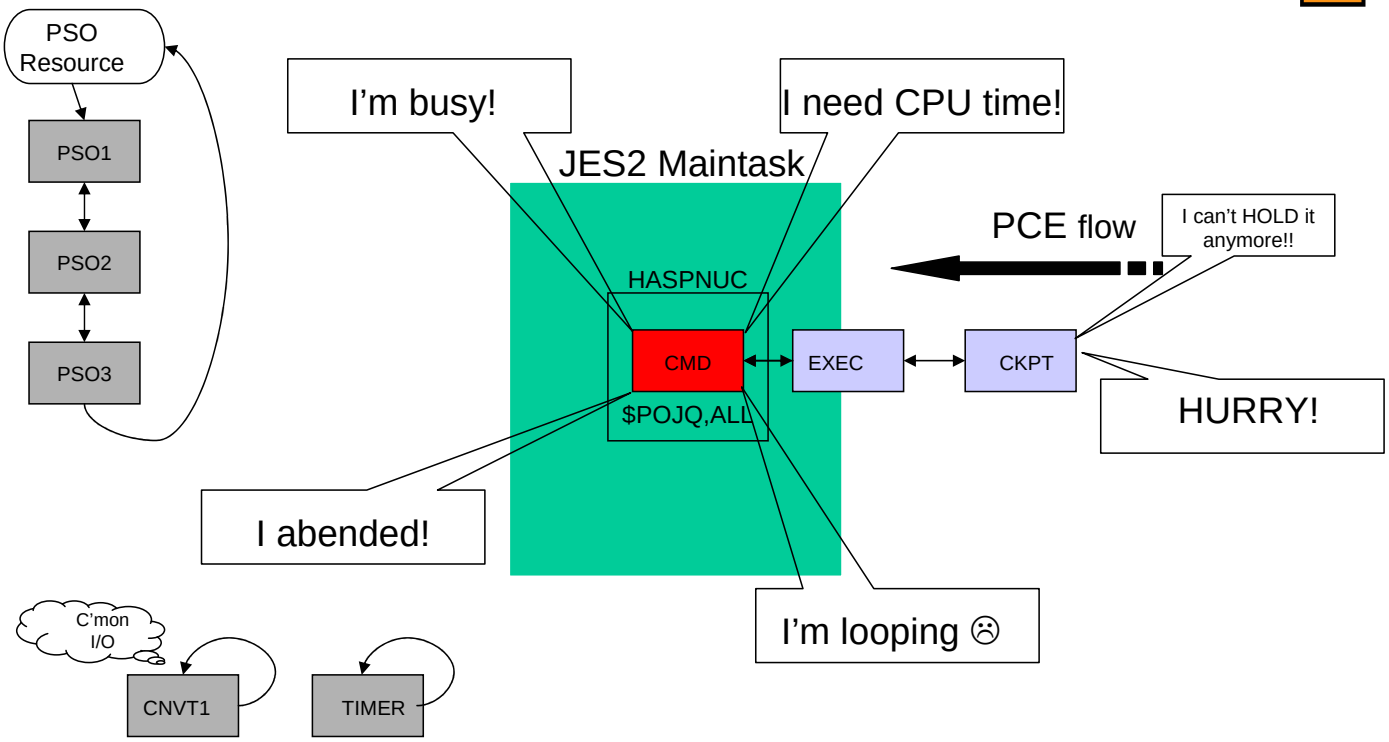
When the MASDEF HOLD= time setting expires the TIMER PCE $POST's the CKPT for the interval expiring. The $POST places the CKPT PCE on the end of the PCE ready queue as it has to line up for its turn at the JES2 maintask

Checkpoint Released



As the PCE's ahead of the CKPT PCE run and $WAIT, the CKPT PCE can get control and perform the final write of the checkpoint dataset. When the writes to the CKPT (dasd or CF) are complete, the DASD reserve or the CF List structure lock is released. Once released the other members of the MAS that are currently waiting on the reserve or CF lock can acquire it and run PCE's under their maintask that require the CKPT. This non CKPT holding membersr can continue to process PCE's up to the point that the PCE requests access to the CKPT (not all PCE's need access).

First Predicament

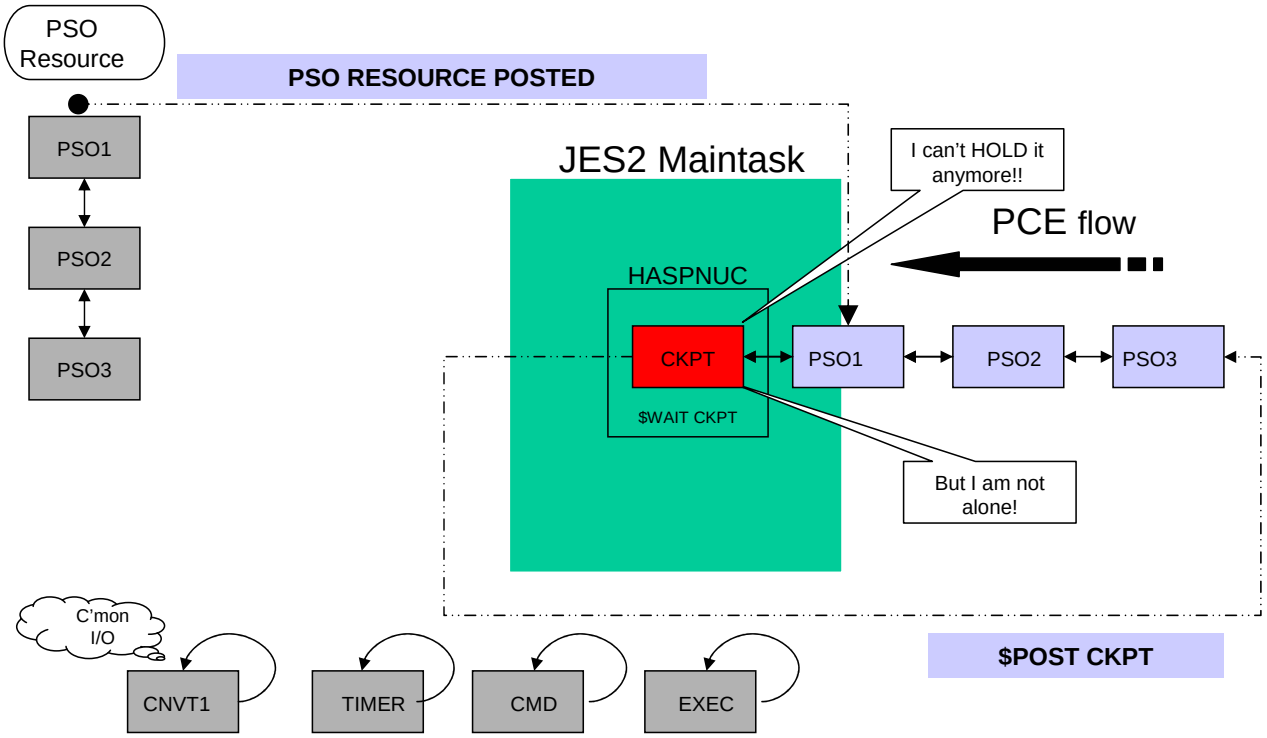


The first predicament that the CKPT PCE can find itself in, is being unable to get dispatched under the maintask. There are several reasons this could occur.

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.

Second Predicament



The MASDEF HOLD= time setting is a guideline not a rule. Even if the CKPT PCE gets control to do the final write it checks to see if there are other PCE's ready to run. In other words it checks to see if it is alone on the PCE READY queue. If it is not then the CKPT PCE places itself at the end of the current ready queue by posting the CKPT resource then $WAITing on it. Similar to waiting on a posted ECB.

The issue here is if there are enough PCE's running so that at least one is always on the ready queue after the CKPT PCE then the release of the CKPT will be delayed. A $HASP263 is less likely to be issued for this reason in z4 as base changes were made to prevent this type of situation from preventing the release of the CKPT.

Who HOLDs it?

- $HASP263 not issued on HOLDing member
- IOS071I 016E,**,*MASTER*, START PENDING not issued on HOLDing member
- $HASP263 LOCK HELD BY MEMBER member_name when CKPT on CF
- $HASP9207 JES CHECKPOINT LOCK HELD DURATION issued on HOLDing system (z4 and up)

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.

Who HOLDs it? cont'd

When CKPT on DASD

RO *ALL,D GRS,DEV=/nnnn

```
RO *ALL,D GRS,DEV=16E
IEE421I RO *ALL,D GRS,DEV=16E 712
SYSR14   RESPONSES ----------------------------------------------------
ISG343I 13.28.38 GRS STATUS 711
DEVICE:016E VOLUME:J2WRK1 RESERVED BY SYSTEM SYSR14
S=SYSTEM  SYSZJES2 J2WRK1SYS2.JESCKPT1                                T
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR   STATUS
SYSR14    JES2                0017       006F9848 EXCLUSIVE   OWN
TST1     RESPONSES ----------------------------------------------------
ISG343I 13.28.38 GRS STATUS 858
DEVICE:016E VOLUME:J2WRK1 NOT RESERVED BY SYSTEM TST1
S=SYSTEM  SYSZJES2 J2WRK1SYS2.JESCKPT1                                T
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR   STATUS
TST1      JES2                0017       006F9848 EXCLUSIVE   OWN
```

Persistent $HASP263's

- Check health of system
  - MVS commands responding?
- Check JES2 CPU usage
  - D A,JES2
  - Wait 60 seconds
  - D A,JES2 look at delta
- Check for outstanding JES2 WTOR's
  - $HASP098 termination options
- Check for indications of JES2 functioning
  - $HASP100, $HASP250, JES2 commands working

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? System 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.

Transient $HASP263's

- 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
- $DPERFDATA(EVENT) shows JES2 events including long PCE dispatches
- $DPERFDATA(*) commands, combined with console dump may be needed to understand the issue

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 trouble some. 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.

Monitor Probe Messages

- $HASP9201 JES2 MAIN TASK WAIT DETECTED AT module+offset
- $HASP9202 POTENTIAL JES2 MAIN TASK LOOP DETECTED NEAR module+offset
- $HASP9203 LONG PCE DISPATCH
- $HASP9204 JES2 MAIN TASK BUSY
- $HASP9205 PCE WAITING FOR BERT LOCK
- $HASP9206 PCE WAITING FOR JOB LOCK
- $HASP9207 JES2 CHECKPOINT LOCK HELD
- $HASP9208 JES2 MAIN TASK LOCAL LOCK WAIT AT module+offset
- $HASP9209 JES2 MAIN TASK NON-DISPATCHABLE AT module+offset
- $HASP9210 JES2 MAIN TASK PAGING WAIT AT module+offset
- $HASP9211 JES2 MAIN TASK NOT RUNNING
- $HASP9212 MVS NOT DISPATCHING JES2 MAIN TASK (OA06186)
- $HASP9301 JES2 MAIN TASK ALERTS CLEARED

To assist in the problem determination with the JES2 asid. The JES2 Heatlh monitor comes in z/OS 1.4. These probe messages attempt to externalize potential issues in the JES2 asid. If $HASP9207 is being issued on the system then the syslog should be scanned for $HASP92xx that may assist in determining why this image is holding the CKPT.

For transient $HASP263 a MSGID slip can capture a dump when $HASP9207 is issued:

SLIP SET,MSGID=$HASP9207,A=SVCD,JOBLIST=(JES2),END

Lockout Options

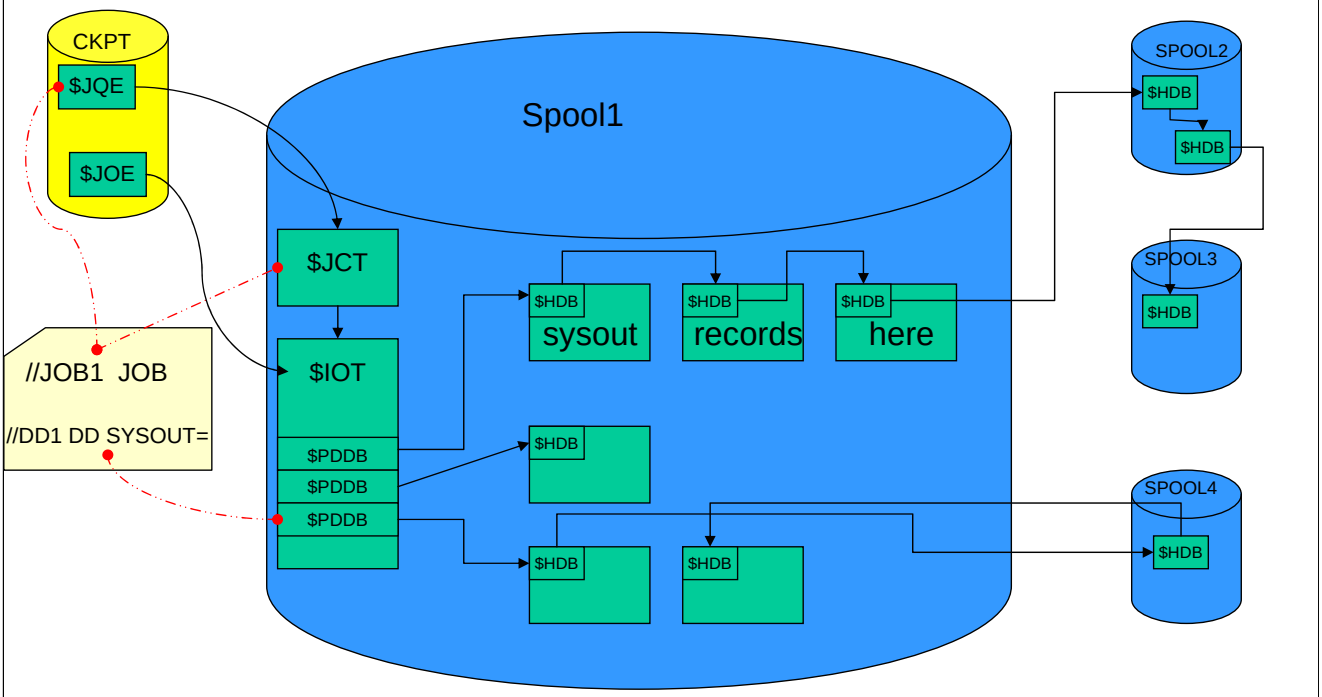
- Console dump JES2 asid first
- $PJES2,ABEND
- $PJES2,ABEND,FORCE
- $ECKPTLOCK,HELDBY=
  - Use with extreme caution
- SYSTEM RESET
  - On a down system it should clear volume RESERVE

These are the options when the problem is isolated to the JES2 asid and waiting it out is not working.

The $ECKPTLOCK,HELDBY can only be used if the CKPT is on a coupling facility. If the system holding the lock is down or going down or is having problems that is taking it down then this command can be issued on an alive member to steal the lock from the sick/dying system. If JES2 on that system tries to access the lock after it has been stolen from it, it will be abended.

The Joy of Spooling

A Job's Spool Structure



This diagram shows the basic job spool structure. It is not complete however it touches on the major control blocks that are on spool and how they are chained. The JOB level information is contained in the $JCT. This contains a lot of the information that makes it into various JES2 SMF records. The $JCT is pointed to by the $JQE in the CKPT. The $JCT points to the primary allocation $IOT. The $IOT contains imbedded in it the $PDDB's. These PDDB's equate to the SYSOUT DD's allocated by the job and essentially hold DD level info (RECFM, LRECL, FCB, UCS etc). Each $PDDB contains the pointer to the first $HDB in a chain of $HDB's which hold the data written to the sysout dataset by the JOB.

From an SDSF perspective, the $JCT information is displayed on the ST panel when you put a ? on the command line to display the alternated fields. Fields such as reader start and stop times are taken from the $JCT. If you place a ? beside the job you get the JDS panel and the information here is taken from the $PDDB's that SDSF has located in the $IOT's it despooled,

One major debugging issue with spooled control blocks is that they are not instorage all the time like the CKPT data ($JQE's , JOE's etc). These spooled control blocks are read and written as needed both in the JES2 and USER address spaces so their presence instorage is transient. The SDSF trace facility can be used to despool most but not all of the spooled control blocks associated with a job if the need arises.

SDSF TRACE Instructions

TRACE ON

this turns on the SDSF tracing function. If the DDname ISFTRACE is not in the tso users logon proc or it has not been allocated via clist or tso command then by default SDSF will allocate a spool dataset.

TRACE nnnnnnnn

where these n's represent a mask of what type of trace entries you want to collect in the SDSF trace. If you issue TRACE FFFFFFFF you get all the SDSF traces possible. This would be similar to turning on all JES2 traceids. From a JES2 point of view we are not interested in all of these traces. TRACE 00000040

- this traces SDSF spool I/O and SRB processing. Whenever SDSF needs spool info for its panels it will trace out the I/O it does and the information retrieved from spool. The control blocks you will see are the $JCT, $IOT and $BUFFER HDB's.

TRACE OFF

- turns the trace off and closes the trace dataset making the collected trace data viewable

It is also important to note that the spooled control blocks span several spool volumes. The volumes that a job has spool space on is recorded in the $JQE in the spools used mask.

Spools: Filling

- $D JQ,SPOOL=(TG>nnnnn)
- $D JQ,SPOOL=(PERCENT>nnnnn)
- SORT TGNUM D on ST panel (no other filters)
- $ACTIVATE LEVEL=z2 provides complete job spool usage in displays

Track groups are the JES2 unit of allocation for spool space. The spool volumes are formatted and carved into trackgroups based on the SPOOLDEF settings.

$HASP050 TGS SHORTAGE and $HASP355 SPOOLS FULL messages issued when a spool (Track Groups) shortage encountered so how do you determine the cause of spools filling? $DJQ commands and SDSF ST panel display track group usage at the job level. If not $ACTIVATEd to z2 mode then maximum displayable TGS per job is 32K and the jobs with greater than that amount will have **** for their TGS counts in $HASP890. If activated then jobs owning more than 32K TG's are accurately reported and filtered by JES2 commands and on SDSF panels.

No big Spool users, now what?

- $DSPL(*) shows each volume full
  - Not a normal condition
  - Implies track groups being rejected for use
- Check logrec for:
  - HASCSRDS Allocation Attempt SYMRECs
  - HASCSRDS Spool Track Group Recovery SYMRECS
  - Abend/error records prior to the first HASCSRDS record

If no one job shows up as owning spool and $DSPOOLDEF shows volumes as full then this is not a normal condition and can imply that TG's are being rejected for use. If caused by program error or control block damage spool can be used up rapidly. The presence of allocation attempt symrecs in logrec in large numbers is a definite indication of this condition.

Samples of the HASCSRDS SYMRECS:

COMPONENT INFORMATION:

COMPONENT ID: SC1BH
COMPONENT RELEASE LEVEL: Z104
SERVICE RELEASE LEVEL: OA05988
DESCRIPTION OF FUNCTION: ALLOCATION ATTEMPT ($STRAK)
PROBLEM ID: SYMTAB
SUBSYSTEM ID: JES2Z104

COMPONENT INFORMATION:

COMPONENT ID: SC1BH
COMPONENT RELEASE LEVEL: Z104
SERVICE RELEASE LEVEL: OA05988
DESCRIPTION OF FUNCTION: SPOOL TRACKGROUP RECOVERY
PROBLEM ID: SYMTABB
SUBSYSTEM ID: JES2Z104

Solutions

- $SSPL to add new volumes
- $P/$C/$PO/$CO to purge jobs
- Spool offload
- Cancel/Force problem address space
  - D GRS,C will show AWAITING SPOOL SPACE owner
- $TSPOOLDEF,GCRATE=FAST to reclaim spool space

Determining the problem address space (if there is one) is the most difficult part. When spool space is not available then jobs that require it will ENQ on the SYSZJES2 AWAITING SPOOL SPACE resource. If logrec shows many ALLOCATION ATTEMPT logrecs then it is highly likely that the problem address space either OWNS this resource or is high on the list of resource waiters. Cancelling/Forcing of these asids can be done in an attempt to remove the problem asid. If uncertain then a dump of JES2 can be taken and IBM contacted for assistance.

Spool volumes can be added to alleviate the problem and the $TSPOOLDEF GCRATE=FAST can be used to reclaim these rejected track groups. The command is I/O intensive so caution should be exercised when using it. Apar OA07284 fixes a problem which can result in invalid TGS shortage messages.

Spool: Emptying

- $PSPL to drain volume
- $DJQ,SPOOL=(VOLUME=volser) to display jobs on volume
- Jobs not automatically moved to another volume
- Jobs must be purged (eg. spool offload by volume)
- $PSPL(volser),C to cancel jobs on that volume

A the volumes that a job has had spool space allocated on is recorded at the JOB ($JQE) level. All of the data may have been physically removed from that volume but the volume cannot drain until the all jobs who once had space on that volume are purged. Apar OA07357 corrects a problem in which the spools used masks for jobs are invalidly updated during warmstart processing. This results in almost all jobs that were processed during warmstart having masks that show the jobs having space on all volumes. If warmstarts are done without the apar on then the volume will likely never drain.

Stubborn Volumes

- Make sure volume is DRAINING or HALTING
- All member warmstart required
- Before starting first member V dddd,OFFLINE
- S JES2
- $HASP424 volser IS NOT MOUNTED then
- $HASP853 REPLY GO, QUIT, OR PURGE
- R xx,PURGE

This process will allow you to remove volumes that refuse to drain. The key is to insure that the volume is either draining or halting before the all systems warmstart is performed. If the volume has gone STATUS=INACTIVE prior to the all member warmstart then the volume will not be allocated by JES2 and it can be physically removed from the system however JES2 will still keep track of the volume.

If you have volumes that show up as STATUS=INACTIVE on a $DSPL(*) command and you want to clean these up you can try the following:

1. Does volume still exist with the formatted SYS1.HASPACE on it?

2. Yes, then $SPL(volser),P to place it into draining status then follow above procedure if it will not still not drain.
3. If the volume does not exist then you will have to do the following:

a) clip a volume to a spool volser other than the one you are trying to get rid of and it should not match an active spool volume

b) sys1.haspace (or your local spool dataset name) will need to be defined as it was when it last existed

```
$DSPOOL(SPOOL1),UNITDATA
$HASP893 VOLUME(SPOOL1)  UNITDATA=(EXTENT=00,TRKRANGE=(002D,
$HASP893                 826D),RECMAX=12,TRKPERCYL=15)
$HASP646 48.1904 PERCENT SPOOL UTILIZATION
Start track  X'002D' = 45
Last track   X'826D' = 33,389
Extent size = 33,389 - 45 + 1 = 33,345
Must reallocate in exactly the same spot
//STEP1    EXEC PGM=IEFBR14
//DD1      DD   DSN=SYS1.HASPACE,DISP=(,KEEP),UNIT=3390,
//            VOL=SER=volser,SPACE=(ABSTR,(33345,45))
```

The above will accomplish this task

c) The new dataset will have to be formatted. $SSPL(volser),FORMAT,P

d) The volser will now have to be clipped to the INACTIVE volser

e) $SSPL(volser),P can now be issued to place the volume into draining status

f) volume can now be removed on an all systems warmstart if it does not go drained at this point.

Shutting down JES2

$PJES2,PRETTYPLEASE

$PJES2: Stage 1

- Runs under JES2 command processor
- Checking the JES2 asid for the “all clear”
- Issues $HASP607 JES2 NOT DORMANT if not
- Checks for:
  - -outstanding I/O and WTO’s
  - -active or ended PCE’s
  - -spool dataset browse or job info SSI’s
  - -datasets yet to be spun
  - -allocated INTRDR’s
- Post’s EXEC PCE to complete shutdown

We can break the JES2 shutdown into three stages. These are not necessarily distinct pieces of processing but really distinguished by the messages you may or may not see and the commands you may or may not be able to issue.

The first part of the JES2 shutdown process runs under the command PCE. It checks the state of the JES2 asid to see if JES2 is dormant.

(apar OW03011 has even more info)

$PJES2,ABEND or $PJES2,TERM can be used to shut down JES2 at this point if need be.

If $HASP607 is not issued then shutdown moves into what I call stage2.

Stage 2

- Stopping of JES2 and WLM initiators
- JES2 commands will not be accepted
- IEE707I command NOT EXECUTED received
- No $HASP714 JES2 TERMINATION PROCESSING AWAITING ADDRESS SPACES issued

You know when you are in stage 2 when you try to issue a JES2 command and receive message IEE707I command NOT EXECUTED. If WLM or JES2 Initiator asids are present they are posted in order to shutdown. If the JES2 shutdown hangs in this stage no message $HASP714 is issued. Apar OA06332 for Logger is a recent apar that can prevent INIT asids from ending.

For diagnosis a console dump of JES2 and MASTER should be taken.

$PJES2,ABEND can be used to terminate JES2 at this point.

D A will still show INIT's active if JES2 is stuck in this stage.

Stage 3

- Request job id tasks posted to terminate
  - syslog and RACF are examples
- ASID's known to JES2 monitored for shutdown
- $HASP714 issued periodically indicating progress
- If no progress for 2 minutes dump taken

$PJES2,ABEND can be used to terminate JES2 at this point.

The diagnostic dump should be saved and sent into IBM for diagnosis

The usual cause is an address space abnormally terminated and did not proceed through EOM.

Avoiding/Anticipating

- Drain Initiators prior to $PJES2
  - $PI
- Commands to help
  - $DPCE(*),ACTIVE>0
  - $DU,STAR
  - $DI,LONG
  - $DA,ALL
  - $DA,X
  - D R,L
  - D A,L

$PI to drain initiators prior to shutdown will allow you to anticipate a hang during shutdown if all INIT's do not drain for some reason (prior abend or other error). It also allows you to issue JES2 commands at this point whereas after the $PJES2 is issued JES2 commands my no longer be accepted.

Sample of $DPCE command issued during shutdown. Has active (non-init and non request jobid) asids are shutdown the ACTIVE count in the EXEC PCE decreases to zero and $HASP003 will be issued when using the ACTIVE filter.

```
$DPCE(*),ACTIVE>0
$HASP653 PCE(EXEC)      COUNT=(1,1,0),ACTIVE=2,TRACE=NO
p tcas
IKT006I  TCAS ENDED
IEF404I TCAS - ENDED - TIME=12.32.28
$HASP395 TCAS     ENDED
$DPCE(*),ACTIVE>0
$HASP653 PCE(EXEC)      COUNT=(1,1,0),ACTIVE=1,TRACE=NO
z net,quick
 IST102I  VTAM IS NOW INACTIVE
 $dpce(*),active>0
        $HASP003 RC=(52),
$HASP003 RC=(52),PCE(*)  - NO SELECTABLE ENTRIES FOUND MATCHING
$HASP003         SPECIFICATION
$DPCE(exec)
$HASP653 PCE(EXEC)      COUNT=(1,1,0),ACTIVE=0,TRACE=NO
```

Diagnostic Helpers

- SYSTEM TRACE, bigger is better
  - TRACE ST,999K to maximize it.
- CSA tracker always a good idea
  - VSM TRACK CSA(ON) SQA(ON) in DIAGxx
- If in doubt, dump JES2 and others
  - SDATA=(PSA,NUC,SQA,LSQA,RGN,LPA,TRT,CSA,GRSQ), DSPNAME=('JES2AUX'.*),JOBNAME=JES2,END
- Keep all JES2 MAS wide dumps
  - $Jxx, $Kxx, $Qxx among others generate remote dumps
- SDSF troubles?
  - Ask the user exactly what they did.

TRACE ST,999K to maximize it.

VSM TRACK CSA(ON) SQA(ON) It is on by default and for debugging purposes it is a very valuable tool.

JES2 has several internal ctraces that wrap along with other diagnostic information so if a JES2 related problem is suspected a console dump of JES2 is never a bad idea. Also if there are any related asids (XWTR, SAPI user, FSS, STC/TSU/JOB) then include those as well.

SAMPLE SDATA for JES2:

SDATA=(PSA,NUC,SQA,LSQA,RGN,LPA,TRT,CSA,GRSQ),

DSPNAME=('JES2AUX'.*),JOBNAME=JES2,END

Additional jobnames can be included on the JOBNAME keyword. ASID= can be used to dump a list of asids.

JES2 monitor asid will automatically be dumped if JES2 asid dumped.

Certain JES2 $ERROR situations will generate MAS wide dumps (a subset of $Jxx, $Kxx, $Qxx among others). It is important to keep all of these dumps since they are taken when one member detects an error whose origin may be on another member in the MAS.

SDSF problems are often (not always) related to the sequence of events the user performed leading up to the problem. Better to get that information up front if possible.

Diagnostic Helpers

- JES2 Performance issues?
  - PERFDATA commands

```
D A,JES2
$T PERFDATA(*),RESET
Wait 10 minutes
$JDHISTORY,HOURS=2 (z4 and OA06186 up)
$DPERFDATA
D A,JES2
```

The following set of commands can be used to get JES2 performance statistics. So for high JES2 CPU usage, JES2 slowdowns and the like this series of commands can be used repeatedly to display JES2 performance information. The $JDHISTORY,HOURS=2 (z4 and OA06186 up) command will give monitor information for the last interval which may also be useful in problem determination.

Quickies

- Abend878's during JES2 startup
  -code REGION=0 for JES2
- Bert Shortages are bad
- Large numbers of duplicate jobs can cause performance problems
- $ZAPJOB use with caution.

We often see abend878's when starting JES2. This occurs typically when the above the line size is restricted via SMF exits or Region on the jobcard. To avoid this situation REGION=0 should be coded. Changes in definitions could cause storage increases and if the region is too small then suddenly JES2 will not start. Typically for JES2, once it has gone through a peak busy period storage should not increase to much after that due to the use of CPOOL's.

BERT shortages are not a good thing. BERT's are needed when processing various control blocks in JES2 and a shortage can cause $WAIT's for berts in PCE's that may not normally wait and this can lead to unforeseen errors. It is best to run with too many than too few. As with all checkpointed CB's their usage is pushed to the front of the virtual storage range to avoid fragmentation so over specifying should not be an issue.

Large numbers of duplicate jobs that could be processed by more than one INIT in the JESPLEX can cause high CPU usage in JES2. $DPERFDATA will typically show high CPU in the EXEC and CKPT PCE's. To avoid this situation insure that the class these jobs are in are only being selected by one INIT in the MAS. Another option is to code XEQCOUNT=1 for that jobclass.

$ZAPJOB should be used with caution since this will "rip it out by the roots" with no regard to whether it is current inuse by a PCE or other JES2 process. This can lead to unexpected and unwanted abends.