SHARE
EDUCATE • NETWORK • INFLUENCE

IPCS for Beginners

Ed Jaffe
Phoenix Software International
Tuesday, February 24, 2026
SHARE 145
Orlando, Florida

Session Overview

- IPCS is an amazing tool used by the best z/OS diagnosticians in the world
- It can be used to process:
  - SVC dumps
  - IEATDUMP transaction dumps
  - SYSMDUMP dumps
  - CTRACE component traces
  - GTF generalized traces
- It can be used to display data found in-memory on your system
- It can be used to display and manage DAE on your system
- This presentation was originally envisioned as a way to help programmers, already familiar with text-only SYSUDUMP and SYSABEND dumps, level-up their skills using IPCS.

Session Overview (*continued…*)

- This is supposed to be a BEGINNER'S class
- I could have drawn a line at mentioning more advanced topics like traces, ACTIVE memory, DAE, etc. and saved them for an intermediate class
- Instead, I decided to do an in-depth introduction of IPCS basic features coupled with a high-level overview of the more advanced topics so you're aware they are there
- I can always give additional talks with in-depth discussion on those topics
  - Example: I've already done the MVS System Trace as a stand-alone presentation
- Don't feel bad if you're unfamiliar with some of the things I present today
- Remember, it was also new to me at one time, and I am an autodidact
- Simply keep my slides around for future reference when needed

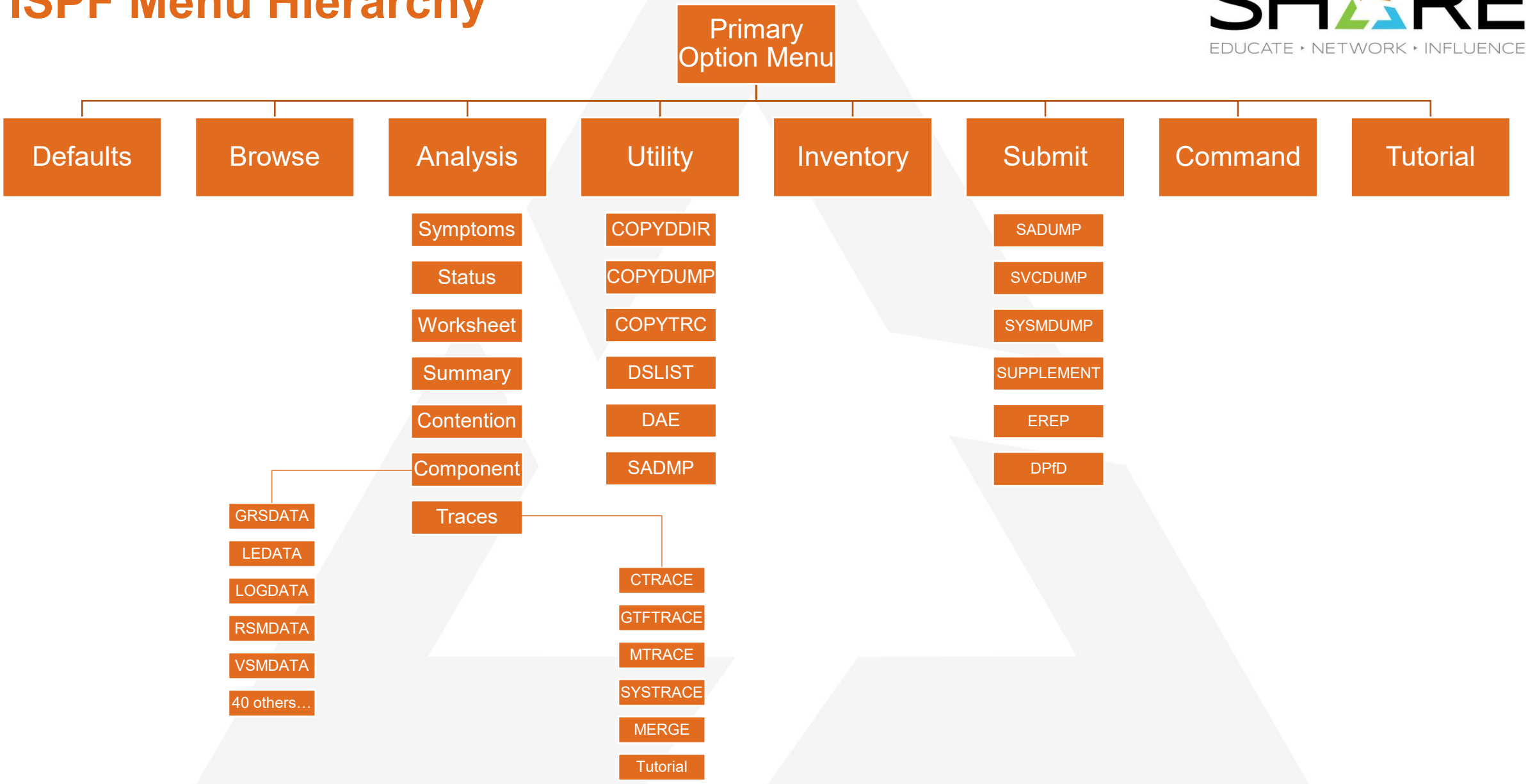
Interfaces to IPCS

- You can externally invoke IPCS:
  - As a command processor from TSO/E READY
  - As an ISPF dialog
    - **All of my examples will be using the ISPF dialog**
  - Recommend 3270 terminal 141 columns or wider (I use 142 and sometimes 160)
  - You can invoke IPCS in batch:
    - To copy or initialize dumps so you don't have to wait for them in real-time
    - You can issue all IPCS commands that are available interactively
  - You can invoke IPCS from a CLIST, REXX, or Python
- There is also an API available to commands issued from within IPCS
  - REXX and CLIST
- We now have an open-source project called Ambitus delivering pyIPCS for Python
  - https://github.com/ambitus/pyIPCS
  - I will demonstrate this tomorrow in my z/OS 3.2 User Experience session

How the ISPF Interface Enhances IPCS

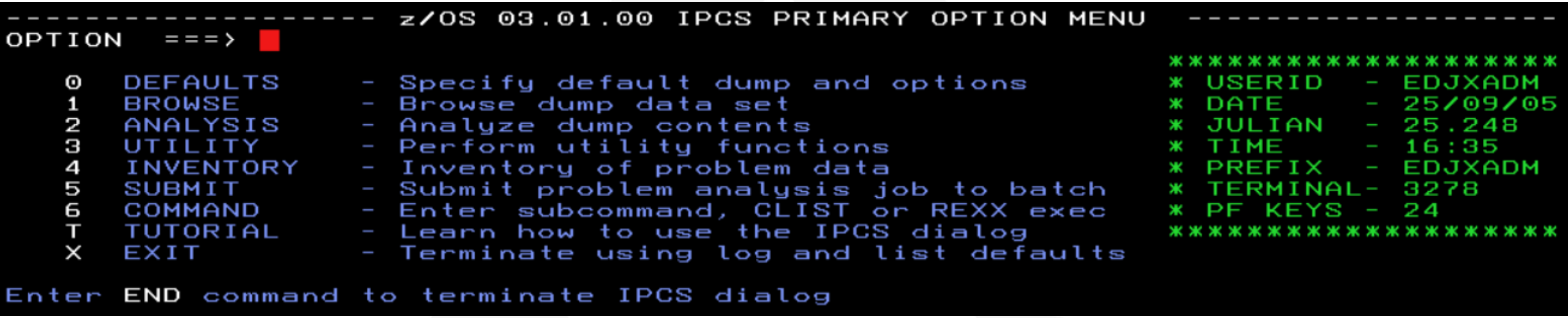
- At its heart, IPCS is a 100% command-driven product. Technically, it is a TSO/E Command Processor, which supports many subcommands. You can run original IPCS from TSO/E READY without ISPF.
- The ISPF interface is primarily a wrapper around the IPCS subcommands, which:
  - Helps to make remembering the commands easier
  - Provides better processing of the reports generated by those commands
- Exceptions are DAE and BROWSE. Those functions are not IPCS subcommands and have no counterparts in traditional IPCS.
- Another advantage to the ISPF interface is that you can have IPCS open on as many split screens as you wish.
- There is a menu navigation tree. If you make a leaf choice that results in an IPCS subcommand being issued, the output is put into a scrollable IPCS Dump Display Reporter Dialog.
- You can also use the **IPCS** primary command to issue an IPCS command directly from any panel in IPCS, and have the result placed into the IPCS Dump Display Reporter Dialog. This behavior is similar to the **TSO** primary command in ISPF.

ISPF Menu Hierarchy



IPCS Primary Option Menu Under ISPF

- The primary option menu was obviously modeled after the original SPF



- On the next several slides I will briefly show the major options available

Option 0 – Defaults

- All of these options and others can be set with the **IPCS SETDEF** command
- I pretty much never use this screen, but I am aware that some people like to set the dump source here
- It's also handy for double-checking your current settings e.g, **MACHINE**

```
------------------------- IPCS Default Values -------------------------------
Command ===>

   You may change any of the defaults listed below.  The defaults shown before
   any changes are LOCAL.  Change scope to GLOBAL to display global defaults.

   Scope    ==> LOCAL    (LOCAL, GLOBAL, or BOTH)

   If you change the Source default, IPCS will display the current default
   Address Space for the new source and will ignore any data entered in
   the Address Space field.

   Source   ==> DSNAME('SYS3.DUMP.D250904.T112635.C4CONDOR.S00001')
   Address Space    ==> ASID(X'00A6')
   Message Routing  ==> NOPRINT TERMINAL NOPDS
   Message Control  ==> NOCONFIRM VERIFY FLAG(SERIOUS)
   Display Content  ==> MACHINE REMARK REQUEST NOSTORAGE SYMBOL NOALIGN

Press ENTER to update defaults.

Use the END command to exit without an update.
```

Option 1 – Browse

- Pressing **<Enter>** on the Entry Panel here takes you to the pointer list

```
------------------------------------------------------ IPCS - ENTRY PANEL ------------------------------------------------------
Command ==>

 CURRENT DEFAULTS:
  Source ==> DSNAME('SYS3.DUMP.D250904.T112635.C4CONDOR.S00001')
  Address space ==> ASID(X'00A6')

 OVERRIDE DEFAULTS:                                                          (defaults used for blank fields)
  Source ==> DSNAME('SYS3.DUMP.D250904.T112635.C4CONDOR.S00001')
  Address space ==> ASID(X'00A6')
  Password       ==>

 POINTER:
  Address        ==>                                                         (blank to display pointer stack)
  Remark         ==>                                                                      (optional text)
```

- You can delete (**D**), edit (**E**), format (**F**), insert (**I**), repeat (**R**) pointers
- You can add new pointers to the end of the list with the **STACK** command
- You can browse the memory contents at a pointer by issuing the **SELECT** command or by issuing the **S** line command against the pointer
- Pointers keep their pointer numbers until a **RENUM** command is issued

```
DSNAME('SYS3.DUMP.D250904.T112635.C4CONDOR.S00001') POINTERS  ------------------------------------------------------------------
Command ===>                                                                                  SCROLL ===> CSR
ASID(X'00A6') is the default address space
PTR   Address                 Address space                                 Data type
00001 00.                     ASID(X'00A6')                                 AREA
      Remarks:
********************************************************* END OF POINTER STACK **********************************************
```

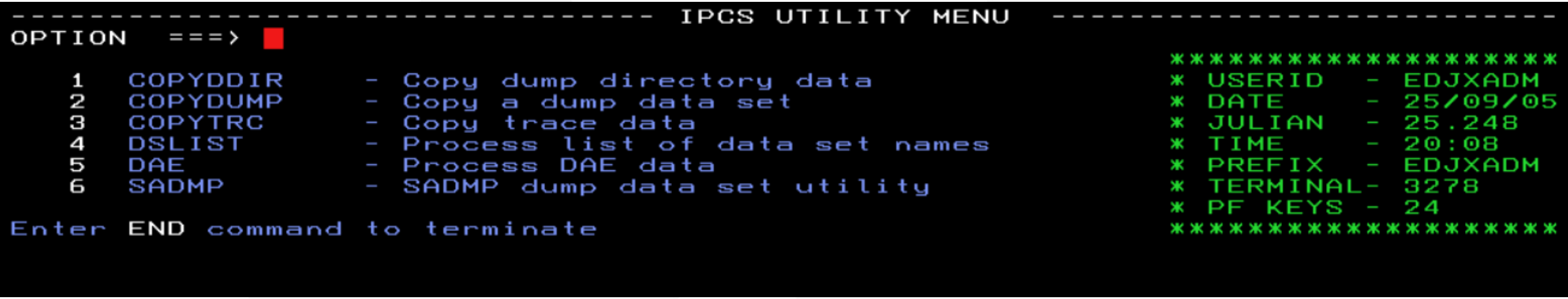
Option 2 – Analysis

- Sub-options on this display
  - **SYMPTOMS** – symptom strings DAE uses to suppress dumps
  - **STATUS** – a panel driven way of invoking the **IPCS STATUS** command
  - **WORKSHEET** – CPU masks and other CPU status information
  - **SUMMARY** – a panel driven way of invoking the **IPCS SUMMARY** command
  - **CONTENTION** – Some GRS contention information
  - **COMPONENT** – useful verb exits written by the z/OS developers
  - **TRACES** – trace formatters for CTRACE, GTFTRACE, MTRACE and SYSTRACE

```
---------------------- IPCS MVS ANALYSIS OF DUMP CONTENTS -------------------------
OPTION  ===>

To display information, specify the corresponding option number.

    1   SYMPTOMS     - Symptoms                                  **********************
    2   STATUS       - System environment summary                * USERID   - EDJXADM
    3   WORKSHEET    - System environment worksheet              * DATE     - 25/09/05
    4   SUMMARY      - Address spaces and tasks                  * JULIAN   - 25.248
    5   CONTENTION   - Resource contention                       * TIME     - 17:43
    6   COMPONENT    - MVS component data                        * PREFIX   - EDJXADM
    7   TRACES       - Trace formatting                          * TERMINAL- 3278
                                                                 * PF KEYS  - 24
                                                                 **********************

Enter END command to terminate MVS dump analysis.
```

Option 3 – Utility

- Sub-options on this display
  - **COPYDDIR** – helps you copy DDIR data sets
  - **COPYDUMP** – helps you copy dump data sets with all kinds of options
  - **COPYTRC** – helps you copy trace data sets with all kinds of options
  - **DSLIST** – like ISPF 3.4 with the option to add dump/trace files to the inventory
    - This is where I typically add dump files to my inventory
  - **DAE** – Formats DAE activity on live system and allows you to request dumps be taken
  - **SADMP** – help you manage your stand-alone dump volumes



Option 4 – Inventory

- Shows all dump and trace data sets in your inventory
- Useful line commands (there are many more)
  - **BR** - Activates the BROWSE option of the IPCS dialog for that source
  - **DD** - Deletes description of the source and, optionally, the source data set
  - **LD** – List dumped storage summary (good place to find your dataspace names)
  - **SD** - Establishes the source as both the local and global IPCS default (SETDEF)

```
IPCS INVENTORY - EDJXADM.DDIR ------------------------------------------------------------
Command ===>

AC Dump Source                                                                  Status
__ DSNAME('SYS3.DUMP.D250904.T112635.C4CONDOR.S00001') . . . . . . . . . CLOSED
   Title=PHOENIX TP MONITOR ERROR RECOVERY ABEND
   Psym=RIDS/EJESML#L RIDS/#UNKNOWN AB/S00C4 VALU/HF004C083 REGS/0C050 REGS/03050 PRCS/00000010
********************************************************** END OF IPCS INVENTORY **************
```

Option 5 – Submit

- Provides several batch job utilities
- To illustrate how infrequently I use these batch job options, note that the job card is left over from when I demonstrated how to use DPfD in preparation for my **z/OS 2.4 User Experience** presentations
  - That was going on seven years ago!!!

```
---------------------- IPCS MVS DUMP BATCH JOB OPTION MENU ----------------------
OPTION  ===>

    1   SADUMP      - Prepare stand alone dump for analysis
    2   SVCDUMP     - Prepare SVC dump for analysis
    3   SYSMDUMP    - Prepare SYSMDUMP for analysis
    4   SUPPLEMENT  - Perform supplementary dump analysis
    5   EREP        - Process software data using EREP
    6   DPfD        - Data Privacy for Diagnostics

JOB STATEMENT INFORMATION:   (Verify before proceeding)

   ===> //DPFD      JOB 1,JAFFE,CLASS=A,MSGCLASS=T,TIME=NOLIMIT
   ===>
   ===>
   ===>
   ===>
   ===>

Enter END to terminate batch job processing.
```

```
**********************
* USERID   - EDJXADM
* DATE     - 25/09/05
* JULIAN   - 25.248
* TIME     - 19:43
* PREFIX   - EDJXADM
* TERMINAL- 3278
* PF KEYS  - 24
**********************
```

Option 6 – Command

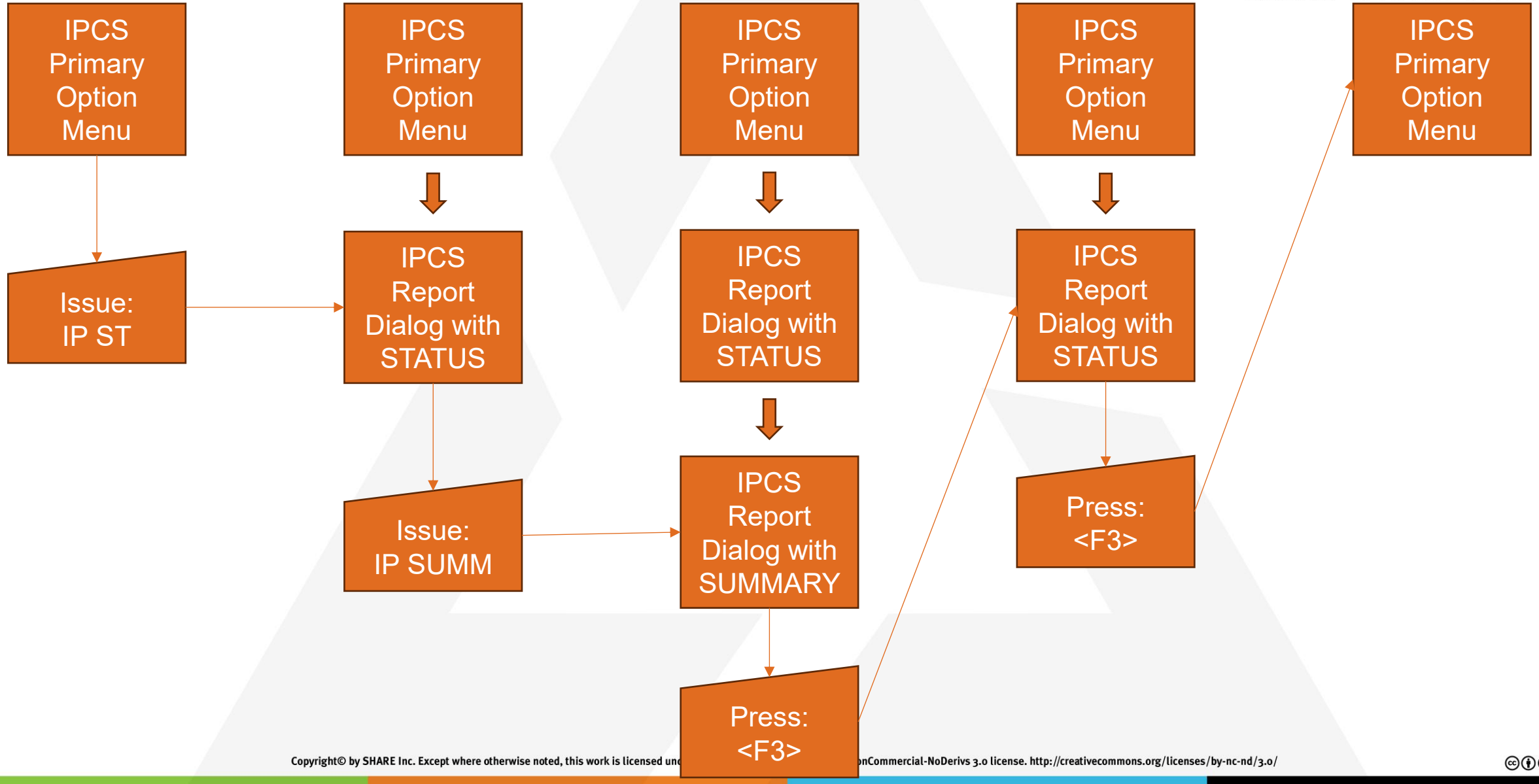
- Provides a 234-character command line, and reminders of the names of some common IPCS commands

```
------------------------- IPCS Subcommand Entry --------------------------------
Enter a free-form IPCS subcommand or a CLIST or REXX exec invocation below:

===>

------------------------ IPCS Subcommands and Abbreviations ---------------------
ADDDUMP                | DROPDUMP, DROPD   | LISTDUMP, LDMP    | RENUM,     REN
ANALYZE                | DROPMAP,  DROPM   | LISTMAP,  LMAP    | RUNCHAIN,  RUNC
ARCHECK                | DROPSYM,  DROPS   | LISTSYM,  LSYM    | SCAN
ASCBEXIT, ASCBX        | EPTRACE           | LISTUCB,  LISTU   | SELECT
ASMCHECK, ASMK         | EQUATE,   EQU, EQ | LITERAL           | SETDEF,    SETD
CBFORMAT, CBF          | FIND,     F       | LPAMAP            | STACK
CBSTAT                 | FINDMOD,  FMOD    | MERGE             | STATUS,    ST
CLOSE                  | FINDUCB,  FINDU   | NAME              | SUMMARY,   SUMM
COPYDDIR               | GTFTRACE, GTF     | NAMETOKN          | SYSTRACE
COPYDUMP               | INTEGER           | NOTE,     N       | TCBEXIT,   TCBX
COPYTRC                | IPCS HELP, H      | OPEN              | VERBEXIT,  VERBX
CTRACE                 | LIST,     L       | PROFILE,  PROF    | WHERE,     W
```

Popular IPCS Commands for Beginners

- These commands must be prefixed with IPCS (or just IP for short)
- **IP**CS **L**IST data-description
- **IP**CS **W**HERE data-description
- **IP**CS **EQ**U symname|**X** data-description|**X**
  - Special symbol X always points to the last-displayed address
  - Symbol names can be up to 31 chars in length and cannot start with a number
  - Character set is A-Z, 0-9, $ (X'5B'), # (X'7B'), @ (X'7C')
- **IP**CS **LISTSYM**|**LSYM**
- **IP**CS **ST**ATUS
- **IP**CS **SUMM**ARY [**FO**RMAT] [**AL**L|**C**URRENT|**ER**ROR] **R**EGS
- **IP**CS **SYSTRACE** in some cases

IPCS Primary Command Result Navigation



IPCS Dump Display Reporter Dialog

- There is a line command field on the left of every line. It accepts line commands up to six characters in length.
- The ending attribute of the line command field is cleverly placed at the first blank on the line following the end of the six-character field. This ensures the line command field is not noticeable and reduces the data width of each line by only one character.
- Here is the output of the **IP ST** command

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------- Line 0 Cols 1 140
Command ===> _                                                                                                   SCROLL ===> CSR
 ********************************************************* TOP OF DATA ***************************************************************

                                MVS Diagnostic Worksheet

 Dump Title: JOBNAME EDJX2     STEPNAME $IKJTEST$IKJTEST SYSTEM 0C4

 CPU Model 8562 Version 00 Serial no. 01A798 Address 0000
 Date: 09/10/2025      Time: 16:57:13.320414 Local

 Original dump dataset: EDJX2.EDJX2.J0657620.D0000009.?
```

- Here's what it looks like when I overtype the line with **X** characters

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------- Line 0 Cols 1 140
Command ===> _                                                                                                   SCROLL ===> CSR
 ********************************************************* TOP OF DATA ***************************************************************
XXXXXX XXXXXX
XXXXXX XXXXXX
XXXXXX XXXXXX                   MVS Diagnostic Worksheet
XXXXXX XXXXXX
XXXXXXXXXXXX XXXXXXE EDJX2     STEPNAME $IKJTEST$IKJTEST SYSTEM 0C4
XXXXXX XXXXXX
XXXXXXXXXX XXXXXXersion 00 Serial no. 01A798 Address 0000
XXXXXX XXXXXX2025      Time: 16:57:13.320414 Local
XXXXXX XXXXXX
XXXXXXXXX XXXXXXataset: EDJX2.EDJX2.J0657620.D0000009.?
```

IPCS Dump Display Reporter Dialog Line Commands

- **D** – delete line(s)
  - Supports a single **D**, **Dn** where '**n**' is the number to be deleted, or a **DD** / **DD** block commands
  - Be careful with this; there is no **UNDO** command
- **X** – exclude line(s)
  - Supports a single **X**, **Xn** where '**n**' is the number to be deleted, or a **XX** / **XX** block commands
- **F** – show first line(s) of an excluded block
  - Supports a single **F** or **Fn** where '**n**' is the number to be shown
- **L** – show last line(s) of an excluded block
  - Supports a single **L** or **Ln** where '**n**' is the number to be shown
- **S** – show excluded line(s) with the leftmost indentation
  - Supports a single **S** or **Sn** where '**n**' is the number to be shown
  - If several lines are indented equally, the first line or first '**n**' lines are shown

IPCS Dump Display Reporter Dialog Primary Commands

- Dump Display Reporter Dialog primary commands are not preceded by **IPCS**
- **CBF**ormat and **W**here commands work just like their IPCS counterparts
- **F**ind – find lines containing string data; redisplayed if excluded
  - Search for ASCII, EBCDIC, hex, picture specifications (like ISPF), addresses, signed binary values
  - Data relationship to the specified value can be EQ, =, NE, ^=, LT, <, GT, >, LE, <= GE, or >=
  - Refer to the documentation for additional capabilities
- **L**ocate – scroll to a specific line in the report
- **MORE** – allows you to resume processing after a virtual memory shortage in your IPCS session
- **REPORT|RPT** subcmd – process some or all of the report as desired
  - A few popular subcmd values shown, others found in the documentation:
    - **B**rowse [start[:end]] – place report into ISPF BROWSE
    - **IPCSPRNT** [start[:end]] – copy report to IPCSPRNT DD
    - **V**iew [start[:end]] – place report into ISPF VIEW
- **RESET** – clears pending line commands and redisplays all excluded lines
- **SORT**– sort the display
  - Specify sort order SORT A or SORT D (A is default)
  - Specify sort key column ranges e.g., SORT 10 20 40 50
  - Combine order and key column ranges e.g., SORT 10 20 A 40 50 D
  - Sort excluded or non-excluded lines e.g., SORT X or SORT NX

Managing Your Dump Inventory

- If you do not yet have a DDIR data set, you can create one by issuing the following TSO/E command (I usually do it from READY):
  - BLSCDDIR VOLUME(volser)
- Add dumps and/or traces to your inventory using the **A** line command from IPCS option 3.4 (DSList)
- Remove dumps and traces from your inventory using the **DD** line command from IPCS option 4 (Inventory)

```
------------------- CONFIRM IPCS DROPDUMP and DELETE ----------------------------
Command ===> _

  You have requested that IPCS delete information related to a data set:

       DSNAME    ==> 'SYS3.DUMP.D250904.T112635.C4CONDOR.S00001'

  Please ensure that both actions shown reflect your wishes.

  1.   Dump directory records referring to the data set may be erased.

       RECORDS   ==> ALL             (ALL, ANALYSIS, TRANSLATION, or NONE)

  2.   The data set, itself, may be deleted.

       DELETE    ==> NO              (YES or NO)

Press ENTER to continue.

Use the END command to exit without deletion.
```

Data Descriptions

- Data descriptions are complex; I am just showing a subset of options.
- The general format is:
  - address [**AS**ID(asid) [**DSP**NAME(dspname)]] [**L**ENGTH(length)] [attribute]
- I use **IP SETDEF LENGTH(8192)** to ensure I never have to specify length.
- The address can be a previously-defined symbol, a relative address, a literal address, a general-purpose register, a floating-point register, or an indirect address.
- When needed, I specify ASID as a hex value i.e., X'02FB' unless it's *MASTER* which is AS(1).
- To see the all the data space names in your dump, issue the **LD** line command against the dump from IPCS Option 4 (Inventory) and then **F DSP** command.
- The only attributes I will cover are **I**NSTRUCTION and **STR**UCTURE(strname)

Indirect Addressing

- An indirect address is a symbolic, relative, or literal address, or a general-purpose register followed by one or more percent signs (%), question marks (?), or exclamation points (!)
- Once IPCS accepts a pointer, it retrieves the contents of that pointer from the dump and interprets it thusly:
  - If the address is followed by **%** then the pointer is interpreted as a 24-bit address
  - If the address is followed by **?** then the pointer is interpreted as a 31-bit address
  - If the address is followed by **!** then the pointer is interpreted as a 64-bit address
- The following admonishment was added to the manual with z/Architecture:

  "It is not recommended that you use registers in indirect addresses. For compatibility with TSO/E TEST, general-purpose registers will be accepted in an address expression, but the resolution of the expression by IPCS will generally prove unsatisfactory."
- But why?
- IPCS treats a register symbol just like any other pointer. It used to point to a fullword in the dump but, in z/Architecture, registers are now eight bytes wide. Therefore, a specification like **10R?** now picks up the high half of the register instead of the low half. **10R!** works if the register contains a "clean" 64-bit address. Otherwise, **10R+4?** is what you should use.

Indirect Addressing Using a Register Specification

- Suppose you have the following registers:

```
General purpose register values
   0-1   00000000_11FF9059   00000000_00000000
   2-3   FFFFFFFF_00000060   FFFFFFFF_00012040
   4-5   FFFFFFFF_862D502C   FFFFFFFF_00AC7940
   6-7   FFFFFFFF_00000000   FFFFFFFF_00012000
   8-9   FFFFFFFF_00AC7940   FFFFFFFF_00FD5A28
  10-11  FFFFFFFF_862E3030   FFFFFFFF_062E402F
  12-13  FFFFFFFF_11FF8060   00000000_11FF9000
  14-15  00000000_F5000000   00000000_00000000
```

- R0, R1, R13, R14 and R15 are all “clean” 64-bit addresses
- **IP L 13R!** will work just fine, and so will **IP L 13R+4?**
- If we wish to use R3 as a 31-bit pointer, we must specify **IP L 3R+4?**
- **THE RULE**: always think of the general-purpose registers as symbols that point to eight-byte areas in the dump containing the register contents

Sample Address Specifications

- IP L CVT
- IP L A
- IP L A.
- IP L 0A
- IP L 345678?+14?
- IP L 10R
- IP L 10R! (if “clean”)
- IP L 10R!+280
- IP L 10R!+280!

IP L 10R!+280!+C

IP L 10R!+280!+C?

IP L MYCB

IP L MYCB+160?+40

IP L ASCB109

IP L 234_56789ABC

IP L 3456789A I

IP L MYCB AS(X’43’)

IP L MYCB AS(X’43’) DSP(MYDSP)

IP W 10R!

IP W 3456789A

CBF CVT STR(CVT)

CBF ASCB109 STR(ASCB)

TIP: For longer specifications that would be tiresome to continually type, such as ASID and DSPNAME, I recommend you create pointers in the BROWSE function. The STACK command is useful for this.

A Few Command Output Samples

SHARE

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------------
Command ===> IP L CVT                                                                                                          SCROLL ===> CSR
********************************************************* TOP OF DATA ************************************************************************

CVT - Communications Vector Table
LIST FD5A28. ASID(X'0001') POSITION(X'-28') LENGTH(X'0528') STRUCTURE(Cvt)
ASID(X'0001') ADDRESS(FD5A00.) KEY(00) ABSOLUTE(7FB92A00.)
-00028 00FD5A00. E2D7F74B F34BF140 C8C2C2F7 F7C5F040  40404040 40404040 40404040 40404040 |SP7.3.1 HBB77E0                    |
-00008 00FD5A20. 00008562 F0F3F840 00000218 00FEC9EC  00FD8264 00FD5818 00000000 00FF4D50 |..e.038 .....I...b.........(&|
+00018 00FD5A40. 00FEFE2E 00FE07DA 00FE0564 01E28BA8  810CC680 00FEAB30 0266B070 00FDF940 |.............S.ya.F.........9|
+00038 00FD5A60. 0125247F 00FD4EF8 00F45000 00FF7E00  00FECA12 00000000 0A0307FE 00FD826C |...".+8.4&...=.............b%|
```

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------------
Command ===> IP CBF CVT STR(CVT)                                                                                               SCROLL ===> CSR
********************************************************* TOP OF DATA ************************************************************************

 CVT: 00FD5A28
    -0028  PRODN.... SP7.3.1   PRODI.... HBB77E0   VERID....                                  MDL..... 8562     RELNO.... 038
    +0000  TCBP..... 00000218  0EF00.... 00FEC9EC  LINK..... 00FD8264  AUSCB.... 00FD5818  BUF..... 00000000 XAPG..... 00FF4D50
    +0018  0VL00.... 00FEFE2E  PCNVT.... 00FE07DA  PRLTV.... 00FE0564  LLCB..... 01E28BA8  LLTRM.... 810CC680 XTLER.... 00FEAB30
    +0030  SYSAD.... 0266B070  BTERM.... 00FDF940  DATE..... 0125247F  MSLT..... 00FD4EF8  ZDTAB.... 00F45000 XITP..... 00FF7E00
    +0048  0EF01.... 00FECA12  VSS..... 0000      VPSM..... 0000      EXIT..... 0A03      BRET..... 07FE     SVDCB.... 00FD826C
    +0058  TPC...... 00FD5840  FLGC0.... 00        ICPID.... 0002      CVT...... 40C3E5E3  CUCB..... 00FD8988 QTE00.... 00FECA46
```

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------------
Command ===> IP L 10R                                                                                                          SCROLL ===> CSR
********************************************************* TOP OF DATA ************************************************************************

REGGEN - General purpose registers
LIST 0E7C. HEADER POSITION(X'+50') LENGTH(X'08') ENTRY(X'+0A') STRUCTURE(Reggen64)
HEADER ADDRESS(0ECC.)
+00050 00000ECC. 00000000 7F2F7000                                                |....".. .                                                    |
********************************************************* END OF DATA ************************************************************************
```

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------------
Command ===> IP L 10R!                                                                                                         SCROLL ===> CSR
********************************************************* TOP OF DATA ************************************************************************

LIST 7F2F7000. ASID(X'00A6') LENGTH(X'2000') AREA
ASID(X'00A6') ADDRESS(7F2F7000.) KEY(80) ABSOLUTE(99C4F000.)
7F2F7000. C5D1C5E2 C5D4D940 065000F3 22E5F6D9  F5D4F04B C4E5D9F3 60D8C5D1 F6F5F0F0 |EJESEMR .&.3.V6R5M0.DVR3-QEJ6500|
7F2F7020. 60F0F761 F2F761F2 F560F1F3 4BF2F040  01020400 01030200 0000001B 00000000 |-07/27/25-13.20 .............|
7F2F7040. 00000000 7F30C100 00000000 7F30C000  00000000 7F30E000 00000000 9329BE1A |....".A.....".{.....".\.....l...|
7F2F7060. 12014064 120184A0 7F2F7000 93212000  00000000 00000000 7F2D8000 00010000 |.. ...d.".../.l.........".......|
```

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------------
Command ===> IP L 10R!+54?                                                                                                     SCROLL ===> CSR
********************************************************* TOP OF DATA ************************************************************************

LIST 7F30E000. ASID(X'00A6') LENGTH(X'2000') AREA
ASID(X'00A6') ADDRESS(7F30E000.) KEY(80) ABSOLUTE(1FC8A000.)
7F30E000. C5D1C5E2 40C7E4C1 D9C440E2 E3C1D9E3  00000000 00000000 00000000 00000000 |EJES GUARD START.............|
7F30E020 LENGTH(X'0FC0')==>All bytes contain X'00'
7F30EFE0. 00000000 00000000 00000000 00000000  C5D1C5E2 40C7E4C1 D9C440C5 D5C44040 |.............EJES GUARD END  |
ASID(X'00A6') ADDRESS(7F30F000.) KEY(??)
```

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------------
Command ===> IP W 11R!                                                                                                         SCROLL ===> CSR
********************************************************* TOP OF DATA ************************************************************************
     ASID(X'00A6') 13212000. AREA(Sp252Key00R31Exy#dq7F40Eb08)+00 IN EXTENDED PRIVATE
     ASID(X'00A6') 13212000. EJESSUBS+00 IN EXTENDED PRIVATE
********************************************************* END OF DATA ************************************************************************
```

MVS Dump Reading Landmine #1

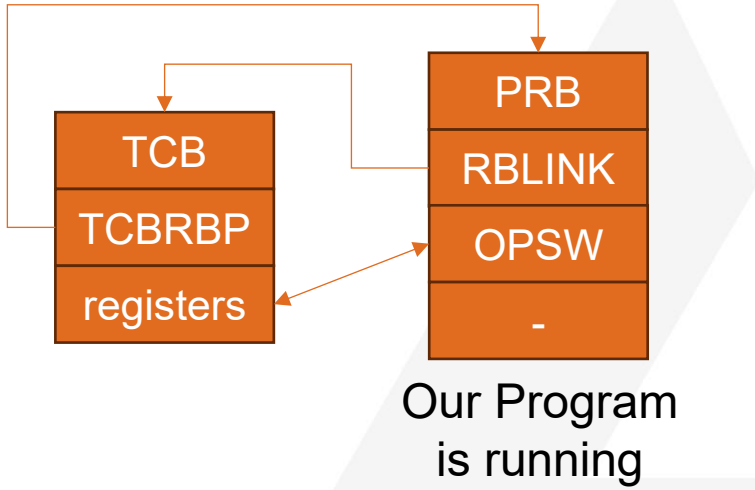
- PSW and failing instruction
  - PSW usually points *after* the failing instruction, not *at* the failing instruction
  - However, it points *at* the failing instruction for a translation exception
    - PIC 0010 (segment), 0011 (page), 002B (region 1st), 003A (region 2nd), 003B (region 3rd)
    - All of these get lumped into 0C4 abend along with the original PIC 0004 (protection)
- Summary dump displays 6 bytes of instruction data on either side of PSW

```
Time of Error Information

  PSW: 47141000 80000000 00000000 1203E712
  Instruction length: 02    Interrupt code: 000D
  Failing instruction text: 00181610 0A0DD507 B8934000
  Breaking event address: 00000000_00000000

  AR/GR 0-1     00000036/00080000_80000000   00000000/00000000_80000FA0
  AR/GR 2-3     00000000/00000000_7F33F410   FFFFFFFF/00000000_44040000
  AR/GR 4-5     FFFFFFFF/00000000_80000000   FFFFFFFF/00000000_1205C406
  AR/GR 6-7     FFFFFFFF/00000000_00000041   FFFFFFFF/00000000_FFFFF000
  AR/GR 8-9     FFFFFFFF/00000000_7F443F00   00000000/00000000_7F5348E8
  AR/GR 10-11   00000000/00000000_7F532000   00000000/00000000_1203C000
  AR/GR 12-13   00000000/00000000_1207B8D0   00000000/00000000_7F444000
  AR/GR 14-15   00000000/00000000_9203E6F6   00000000/00000000_00000000

  Home ASID: 003B     Primary ASID: 003B     Secondary ASID: 003B
  PKM: 00C0           AX: 0000               EAX: 0000

  This Task's ASID/TCB: 003B/00AAB038

  RTM was entered because a task requested ABEND via SVC 13.
  The error occurred while an enabled RB was in control.
  No locks were held.
  No super bits were set.
```

PSW NSI points here

MVS Dump Reading Landmine #2

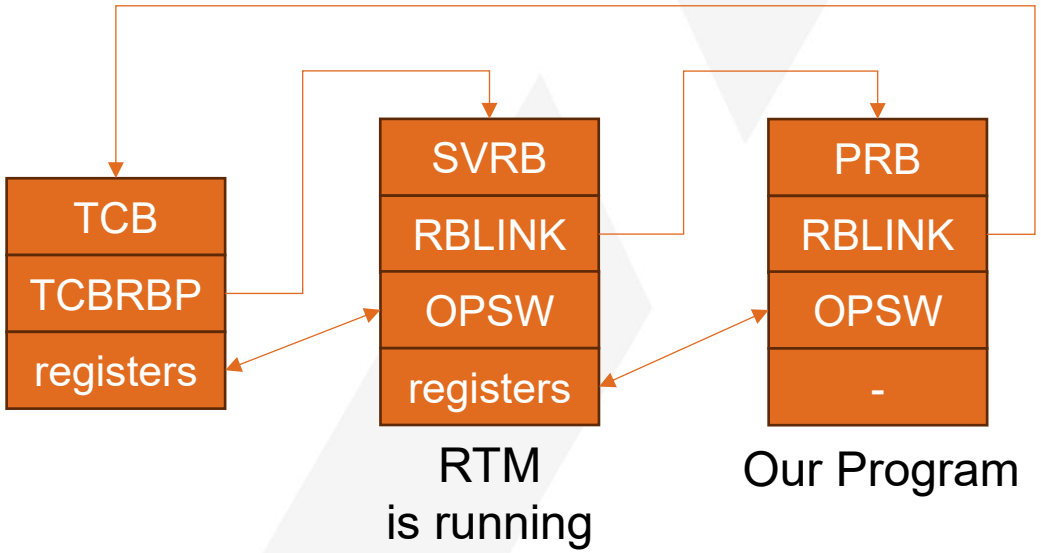
- Pairing up PSW and registers in Request Blocks
  - A somewhat confusing design
  - PSWs are stored in request blocks
  - Current registers are always stored in the TCB
  - This diagram illustrates normal running condition, with just one PRB



Our Program
is running

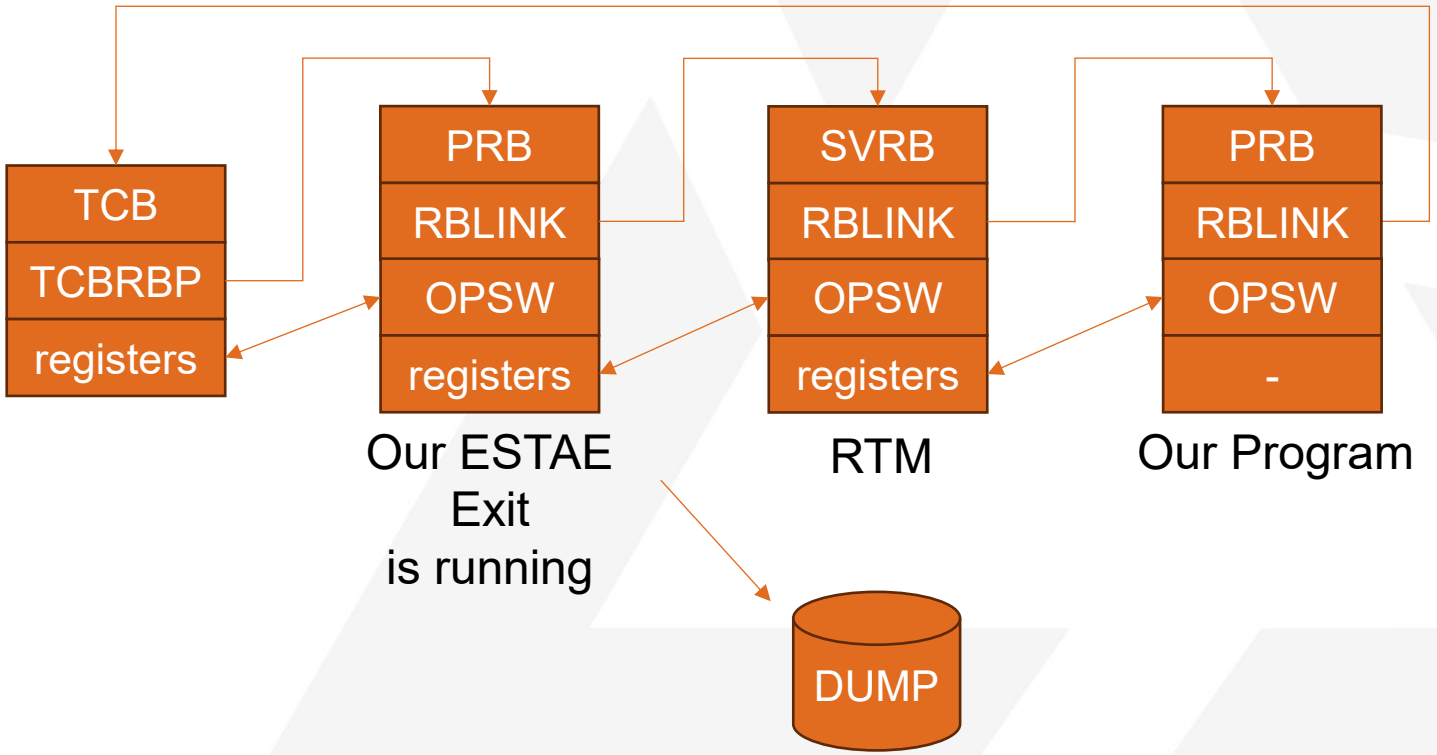
MVS Dump Reading Landmine #2 (*continued…*)

- Pairing up PSW and registers
  - Now consider how things look when the program abends
  - RTM gets control in an SVRB to handle recovery/termination



MVS Dump Reading Landmine #2 (*continued…*)

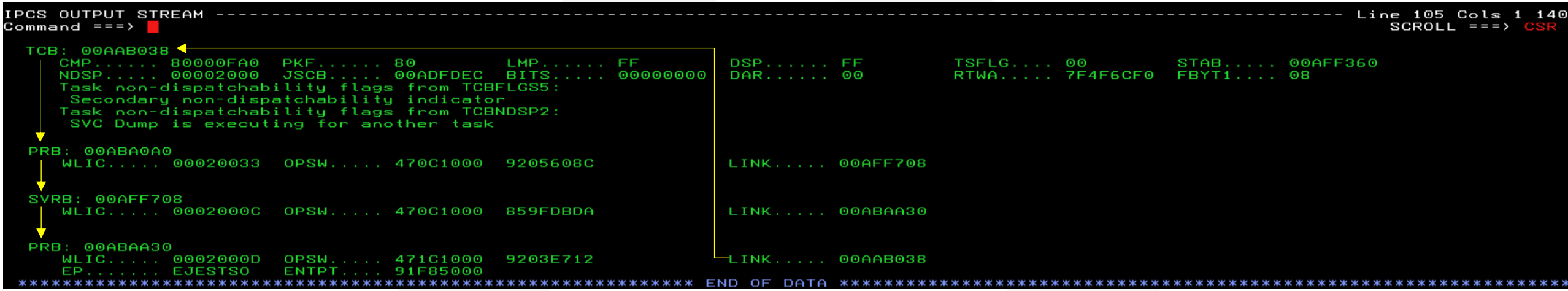
- Pairing up PSW and registers
  - RTM issues a SYNCH macro to call our ESTAE exit
  - Our ESTAE exit issues the SDUMPX macro to take the SVC dump



NOTE: If there is no ESTAE in effect, RTM will SYNCH to its own code under an SVRB to take the dump, so the arrangement of the RBs will be essentially the same.

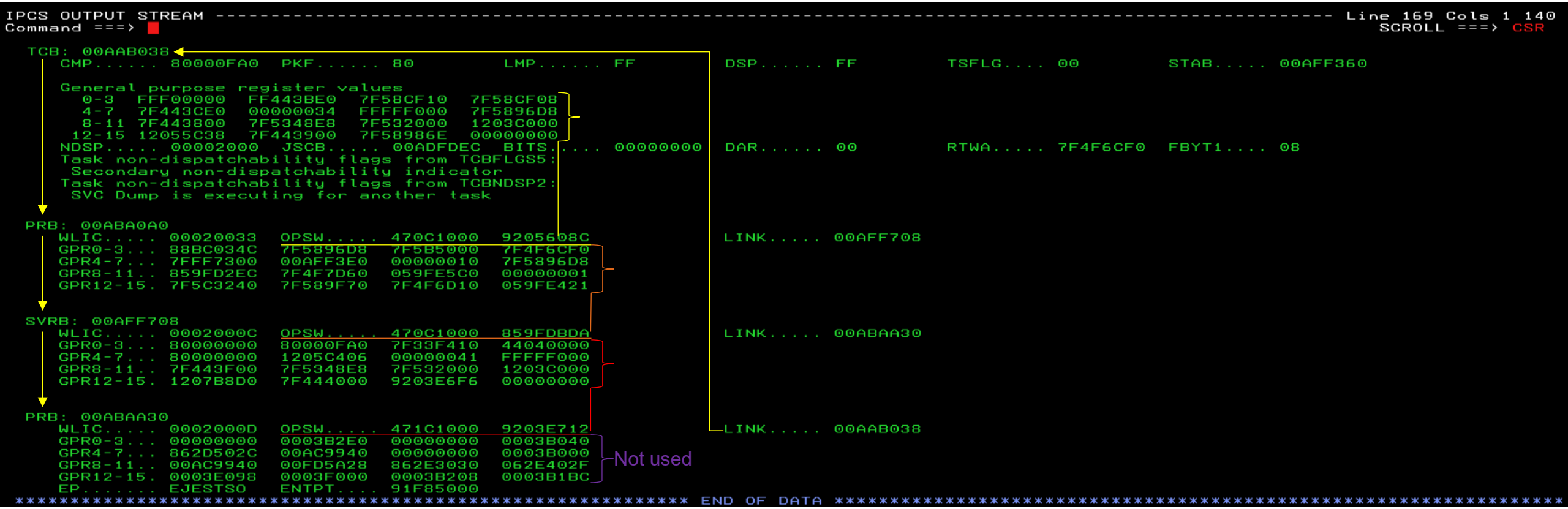
MVS Dump Reading Landmine #2 (*continued…*)

- TCB and RBs as represented in IPCS SUMMARY report
- Not confusing at all because registers are not formatted



MVS Dump Reading Landmine #2 (*continued…*)

- TCB and RBs as represented in IPCS SUMMARY REGS report



- **THE RULE:** Never pair up the OPSW and registers found in an RB
- The registers, associated with the OPSW in an RB, are in the control block (TCB or RB) that points to that RB

Software-Defined Save Areas (F1, F4, F5, F7, F8)

- Clients on z/OS V2R5 and higher can use **IP BLSXFnSA data-description** commands

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------
Command ===> IP BLSXF1SA 7FF18008                                                              SCROLL ===> CSR
********************************************************** TOP OF DATA ***********************************************
 F1SAFMT version 02/03/2023

 Formatting passed address 7FF18008 as a type F1 savearea.

 WRD# |  Dec  |  Hex  | Contents | Desc.                    | Save area contents points to a word containing:
   0  |  0000 |  0000 | 00000000 | Used by languages        | N/A
   1  |  0004 |  0004 | 00000000 | Previous save area addr. | 00000000
   2  |  0008 |  0008 | 7F475698 | Next save area addr.     | 00000000
   3  |  0012 |  000C | 814A648C | Register 14              | Unavail
   4  |  0016 |  0010 | 00000000 | Register 15              | 00000000000000000000000000000000 = |................|
   5  |  0020 |  0014 | 00000150 | Register 0               | 47044001800000000000000AD60C5E = |.. ..........O.;|
   6  |  0024 |  0018 | 7F46EEB0 | Register 1               | Unavail
   7  |  0028 |  001C | 00000505 | Register 2               | 00000000000000000000000000000000 = |................|
   8  |  0032 |  0020 | 0000E503 | Register 3               | 40404040405C40404040C39695A38189 = |      *    CONTAI|
   9  |  0036 |  0024 | 00AFD6A0 | Register 4               | 00AFFF980000000000000000DD3FD0 = |...Q...........}|
  10  |  0040 |  0028 | 8AD22742 | Register 5               | 5820704858E0202807FED7C1E3C3C840 = |.....\....PATCH |
  11  |  0044 |  002C | 00000000 | Register 6               | 00000000000000000000000000000000 = |................|
  12  |  0048 |  0030 | 00F90800 | Register 7               | C1E2C3C200F9030000F9120000000000 = |ASCB.9...9......|
  13  |  0052 |  0034 | 814A5DD2 | Register 8               | Unavail
  14  |  0056 |  0038 | 7F46EEB0 | Register 9               | Unavail
  15  |  0060 |  003C | 00000050 | Register 10              | 00000000000000000000000000000000 = |................|
  16  |  0064 |  0040 | 00001960 | Register 11              | Unavail
  17  |  0068 |  0044 | 7F46EF30 | Register 12              | Unavail
  18  |  0072 |  0048 |   Savearea end X48 = 72 bytes       | N/A


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

Hardware-Defined Linkage Stack Entries

- BAKR state entry created when BAKR instruction issued
  - TARG field contains an actual program address to which control is passed
- PC state entry created when PC instruction issued for a “stacking” PC
  - TARG field contains the PC number issued rather than the actual target address e.g,. 00000000 000030B for STORAGE OBTAIN

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------
Command ===>                                                                                  SCROLL ===> CSR
 LINKAGE STACK ENTRY  02  FROM TCB.  LSED: 7F603C28
  LSE: 7F603B08
     GENERAL PURPOSE REGISTER VALUES
     00-01.... 00000000  92005000  00000000  0003B2E0
     02-03.... 00000000  91F86000  00000000  000000FC
     04-05.... FFFFFFFF  862D502C  FFFFFFFF  00AC9940
     06-07.... FFFFFFFF  00000000  FFFFFFFF  0003B000
     08-09.... FFFFFFFF  00AC9940  FFFFFFFF  00FD5A28
     10-11.... FFFFFFFF  862E3030  FFFFFFFF  91F85000
     12-13.... FFFFFFFF  0003E098  00000000  0003F000
     14-15.... 00000000  91F8521E  00000000  92005000
     ACCESS REGISTER VALUES
     00-03.... 00000000  00000000  FFFFFFFF  FFFFFFFF
     04-07.... FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
     08-11.... FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
     12-15.... FFFFFFFF  FFFFFFFF  00000000  05794FF2
     PKM...... 00C0      SASN..... 003B      SINS..... 00000002  EAX..... 0000      PASN..... 003B      PINS..... 00000002
     PSW...... 47140000  80000000            PSWE..... 00000000  11F8521E
     TARG..... 00000000  92005004            MSTA..... 00000000  00000000
     TYPE..... 8C
      BAKR STATE ENTRY
     RFS...... 0378      NES...... 0000
```

After BAKR R14,0 PSW contains original R14 to be used as return address when the PR is issued

```
LIST 12005000. ASID(X'003B') LENGTH(X'1000') INSTRUCTION
ASID(X'003B') ADDRESS(12005000.) KEY(00) ABSOLUTE(1B556000.)
12005000 | B240 00E0      | BAKR    R14,0
12005004 | C0C0 0000 17FE | LARL    R12,*+X'2FFC'
```

Component Data Analysis – Option 2.6

- 45 functions altogether
- Only a handful of them are appropriate to most diagnosticians
  - **GRSDATA** – to see the status of ENQs held and/or contention if necessary
  - **LEDATA** – to help diagnose abends in LE-enabled HLASM and HLLs
  - **LOGDATA** – summary of other recent abends that have occurred on the system
  - **RSMDATA** – information about 64-bit virtual and real memory use
  - **VSMDATA** – information about 24- and 31-bit virtual memory use

```
------------------- IPCS MVS DUMP COMPONENT DATA ANALYSIS ----------------------
OPTION ===>                                              SCROLL ===> CSR

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      Abstract
_  ALCWAIT   Allocation wait summary
_  AOMDATA   AOM analysis
_  APPCDATA  APPC/MVS Data Analysis
_  ASCHDATA  APPC/MVS Scheduler Data Analysis
_  ASMCHECK  Auxiliary storage paging activity
_  ASMDATA   ASM control block analysis
_  AVMDATA   AVM control block analysis
_  BHIDATA   BHI control block analysis
_  COMCHECK  Operator communications data
_  COUPLE    XCF Coupling analysis
_  CSFDATA   ICSF control block analysis
```

Let's Debug a Problem Together

- This program copies the EXEC parm to an area of memory using the MVCL instruction
- The programmer accidentally sets the target length equal to the source length instead of the length of the target area
  - LR R1,R15 should have been
  - LHI R1,L'ParmData
- This allows a memory overlay when the parm length exceeds eight characters
- [Aside: The most observant of you might have already noticed I forgot to set the eyecatcher… ☹]

```
******************************** Top of Data ********************************
EXAMPLE  RSECT ,                       Define reentrant CSECT
EXAMPLE  AMODE 31                      Indicate AMODE(31)
EXAMPLE  RMODE 31                      Indicate RMODE(31)
EXAMPLE  LOCTR ,                       Define LOCTR for the code

CONST            LOCTR ,               Define LOCTR for constants
Constants        DS 0D                 Align to doubleword

* Module Work Area
WkArea           DSECT ,               Dynamic work area
SaveArea         DS  18F               18 fullword standard save area
EyeCatch         DS  CL8               Eyecatcher (WKAREA)
ParmData         DS  CL8               Data from EXEC Card
ImportantPtr     DS  A                 Some important pointer
                 DS 0D                 Align to doubleword
WkArea_Len       EQU *-WkArea          Length of work area

         ASMDREG ,                     Define register equates
         SYSSTATE ARCHLVL=OSREL        Set architectural level

* Mainline Code
EXAMPLE  LOCTR ,                       Resume LOCTR for the code
         BAKR  R14,0                   Save the registers
         LARL  R12,Constants           Point R12 to constants
         USING Constants,R12           <Synchronize base register
         LHI   R2,WkArea_Len           Get work area length
         STORAGE OBTAIN,               Acquire work area                +
               LENGTH=(2)              ..
         LR    R13,R1                  Place address in R13
         USING WkArea,R13              <Synchronize base register
         LR    R0,R1                   Zero the work area
         LR    R1,R2                   ..
         XR    R15,R15                 ..
         MVCL  R0,R14                  ..

* Get Parm Data from EXEC Card
         EREG  R1,R1                   Get original R1
         L     R1,0(,R1)               Point to parm len+data
         LA    R14,2(,R1)              Get parm address
         LH    R15,0(,R1)              Get parm length
         LA    R0,ParmData             Get target address
         LR    R1,R15                  Get target length
         ICM   R15,B'1000',=C' '       Set blank pad
         MVCL  R0,R14                  Copy the parm data

         XR    R15,R15                 Set retcode = 0
         L     R14,ImportantPtr        Load important pointer
         CLC   0(8,R14),=C'ALL_GOOD'   Check important eyecatcher
         JE    AOK                     Jump if AOK
         LHI   R15,4                   Set retcode = 4
AOK      DC    0H
         PR    ,                       Return to caller

* Place Literal Pool at End of Constants LOCTR
CONST            LOCTR ,               Resume LOCTR for constants
                 LTORG ,               Define literal pool

         END   EXAMPLE
****************************** Bottom of Data ******************************
```

Testing the Program With Different PARM Values

- All is well until the parm length exceeds eight, then BOOM!

```
 READY
call *(example)
 READY
call *(example) 'test'
 READY
call *(example) 'test1234'
 READY
call *(example) 'test12345'
 IKJ56641I EXAMPLE  ENDED DUE TO ERROR+
 READY
?
 IKJ56641I SYSTEM ABEND CODE 0C4    REASON CODE 00000010
 READY
```

- I pressed <Enter> and the dump was taken to SYSMDUMP, which I had pre-allocated to a data set with RECFM=FBS LRECL=4160 BLKSIZE=0
- We will process that dump from an unprivileged userid called **JOEUSER**

Getting my Dump Directory Created

- If you don't have the full complement of SBLSxxxx data sets allocated to your session, do it yourself or have your SysAdmin add them to your logon CLIST or REXX
- BLSCDDIR is in SBLSCLI0 which is allocated to my session

```
 READY
blscddir volume(mvspv1)
 Dump directory name 'JOEUSER.DDIR' will be used
 Dump directory space will be allocated in units of 15000 records
 Dump directory space will be allocated on volume MVSPV1
 DEFINE CLUSTER(NAME('JOEUSER.DDIR') VOLUME(MVSPV1) INDEXED REUSE SHAREOPTIONS(1
,3)) INDEX(NAME('JOEUSER.DDIR.I') RECORDS(150,150) CONTROLINTERVALSIZE(4096)) DA
TA(NAME('JOEUSER.DDIR.D') RECORDS(15000,15000) KEYS(128,0) RECORDSIZE(2800 3072)
 BUFFERSPACE(1024000)) /*                                  @03C*/
 IDC0508I DATA ALLOCATION STATUS FOR VOLUME MVSP12 IS 0
 IDC0509I INDEX ALLOCATION STATUS FOR VOLUME MVSP12 IS 0
 IDC0181I STORAGECLASS USED IS BASE
 IDC0181I MANAGEMENTCLASS USED IS STANDARD
 IDC0181I DATACLASS USED IS KEYEDNC
 IPCSDDIR 'JOEUSER.DDIR' /*                                @P1C*/
 ALLOCATE FILE(IPCSDDIR) REUSE DSNAME('JOEUSER.DDIR') SHR /* @P1C*/
 Dump directory 'JOEUSER.DDIR' allocated to FILE(IPCSDDIR)
 READY
```

Adding the Dump to My Inventory

- IPCS 3.4 DSList works similarly to ISPF 3.4 DSList
- Use a mask that makes sense, then add the data set(s) to your inventory

```
---------------------- IPCS Data Set List Utility ------------------------------
Command ===>

Enter the parameter below:

   DSNAME LEVEL   ===> JOEUSER

* The following line commands will be available when the list is displayed:

A - Add data set to IPCS inventory     V - View data set using IPCS
B - Browse data set using ISPF         = - Repeat last command
D - Delete data set                    IPCS subcommand, CLIST or REXX exec
E - Edit data set using ISPF
```

```
IPCS DSLIST - JOEUSER ------------------------------------------------------------------------------------------------ Row 1 of 21
Command ===>                                                                                                  Scroll ===> PAGE

Command  DSNAME                                                  Message
         JOEUSER . . . . . . . . . . . . . . .
         JOEUSER.A.CNTL . . . . . . . . . . . .
         JOEUSER.ASMLANGX . . . . . . . . . . .
         JOEUSER.BRODCAST . . . . . . . . . . .
         JOEUSER.CLIST . . . . . . . . . . . . .
         JOEUSER.DDIR . . . . . . . . . . . . .
         JOEUSER.DDIR.D . . . . . . . . . . . .
         JOEUSER.DDIR.I . . . . . . . . . . . .
         JOEUSER.DITPROF . . . . . . . . . . . .
         JOEUSER.EXEC . . . . . . . . . . . . .
         JOEUSER.ISP57460.SPFLOG1.LIST . . . .
         JOEUSER.LINKLIB . . . . . . . . . . . .
         JOEUSER.MVSA0.ISPPROF . . . . . . . . .
         JOEUSER.MVS60.ISPPROF . . . . . . . . .
         JOEUSER.MVS60.ZHSV04R1.DUMP.TSOFILE
         JOEUSER.MVS60.ZHSV04R1.OLPQ.TSOFILE
         JOEUSER.MVS60.ZHSV04R1.PROF.TSOFILE
         JOEUSER.MVS70.ISPPROF . . . . . . . . .
         JOEUSER.OMVS.ZFS . . . . . . . . . . .
         JOEUSER.OMVS.ZFS.DATA . . . . . . . . .
a        JOEUSER.SYSMDUMP . . . . . . . . . . .  Added to IPCS inventory
********************************************** END OF DATA SET LIST ********************************************************
```

A=Add

```
BLS18124I TITLE=JOBNAME EDJX2     STEPNAME $IKJTEST$IKJTEST SYSTEM 0C4
BLS18255I Dump Init     Elapsed Time          CPU Time
          Input I/O     00:00:00.034482       00:00:00.020013
***
```

Initialize the Data Set With IPCS STATUS

- IPCS Option 4 is Inventory
- SD line command = SETDEF

```
IPCS INVENTORY - JOEUSER.DDIR ------------------------------------------------------------------------------
Command ===>                                                                          SCROLL ===> PAGE

AC Dump Source                                                                        Status
sd DSNAME('JOEUSER.SYSMDUMP') . . . . . . . . . . . . . . . . . . . . . . . . . . . ADDED
   Title=JOBNAME EDJX2    STEPNAME $IKJTEST$IKJTEST SYSTEM 0C4
   Psym=RIDS/EXAMPLE#L RIDS/#UNKNOWN AB/S00C4 VALU/H07E000C0 REGS/C0010 PRCS/00000010
********************************************** END OF IPCS INVENTORY **********************************************
```

- I am issuing IP STATUS, which is also available from IPCS Option 2.2

```
IPCS INVENTORY - JOEUSER.DDIR ------------------------------------------------------------------------------
Command ===> IP STATUS                                                                SCROLL ===> PAGE

AC Dump Source                                                                        Status
__ DSNAME('JOEUSER.SYSMDUMP') . . . . . . . . . . . . . . . . . . . . . . . . . . . ADDED
   Title=JOBNAME EDJX2    STEPNAME $IKJTEST$IKJTEST SYSTEM 0C4
   Psym=RIDS/EXAMPLE#L RIDS/#UNKNOWN AB/S00C4 VALU/H07E000C0 REGS/C0010 PRCS/00000010
********************************************** END OF IPCS INVENTORY **********************************************
```

- Always answer ‘Y’ to this if prompted
- The situation where ‘N’ is appropriate is for an advanced class

```
 IKJ56650I TIME-11:26:42 AM. CPU-00:00:10 SERVICE-64183 SESSION-00:28:04 SEPTEMBER 10,2025
 BLS18122I Initialization in progress for DSNAME('JOEUSER.SYSMDUMP')
 BLS18124I TITLE=JOBNAME EDJX2    STEPNAME $IKJTEST$IKJTEST SYSTEM 0C4
 BLS18223I Dump written by z/OS 03.01.00-0 SYSMDUMP - level same as IPCS level
 BLS18558I This redactable dump has not been post-processed to protect sensitive data
 BLS18222I z/Architecture mode system
 BLS18160D May summary dump data be used by dump access?  Enter Y to use, N to bypass.
y
 BLS18255I Dump Init     Elapsed Time          CPU Time
           Input I/O   00:00:00.520273     00:00:00.392049
           DDIR        00:00:00.055268     00:00:00.048865
 BLS18123I 65,729 blocks, 273,432,640 bytes, in DSNAME('JOEUSER.SYSMDUMP')
 IKJ56650I TIME-11:31:49 AM. CPU-00:00:10 SERVICE-67190 SESSION-00:33:11 SEPTEMBER 10,2025
 BLS18224I Dump of z/OS 03.01.00-0 - level same as IPCS level
 ***
```

IPCS Status Report

- FIND ‘TIME OF ERROR’ to find the “good” stuff you want

```
IPCS OUTPUT STREAM ---------- FOUND: LINE 184 COL 1
Command ===>                                  SCROLL ===> CSR

Time of Error Information

   PSW: 07850000 80000000 00000000 11FF8050
   Instruction length: 06    Interrupt code: 0010
   Failing instruction text: 17FF58E0 D058D507 E000C000
   Translation exception address: 00000000_75000800
   Breaking event address: 00000000_01121222

   AR/GR 0-1     00000000/00000000_11FF9059   00000000/00000000_00000000
   AR/GR 2-3     FFFFFFFF/FFFFFFFF_00000060   FFFFFFFF/FFFFFFFF_00012040
   AR/GR 4-5     FFFFFFFF/FFFFFFFF_862D502C   FFFFFFFF/FFFFFFFF_00AC7940
   AR/GR 6-7     FFFFFFFF/FFFFFFFF_00000000   FFFFFFFF/FFFFFFFF_00012000
   AR/GR 8-9     FFFFFFFF/FFFFFFFF_00AC7940   FFFFFFFF/FFFFFFFF_00FD5A28
   AR/GR 10-11   FFFFFFFF/FFFFFFFF_862E3030   FFFFFFFF/FFFFFFFF_062E402F
   AR/GR 12-13   FFFFFFFF/FFFFFFFF_11FF8060   FFFFFFFF/00000000_11FF9000
   AR/GR 14-15   00000000/00000000_F5000000   00000000/00000000_00000000

   Home ASID: 006D      Primary ASID: 006D      Secondary ASID: 006D
   PKM: 00C0            AX: 0000                EAX: 0000

   This Task's ASID/TCB: 006D/00AB8D30

   RTM was entered because of a program check interrupt.
   The error occurred while an enabled RB was in control.
   No locks were held.
   No super bits were set.

RECOVERY ENVIRONMENT

   Recovery routine type: ESTAI recovery routine
   Recovery routine entry point: 862D502C
   An SVC dump was scheduled by a previous recovery routine.
   The RB associated with this exit was not in control at the time of error.
   User requested no I/O processing.
   IFLG bits 1-3 are valid.

   NO DATA EXISTS IN THE VARIABLE RECORDING AREA

IEA24013I FORMATTING COMPLETED SUCCESSFULLY.



CPU STATUS:
PSW=07850000 80000000 00000000 11FF8050
     (Running in PRIMARY, key 8, AMODE 31, DAT ON, PROBLEM STATE)
     Disabled for PER
    ASID(X'006D') 11FF8050. AREA(Sp252Key00R31Exy#dq7F593Cd0)+50 IN EXTENDED PRIVATE
    ASID(X'006D') 11FF8050. EXAMPLE+50 IN EXTENDED PRIVATE
   ASCB109 at FB9280, JOB(EDJX2), for the home ASID
   ASXB109 at AFD000 and TCB109G at AB8D30 for the home ASID
   HOME ASID: 006D PRIMARY ASID: 006D SECONDARY ASID: 006D
********************************************************** END OF DATA **********************************************************
```

Time of Error Information:

- Key, State, ASC mode, Addressing mode, NSI
- Instruction length, Interrupt code
- The failing instruction text (PSW NSI points to the middle)
- Translation exception address
- Breaking event (last branch) address
- Access registers, general purpose registers
- Primary, secondary & home ASIDs (advanced)
- PKM/AX/EAX – advanced
- This task’s ASID & TCB address
- RTM entry reason and other flags

Recovery environment information

- In this case, recovery provided by the system

CPU Status Information

- Individual PSW bits broken down
- Module name and offset where error occurred

Investigating the Problem

- Looking at TEA, we can see right away the address in R14 is the problem.
- We must ask, how did it get polluted with “garbage?”
- We need to look at how it was loaded, and if it was manipulated in any way since last being loaded?

```
0000004A 17FF                           170          XR    R15,R15              Set retcode = 0
0000004C 58E0 D058                00000058  171          L     R14,ImportantPtr     Load important pointer
00000050 D507 E000 C000 00000000  00000060  172          CLC   0(8,R14),=C'ALL_GOOD' Check important eyecatcher
00000056 A784 0004                0000005E  173          JE    AOK                  Jump if AOK
0000005A A7F8 0004                          174          LHI   R15,4                Set retcode = 4
0000005E                                    175 AOK      DC    0H
0000005E 0101                               176          PR    ,                    Return to caller
```

- Even when debugging without a source code listing (certainly not an ideal situation), you can use IPCS to decode instructions:

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------
Command ===> IP L PSW+8!-C I                                                                    SCROLL ===> CSR
********************************************************** TOP OF DATA ***********************************************

 LIST 11FF8050. ASID(X'006D') POSITION(X'-0C') LENGTH(X'1000') INSTRUCTION
 11FF8044 | BFF8 C00C      | ICM     R15,X'8',X'C'(R12)
 11FF8048 | 0E0E           | MVCL    R0,R14
 11FF804A | 17FF           | XR      R15,R15
 11FF804C | 58E0 D058      | L       R14,X'58'(,R13)
 11FF8050 | D507 E000 C000 | CLC     X'0'(X'8',R14),X'0'(R12)
 11FF8056 | A784 0004      | BRC     X'8',*+X'8'
 11FF805A | A7F8 0004      | LHI     R15,X'4'
 11FF805E | 0101           | PR
 11FF8060 LENGTH(X'0FE4')==>Displayed as AREA
 11FF8060. C1D3D36D C7D6D6C4 00000002 40800000  00000000 00000000 00000000 00000000 |ALL_GOOD.... ................|
 11FF8080 LENGTH(X'0FC0')==>All bytes contain X'00'
 11FF9040. 00000000                                                                 |....                            |
********************************************************** END OF DATA ***********************************************
```

PSW is a symbol that points to a 16-byte area in the dump where the z/Architecture PSW is stored. Use the IPCS LSYM command to see all of the symbols that are automatically set by IPCS as well as those you created using the EQ command.

Solving the Problem

- Fundamentally, the approach is to examine the code and “play computer” to reproduce at your desk how R14 ended up with an invalid address.
- In our contrived example, R14 is loaded from a fullword in memory immediately before it is used as a base register for the CLC instruction.
- Let’s use IPCS to show us the memory before and after the fullword in question. Why? Because most overlays occur because of erroneous updates to adjacent areas, and not just randomly.
- For random overlays, unless you get really lucky, you will probably need to reproduce with a storage-alteration SLIP trap in place, the use of which is beyond the scope of today’s presentation.

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------
Command ===> IP L 13R!+58                                                                          SCROLL ===> CSR
 ********************************************************* TOP OF DATA ***************************************************

 LIST 11FF9000. ASID(X'006D') POSITION(X'+58') LENGTH(X'2000') AREA
 11FF9058.                                    F5000000 00000000 |                              5.......|
 11FF9060 LENGTH(X'0FA0')==>All bytes contain X'00'
```

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------
Command ===> IP L 13R!                                                                             SCROLL ===> CSR
 ********************************************************* TOP OF DATA ***************************************************

 LIST 11FF9000. ASID(X'006D') LENGTH(X'2000') AREA
 11FF9000 LENGTH(X'40')==>All bytes contain X'00'
 11FF9040. 00000000 00000000 00000000 00000000  E3C5E2E3 F1F2F3F4 F5000000 00000000 |................TEST12345.......|
 11FF9060 LENGTH(X'0FA0')==>All bytes contain X'00'
```

AHH-HAAAAAA!

Browsing Memory

- I keep an IPCS BROWSE session going as a split screen while debugging
  - Go to any location (**Locate**) or scroll through <F7> <F8> any address space or data space
  - Perform character or hex **Find** operations to “brute force” locate something
  - Create symbols (**EQuate**) to remember addresses and use them in other commands
  - Create pointers (**STACK**) to make it easier to later find and process important major areas

```
------------------------------------------------------ IPCS - ENTRY PANEL ------------------------------------------------------
Command ==>

 CURRENT DEFAULTS:
  Source ==> DSNAME('JOEUSER.SYSMDUMP')
  Address space ==> ASID(X'006D')

 OVERRIDE DEFAULTS:                                                          (defaults used for blank fields)
  Source ==> DSNAME('JOEUSER.SYSMDUMP')
  Address space ==>
  Password       ==>

 POINTER:
  Address        ==>                                                         (blank to display pointer stack)
  Remark         ==>                                                                   (optional text)
```

```
DSNAME('JOEUSER.SYSMDUMP') POINTERS  ------------------------------------------------------------------------------------------
Command ===>                                                                                          SCROLL ===> CUR
ASID(X'006D') is the default address space
PTR   Address              Address space                                  Data type
s0001 00.                  ASID(X'006D')                                  AREA
      Remarks:
********************************************** END OF POINTER STACK **********************************************************
```

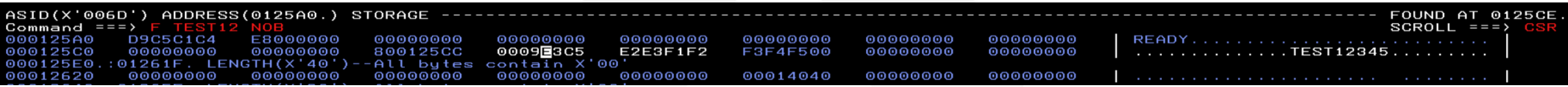
```
ASID(X'006D') ADDRESS(00.) STORAGE ---------------------------------------------------------------------------------------------
Command ===>                                                                                          SCROLL ===> CSR
00000000    00000000    00000000    00000000    00000000    00FD5A28    00000000    7FFFF000    7FFFF000  | ..............!.....".0.".0. |
00000020    7FFFF000    7FFFF000    7FFFF000    7FFFF000    00000000    00000000    7FFFF000    7FFFF000  | ".0.".0.".0.".0........".0.".0. |
00000040    00000000    00000000    00000000    00FD5A28    00000000    00000000    00000000    00000000  | ............!.............. |
00000060.:7F. LENGTH(X'20')--All bytes contain X'00'
00000080    00000000    00001005    00020001    00020011    00000001    00004070    00000000    178BA53E  | ..................... .....v. |
000000A0    00010001    00000000    000001EF    890FF400    00000000    00000000    00010ED1    02721D08  | ............i.4.........J.... |
000000C0    28000000    00000000    FBFFFFFB    FEFFFF7C    FFDEE000    00000000    FF36EFD8    00000000  | ..............®..\........Q.... |
000000E0.:FF. LENGTH(X'20')--All bytes contain X'00'
00000100    00000000    00000000    00000000    00000000    00000000    01DC7424    00000000    00000000  | ..............................  |
00000120    00000000    00000000    00000000    00000000    07040000    80000000    00000000    0587CA0E  | ..........................g.. |
00000140    07641001    80000000    00000000    12266514    07042001    80000000    00000000    05872DDA  | ..........................g.. |
00000160    00000000    00000000    00000000    00000000    07642001    80000000    00000000    108C69EC  | ..............................  |
00000180.:019F. LENGTH(X'20')--All bytes contain X'00'
```

Line Commands Available While Browsing Memory

- Every four-byte word shown on the display has a command field
- **%** – evaluate the address in the word as a 24-bit address, add it to the pointer stack, and scroll to that address in memory
- **?** – evaluate the address in the word as a 31-bit address, add it to the pointer stack, and scroll to that address in memory
- **!** – evaluate the address in two adjacent words as a 64-bit address, add it to the pointer stack, and scroll to that address in memory
- **L** – evaluate the address in the word as a 24-bit address and add it to the pointer stack
- **H** – evaluate the address in the word as a 31-bit address and add it to the pointer stack

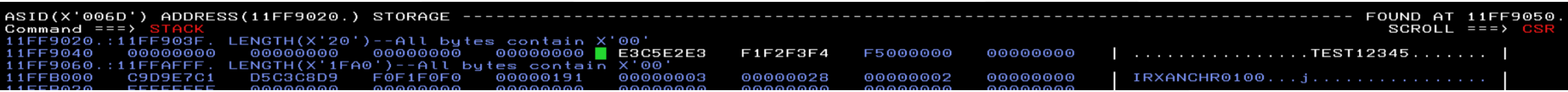
Using FIND Command While Browsing Memory

- I issued **Find TEST12 NOBreak** to “brute force” find the string **TEST12**
- This looks to be where the parameter is set up for the program:

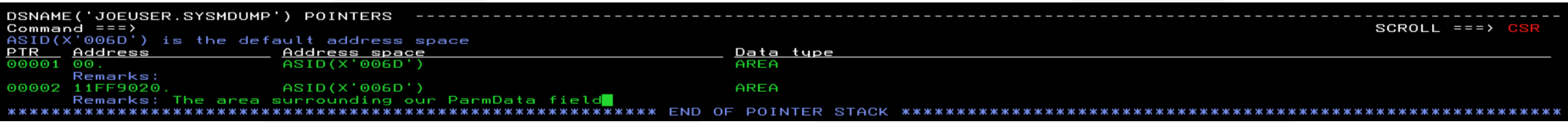


- <F5> This is the area in memory where the TSO/E command is stored:

- <F5> Finally, we’ve got our eight-byte parm data field and one-byte overlay



- Lastly, an example of setting the remark field after using the **STACK** command



Additional Commands Available While Browsing Memory

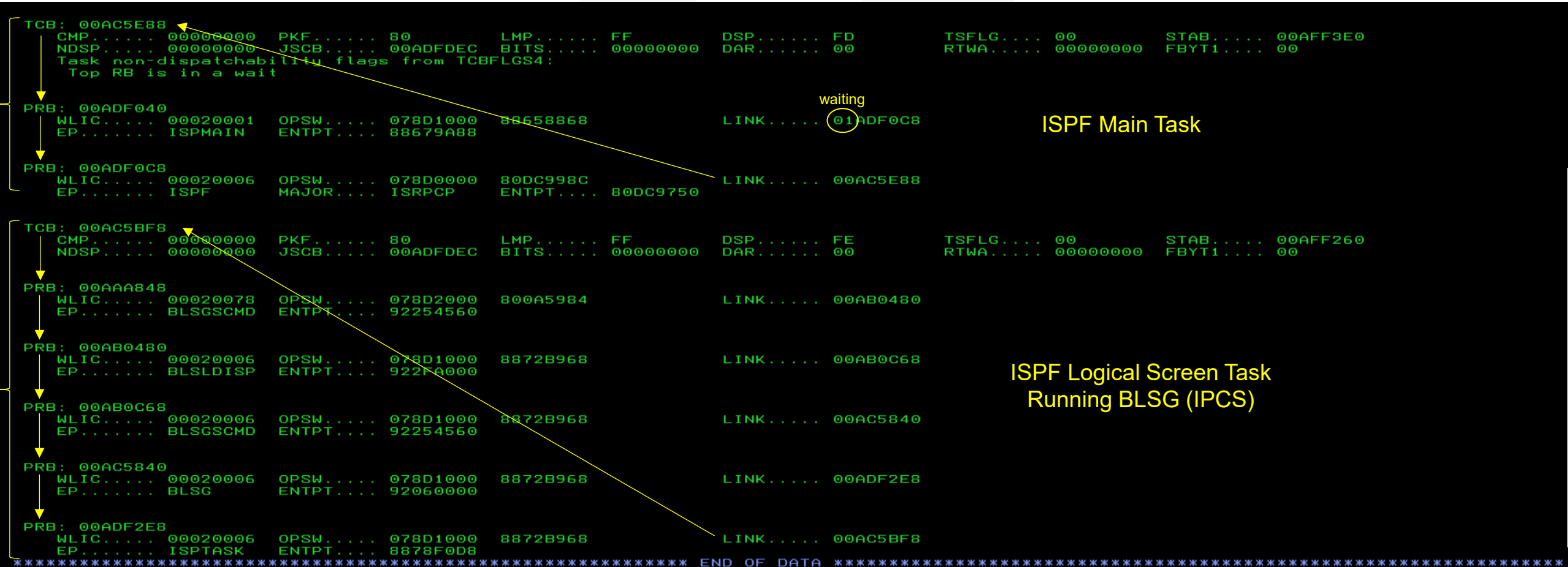
- BROWSE commands are not prefixed with **IPCS**
- We already showed the **Find** and **STACK** commands
- The other commands are **ASCII**, **ALIGN**, **CANcel**, **CBFormat**, **CONDENSE**, **EBCDIC**, **EQuate**, **Locate**, **NOALIGN**, **OPCODE**, **RENum**, **RESET**, **VERBOSE** and **Where**
  - ASCII and EBCDIC change the character data on the right from/to ASCII or EBCDIC
  - ALIGN and NOALIGN control whether you can position the display to start on other than a 16- or 32-byte boundary – personally, I hate the NOALIGN behavior, but that's just me…
  - CANCEL kills the browse task – I've never used this ever; I use <F3> to go back
  - CBFORMAT, EQUATE and WHERE are identical to their IPCS counterparts
  - CONDENSE and VERBOSE control the suppression of rows with identical data
  - LOCATE positions the browser to the requested address (subject to [NO]ALIGN)
  - OPCODE is practically useless; use **IP L X I** instead
  - RENUM renumbers the pointer list
  - RESET removes all pending primary and line commands – similar to ISPF RESET

SHARE
EDUCATE • NETWORK • INFLUENCE

The Extra Bits

Using Data Source = ACTIVE

- Many IPCS functions will operate on a live system without needing a dump
- Issue **IP SETDEF ACTIVE** to set your source to use the live system
  - By default, you are looking at your own address space
  - This is the bottom of the **KEY FIELDS** report generated by the **IP SUMMARY** command



Looking Beyond Your Program Using Data Source = ACTIVE

- By default, ACTIVE provides access to your own private area as well as non-fetch-protected common memory areas
- Users with higher access privilege requirements can be granted READ access to the following resources in the FACILITY class as needed:
  - **BLSACTV.ADDRSPAC** – additionally provides access to:
    - fetch-protected memory i.e., all memory visible to a key 0 application
    - all data spaces owned by the user's ASID
  - **BLSACTV.SYSTEM** – additionally provides access to:
    - ABSOLUTE storage aka REAL memory
    - any other ASID's private area
    - the data spaces owned by other ASIDs
- Functions include (nearly) the full complement of IPCS functions such as control block formatting, searching through memory, etc.
- It is an EXTREMELY powerful diagnostic tool!

Traces

- Component Trace (CTRACE)
  - Specific traces written by product developers
  - Option 2.7.1 or **IPCS CTRACE** command
- Generalized Trace Facility Trace (GTF)
  - Numerous traces; often used to capture PER SLIP events
  - Option 2.7.2 or **IPCS GTF** command
- Master Trace (MTRACE)
  - Shows system messages i.e., WTOs leading up to when the SVC dump was taken
  - Option 2.7.3 or **IPCS VERBX MTRACE** command
- System Trace (SYSTRACE)
  - Documents detailed activity occurring on every CP and zIIP right up to when the dump was taken
  - Option 2.7.4 or **IPCS SYSTRACE** command

Component Trace (CTRACE)

- Component Trace is very specialized; you might never work with one
- Often the trace is captured only in the address space of the component for which the trace applies
- If you attempt to access a component trace in any “normal” dump, the buffers are generally not there

```
IPCS OUTPUT STREAM ---------------------------------------------------------------------------------------- Line 0 Cols 1 140
Command ===>                                                                                              SCROLL ===> CSR
 ********************************************************** TOP OF DATA ***********************************************************

 COMPONENT TRACE QUERY SUMMARY

 ITT10003I There are no trace buffers in the dump for COMP(SYSXES)  SUB((GLOBAL))
  COMP(SYSXES)  SUBNAME((GLOBAL))

  Trace buffers are not in the dump.
              Buffer size: 0168K

              Options: CONNECT,CONFIG,RECOVERY

              MinOpts: DEFAULT TRACING ONLY
 ********************************************************** END OF DATA ************************************************************
```

Generalized Trace Facility Trace (GTF)

- GTF trace is typically used when trying to debug a reproduceable problem
- A sysadmin specifies the trace parameters and starts GTF as a started task
- Wraparound trace buffers are in the GTF address space and trace records can optionally be written to a DASD data set

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------------------------------
Command ===> IP GTF                                                                                                    SCROLL ===> CSR
********************************************************** TOP OF DATA ***********************************************************************
   **** GTFTRACE DISPLAY OPTIONS IN EFFECT ****
  SSCH=ALL  IO=ALL  CCW=SI
  SVC=ALL  PI=ALL  IOX=ALL
  EXT  DSP  SLIP  RNIO  SRM  RR

  Dumped GTF data for GTFPHX   7130

  **** GTF DATA COLLECTION OPTIONS IN EFFECT: ****
  Minimum tracing for IO, SSCH, SVC, PI, EXT, and FRR events
  All GTRACE events requested
  All events associated with the execution should be traced
  All DISPATCHER events traced
  PCI events are to be traced
  System resource manager events traced

             **** GTF TRACING ENVIRONMENT ****
      Release: SP7.3.1   FMID: HBB77E0   System name: MVS60
      CPU Model: 8562  Version: 00  Serial no. 01A798

 SVC    CODE.... 060       ASCB.... 00FC8A00 CPU..... 0000     PSW..... 07041000 8000003C 00000000 05987A10          TCB..... 00AE1CE8
                           R15..... 00000000 R0..... 00000100 R1..... 7F317FA0
                GMT-09/13/2025 20:48:13.592255  LOC-09/13/2025 13:48:18.592255

 SVCR   CODE.... 060       ASCB.... 00FC8A00 CPU..... 0000     PSW..... 07041000 8000003C 00000000 05987A10          TCB..... 00AE1CE8
                           R15..... 00000000 R0..... 00000071 R1..... 7F317FA0
                GMT-09/13/2025 20:48:13.592271  LOC-09/13/2025 13:48:18.592271

 PI     CODE.... 048       ASCB.... 00FC8A00 CPU..... 0000     PSW..... 07045000 80000030 00000000 11FF8368          TCB..... 00AE1CE8
                           VPH..... 00000000 VPA..... 00506401 R15..... 00000000 R1..... 1281C950
                GMT-09/13/2025 20:48:13.592287  LOC-09/13/2025 13:48:18.592287

 PI     CODE.... 049       ASCB.... 00FC8A00 CPU..... 0000     PSW..... 07044000 80000031 00000000 11FF915E          TCB..... 00AE1CE8
                           VPH..... 00000000 VPA..... 00506401 R15..... 7F5D6074 R1..... 00000000
                GMT-09/13/2025 20:48:13.592324  LOC-09/13/2025 13:48:18.592324

 SVC    CODE.... 001       ASCB.... 022E0B80 CPU..... 0002     PSW..... 07041000 80000001 00000000 1200C486          TCB..... 00AF3A60
                           R15..... 7F3CCA48 R0..... 00000001 R1..... 80C335B8
                GMT-09/13/2025 20:48:13.592361  LOC-09/13/2025 13:48:18.592361
```

Master Trace (MTRACE)

- Master Trace appears in SVC dumps
- Controlled by the TRACE MT command
- Default buffer size is 24K, which is waaay too small
- We specify TRACE MT,256K

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------------------
Command ===> IP VERBX MTRACE                                                                  SCROLL ===> CSR

 *** MASTER TRACE TABLE ***

 TAG   IMM DATA   |------------------------------------------------------ MESSAGE DATA ------------------------------------------------------|

 0001  00000028   NC0000000 MVS60     2025256 08:50:15.14 INSTREAM 00000290   LOGON
 0001  00000049   N 0000000 MVS60     2025256 08:50:15.45 JES3     00000090   IAT6100 ( DEMSEL ) JOB JOEUSER  (J0658468), PRTY=15, ID=JOEUSER
 0001  0000008F   N 0000000 MVS60     2025256 08:50:15.50 JOEUSER  00000281   IEF125I JOEUSER - LOGGED ON - TIME=08.50.15
 0001  0000008F   M 0020000 MVS60     2025256 08:50:22.27 JOEUSER  00000281   IEC161I 056-084,JOEUSER,$IKJTEST$IKJTEST,IPCSDDIR,,,JOEUSER.DDIR,
               068
 0001  0000008F   E                                             068 00000081   IEC161I JOEUSER.DDIR.D,CATALOG.USRCAT01.VMVSSY1
 0001  0000008F   M 0020000 MVS60     2025256 08:50:22.27 JOEUSER  00000281   IEC161I 056-084,JOEUSER,$IKJTEST$IKJTEST,IPCSDDIR,,,JOEUSER.DDIR,
               069
 0001  0000008F   E                                             069 00000081   IEC161I JOEUSER.DDIR.I,CATALOG.USRCAT01.VMVSSY1
 0001  0000008F   M 0020000 MVS60     2025256 08:50:22.28 JOEUSER  00000281   IEC161I 062-086,JOEUSER,$IKJTEST$IKJTEST,IPCSDDIR,,,JOEUSER.DDIR,
               070
 0001  0000008F   E                                             070 00000081   IEC161I JOEUSER.DDIR.D,CATALOG.USRCAT01.VMVSSY1
 0001  00000001   N 0000000 MVS60     2025256 08:55:20.17          00000281   IEA989I SLIP TRAP ID=X422 MATCHED.  JOBNAME=BPXOINIT, ASID=0035.
 0001  0000000B   NC0000000 MVS60     2025256 09:00:00.20 INTERNAL 00000290   8I C
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8506 JSAM BUFFER USAGE
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8725 TOTAL NUMBER OF JSAM BUFFERS ..... 00992
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8508 CURRENT NUMBER IN USE ..... 00025
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8510 MAXIMUM NUMBER USED ..... 00093
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8722 PRIMARY EXTENT SIZE ..... 00992
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8723 SECONDARY EXTENT SIZE ..... 01984
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8724 SECONDARY EXTENTS ALLOWED ..... 0016
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8501 CURRENT SECONDARY EXTENTS IN USE ..... 0000
 0001  00000049   NR0000000 MVS60     2025256 09:00:00.20 JES3     00000290   IAT8512 NUMBER OF AWAITS FOR AVAILABLE BUFFER .... 0000
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.31 JES3     00000290   IRR010I  USERID SYSOPER  IS ASSIGNED TO THIS JOB.
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.34 JES3     00000090   IAT6100 (J0658455) JOB AUTOELIC (J0658470), PRTY=13, ID=EJES
                    NET-ID=*NONE     SUB=J0656054
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.34 JES3     00000281   IAT6118 0001 CARDS FLUSHED
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.34 JES3     00000090   IAT6100 (J0658458) JOB AUTOPLIC (J0658469), PRTY=13, ID=SYSOPER
                    NET-ID=*NONE     SUB=J0656054
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.34 JES3     00000281   IAT6118 0001 CARDS FLUSHED
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.35 JES3     00000281   IAT6306 JOB (J0658472) IS INTRDR, CALLED BY MAIN
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.41 JES3     00000290   IRR010I  USERID SYSOPER  IS ASSIGNED TO THIS JOB.
 0001  00000049   N 0000000 MVS60     2025256 09:00:00.42 JES3     00000290   IRR010I  USERID SYSOPER  IS ASSIGNED TO THIS JOB.
```

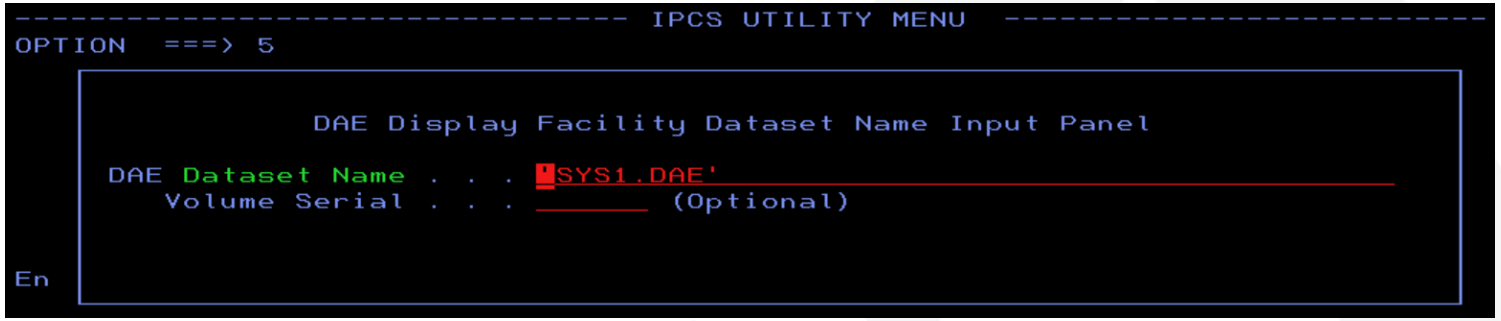
System Trace (SYSTRACE)

- One of the most important traces for debugging abends
- IBM adds TRT to dump specifications in IEACMD00
  - COM='CHNGDUMP SET,SDUMP=(LSQA,TRT,XESDATA),ADD’
- IEADMCxx, IEADMPxx, and IEADMRxx should ideally specify:
  - SDATA=ALLSDATA,PDATA=ALLPDATA
- I follow a three-step process to find my problems in the system trace:
  - Issue **IP SYSTRACE**
  - Issue **F BEFORE**
  - Issue one of the following:
    - **F ‘*’** to stop at all exceptions from this point forward – there could be several before you hit yours
    - **F RCVY** to find recovery processing – usually yours is first, but not always
    - **F mypswnsi** to find the event shown in the dump – usually it’s the only one, but not always
  - Back up half a page and inspect the trace to see what happened

System Trace (SYSTRACE)

Option 3.5 – Dump Analysis and Elimination (DAE)

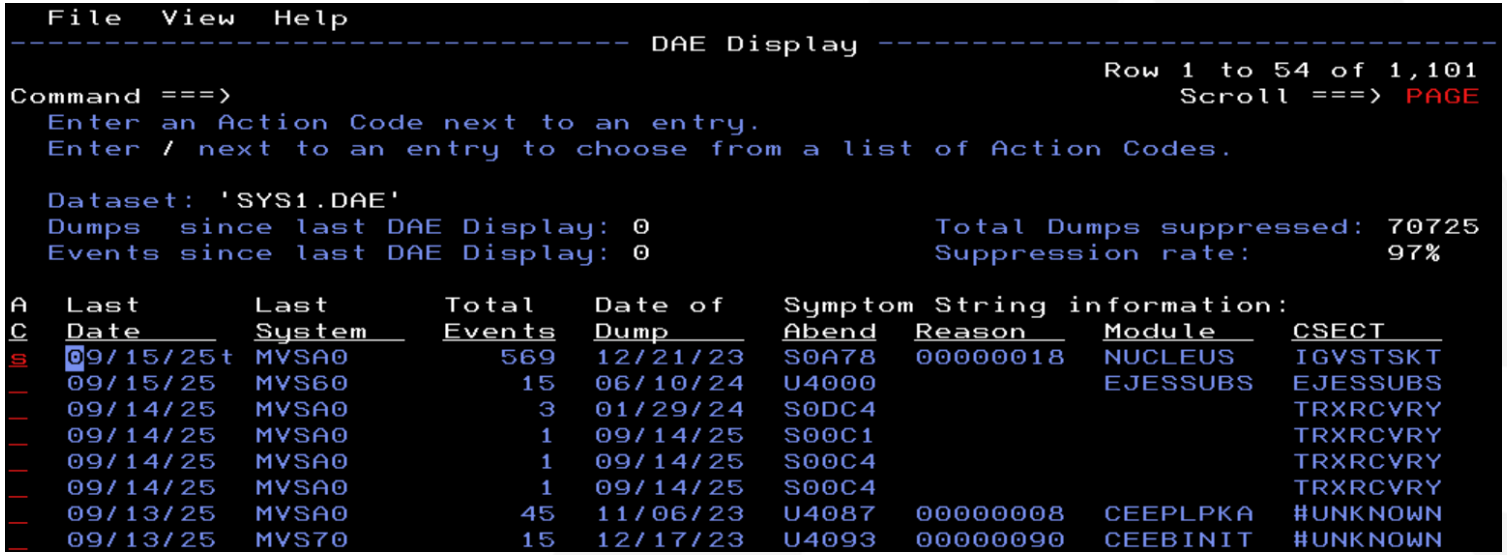
- Upon initial entry, you are prompted for the DAE data set name



- Pressing <Enter> brings you to the DAE display (next slide)
- You can change to a different data set later by issuing the LISTD command or by exiting and re-invoking the DAE function

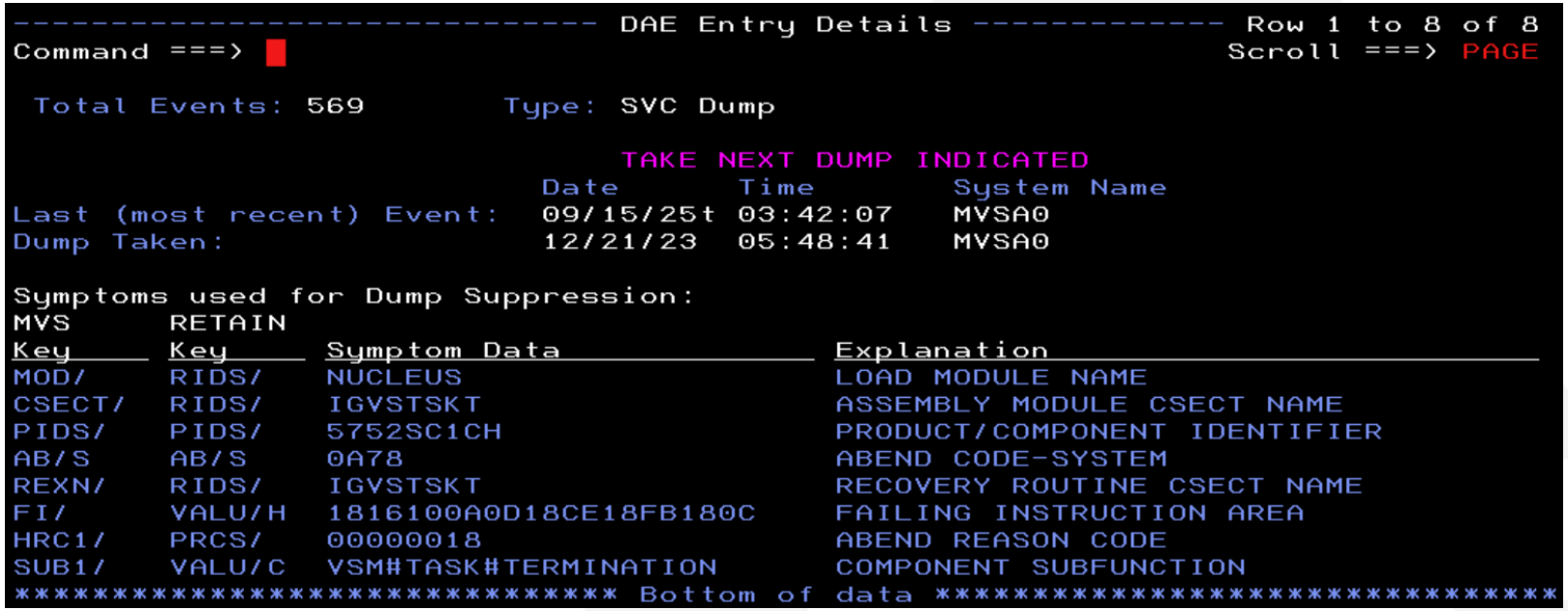
DAE Display

- Issue **Find string** to find any string on the display



- Line commands are **S**how details, **T**ake next dump, **V**iew dump in IPCS
- The little ‘t’ next to the date on the first line indicates “Take Next Dump” is already active for that entry

DAE Display Details



Things to Check Out On Your Own

- I discussed only a subset of IPCS commands and only a subset of options for those commands. Hopefully, this gives you enough to be productive. Be sure to research additional functions when you find time.
- I discussed reading the MVS system trace in a session appropriated entitled, **SHARE'd Knowledge: The MVS System Trace** in Atlanta back in March 2023. Those slides should be available for download by the membership.
- I discussed 71 undocumented IPCS commands in the session I gave last year (2025) in Cleveland called **A Treasure Trove of Undocumented IPCS Commands**. Those slides should be available for download as well. To find those undocumented commands ☺:
  - Navigate to IPCS hidden option 2.6i to find the first 23 of them
  - Issue **IP IEAVHELP** command to see the remaining 48 undocumented commands

SHARE
EDUCATE • NETWORK • INFLUENCE

Questions?

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation