Session 21585
Mining z/OS Debugging Nuggets

z/OS Core Technologies – August 10th, 2017

Patty Little plittle@us.ibm.com
John Shebey jshebey@us.ibm.com
IBM Poughkeepsie

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- **MVS**
- **OS/390®**
- **z/Architecture®**
- **z/OS®**

* Registered trademarks of IBM Corporation

Table of Contents

Getting Started

IEAVDUMP**

- Gives an overview of an SVC Dump
  - Dump information
    - Type of dump, time of dump, dump title, dump reason code
    - ASIDs in dump, types of storage in dump
  - System information
    - System name, z/OS release, machine type and model
    - Time of last IPL, CPU overview, size of system trace, time zone offset

** Part of IPCS 2.6i toolkit

IEAVDUMP Example

```
 IEAVDUMP VERSION HBB7770

 -------- GENERAL DUMP INFORMATION FOLLOWS --------
 DUMP TITLE:       SLIP DUMP ID=CAT1
 DUMP TYPE:        SLIP DUMP OF Z/OS HBB7790, SNAME ABCD
 DUMP TAKEN:       DEC 12 2016, 11:33:07 (LOCAL)
 SLIP TRAP:        SLIP SET,IF,ACTION=(SVCD),RANGE=(1901A3B2),JOBNAME=CATALOG,JOBLIST=(C

 DUMP OF ASIDS:
   X'002E'  JOBNAME: CATALOG
   X'0021'  JOBNAME: VLF

 ELAPSED GLOBAL DATA CAPTURE (GDC) TIME: 0.71 SECONDS (BEGAN AT DEC 12 2016, 11:33:07)
   USE VERBX IEAVTSFS FOR MORE DETAILS ABOUT DUMP CAPTURE
   SYSTEM WAS NOT QUIESCED DURING GDC
DUMP ASSOCIATED WITH LOGREC ERRORID: N/A
```


This excerpt from the IEAVDUMP report, highlighting from top to bottom, shows the dump title, the type of dump, the z/OS release, the system name, the date/time of the dump, and the ASIDs included in the dump.

Additional information that is not highlighted includes the SLIP trap when the dump is a SLIP dump, whether or not the system was quiesced during global storage capture (and if so, for how long), and an associated logrec errorID if applicable.

The version number displayed at the top of the IEAVDUMP report is the version number for the IEAVDUMP exec, not the version number of the system on which the dump was produced.

IEAVDUMP example (cont)

```
-------- SYSTEM SOFTWARE INFORMATION FOLLOWS --------
SYSTEM IPLED ON:      OCT 22 2016, 03:35:51 (LOCAL)
SYSTRACE SIZE:        1024K PER CPU
GMT DELTA:            -5.00 HOURS
ENVIRONMENT:          LPAR
ARCH:                 64 BIT ENVIRONMENT
LLA SERVICES AVAILABLE:   YES
VLF SERVICES AVAILABLE:   YES
SECURITY PRODUCT:         TOP SECRET

-------- CPU INFORMATION FOLLOWS ---------------
CPU TYPE:  002964
CPU MODEL: N63

            |  # ONLINE  | NOTE:
NORMAL CPUS |          9  |
Z/AAPS      |          0  | A ZAAP IS IN THE CONFIGURATION.
Z/IIPS      |          4  | A ZIIP IS IN THE CONFIGURATION.
   ***  SEE IEAVCPUI FOR DETAILED INFORMATION ***
```


This excerpt from the IEAVDUMP report, highlighting from top to bottom, shows the date/time that the system was last IPLed, the size of a system trace trace buffer, the time zone offset, machine type/model, and the distribution of CPs (normal, z/AAPs, and z/IIPs). Note that more detailed information about the distribution of CPUs can be obtained via the IPCS option 2.6i IEAVCPUI exec, to be discussed shortly.

Additional information that is not highlighted includes the security environment, as well as verification that LLA and VLF services are available (i.e. functioning).

The version number displayed at the top of the IEAVDUMP report is the version number for the IEAVDUMP exec, not the version number of the system on which the dump was produced.

IEAVDUMP example (cont)

```
-------- ADDITIONAL DETAIL ABOUT THE DUMP DATA --------
SDRSN FIELD IS ALL ZEROES.
          SEE IEA611I OR IEA911E, IN HARDCOPY SYSLOG, FOR FINAL SDRSN,
          INCLUDING EXCEPTIONAL CONDITIONS THAT OCCURRED DURING THE DUMP WRITING PHASE
SDATA REQUESTED: OUTPUT OBTAINED FROM THE FOLLOWING IPCS COMMAND.
             IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS)

  . . . . . Lines omitted . . . . .

   ==> FLAGS SET IN SDUSDATA:
    Dump current PSA.
    Dump SQA.
    Dump rgn-private area.
    Dump trace data.
    Dump CSA.
    Dump SWA.
    Dump summary dump data.
    Dump all nucleus.
```


This excerpt from the IEAVDUMP report, highlighting from top to bottom, shows the SVC DUMP reason code and the areas of storage included in the SVC dump.

IEAVCPUI**

- Gives detailed system CPU information
  - Logical CPU numbers
  - CPU types
    - Standard, z/IIP, or z/AAP
    - Polarity: Vertical High, Medium, or Low (a.k.a. Discretionary)
  - CPU-related information
    - PSW of CP at time of dump
    - Enabled wait status, parked status
    - Addresses of CPU-related control blocks: PSA, PCCA, LCCA

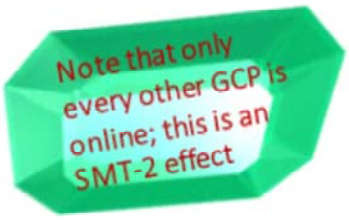
** Part of IPCS 2.6i toolkit

"Polarity" of a CP is determined by the weight and logical CPU quantities assigned to this LPAR. A vertical high CP is effectively dedicated to this LPAR. A vertical medium CP is shared with other LPARs. A vertical low CP (or discretionary CP) only is able to run when there is demand on this LPAR and other LPARs are not using all of their allotted CPU resource.

Vertical low CPs may be parked by WLM, meaning they are not available to run workload for this LPAR at this time. WLM regularly evaluates whether to park or unpark a CP based on system demand and availability of physical CPs across the CEC.

While each CP's PSA is addressable at location 0 by the unit of work executing on it, the PSA control block actually lives in ESQA storage. That ESQA address is provided in this report.

IEAVCPUI example



CPUN	CPULA	TYPE	DISC	CAP	POL	PSW (PSAPSWSV16)
0000	4000	CP	NO	NO	HIGH	07851000 80000000 00000000 1AA1832A
0001		CP				OFFLINE
0002	4002	CP	NO	NO	HIGH	07060000 00000000 00000000 00000000
0003		CP				OFFLINE
0004	4004	CP	NO	NO	MED	07060000 00000000 00000000 00000000
0005		CP				OFFLINE
0006	4006	CP	YES	NO	LOW	07060000 00000000 00000000 00000000
0007		CP				OFFLINE
0008	4008	CP	YES	NO	LOW	07060000 00000000 00000000 00000000
0009		CP				OFFLINE
000A	400A	ZIIP	NO	NO	MED	07060000 00000000 00000000 00000000
000B	400B	ZIIP	NO	NO	MED	07060000 00000000 00000000 00000000
000C	400C	ZIIP	YES	NO	LOW	07060000 00000000 00000000 00000000

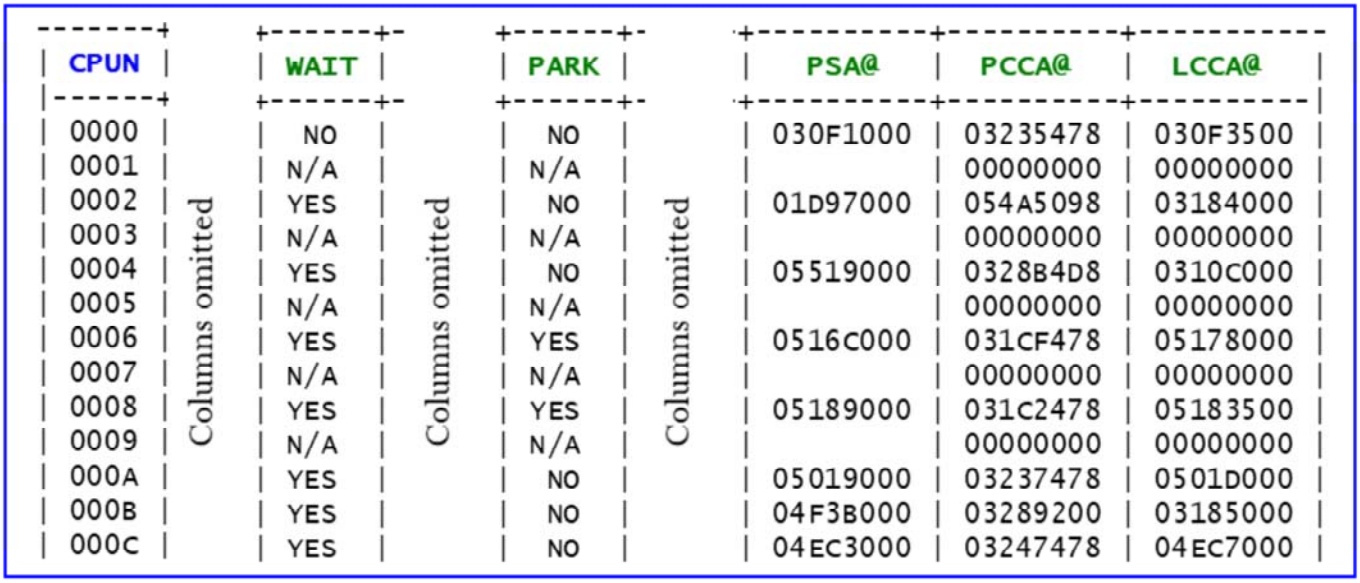
Columns omitted

More columns on next slide

This excerpt from the IEAVCPUI shows a system with 5 general (standard) CPs, 2 of which are vertical highs, 1 of which is a vertical medium, and 2 of which are vertical lows (or DISCretionaries). The system also has 3 z/IIPs, 2 of which are vertical mediums, and 1 of which is a vertical low (or DISCretionary).

Note only every other standard CP is online. This is indicative of an SMT-2 environment.

IEAVCPUI example (cont)



CPUN	WAIT	PARK	PSA@	PCCA@	LCCA@
0000	NO	NO	030F1000	03235478	030F3500
0001	N/A	N/A		00000000	00000000
0002	YES	NO	01D97000	054A5098	03184000
0003	N/A	N/A		00000000	00000000
0004	YES	NO	05519000	0328B4D8	0310C000
0005	N/A	N/A		00000000	00000000
0006	YES	YES	0516C000	031CF478	05178000
0007	N/A	N/A		00000000	00000000
0008	YES	YES	05189000	031C2478	05183500
0009	N/A	N/A		00000000	00000000
000A	YES	NO	05019000	03237478	0501D000
000B	YES	NO	04F3B000	03289200	03185000
000C	YES	NO	04EC3000	03247478	04EC7000

Columns omitted

This excerpt from the IEAVCPUI report shows the enabled wait status and parked status of each CP. It also shows the address of the CP's associated PSA, PCCA, and LCCA control blocks.

IPCS 2.6i Toolkit

- Great diagnostic stuff if you know how to use it!
  - L2 has developed these EXECs and found them so useful that we have made them available to you (shipped as compiled REXX execs with no formal documentation).
- Disclaimers
  - "Level 2 toolkit functions are intended to be used as directed by service personnel. " **(But we're empowering you!)**
  - "This CLIST is intended for IBM diagnostic support personnel. **The code contained is to be used as-is and is not supported in any way."**
- Words of assurance
  - L2 uses many of these constantly. They are safe! They present no danger to your system.
  - There are some real nuggets out there!
- Notes
  - HELP can be displayed for each EXEC, e.g. IP IEAVCPUI HELP [degree of helpfulness varies ☺]
  - You must max to the bottom of a report before you can exit it.

These EXECs are very casually supported. They are not described in the IPCS Commands manual or any other formal IBM documentation. If you find an error in a report, L2 has probably found it too and reported it to development for future correction. However, you are welcome to let L2 know if you'd like, understanding that the issue will be noted but not APARed or otherwise publicly documented.

IPCS 2.6i Panel


```
To display information, specify "S option name" or enter S to the left
of the option desired.  Enter ? to the left of an option to display
help regarding the component support.

S Name     Exec     Abstract
_ ALET2DSP IAXAR2D  DataSpace Name associated with input AR/ALET
_ CMD2FILE BLSXC2FI Writes output from an input IPCS cmd to output dataset
_ CPUINFO  IEAVCPUI displays high level CPU information
_ DAEINFO  IEAVDAE  Formats DAE information that resides in storage
_ DATAINFO BLSXDINF List and Where in storage addresses
_ DISPINFO IEAVDISP Dispatchability and lock contention information
_ DUMPINFO IEAVDUMP General and environment information about the dump
_ FRRSDATA IEAVFRRS Validity check FRR stacks
_ IOSBLKS  IOSFSMGB Information about IOS blocks
_ IPLPARMS BLSXIPLP Values used during system initialization (IPL)
_ LOCKSTAT IEAVLOCK Information about locks held at time of dump
_ LOGDATAS IEAVLOGD One line summary of input LOGREC Dataset or LOGDATA
_ SAVEAREA BLSXSAVA Maps standard savearea chain. Input save area addr
```


The NAME column is the “option name” that is mentioned in the upper paragraph. It is the name that the exec was known as internally by L2. You will find references to this original name in the execs’ help files. The EXEC column is the formal name that you should use when invoking the REXX exec via “IP execname”. The abstract gives a one-line description of the purpose of the exec. Using IP execname HELP may provide additional detail.

Any of these execs may be invoked via the line command: IP execname as an alternative to selecting it off of the 2.6.i panel.

IPCS 2.6i Panel (cont)


```
S Name     Exec     Abstract
_ SCTSIOT  IEFDDSUM Displays all DDs and DSNs in a job
_ SHOWVCM  IEAVSVCM Displays HiperDispatch information
_ SIOTPLUS IEFSIOTP Maps some fields in SIOT, EDL, DDWA, VOLUNIT, etc.
_ SLIPDATA IEAVSLIP Display SLIP control block data
_ SLOTCNT  ILRSLOTC Auxiliary slot usage information
_ SUMTRACE BLSXSUMT CPU usage information based on entries in SYSTRACE
_ SVC99RB  IEFSVC99 Maps dynamic allocation request block and text units
_ TCBMAP   IEAVTCBM Picture of TCB structure of the default ASID
_ VSMINFO  IGVVSMIN Map of Virtual Storage boundaries
_ WEBINFO  IEAVWEBI Overview of work (ie. WEBs)
_ WJSIPAMT BPXWAMT  Displays the automount rules set for this system
********************************* END OF LIST ******************************
```

Retrieving Lengthy IPCS Commands

- RETP (enter on command line without IP prefix)
  - Retrieve panel with up to last 25 commands issued

```
 ---------- Retrieve ----------
|    Options  Help               |
|  -------------------------     |
|     ISPF Retrieve Panel        |
|                                |
| Select the command             |
| to be retrieved                |
|                    More:    + |
|     1.  =1                     |
|     2.  cbf ascb1              |
|     3.  f psa                  |
```

- KEYS (enter on command line without IP prefix) – setup PF keys
  - Next to PF12, type RETRIEVE
    - PF12 key will then cycle through most recent commands

The RETP command is an ISPF command that is particularly useful in an IPCS session to retrieve lengthy IPCS commands that might have been forgotten or that you need to re-execute. For this reason, it's recommended that the minimum number of characters for a command to be saved should be set to 6. Otherwise, the 25 command slots can fill up with short commands that are faster to type than to retrieve via RETP. The instructions below describe how to set the minimum number of characters to 6.

-

From the "Options" menu (move cursor under Options and press ENTER), select option 1 (Set minimum number of characters saved in retrieve stack) by typing '1' and pressing ENTER. On the next panel, type the minimum number of characters (6) and press ENTER.

-

Note that this change affects not only saved IPCS commands, but also other commands such as TSO and ISPF commands.

Data Reduction

REPORT VIEW

- REPORT VIEW allows you to view an IPCS report in ISPF Edit mode
  - Format desired IPCS report
  - Max to the bottom of the report to buffer the data (must do this!)
  - Enter REPORT VIEW on command line
- Use ISPF Edit primary/line commands to massage/filter the data
  - EXCLUDE (X) lines
  - FIND (F) lines
  - DELETE (D) lines
  - SORT lines by columns in ascending/descending order
  - COLS to show column numbers
- Modified output can even be saved into a new dataset
  - Block off text you want to save with CC line command
  - From ISPF command line, type CREATE to specify output dataset

REPORT VIEW will only format data that has been buffered. Therefore it is important that, after you enter your IPCS command, you max to the bottom of the output before entering REPORT VIEW on the command line.

Commonly Used ISPF Edit Commands

Edit Primary Commands

• **F text**	Find text string
F text nn	Find text in column nn
F text ALL	Find all instances of text string
• **X text ALL**	Exclude (hide) all lines with text string
X text ALL nn	Exclude (hide) all lines with text string in column nn
X ALL	Exclude all lines
• **RESET**	Show all excluded lines
DEL ALL X	Delete all excluded lines
• **HEX ON/OFF**	Show/hide hex characters
COLS	Show/hide ruler with columns
SORT x y A/D	Sort data based on contents of cols x to y in Ascending/Descending order
CREATE	Create new dataset with text blocked off by CC/CC line command

Edit Line Commands

• **D**	Delete this line
Dn	Delete n lines starting at this line
DD	Deletes block of lines starting with first DD and ending with second DD
• **X**	Exclude this line
Xn	Exclude n lines starting at this line
XX	Exclude block of lines starting with first XX and ending with second XX
• **C**	Copy the content of the line
Cn	Copy the content of the line 'n' times
CC	Copy block of lines starting with first CC and ending with second CC
• **F**	Show first line of excluded text
Fn	Show first n lines of excluded text
L	Show last line of excluded text
Ln	Show last n lines of excluded text

The edit primary commands are entered on the ISPF command line. The edit line commands are entered in the leftmost column of the ISPF edit session (in row numbers).

IBMMAINFRAMES.com forum post with common ISPF edit commands:

http://ibmmainframes.com/about9529.html

REPORT VIEW example

- Issue IPCS command (e.g. IP OMVSDATA DETAIL)
- Scroll max to bottom of report (type M on command line, followed by PF8)
- Type REPORT VIEW on command line

```
IPCS OUTPUT STREAM ---------------------------------------------------------------------------------------- Line 276744 Cols 1 130
       Program Name: N/A
       Space switch stack:
         IP CBF 000000007D570000. ASID(X'0010') STR(BPXSTACK)

       Thread Attributes:
                         undetached
                         heavy


       Signal Data (Thread Level):  N/A

       Serialization Data:  N/A

 ********************************************************* END OF DATA ************************************************************

Command ===> REPORT VIEW                                                                                    SCROLL ===> CSR
```

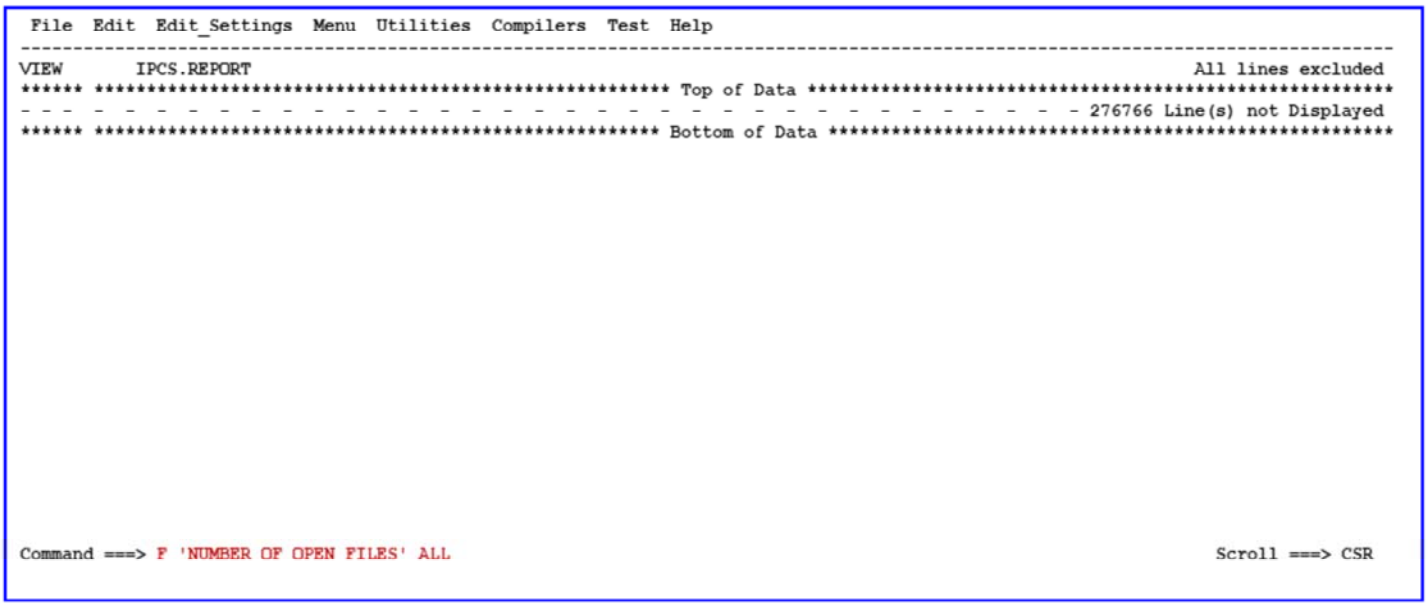
REPORT VIEW example (cont)

- Exclude all lines by typing X ALL on command line

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------------------------------------------------------
VIEW       IPCS.REPORT                                                                                  Columns 00001 00124
****** ***************************************************** Top of Data ********************************************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000001   * * * * OPENMVS REPORT  * * * *
000002
000003    Report(s):              PROCESS
000004
000005    Level(s):               DETAIL
000006
000007    Filter(s):              NONE
000008
000009
000010
000011 Suppression-On-Protection is installed
000012
000013 Kernel status:             Active
000014
000015 Kernel address space name: OMVS
000016
000017 Kernel address space ID:   X'0010'
000018
000019 Kernel stoken:             0000004000000001

Command ===> X ALL                                                                          Scroll ===> CSR
```

REPORT VIEW example (cont)

- Issue F 'NUMBER OF OPEN FILES' ALL to look for all instances of 'NUMBER OF OPEN FILES'



REPORT VIEW example (cont)

- Issue DEL ALL X to delete all excluded lines

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------------------------------------------------------
VIEW       IPCS.REPORT                                                                          253 CHARS 'NUMBER OF OPE
****** ********************************************************* Top of Data ********************************************************
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 805 Line(s) not Displayed
000804     Number of open files for this process:            1  Token: 005C20A0
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 953 Line(s) not Displayed
001758     Number of open files for this process:            0  Token: 005C2750
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 745 Line(s) not Displayed
002504     Number of open files for this process:            0  Token: 005C2E00
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 743 Line(s) not Displayed
003248     Number of open files for this process:            0  Token: 005C34B0
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 741 Line(s) not Displayed
003990     Number of open files for this process:            0  Token: 005C3B60
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 1236 Line(s) not Displayed
005227     Number of open files for this process:            1  Token: 005C4210
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 769 Line(s) not Displayed
005997     Number of open files for this process:            2  Token: 005C5620
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 911 Line(s) not Displayed
006909     Number of open files for this process:            1  Token: 005C48C0
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 796 Line(s) not Displayed
007706     Number of open files for this process:           10  Token: 005C6380
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 1166 Line(s) not Displayed
008873     Number of open files for this process:            0  Token: 005C6A30
- - - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 755 Line(s) not Displayed

Command ===> DEL ALL X                                                                                    Scroll ===> CSR
```


Note that 253 instances of ‘NUMBER OF OPEN FILES’ were found.

REPORT VIEW example (cont)

- Issue SORT 50 57 D to sort the number of open files per process in Descending order

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------------
VIEW        IPCS.REPORT                                                                          Columns 00001 00124
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+----1----+----2----
****** ********************************************************* Top of Data ********************************************************
000001      Number of open files for this process:              1  Token: 005C20A0
000002      Number of open files for this process:              0  Token: 005C2750
000003      Number of open files for this process:              0  Token: 005C2E00
000004      Number of open files for this process:              0  Token: 005C34B0
000005      Number of open files for this process:              0  Token: 005C3B60
000006      Number of open files for this process:              1  Token: 005C4210
000007      Number of open files for this process:              2  Token: 005C5620
000008      Number of open files for this process:              1  Token: 005C48C0
000009      Number of open files for this process:             10  Token: 005C6380
000010      Number of open files for this process:              0  Token: 005C6A30
000011      Number of open files for this process:              5  Token: 005C70E0
000012      Number of open files for this process:             13  Token: 01E17D30
000013      Number of open files for this process:              1  Token: 005C84F0
000014      Number of open files for this process:            543  Token: 01E1A550
000015      Number of open files for this process:              0  Token: 005C7E40
000016      Number of open files for this process:             17  Token: 005CC120
000017      Number of open files for this process:              5  Token: 01E03580
000018      Number of open files for this process:              0  Token: 005CAD10
000019      Number of open files for this process:              4  Token: 005C9250
000020      Number of open files for this process:             94  Token: 01E14100

Command ===> SORT 50 57 D                                                          Scroll ===> CSR
```

REPORT VIEW example (cont)

- Block off lines of output with CC line command; issue CREATE to store output in new dataset

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------------------------------------------------------
VIEW       IPCS.REPORT                                                                                  Columns 00001 00124
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0----+----1----+----2---
****** ***************************************************** Top of Data ********************************************************
CC0001     Number of open files for this process:           543  Token: 01E1A550
000002     Number of open files for this process:           377  Token: 01E05040
000003     Number of open files for this process:           363  Token: 01E0EA10
000004     Number of open files for this process:           358  Token: 01E1B2B0
000005     Number of open files for this process:           349  Token: 005D1EC0
000006     Number of open files for this process:           333  Token: 01E32FE0
000007     Number of open files for this process:           322  Token: 005BE470
000008     Number of open files for this process:           233  Token: 01E434B0
000009     Number of open files for this process:           139  Token: 005DE0B0
000010     Number of open files for this process:           116  Token: 005D6F00
000011     Number of open files for this process:           113  Token: 005F6B40
000012     Number of open files for this process:            94  Token: 01E14100
000013     Number of open files for this process:            90  Token: 005CFD50
000014     Number of open files for this process:            66  Token: 01E50AB0
000015     Number of open files for this process:            56  Token: 005E5FC0
000016     Number of open files for this process:            39  Token: 01E085C0
000017     Number of open files for this process:            39  Token: 01E11F90
000018     Number of open files for this process:            39  Token: 01E2A370
000019     Number of open files for this process:            35  Token: 005F1B00
CC0020     Number of open files for this process:            28  Token: 005CB3C0

Command ===> CREATE                                                                          Scroll ===> CSR
```


At a glance, I can see a distribution of file usage by process (with the largest users on top).

IEAVLOGD**

- Reduces LOGREC records to one-line entries
  - Software Records
  - Symptom Records
- Works against:
  - A formatted LOGREC data set
  - In-core LOGREC in an SVC dump or Standalone dump
- Reports lines in order that LOGREC records are encountered
- One-line entries are sortable via various columns

** Part of IPCS 2.6i toolkit

IEAVLOGD

- Line for software record includes:
  - System name
  - Date/time
  - CPU
  - ASID/jobname
  - Sequence number
  - ABEND code
  - Indication of whether dump was taken
  - Register 15
  - PSW
  - Cross memory environment

Invoking IEAVLOGD

Enter:

```
------------------ IPCS MVS LEVEL 2 TOOLKIT ------
OPTION ===> IP IEAVLOGD
```

Prompt:

```
PLEASE ENTER EITHER 'DUMP' OR 'DA(....)':
```

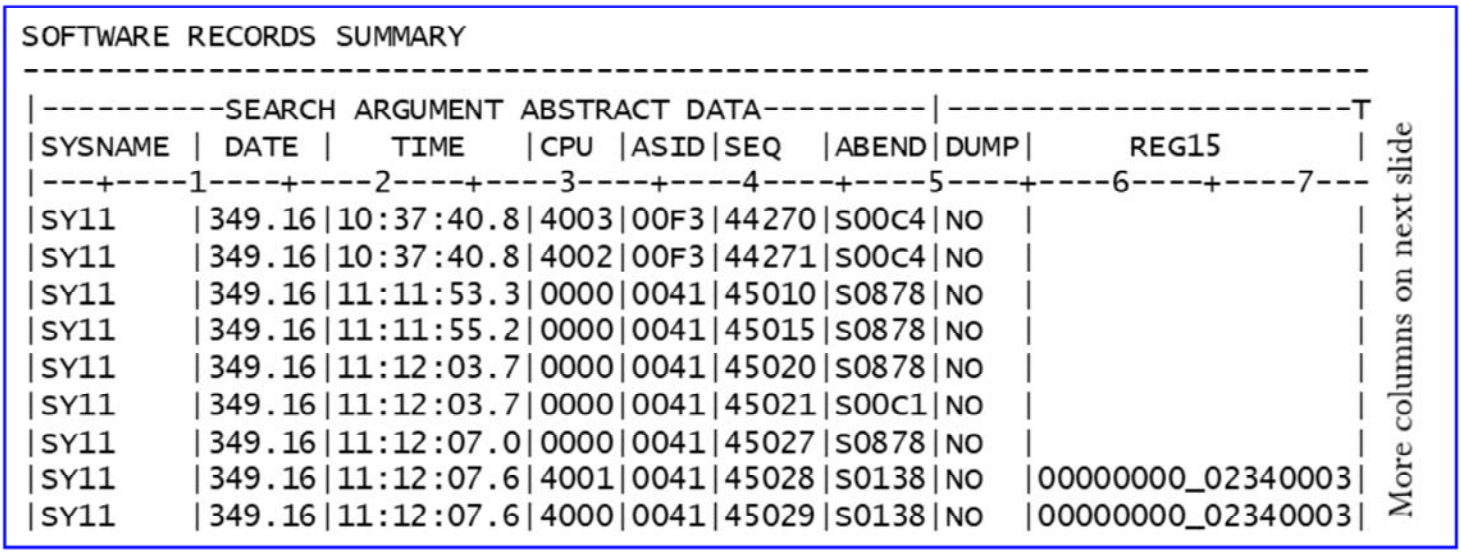
Response:

```
DUMP
  OR
DA('PATTY.MYLOGREC.DATASET.FMT')
```


The command can also be entered all at once:
IP IEAVLOGD DUMP or
IP IEAVLOGD DA('PATTY.MYLOGREC.DATASET.FMT')

It can be entered from any IPCS panel.

IEAVLOGD example

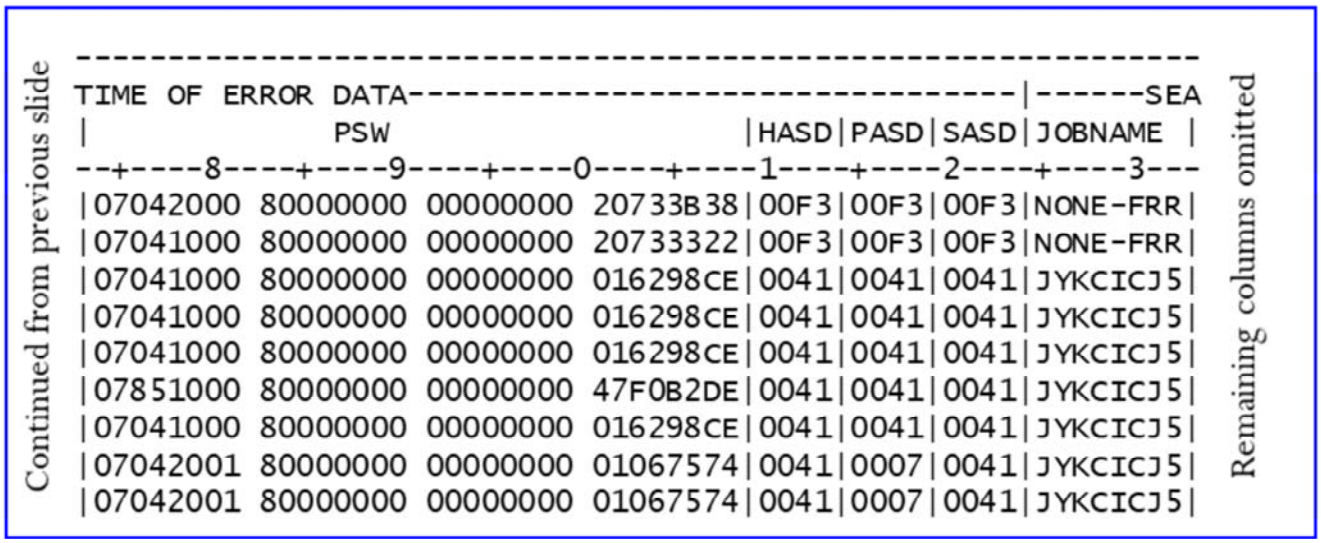


SOFTWARE RECORDS SUMMARY

SEARCH ARGUMENT ABSTRACT DATA								
SYSNAME	DATE	TIME	CPU	ASID	SEQ	ABEND	DUMP	REG15
SY11	349.16	10:37:40.8	4003	00F3	44270	S00C4	NO	
SY11	349.16	10:37:40.8	4002	00F3	44271	S00C4	NO	
SY11	349.16	11:11:53.3	0000	0041	45010	S0878	NO	
SY11	349.16	11:11:55.2	0000	0041	45015	S0878	NO	
SY11	349.16	11:12:03.7	0000	0041	45020	S0878	NO	
SY11	349.16	11:12:03.7	0000	0041	45021	S00C1	NO	
SY11	349.16	11:12:07.0	0000	0041	45027	S0878	NO	
SY11	349.16	11:12:07.6	4001	0041	45028	S0138	NO	00000000_02340003
SY11	349.16	11:12:07.6	4000	0041	45029	S0138	NO	00000000_02340003

More columns on next slide

IEAVLOGD example (cont)



VSM Nuggets

- How to:
  - Identify what area of storage an address lives in
    - IGVVSMIN
  - Identify what subpool an address lives in
    - VERBX VSMDATA 'SUM' (see appendix)
  - Identify who obtained storage in SQA or CSA
    - VERBX VSMDATA 'OWNCOM DETAIL SORT(ADDR) CONTENT(NO)'

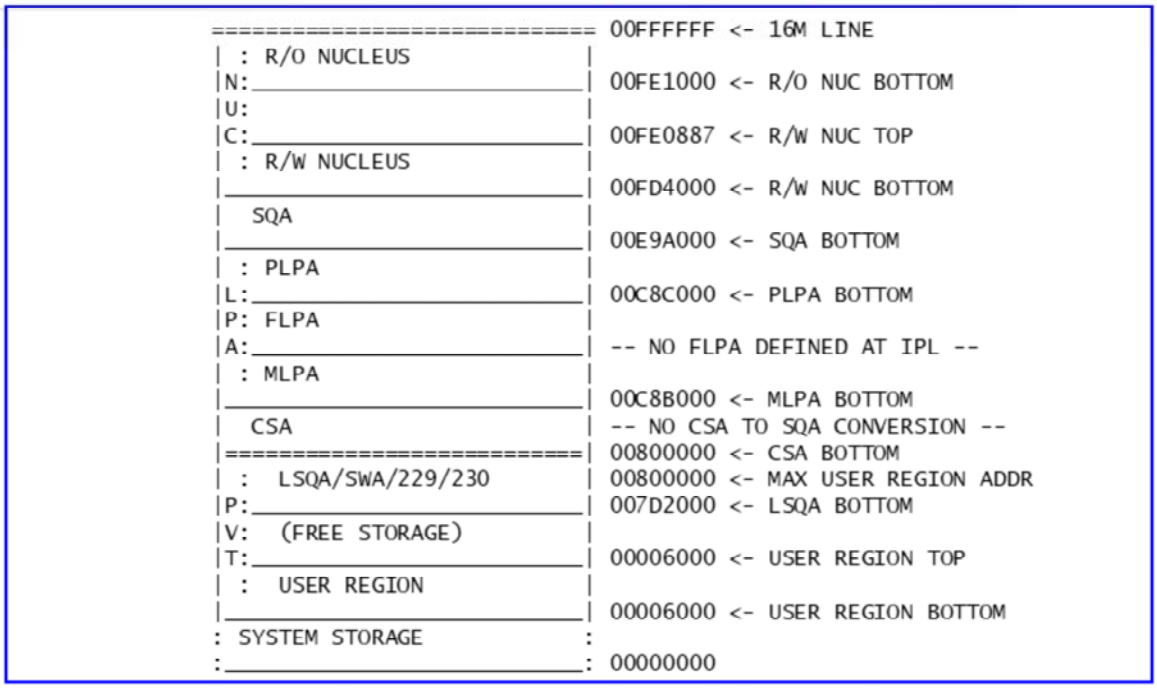
IGVVSMIN**

- Provides a map that breaks down an address space's below-the-bar and below-the-line storage into:
  - Bottom and top of private storage
    - Also notes user region max, current user region top, and current LSQA bottom
  - CSA
    - Also notes any CSA-to-SQA conversion
  - LPA, broken down into Modifiable, Fixed, and Pageable LPA
  - SQA
  - Nucleus broken down into R/W nucleus and Read Only nucleus
- Provides helpful nuggets about subpool numbers, LPA definitions

** Part of IPCS 2.6i toolkit

When the report identifies "LSQA bottom", it really is referring the to bottom of authorized private storage which includes LSQA, SWA, and high private.

IGVVSMIN map excerpt



IGVVSMIN subpool info (excerpt)

```
SUBPOOL INFORMATION:
 EXTENDED PRIVATE STORAGE
  ABOVE LINE - HIGH PRIVATE (ELSQA AND ESWA SUBPOOLS):
   ELSQA & ESWA SUBPOOLS ARE ALLOCATED FROM THE TOP DOWN
    ELSQA SUBPOOLS:  203-225, 233, 234, 235, 253, 254, 255
    ESWA  SUBPOOLS:  229, 230, 249, 236, 237
  ABOVE LINE - LOW PRIVATE (USER SUBPOOLS):
   USER SUBPOOLS ARE ALLOCATED FROM THE BOTTOM UP
    USER SUBPOOLS :  0-132, 240, 250, 251 AND 252
 ECSA SUBPOOLS    :  227, 228, 231, 241
 ESQA SUBPOOLS    :  239, 245, 246, 247 AND 248
 SQA  SUBPOOLS    :  226, 239 AND 245
 CSA  SUBPOOLS    :  227, 228, 231 AND 241
```

IGVVSMIN LPA info (excerpt)

```
LPA INFORMATION:
 WHAT DETERMINES IF A MOD IS IN FLPA, MLPA OR PLPA?
  FLPA MODS ARE SPECIFIED IN IEAFIXXX MEMBERS AT IPL.
  MLPA MODS ARE SPECIFIED IN IEALPAXX MEMBERS AT IPL.
  PLPA MODS ARE SPECIFIED IN LPALSTXX OR PROGXX MEMBERS AT IPL
```

VERBX VSMDATA 'OWNCOMM DETAIL SORT(ADDR) CONTENT(NO)'

- Provides VSM information about who owns an area of global (CSA/SQA) storage, and when it was obtained
  - Sorted in increasing address order

```
ASID Job Name Id       St T Address  Length   Ret Addr MM/DD/YYYY HH:MM:SS
---- -------- -------- -- - -------- -------- -------- ---------- --------
0053 JYKCICG0 JO530102 Ac C 00BA4D60 00000150 200655A4 12/11/2016 10:49:57
0053 JYKCICG0 JO530102 Ac C 00BA4EB0 00000150 200655A4 12/11/2016 10:49:57
006A FIT0BMHR JO239196 OG C 00BA6B50 000014B0 00008382 12/05/2016 10:01:56
0090 JXH0839A JO185121 OG C 00BA8000 00001000 216B8374 12/05/2016 02:11:12
027A JYKCICJ1 JO530250 Ac C 00BA9040 00000150 200655A4 12/11/2016 10:58:37
027A JYKCICJ1 JO530250 Ac C 00BA9190 00000150 200655A4 12/11/2016 10:58:37
```

The CSA storage at address BA6B50 for length 14B0 was obtained at 10:01:56 on December 5th by code at address 8382 in job FIT0BMHR in ASID x'6A'. That job is no longer active on the system.

The "T" column is the storage type. "C" indicates "CSA". "S" indicates "SQA".

The "St" column is the storage status. "Ac" stands for "active", which means that the address space which obtained the storage is still up and running. "OG" stands for "Owner Gone" which means that the address space which obtained the storage is now gone. Owner Gone storage is often called Orphaned Storage.

"Ret Addr" indicates the address of the code that obtained the storage.

Nuggets from the system trace table

- **SYSTRACE** formats information from the system trace table
  - **SYSTRACE ASID(X'yy') TCB(X'zzzzzz')** to format entries for a specific TCB
  - **SYSTRACE ALL START(mm/dd/yy,hh.mm.ss.dddddd)**
    STOP(mm/dd/yy,hh.mm.ss.dddddd) TIME(LOCAL)
    to format all entries in a particular time range
  - **SYSTRACE CPU(X'yy') ALL TI(LO)**
    to format all entries on a specific CP
  - **SYSTRACE CPUTYPE(STANDARD) ALL**
    to format all entries on all standard (general) CPs
  - **SYSTRACE STATUS TIME(LOCAL)** for a system trace summary

When an address space has lots of simultaneous activity for different TCBs, it is helpful to be able to filter the trace entries by TCB to get a clearer picture of what is going on under your TCB of interest. While not shown above, it is also possible to format activity for a specific SRB by filtering on its WEB address. (For an SRB entry, the WEB address is reported under the "WU-addr" field of the SYSTRACE report: IP SYSTRACE WEB(X'zzzzzzzz') TI(LO) .)

System trace tables are getting larger and larger, and some systems are running with many CPs. This can make for a very large system trace table. You can narrow down how big a timeframe is formatted by specifying a START and STOP time.

Advanced debugging sometimes requires focusing on activity on a single CPU. The SYSTRACE CPU parameter gives you this filtering capability. Be aware that if you don't specify "ALL", it defaults to showing you just the entries for the current address space on the specified CP, not all activity on the specified CP.

Sometimes it is helpful to eliminate z/IIPs from the picture. Specifying CPUTYPE(STANDARD) will show just entries from the general (standard) CPs. While not demonstrated above, you can also specify CPUTYPE(ZIIP) to look at just z/IIP workload.

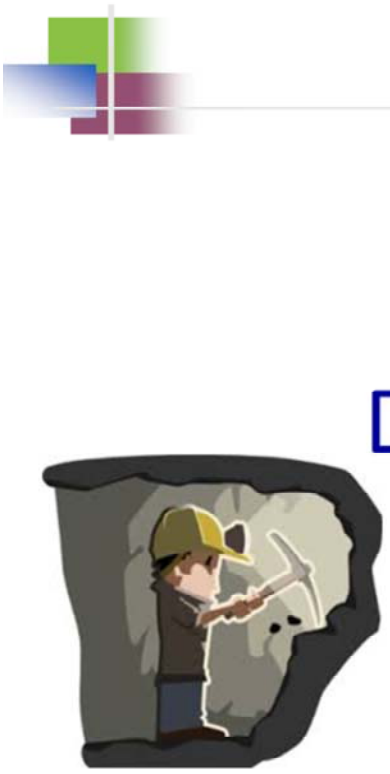
SYSTRACE STATUS TIME(LO)

```
TRACE services are available
TRACE is active
ST=(On,0001M,00006M) AS=On BR=Off EX=On MO=Off

The earliest timestamp in SYSTRACE is from CPU 0004: 12/14/2016 15:24:31.308528
The latest   timestamp in SYSTRACE is from CPU 0005: 12/14/2016 15:25:40.376142
TRACE data reporting from all CPUs starts at         12/14/2016 15:25:40.201513 (CPU 0000)
TRACE data reporting from all CPUs ends at           12/14/2016 15:25:40.375447 (CPU 0001)

------+------+------+------+-----------------------------+--------------------------
 CPU  | Type | Pol  | Park | SYSTRACE First Local Time   | SYSTRACE Last Local Time
------+------+------+------+-----------------------------+--------------------------
 0000 | CP   | High | No   |  12/14/2016 15:25:40.201513 | 12/14/2016 15:25:40.376141
 0001 | CP   | High | No   |  12/14/2016 15:25:40.135381 | 12/14/2016 15:25:40.375447
 0002 | CP   | Med  | No   |  12/14/2016 15:25:40.062828 | 12/14/2016 15:25:40.375504
 0003 | CP   | Med  | No   |  12/14/2016 15:25:39.999764 | 12/14/2016 15:25:40.375524
 0004 | CP   | Low  | Yes  |  12/14/2016 15:24:31.308528 | 12/14/2016 15:25:40.376076
 0005 | zIIP | High | No   |  12/14/2016 15:25:38.633241 | 12/14/2016 15:25:40.376142
```


Useful information:

- Size of trace buffer
- What range of time has all CPs represented (reporting)
- CPU numbers, CPU types, and CPU polarity

SYSTRACE STATUS can be handy for getting a quick peek at much time is covered by each CP in the system trace, and in what time range all CPs are represented. When looking at a portion of the trace where not all time ranges are represented, a “-” (hyphen) will appear between the CPU number and the ASID, making it clear that the picture you are looking at may be incomplete due to missing trace entries from unrepresented CPs. This is an effect of different workloads on different CPs generating different volumes of records and therefore filling up at different rates. CPs running work that is writing fewer entries will have a longer, older history in its trace buffer.



Digging Even Deeper

LIST Nuggets

- Useful (but not well-known) operands of LIST command
  - INSTR (I)
    - Interpret data as Assembler instructions (with machine code)
    - Useful for interpreting failing instruction when debugging an abend, or for looking at OEM code for which a listing is unavailable
  - DISPLAY
    - Display the key of a page of storage, including whether the storage is fetch-protected
    - Particularly useful when debugging an ABEND0C4 PIC4 (Protection Exception) due to a mismatch between the storage key and the PSW key

The IPCS LIST command with the INSTR (I) option will actually interpret data as an Assembler instruction.

There is a storage key associated with each page of storage. The IPCS LIST command with the DISPLAY option will display the key of page on which the specified address resides. It also displays the fetch-protect status of the page.

LIST examples

LIST 01010020. ASID(X'0136') LENGTH(X'12') I

```
LIST 01010020. ASID(X'0136') LENGTH(X'12') INSTRUCTION
01010020 | 4770 5010       | BC     X'7',X'10'(,R5)
01010024 | 58F0 631C       | L      R15,X'31C'(,R6)
01010028 | 05EF            | BALR   R14,R15
0101002A | 5870 A0C0       | L      R7,X'C0'(,R10)
0101002E | 4190 7010       | LA     R9,X'10'(,R7)
```

LIST 97F5E0 ASID(1) DISPLAY

```
LIST 0097F5E0 CPU(X'00) ASID(X'0001') LENGTH(4) AREA
CPU(X'00) ASID(X'0001') ADDRESS(0097F5E0) KEY(06) ABSOLUTE(01CDB5E0)
0097F5E0.  00000000
```

The **first nibble** is the **key**.
The **first bit of the second nibble** is the **fetch-protect** bit.

Key 0

Not Fetch-Protected
x'6'= b'0110'

In the output from the IPCS LIST command with DISPLAY, a two-nibble value is presented: KEY(xx). The first nibble is the key. The second nibble is a sequence of bits, the first of which identifies whether or not the page is fetch protected. The next two bits are the Reference and Change bits respectively. The last bit is undefined.

FIND Nuggets

- Useful (but not-well-known) operands of FIND command
  - BOUNDARY(n) or BDY(n)
    - Search for a string at a specific boundary n
    - Can be used to speed up FIND

```
Examples:
BDY(2) - half-word boundary
BDY(4) - word boundary
BDY(8) – double-word boundary
```

  - BDY (bdy,index)
    - Divide storage into strings 'bdy' bytes long. For each string, FIND will compare search argument with storage once starting with 'index' into each string.
    - Default for index is 1 (or offset 0) if it is not coded
  - MASK
    - Storage is ANDed with MASK before compare
    - Useful when search argument not in contiguous storage, or not in multiples of byte

The FIND command can be used to find a string of data in storage. The BOUNDARY keyword can speed up FIND if you know that the search argument starts on a certain boundary. Note that INDEX of 1 means offset 0.

The MASK keyword of FIND can be used when the search argument is not contiguous, or not in a multiple of bytes. This is very useful when debugging an overlay problem, when you need to find out where the data causing the overlay came from.

FIND examples

BDY (32,5) - search a table containing 32-byte entries. For each entry, only compare the search argument with the second word.

```
ASID(X'0001') ADDRESS(04553000.) STORAGE ------------------- FOUND AT 04553024.
Command ===> F x'05235D70' BDY(32,5) NOB                     SCROLL ===> CSR
04553000   047749C0   04F77380   FF02C010   00000000   | ...{.7....{..... |
04553010   00000000   1F00001E   02EBD000   80009300   | ..........}...1. |
04553020   04B61EC0   05235D70   FF02C010   00000000   | ...{..}...{..... |
04553030   00000000   1F000053   007D1000   80007300   | .........'...... |
```

Search for a word ending with x'F14'.

```
ASID(X'0001') ADDRESS(F45FB0.) STORAGE --------------------- FOUND AT F45FC4.
Command ===> FIND X'00000F14' MASK(X'00000FFF') BDY(4)       SCROLL ===> CSR
00F45FB0   0140015F   008E015F   0000B138   812B2A94   | . .^...^....a..m |
00F45FC0   81234598   85261F14   0250C0E8   0250C0E8   | a..q.....&{Y.&{Y |
00F45FD0   02575458   012B39FE   00000008   025758C0   | ..............{ |
```


In the first example, NOB (NOBREAK) is used to request IPCS to keep searching in storage and not stop at a page break (when storage for a page is not available).

RUNCHAIN

- Process a chain of control blocks in order to:
  - look for a specific one on the chain
  - determine the length of the chain
  - confirm an error scenario
- Basic Parameters
  - ADDRESS
  - LINK
  - NULL
  - NAME
  - AMASK
  - MASK
  - EXEC
  - CHAIN
  - SORTBY

The RUNCHAIN subcommand is useful when you need to run a chain of control blocks. In most cases we need to find out the length of the chain. In some cases we need to find an element on the chain. The basic parameters of RUNCHAIN will be discussed in the following pages.

RUNCHAIN common parameters

- ADDRESS( ) - address of first control block in chain (use ASID parm if needed).
- LINK (x:y) - beginning and ending offset of forward pointer in control block.
  If x is used alone, forward pointer is 4 bytes long.
- NULL( ) - value in LINK that indicates end of chain (default is 0).
  Useful when last element of chain points back to first element or trailer.
- AMASK( ) - value to AND with LINK to form next pointer. Default is x'00FFFFFF' if chain originates below the line and x'7FFFFFFF' if chain originates above the line. So, if chain originates below the line, but elements can reside above the line, AMASK(x'7FFFFFFF') should be used.
- MASK( ) - value to AND with LINK before comparing to NULL.

Example: RUNC ADDR(FD4800.) LINK(4)

```
LIST FD4800. ASID(X'0001') LENGTH(X'04') AREA

LIST FB0100. ASID(X'0001') LENGTH(X'04') AREA

LIST FB7200. ASID(X'0001') LENGTH(X'04') AREA
```


ADDRESS, LINK and NULL are the common parameters of RUNCHAIN. Before you issue RUNCHAIN, remember to check on the offset of the forward pointer in each element and how the chain ends.

RUNCHAIN (NAME parameter)

- **NAME( )** - name for each control block. IPCS will add sequence number.
  - Useful if you need to identify position of element on chain.

Example: RUNC ADDR(FD4800.) LINK(4) NAME(CB)

```
CB001
LIST FD4800. ASID(X'0001') LENGTH(X'04') AREA

CB002
LIST FB0100. ASID(X'0001') LENGTH(X'04') AREA

CB003
LIST FB7200. ASID(X'0001') LENGTH(X'04') AREA

CB004
LIST FB7080. ASID(X'0001') LENGTH(X'04') AREA

CB005
LIST FAF280. ASID(X'0001') LENGTH(X'04') AREA
```


The NAME parameter is useful when you need to identify the position of any element on the chain.

Caution: When specifying a NAME, avoid choosing names used by IPCS (for example: ASCB), since that will cause values associated with those existing symbols to be replaced. LSYM can be used to see what NAMEs are currently in use.

RUNCHAIN (CHAIN parameter)

- **CHAIN( )** - specifies maximum number of blocks to be processed. Default is 999.
  - Scroll max to bottom of report. If number of blocks processed is 999, use larger CHAIN value to try to get to end of chain.

Example: RUNC ADDR(FD4800.) LINK(4) CHAIN(9999)

```
CB416
LIST F85100. ASID(X'0001') LENGTH(X'04') AREA

CB417
LIST F65A00. ASID(X'0001') LENGTH(X'04') AREA

CB418
LIST F97B80. ASID(X'0001') LENGTH(X'04') AREA
BLS18094I 418 blocks processed
```


The CHAIN parameter should be used if the chain is longer than 999 elements. If you increase CHAIN several times, and you still cannot see the end of the chain, the chain could be circular. Issue a FIND on the address of the first element of the chain. If you find it twice, the chain is circular. Another way is to scroll to the bottom of the output and issue a FIND PREV on the last element. If you find it twice, the chain is circular.

RUNCHAIN (EXEC parameter)

- **EXEC(( ))** - execute a CLIST, REXX EXEC, or IPCS command for each control block on the chain. Note the (( )).

- Useful if you need more information about each element.

Example: RUNC ADDR(FD4800.) LINK(4) EXEC((CBF X STR(ASCB)))

```
LIST FD4800. ASID(X'0001') LENGTH(X'04') AREA

 ASCB: 00FD4800
    +0000  ASCB..... ASCB      FWDP..... 00FB0100  BWDP..... 00000000
    +0020  JSSEQ.... 00000002  ASID..... 0001      SRMF..... 00
    +002C  TCBE..... 00000000  LDA...... 7F6C6E00  RSMF..... C0
    +0040  EJST..... 000002A0  1EB79E48            EWST..... D1B087EF
    +0058  UBET..... 00000000  TLCH..... 00000000  DUMP..... 008EB918
```


The EXEC parameter allows you to execute a CLIST, REXX EXEC, or IPCS subcommand for each element of the chain. This is useful if you need to obtain more information about each element.

BEAR - Breaking Event Address Register

- A 64-bit register containing the **address of the last instruction that causes a break in sequential execution**
  - For example, a branch or a LPSW instruction
- Content of BEAR stored in PSA by H/W when any program check occurs. This is propagated by z/OS to:
  - SDWA (available in ST FAILDATA or VERBX LOGDATA)
  - RTM2WA (available in SUMM FORMAT)
- Useful when diagnosing an ABEND0C1 (or other program check abend), especially one due to a wild branch.

BEAR is an enhancement in z/Architecture since the z9 machines (a while ago). Basically, the machine remembers the address of the last instruction that causes a break in sequential execution (or in common terms, a branch) and surfaces this information in a program interrupt. If this program interrupt is not resolvable, resulting in an error condition, z/OS will save the contents of BEAR in the SDWA or the RTM2WA.

Finding BEAR in a dump

ST FAILDATA or VERBX LOGDATA

```
TIME OF ERROR INFORMATION

  PSW: 07040001 80000000 00000000 0AF8D286
  INSTRUCTION LENGTH: 06   INTERRUPT CODE: 0010
  FAILING INSTRUCTION TEXT: 17884280 3015E320 60180004
  TRANSLATION EXCEPTION ADDRESS: 00000008_004FF800

  BREAKING EVENT ADDRESS: 00000000_0AF8C754
```

SUMM FORMAT

```
RTM2WA: 7FFAFE10
   +0000  ID....... RTM2      ADDR..... 7FFAFE10  SPID..... FF    LGTH..... 0011F0
   +0014  VRBC..... 009FD550  ASC...... 00F882A0  CCF...... 84    CC....... 0C1000
.....
....   Lines omitted here
....
   +06C8  TRNE..... 00000000  072FF800
   +06D0  BEA...... 00000000  0AF8B552
   +06D8  PSW1..... 07040001  80000000  00000000  0AF8D180
```


You can also find BEAR in an RTM2WA (if available) under the failing TCB in SUMM FORMAT.

System Trace aids in diagnosing a performance problem

- **SYSTRACE PERFDATA** for a performance view from system trace entries
- **SPIN** entries in SYSTRACE report

SYSTRACE PERFDATA

- **Provides a "performance breakdown" on system trace data**
  - SRB, TCB, and total CPU time used per address space in trace
  - Breakdown of SRB usage by address space
  - Breakdown of TCB usage by address space
  - LOCAL and CMS lock suspensions
  - I/O times
- See
  - https://share.confex.com/share/119/webprogram/Session11721.html for additional details

Like any debugging tool used to diagnose performance issues, SYSTRACE PERFDATA is not to be considered a silver bullet. However, in cases where it would be helpful to know what jobs are using the most CP during a small snapshot in time, SYSTRACE PERFDATA may help. Users must have some awareness of the system's normal CPU usage in order to effectively analyze SYSTRACE PERFDATA output.

SYSTRACE PERFDATA excerpts

CPU Summary

CPU#	Went from	To	Seconds	SRB Time	TCB Time	Idle Time	CPU Overhead
00	16:02:24.880423	16:02:28.200541	3.320117	1.370858	1.943048	0.000000	0.040452
01	16:02:24.880437	16:02:26.878804	1.998367	1.371296	0.621700	0.000000	0.036751
			5.318485	2.742155	2.564749	0.000000	0.077204

This slide demonstrates how SYSTRACE PERFDATA breaks down how much time each CP spent executing SRB mode work, executing TCB mode work, or idle.

SYSTRACE PERFDATA excerpts

Summary for each address space in system trace:

```
Found 53 address spaces in SYSTRACE.
Found 145 SRB and SSRB PSWs in SYSTRACE.

CPU breakdown by ASID:

ASID Jobname    SRB Time     TCB Time    Total Time
---- -------- ------------ ------------ ------------
006B ABCDEFGH     0.001601     0.012151     0.013752
0060 CICSRGNA     0.005703     0.202266     0.207969
0078 SOMEBAT1     0.000994     0.118045     0.119040
0080 SOMEBAT6     0.006793     0.057610     0.064404
0064 PBLTEST      0.008914     0.045282     0.054196
0085 CICSRGNB     0.000725     0.074175     0.074901
0063 MYJOB        0.001004     0.112320     0.113325
          . . . . .
```

SYSTRACE PERFDATA excerpts

SRB and TCB breakdowns for each address space:

SRB breakdown by ASID:

ASID	Jobname	SRB PSW	# of SRBs	Time
006B	ABCDEFGH	070C0000 81174100	97	0.001601

ASID	Jobname	SRB PSW	# of SRBs	Time
0060	CICSRGNA	070C0000 813C1348	273	0.003070
0060	CICSRGNA	070C0000 8102E876	20	0.000046
0060	CICSRGNA	070C0000 886D0656	7	0.000058
				0.003174

TCB breakdown by ASID:

ASID	Jobname	TCB Adr	# of DSPs	Time
006B	ABCDEFGH	009EB748	97	0.012151

ASID	Jobname	TCB Adr	# of DSPs	Time
0060	CICSRGNA	009FA4E0	733	0.201798
0060	CICSRGNA	009C0E88	20	0.000319
0060	CICSRGNA	009FAB20	22	0.000148
				0.202266

SYSTRACE PERFDATA excerpts

Lock Events:

Lock	ASID	TCB/SRB	Type	PSW Adr	Suspended at	Resumed at	Suspend Time
CEDQ	0009	009F6638	TCB	9276C4D8	16:02:25.532319	16:02:25.532422	0.000102
CEDQ	0001	0099D0E8	TCB	9276C4D8	16:02:25.532572	16:02:25.532610	0.000037
						2 suspends	0.000139

SSCH to I/O times:

Device	SSCH Issued	I/O Occurred	Duration
2080	16:02:24.880476	16:02:24.880680	0.000203
2080	16:02:24.881121	16:02:24.881347	0.000225
2080	16:02:24.882362	16:02:24.882596	0.000233
2080	16:02:24.882970	16:02:24.883410	0.000440
2080	16:02:24.883555	16:02:24.883907	0.000352
			0.001453

Events for 2080 :		5
Quickest I/O	:	0.000203
Slowest I/O	:	0.000440
Total	:	0.001453
Average	:	0.000290

SYSTRACE SPIN entries

- **SPIN LKX/S** entry written after spinning for lock for 1 second
- **SPIN LKX/P** entry written at end of spin

```
PR   ASID WU-ADDR- IDENT  CD/D PSW-- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD Time Local
                                              UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE PSALOCAL PASD SASD Date-12/14/2
0001 0005 02391380  SPIN LKX/S       8174C4B8 01000000 00004002 021C3C04  88000000 00000000 0003 0005 15:24:31.308
                                              00004000 000002F8 00FF15C2  00000000

0001 0005 02391380  SPIN LKX/P       8174C4B8 02482AA2 00004002 021C3C04  88000000 00000000 0003 0005 15:24:32.561
                                              00004000 000002F8 00FF15C2  00000000
```

- **Unique 4:** Lock being requested (map via PSACLHS in MVS Data Areas)
- See: MVS Diagnosis: Tools and Service Aids
  AND
  https://share.confex.com/share/119/webprogram/Session11721.html
  for more information about SPIN system trace entries, including other types of events which can generate spin entries.

The Unique-2 field on a SPIN/LKX entry indicates the CP where the lock holder is executing. Ignore the '4' as this is actually a flag bit that has been set. In the example on this slide, if you ignore the '4', you see that the lock holder is executing on CP2.

Captured dumps and traces

- When a system crashes with a flurry of abends, there may be some extra nuggets in the SADump
  - **COPYCAPD**
    - **Checks a SADump for SVC dumps** that were captured as the system crashed
    - Normally this captured storage would be freed after the dump was written, but the system crashed before this happened
  - **SYSTRACE TTCH(LIST) TI(LO)**
    - **Checks a SADump for system trace table snapshots** taken as work units entered RTM
    - Normally these captured system trace snapshots are freed when the work units exit RTM, but the system crashed before this happened

COPYCAPD [SADumps only]

COPYCAPD

```
Number  Time stamp           Title
------  -------------------  ------------------------------------------------------------

     1  01/24/2017 07:19:20  COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPXPRTRM+????,

1 captured dump processed
```


This shows that there was one captured dump in this SADump. On the next slide we see how to extract it into a dump data set which can then be initialized just like any SVC dump. The success of this initialization and the usability of the dump will depend on how much got written before the system crash occurred.

COPYCAPD [SADumps only]

COPYCAPD 1 OUTDSN('PATTYL.CAPDUMP')

```
Number  Time stamp           Title
------  -------------------  ------------------------------------------------------------

     1  01/24/2017 07:19:20  COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPXPRTRM+????,

 1 captured dump processed

Open volume 1 of DSNAME('PATTYL.CAPDUMP')
BLS18169I Dump 1 is being copied
DATA SPACE ASID(X'0005') DSPNAME(00055SDU) STOKEN(X'8000C60000001817')
DATA SPACE ASID(X'0005') DSPNAME(00053SDU) STOKEN(X'8000990000001813')
DATA SPACE ASID(X'0005') DSPNAME(00054SDU) STOKEN(X'80009A0000001814')
IEA11005I 1 section was not accessible.
IEA11005I 4 SUMDUMP pages were not accessible.
BLS18170I 193,440 records 804,710,400 bytes, copied
```

Dump data set does not have to be pre-allocated but sometimes it helps avoid ABENDx37 errors.

SYSTRACE TTCH(LIST) [SADumps only]

SYSTRACE TTCH(LIST) TI(LO)

```
 TTCH       ASID  TCB        TIME
*7F615000   00D3  009C4168   01/24/2017 07:19:20.068905
 7EFFC000   0006  00000000   01/24/2017 07:19:18.709728
 7F0B4000   0001  00000000   01/24/2017 07:19:16.998678
 7F16C000   0001  00000000   01/24/2017 07:19:04.631369
```

To format the 2nd trace buffer in the list above

- SYSTRACE TTCH(X'7EFFC000') ALL TI(LO)

NOTE: An "*" beside an entry indicates this is not a full-sized system trace buffer snapshot, meaning it does not have as long of a history (just 64K) as a full-sized buffer snapshot. In order not to deplete system resources, RTM caps the number of full-sized system trace buffer snapshots that can be in flight simultaneously.

Other SYSTRACE filtering options may be used in conjunction with TTCH(X'yyyyyyyy') as well.

New Nuggets

ESTA & ESTR system trace entries (R2.1)

```
0000 0239 008FEA30 *SVC      D 00000000_0152E3C2  008A0CA0 0089F000 0089EFB8
                                 07042000 80000000
 - -   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0000 0239 008FEA30 *RCVY  ESTA          00E98336  0089D8D0 00000000 008A0C80
                                                  00000000 008FF610
0000 0239 008FEA30  SVC      C 00000000_09471434  09471AFD 00E98336 0089D8D0
                                 07041000 80000000
```

SDWA

RTM2 is entered via SVC D

RTM2 traces RCVY entry for **ESTAE/ARR getting control**

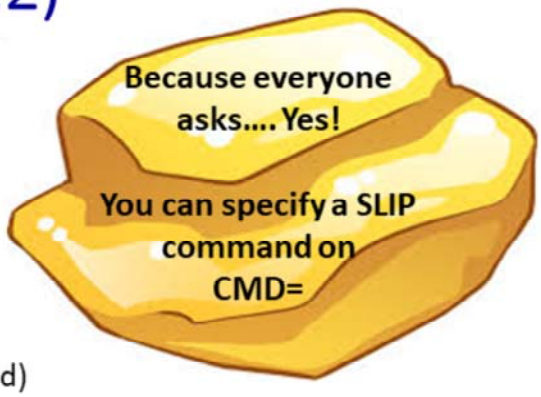
RTM2 passes control to the ESTAE/ARR via SVC C SYNCH

ESTA & ESTR system trace entries (R2.1)

```
0000 0239 008FEA30  SVCR      C 00000000_09471434  0000000C 00000010 0089D8D0
                                07041000 80000000
 - -   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0000 0239 008FEA30 *RCVY  ESTR 00000000 00E9866C  00E98336 008FF610

 - -   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0000 0239 008FEA30  SVCR     11 00000000_00E9866C  008A0CA0 0089F000 008A0C80
                                07042000 00000000
0000 0239 008FEA30  DSP         00000000_00E9866C  00000000 0089F000 008A0C80
                                07042000 00000000
```

- SVCR C indicates ESTAE-type recovery routine completing.
- **RCVY ESTR** indicates retry requested and shows the **retry address,** as well as the **recovery routine address**
- The SVCR xx entry is a "retry effect", not an actual SVC return. It should be ignored.
- The **DSP** entry indicates code getting dispatched at the retry point.

SLIP A=CMD (R2.2)



- You can now request a command via a SLIP!
  - Allowed for both PER and non-PER SLIP traps
  - Up to 8 commands may be issued
  - May be combined with other SLIP actions
    - Cannot be combined with WAIT, IGNORE
    - Cannot be used as a REMOTE action (use ROUTE instead)
- Example: SLIP SET,COMP=0C4,A=CMD,CMD='D T',END
- See SLIP section of MVS System Commands for more details.

Appendix

VSM: Identify subpool and key
RUNCHAIN: Sorting output

VERBX VSMDATA 'SUM ASID(dddd)'

- Provides VSM (virtual storage manager) information, for both global (common) and local (private) storage, for specified ASID
  - For just global storage: VERBX VSMDATA 'SUM NOASID'
  - For just local storage: VERBX VSMDATA 'SUM NOG ASID(dddd)'
- Includes storage map (not as detailed as IGVVSMIN), summary of subpool sizes
- Formats VSM data structures that describe allocated and free areas of storage
- Designed for sortability
- Can be used to identify what subpool/key a piece of storage is in

Note that dddd is the *decimal* ASID number.

Using VERBX VSMDATA to identify subpool and key

- Assumption: You have an address for which you want to determine the subpool number/key
  - IP VERBX VSMDATA 'SUM'
  - SORT 115 122
  - FIND SP/K
  - Scroll or search until you find the VSM control block representing the range of storage where your address lives
- VSM control block notes
  - Each VSM control block has an ADDR/SIZE format describing storage area it represents

If the VSM control block name contains an "F", it represents free storage; otherwise it represents subpool-assigned storage

A VSM control block represents either an allocated (subpool-assigned) or free area of storage. The ADDR field indicates the starting address of the area, and the SIZE field indicates the length of the area.

Sorted VSMDATA report (excerpt)

What subpool/key does address **7F0100** live in?

```
DQE:  Addr 007E9000 Size     2000                                    TCB: 007FF6C8 SP/K: 237/ 1
                                  FQE: Addr 007E9000 Size     AB0    TCB: 007FF6C8 SP/K: 237/ 1
DQE:  Addr 007EB000 Size     1000                                    TCB: 007F8680 SP/K: 230/ 0
DQE:  Addr 007EC000 Size     2000                                    TCB: 007FF6C8 SP/K: 237/ 1
DQE:  Addr 007EB000 Size     1000                                    TCB: 007F8680 SP/K: 230/ 0
                                  FQE: Addr 007EC000 Size     F98    TCB: 007FF6C8 SP/K: 237/ 1
DQE:  Addr 007EE000 Size     2000                                    TCB: 007FF6C8 SP/K: 237/ 1
                                  FQE: Addr 007EF000 Size     FD8    TCB: 007FF6C8 SP/K: 237/ 1
DQE:  Addr 007F0000 Size     1000                                    TCB: 007FDD40 SP/K: 230/ 9
                                  FQE: Addr 007F0000 Size     F70    TCB: 007FDD40 SP/K: 230/ 9
```

- Address 7F0100 falls within the Addr/Size range of this FQE so it is **free**.
- However, the FQE is associated with a DQE with an Addr/Size range that encompasses the range of the FQE.
- **Conclusion:** The storage at 7F0100 is free; however, it resides on a page that contains GETMAINed storage assigned to **SP230 key9**

A VSM control block represents either an allocated (subpool-assigned) or free area of storage. The ADDR field indicates the starting address of the area, and the SIZE field indicates the length of the area.

VSM control blocks that represent allocated (that is, subpool-assigned) storage are:

- DQEs represent allocated pages of storage in RGN, SWA, high private, and CSA
- AQATs represent allocated pages of storage in LSQA and SQA

VSM control blocks that represent free storage are:

- FQEs represent free fragments (less than a page) of storage in RGN, SWA, high private, and CSA
- DFEs represent free fragments of storage in LSQA and SQA
- FBQEs represent free pages of storage. There is a pool of free pages for CSA and for each private storage area.

RUNCHAIN (SORTBY parameter)

- **SORTBY**(sortkey ASCENDING|DESCENDING)
  - Run and sort chain of elements in a specific order.
  - sortkey - can be range of offsets in elements, or attribute of elements.
  - ASCENDING | DESCENDING - sort order. Default is ascending

Example: RUNC ADDR(FC7380.) LINK(4) SORTBY(42:43) EXEC((LIST (x+24, X+B0?, X+AC?) LEN(8)))

```
Address space control block
LIST FC3D00. ASID(X'0001') LENGTH(X'0180') STRUCTURE(Ascb)

Address space control block
LIST FC3D00. ASID(X'0001') POSITION(X'+24') LENGTH(X'08') STRUCTURE(Ascb)
+00024 00FC3D24. 00690000 000100E2                                  |.......S        |

LIST FA6418. ASID(X'0001') LENGTH(X'08') AREA
00FA6418. C2D7E7C1 E2404040                                        |BPXAS           |

LIST FA6BD4. ASID(X'0001') LENGTH(X'08') AREA
00FA6BD4. E2E8E2D3 D6C7C440                                        |SYSLOGD         |
```


The SORTBY parameter allows you to sort the elements of the chain in a certain order.

The example is valuable when trying to understand a high CPU problem where the "hung" job is not getting CPU, and you want to check the relative dispatching priorities between a "hung" job and the jobs that are shown executing in SYSTRACE ALL output.

Questions?

Session 21585
Mining z/OS Debugging Nuggets