Session 16651
z/OS Debugging: Old Dogs and New Tricks, the Sequel

MVS Core Technologies Project – March 5th, 2015

John Shebey & Patty Little
IBM Poughkeepsie

jshebey@us.ibm.com
plittle@us.ibm.com

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

Agenda

- **System Trace debugging** – latest concepts
- **SLIP PVTMOD –** tall tales
- **z/OS 'Tools and Toys' Rexx Execs** – do you know?

System Trace Debugging

Latest concepts and enhancements

System Trace basics revisited

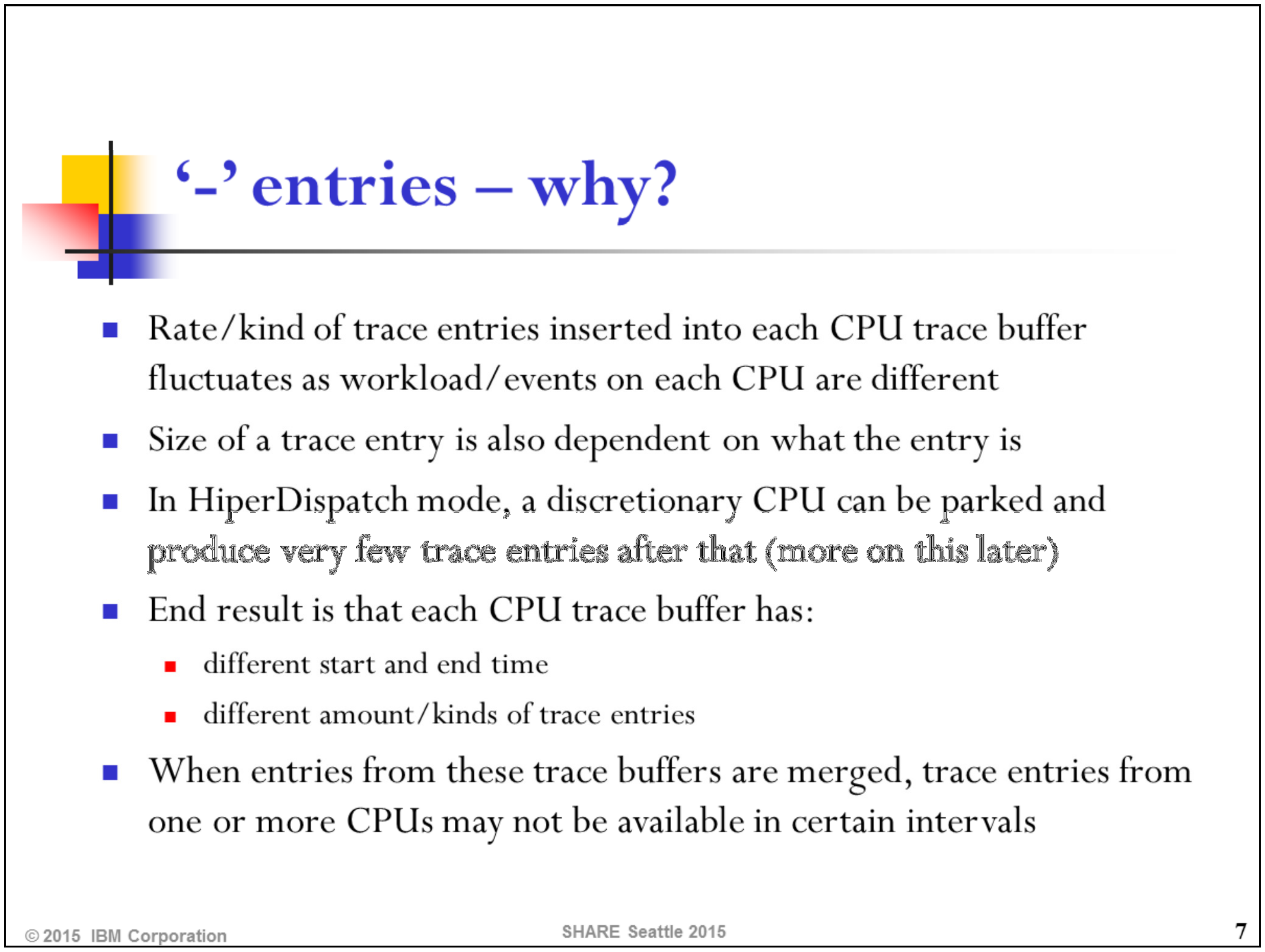
- Consists of **trace buffers (one per logical CPU)**, residing in the Trace Address Space (ASID 4)
- System trace entries are inserted **continuously** by each CPU into its trace buffer
- Contains detailed system activity
- Default size of 1 M per CPU
- **IPCS SYSTRACE** subcommand is used to format system trace entries in a dump
    - Trace entries from the buffers are merged and presented chronologically

'-' entries and related messages

```
0001-00B2 008BA3D0  SVC      1 00000000_39A87DC4  39781370 80000001 C6805778
                               07851000 80000000
0001-0001 00000000  WAIT
******** Trace data is not available from all processors before this time.
0000 00B2 009C37D8  SVC     78 00000000_396F7740  00000002 00000208 00000000
                               07850000 80000000
0000 00B2 009C37D8  SVCR    78 00000000_396F7740  00000000 00000208 397F62F0
0001 0066 33483B80  SRB        00000000_013A443E  00000066 32AECFAC B2AECF80
                               07040000 80000000  009C2D00 00

 many lines omitted here..........

0000 0005 03917900  PC     ...    0      38D07A64              00503
******** Trace data is not available from all processors after this time.
0001-0010 009F79D8  SVC      1 00000000_38EBFF2E  80000000 00000001 C7140BD8
                               07040000 80000000
```

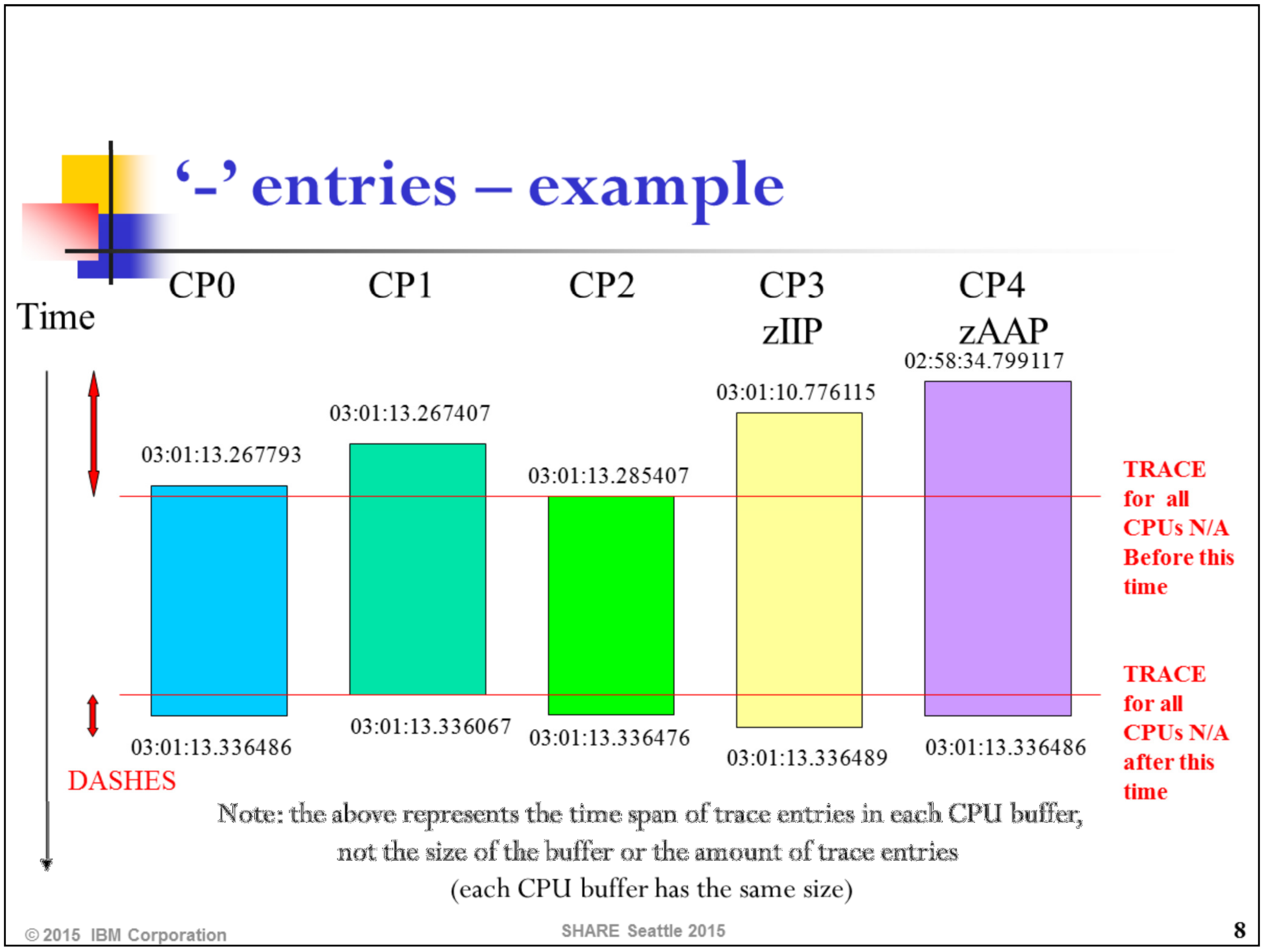
'-' entries indicate trace entries from one or more CPUs are not available in this section

2nd message 'Trace….after this time' is issued for SVC dumps but not standalone dumps



'-' entries – why?

- Rate/kind of trace entries inserted into each CPU trace buffer fluctuates as workload/events on each CPU are different
- Size of a trace entry is also dependent on what the entry is
- In HiperDispatch mode, a discretionary CPU can be parked and produce very few trace entries after that (more on this later)
- End result is that each CPU trace buffer has:
  - different start and end time
  - different amount/kinds of trace entries
- When entries from these trace buffers are merged, trace entries from one or more CPUs may not be available in certain intervals



'-' entries – example

Note: the above represents the time span of trace entries in each CPU buffer, not the size of the buffer or the amount of trace entries (each CPU buffer has the same size)

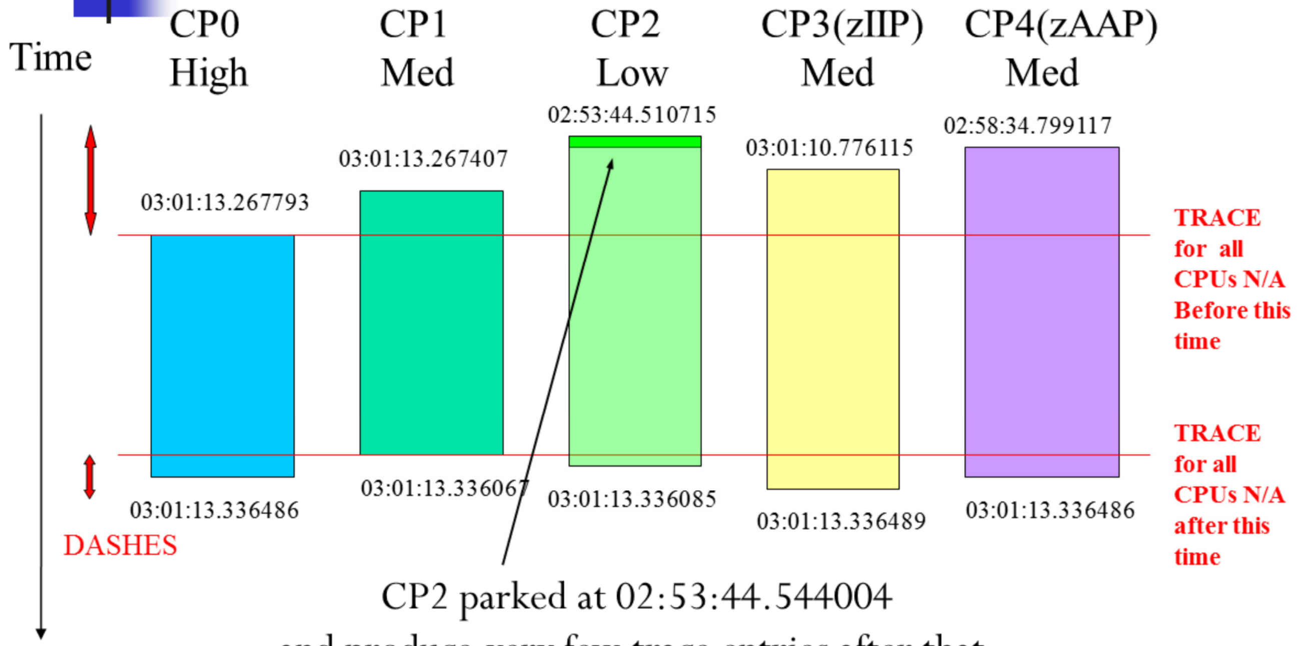
'-' entries – should I care?

- It depends on the problem you are investigating
- In general, you should be aware that **the complete picture or event history** may not be available in the section of the system trace with '-' entries
- If the problem is related to a single work unit or process, and there is no connection with other work/processes running on other CPUs, you may not need to pay attention to whether the trace entries being reviewed have '-' or not
- But if the problem can be related to, or caused by events on other CPUs, you should try to limit your investigation in the section of the trace table with no '-'

Parked CPUs and System Trace

- In HiperDispatch mode, logical processors (CPUs known to z/OS) can be one of the following:
  - High – receiving 100% share of a physical processor
  - Medium – receiving greater than 0% and up to 100% of a physical processor
  - Low or Discretionary – receiving 0% or very low amount of a physical processor
- Low or Discretionary CPUs can be **parked**:
  - CPU will be in a dummy or no-work wait
  - No work will be dispatched by z/OS on this CPU
  - Interrupts can still be taken but then CPU will go back to wait
  - Very few system trace entries will be generated after being parked

Parked CPU – example

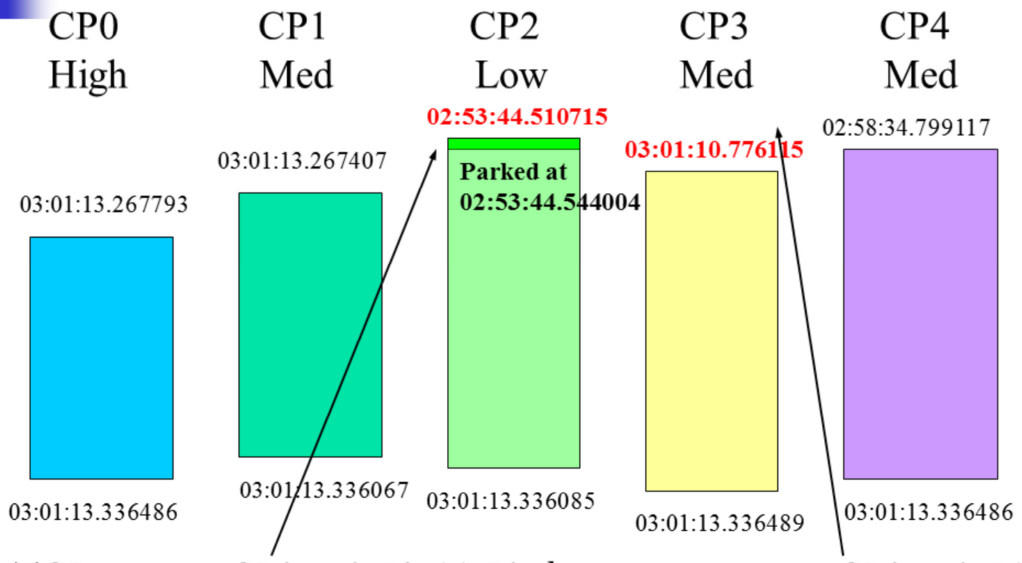


CP2 parked at 02:53:44.544004
and produce very few trace entries after that

Parked CPU – debugging considerations

- System trace entries from parked CPUs are usually much earlier than the time of the dump
  - Consider whether they are relevant to the problem at all, since they are from long time ago
  - Events represented by these entries may not be the **only** (or **last**) activity of a work unit or process (see example next page)
- When reviewing system status in a standalone dump using IPCS subcommands such as STATUS:
  - Many CPUs present may not mean a lot of processing power since some of them can be parked

Parked CPU considerations

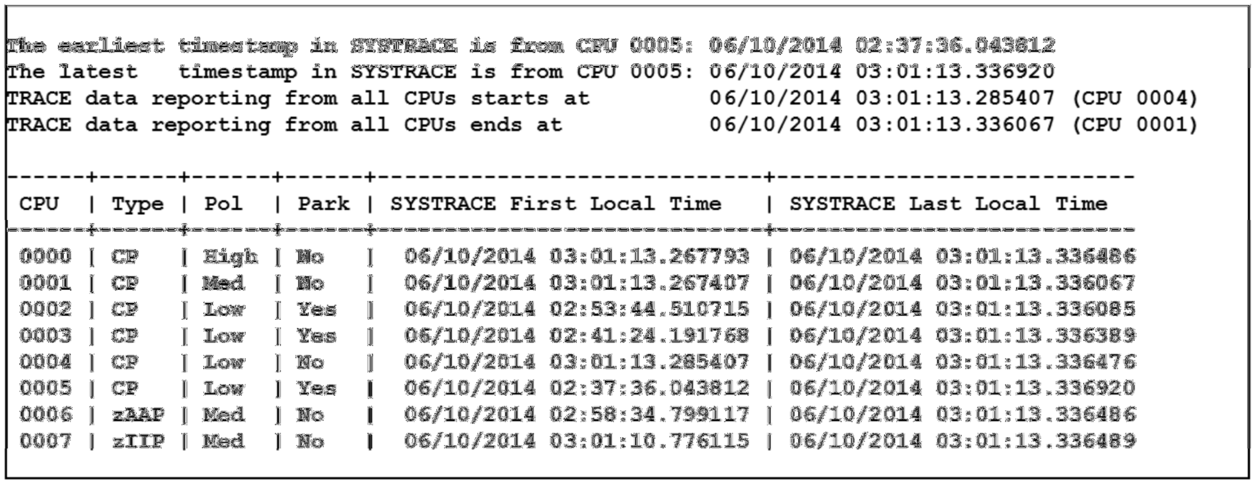


If a TCB runs on CP2 at 2:53:44.52 then runs again on CP3 at 2:54:00, the trace table in the dump will only show its activity on CP2 and not CP3

IP SYSTRACE STATUS TIME(LOCAL)

- New parameter STATUS for IPCS SYSTRACE in z/OS V2.1
- Displays the following information for each CPU at the time of the dump:
  - CPU number and type
  - Parked status and polarity for physical processor share, if running in HiperDispatch mode
  - Start and end time of this CPU in the system trace
- Also displays the start and end time in the trace when all CPUs are available (in SVC dumps, for standalone dumps only the start time is displayed)

IP SYSTRACE STATUS TIME(LOCAL)



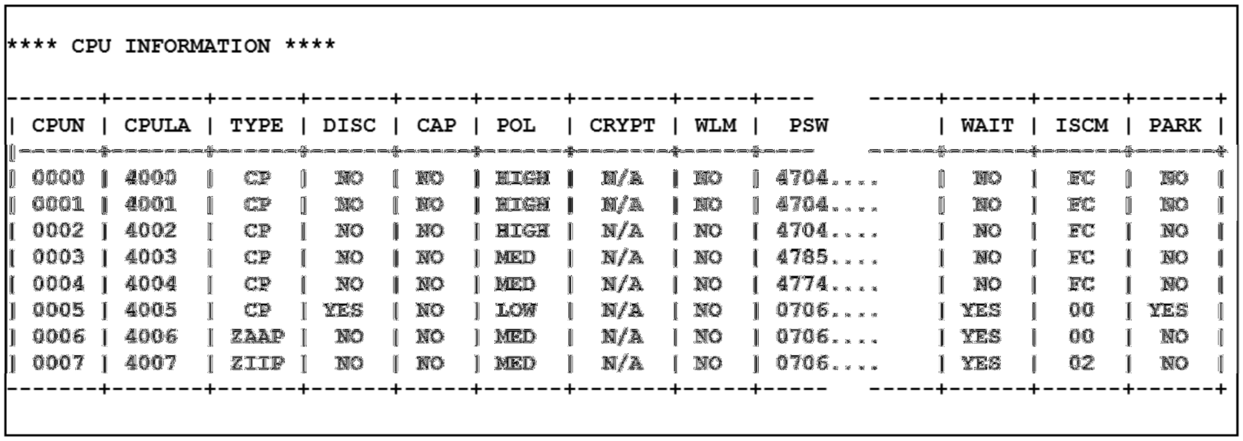
```
The earliest timestamp in SYSTRACE is from CPU 0005: 06/10/2014 02:37:36.043812
The latest   timestamp in SYSTRACE is from CPU 0005: 06/10/2014 03:01:13.336920
TRACE data reporting from all CPUs starts at         06/10/2014 03:01:13.285407 (CPU 0004)
TRACE data reporting from all CPUs ends at           06/10/2014 03:01:13.336067 (CPU 0001)

------+------+------+------+-----------------------------+--------------------------
 CPU  | Type | Pol  | Park | SYSTRACE First Local Time   | SYSTRACE Last Local Time
------+------+------+------+-----------------------------+--------------------------
 0000 | CP   | High | No   |  06/10/2014 03:01:13.267793 | 06/10/2014 03:01:13.336486
 0001 | CP   | Med  | No   |  06/10/2014 03:01:13.267407 | 06/10/2014 03:01:13.336067
 0002 | CP   | Low  | Yes  |  06/10/2014 02:53:44.510715 | 06/10/2014 03:01:13.336085
 0003 | CP   | Low  | Yes  |  06/10/2014 02:41:24.191768 | 06/10/2014 03:01:13.336389
 0004 | CP   | Low  | No   |  06/10/2014 03:01:13.285407 | 06/10/2014 03:01:13.336476
 0005 | CP   | Low  | Yes  |  06/10/2014 02:37:36.043812 | 06/10/2014 03:01:13.336920
 0006 | zAAP | Med  | No   |  06/10/2014 02:58:34.799117 | 06/10/2014 03:01:13.336486
 0007 | zIIP | Med  | No   |  06/10/2014 03:01:10.776115 | 06/10/2014 03:01:13.336489
```

An example from a SVC dump

IP IEAVCPUI

- A z/OS 'Tools and Toys' Rexx Exec – as-is, not documented, not supported, not warranted
- Displays information about the CPUs in the dump
- Available at all supported z/OS releases

IP IEAVCPUI example



```
**** CPU INFORMATION ****

-------+-------+------+------+-----+------+-------+-----+----    -----+------+------+------+
| CPUN | CPULA | TYPE | DISC | CAP | POL  | CRYPT | WLM |  PSW          | WAIT | ISCM | PARK |
|------+-------+------+------+-----+------+-------+-----+----    -----+------+------+------+
| 0000 | 4000  |  CP  |  NO  | NO  | HIGH |  N/A  | NO  | 4704....      |  NO  |  FC  |  NO  |
| 0001 | 4001  |  CP  |  NO  | NO  | HIGH |  N/A  | NO  | 4704....      |  NO  |  FC  |  NO  |
| 0002 | 4002  |  CP  |  NO  | NO  | HIGH |  N/A  | NO  | 4704....      |  NO  |  FC  |  NO  |
| 0003 | 4003  |  CP  |  NO  | NO  | MED  |  N/A  | NO  | 4785....      |  NO  |  FC  |  NO  |
| 0004 | 4004  |  CP  |  NO  | NO  | MED  |  N/A  | NO  | 4774....      |  NO  |  FC  |  NO  |
| 0005 | 4005  |  CP  | YES  | NO  | LOW  |  N/A  | NO  | 0706....      | YES  |  00  | YES  |
| 0006 | 4006  | ZAAP |  NO  | NO  | MED  |  N/A  | NO  | 0706....      | YES  |  00  |  NO  |
| 0007 | 4007  | ZIIP |  NO  | NO  | MED  |  N/A  | NO  | 0706....      | YES  |  02  |  NO  |
-------+-------+------+------+-----+------+-------+-----+-----   -----+------+------+------+
```

An example using the previous SVC dump

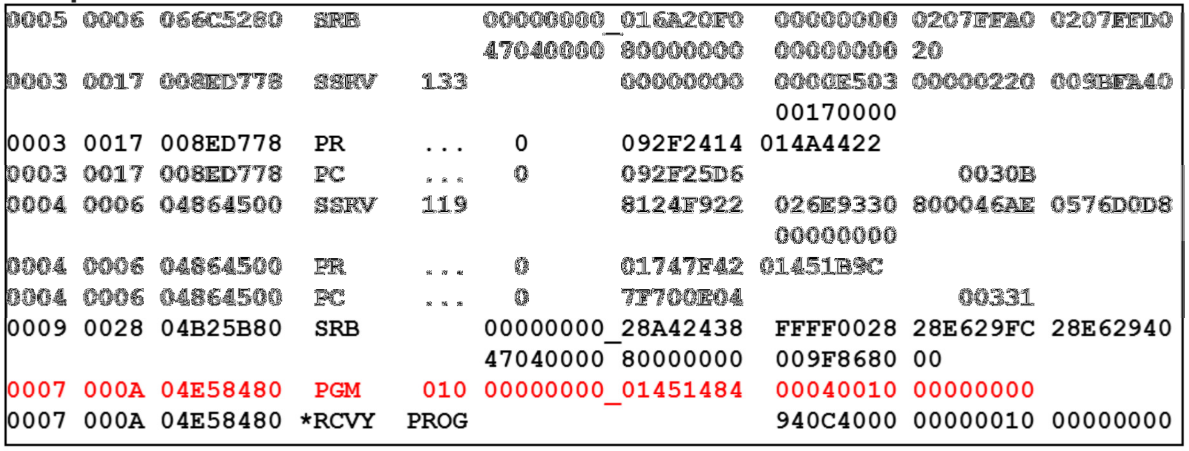
System Trace - Multi-CPU debugging considerations

- Consider the following:
  - z/OS now supports up to 100 CPUs per LPAR (or per image)
  - IPCS SYSTRACE default output is sorted by time
  - Trace entries from many CPUs are inter-mixed with each other
- Suppose you are investigating an error in the system trace that can result from some erroneous activity on one or more CPUs at around the same time (for example, a serialization issue), what would you do?
  - Scan backwards from the current entry and dig up the activity of each CPU?
  - Direct the SYSTRACE output to a file then sort by CPU number?

IP SYSTRACE SORTCPU(date,time,N) TIME(LOCAL)

- New parameter SORTCPU for IPCS SYSTRACE in z/OS V1.12
- Displays the trace entries of each CPU separately (in CPU ascending order):
  - N = number of entries before and after a specific time (default of N=10)
  - Specific time to be provided via date and time
  - Date is in format of mm/dd/yy
  - Time is in format of hh:mm:ss:dddddd
  - If no date and time are supplied, all entries are shown

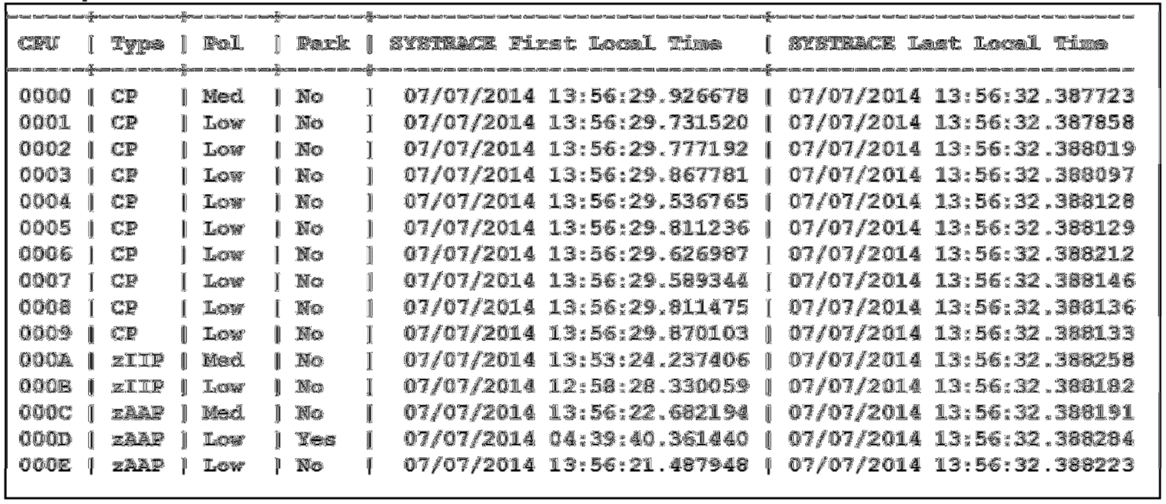
IP SYSTRACE SORTCPU example



```
0005 0006 066C5280  SRB        00000000_016A20F0  00000000 0207FFA0 0207FFD0
                               47040000 80000000  00000000 20
0003 0017 008ED778  SSRV  133           00000000  000CE503 00000220 009BFA40
                                                  00170000
0003 0017 008ED778  PR    ...    0      092F2414 014A4422
0003 0017 008ED778  PC    ...    0      092F25D6             0030B
0004 0006 04864500  SSRV  119           8124F922  026E9330 800046AE 0576D0D8
                                                  00000000
0004 0006 04864500  PR    ...    0      01747F42 01451B9C
0004 0006 04864500  PC    ...    0      7F700E04             00331
0009 0028 04B25B80  SRB        00000000_28A42438  FFFF0028 28E629FC 28E62940
                               47040000 80000000  009F8680 00
0007 000A 04E58480  PGM   010  00000000_01451484  00040010 00000000
0007 000A 04E58480 *RCVY PROG                     940C4000 00000010 00000000
```

System trace from SVC Dump taken for an ABEND0C4
Timestamp for PGM 10 entry is 13:56:32.246684245

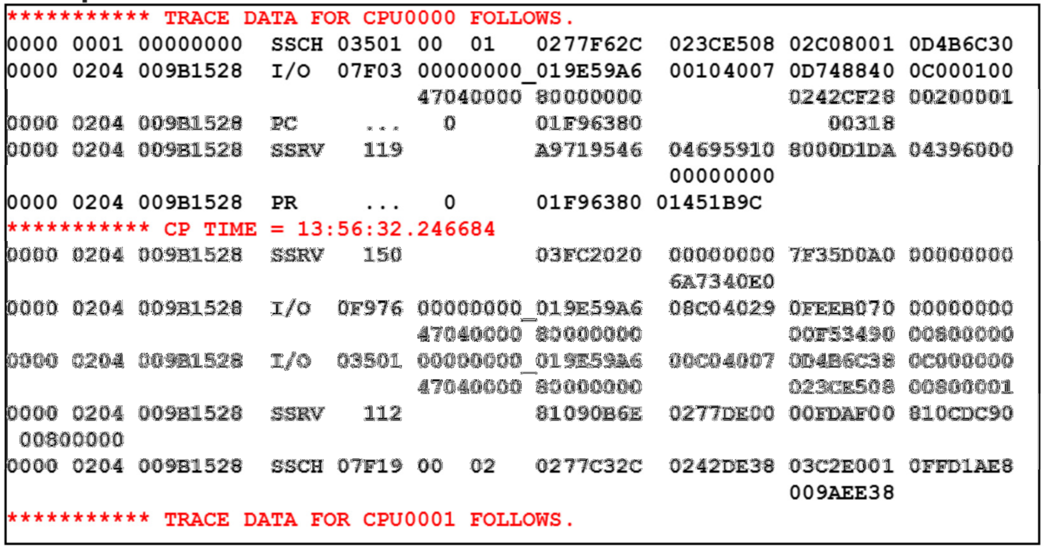
IP SYSTRACE SORTCPU example…



```
-----+-------+-------+-------+--------------------------------+----------------------------
CPU  | Type  | Pol   | Park  | SYSTRACE First Local Time      | SYSTRACE Last Local Time
-----+-------+-------+-------+--------------------------------+----------------------------
0000 | CP    | Med   | No    |  07/07/2014 13:56:29.926678 | 07/07/2014 13:56:32.387723
0001 | CP    | Low   | No    |  07/07/2014 13:56:29.731520 | 07/07/2014 13:56:32.387858
0002 | CP    | Low   | No    |  07/07/2014 13:56:29.777192 | 07/07/2014 13:56:32.388019
0003 | CP    | Low   | No    |  07/07/2014 13:56:29.867781 | 07/07/2014 13:56:32.388097
0004 | CP    | Low   | No    |  07/07/2014 13:56:29.536765 | 07/07/2014 13:56:32.388128
0005 | CP    | Low   | No    |  07/07/2014 13:56:29.811236 | 07/07/2014 13:56:32.388129
0006 | CP    | Low   | No    |  07/07/2014 13:56:29.626987 | 07/07/2014 13:56:32.388212
0007 | CP    | Low   | No    |  07/07/2014 13:56:29.589344 | 07/07/2014 13:56:32.388146
0008 | CP    | Low   | No    |  07/07/2014 13:56:29.811475 | 07/07/2014 13:56:32.388136
0009 | CP    | Low   | No    |  07/07/2014 13:56:29.870103 | 07/07/2014 13:56:32.388133
000A | zIIP  | Med   | No    |  07/07/2014 13:53:24.237406 | 07/07/2014 13:56:32.388258
000B | zIIP  | Low   | No    |  07/07/2014 12:58:28.330059 | 07/07/2014 13:56:32.388182
000C | zAAP  | Med   | No    |  07/07/2014 13:56:22.682194 | 07/07/2014 13:56:32.388191
000D | zAAP  | Low   | Yes   |  07/07/2014 04:39:40.361440 | 07/07/2014 13:56:32.388284
000E | zAAP  | Low   | No    |  07/07/2014 13:56:21.487948 | 07/07/2014 13:56:32.388223
```

IP SYSTRACE STATUSTIME(LOCAL) shows 15 CPUs

IP SYSTRACE SORTCPU example…



```
*********** TRACE DATA FOR CPU0000 FOLLOWS.
0000 0001 00000000  SSCH 03501 00  01   0277F62C  023CE508 02C08001 0D4B6C30
0000 0204 009B1528  I/O  07F03 00000000_019E59A6  00104007 0D748840 0C000100
                           47040000 80000000           0242CF28 00200001
0000 0204 009B1528  PC     ...   0      01F96380            00318
0000 0204 009B1528  SSRV   119          A9719546  04695910 8000D1DA 04396000
                                                  00000000
0000 0204 009B1528  PR     ...   0      01F96380 01451B9C
*********** CP TIME = 13:56:32.246684
0000 0204 009B1528  SSRV   150          03FC2020  00000000 7F35D0A0 00000000
                                                  6A7340E0
0000 0204 009B1528  I/O  0F976 00000000_019E59A6  08C04029 0FEEB070 00000000
                           47040000 80000000           00F53490 00800000
0000 0204 009B1528  I/O  03501 00000000_019E59A6  00C04007 0D4B6C38 0C000000
                           47040000 80000000           023CE508 00800001
0000 0204 009B1528  SSRV   112          81090B6E  0277DE00 00FDAF00 810CDC90
 00800000
0000 0204 009B1528  SSCH 07F19 00  02   0277C32C  0242DE38 03C2E001 0FFD1AE8
                                                           009AEE38
*********** TRACE DATA FOR CPU0001 FOLLOWS.
```

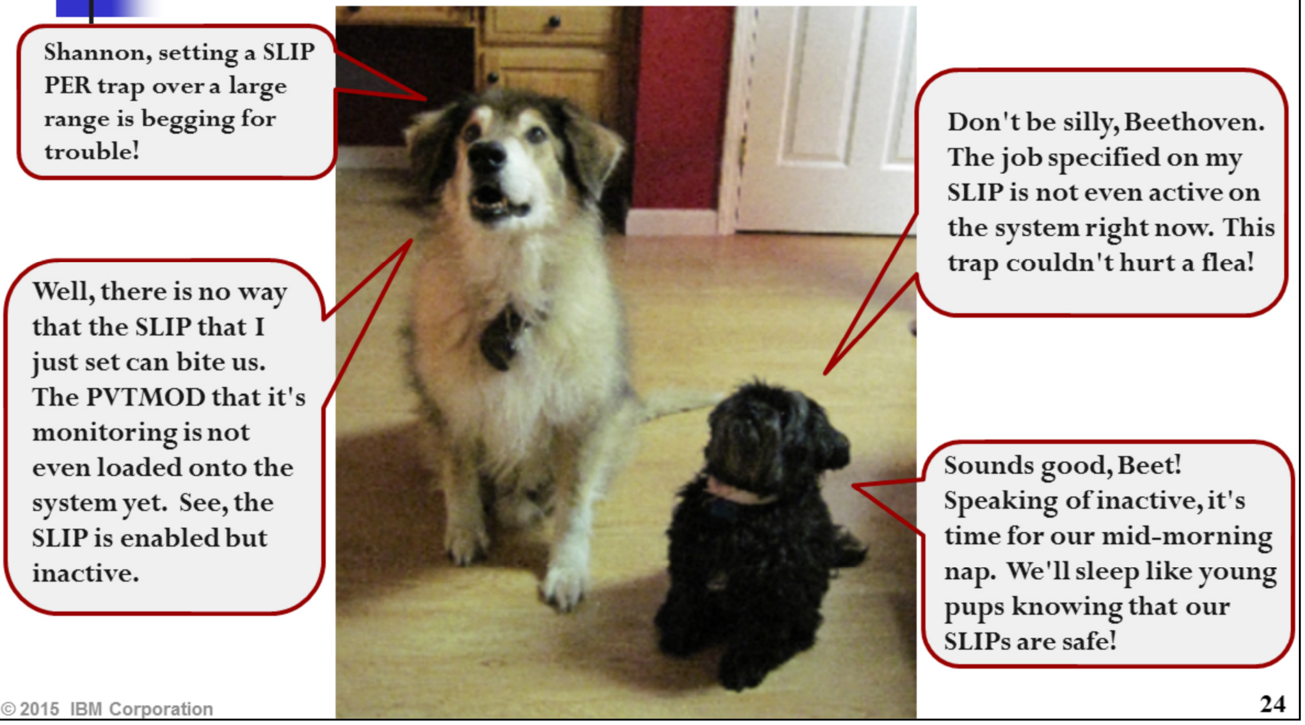
IP SYSTRACE SORTCPU(07/07/14, 13:56:32.246684,5) TIME(LOCAL)

to investigate activity on all other CPUs at the time of the error

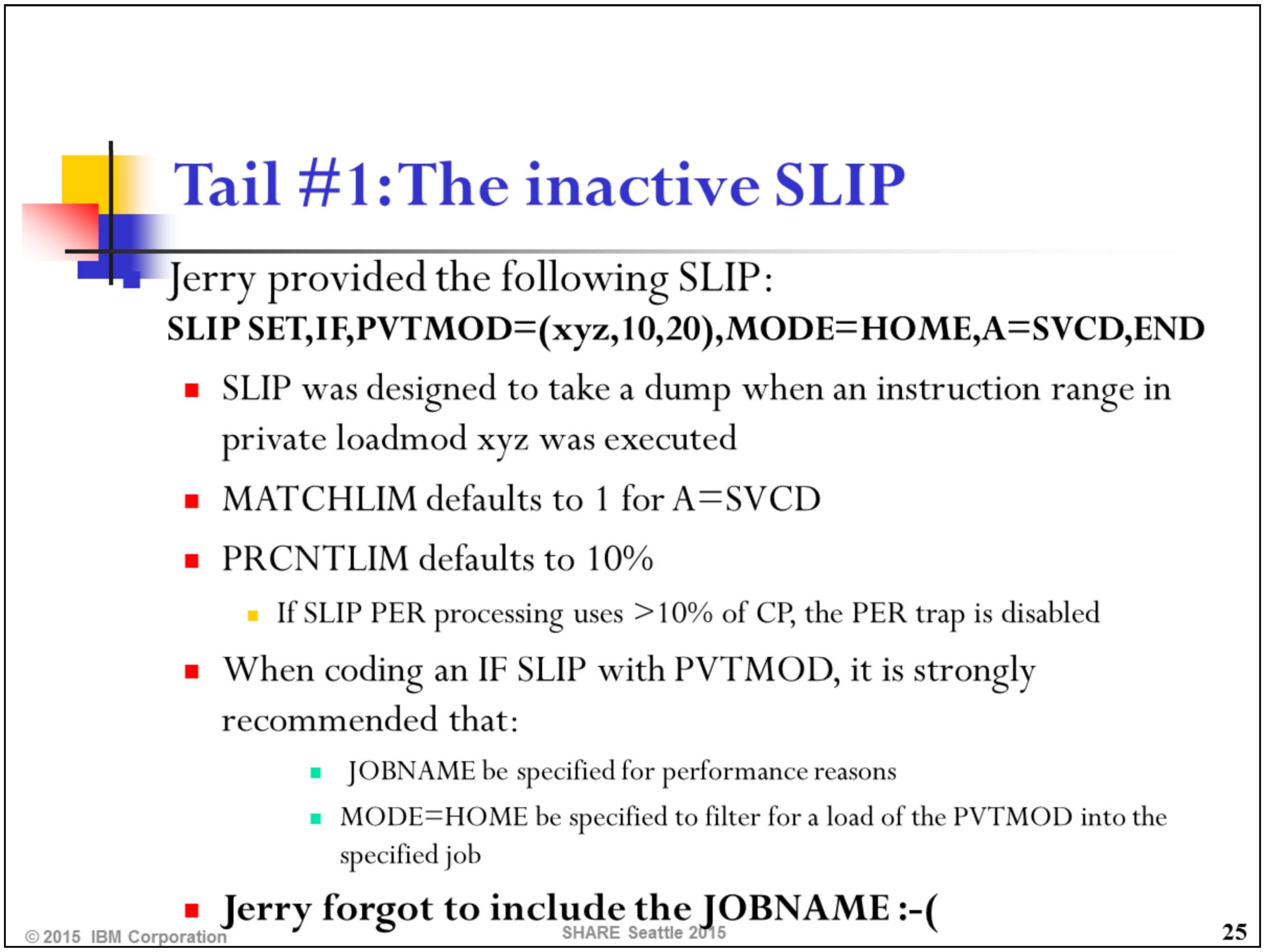
SLIP PVTMOD

Old dogs' tall tails

Old dogs' tall tails about SLIPs


Shannon, setting a SLIP PER trap over a large range is begging for trouble!
Don't be silly, Beethoven. The job specified on my SLIP is not even active on the system right now. This trap couldn't hurt a flea!
Well, there is no way that the SLIP that I just set can bite us. The PVTMOD that it's monitoring is not even loaded onto the system yet. See, the SLIP is enabled but inactive.
Sounds good, Beet! Speaking of inactive, it's time for our mid-morning nap. We'll sleep like young pups knowing that our SLIPs are safe!

SLIP processing is powerful, complex, and subtle. It's an extremely valuable tool in the debugger's tool kit. Used carefully, it can be a tremendous aid in catching difficult problems, gathering detailed documentation, and even saving systems. However, with power and complexity comes caution and responsibility. It is easy and even reasonable to assume that a SLIP for a PVTMOD or Job that is not even on the system can have no potential for system impact, but this assumption is false and underestimates the intricacies of SLIP. While we may understand how a SLIP trap works externally, internally SLIP has constraints related to performance, environment, and complexity that defy our expectations. Therefore, it is best practice to code SLIPs to be as conservative and safe as possible and to be vigilant to the system's performance any time you place a new PER SLIP trap on the system.



Tail #1:The inactive SLIP

Jerry provided the following SLIP:

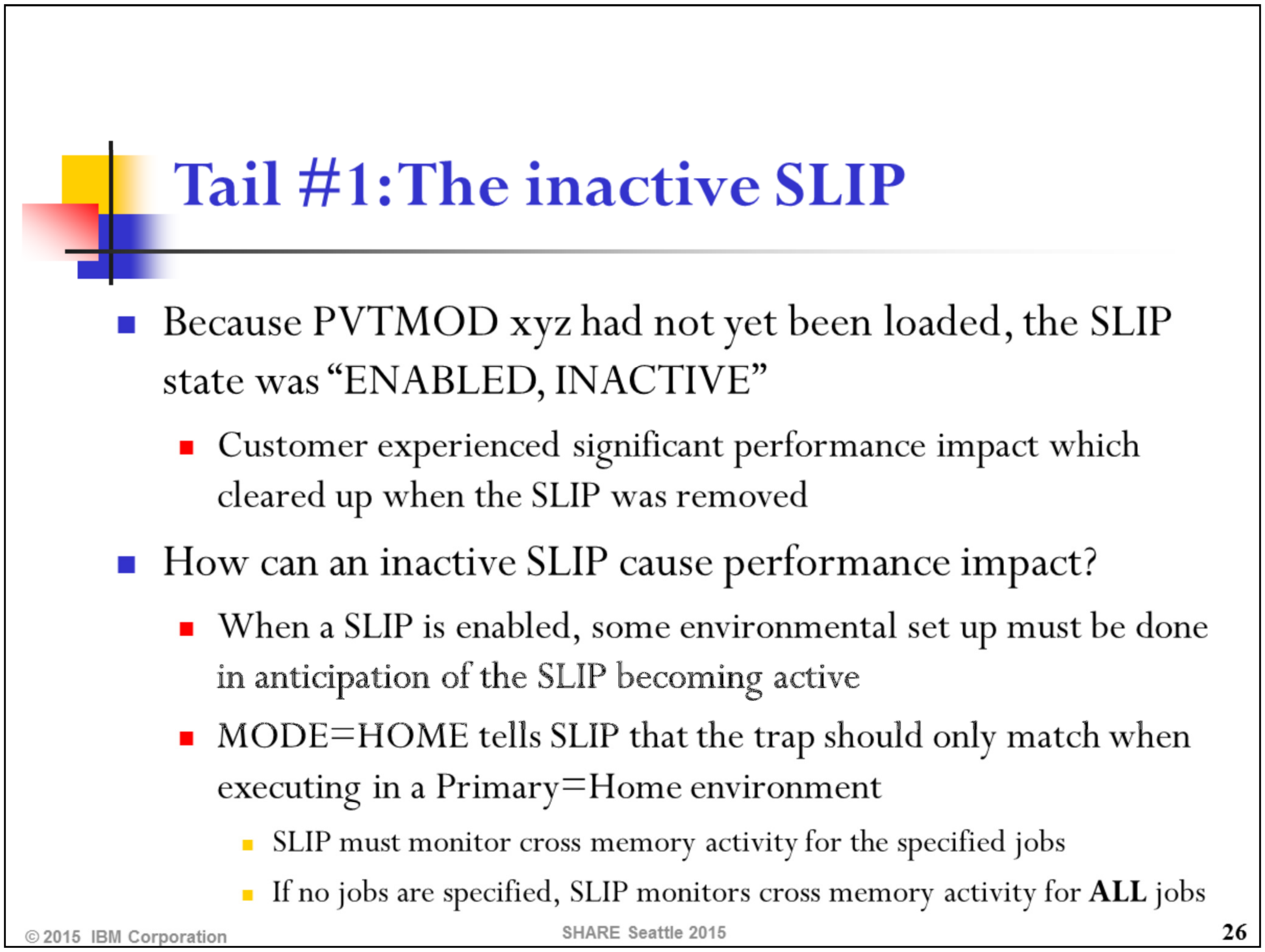
SLIP SET,IF,PVTMOD=(xyz,10,20),MODE=HOME,A=SVCD,END

- SLIP was designed to take a dump when an instruction range in private loadmod xyz was executed
- MATCHLIM defaults to 1 for A=SVCD
- PRCNTLIM defaults to 10%
  - If SLIP PER processing uses >10% of CP, the PER trap is disabled
- When coding an IF SLIP with PVTMOD, it is strongly recommended that:
  - JOBNAME be specified for performance reasons
  - MODE=HOME be specified to filter for a load of the PVTMOD into the specified job
- **Jerry forgot to include the JOBNAME :-(**

SLIP PRCNTLIM (PL) processing monitors how much time is being spent in SLIP as a result of a PER trap being enabled. If this time spent in SLIP is greater than a specified percentage of the total CPU, SLIP processing will automatically disable the PER trap. The default for PERCNTLIM is 10%, but it can be set as low as 1%. It is a good idea to code a conservative PRCNTLIM, especially on systems which are very sensitive to performance impact.

Note that PER processing is highly efficient. Intelligent and conservative SLIP trap design will ensure that the trap performs efficiently and without impact.

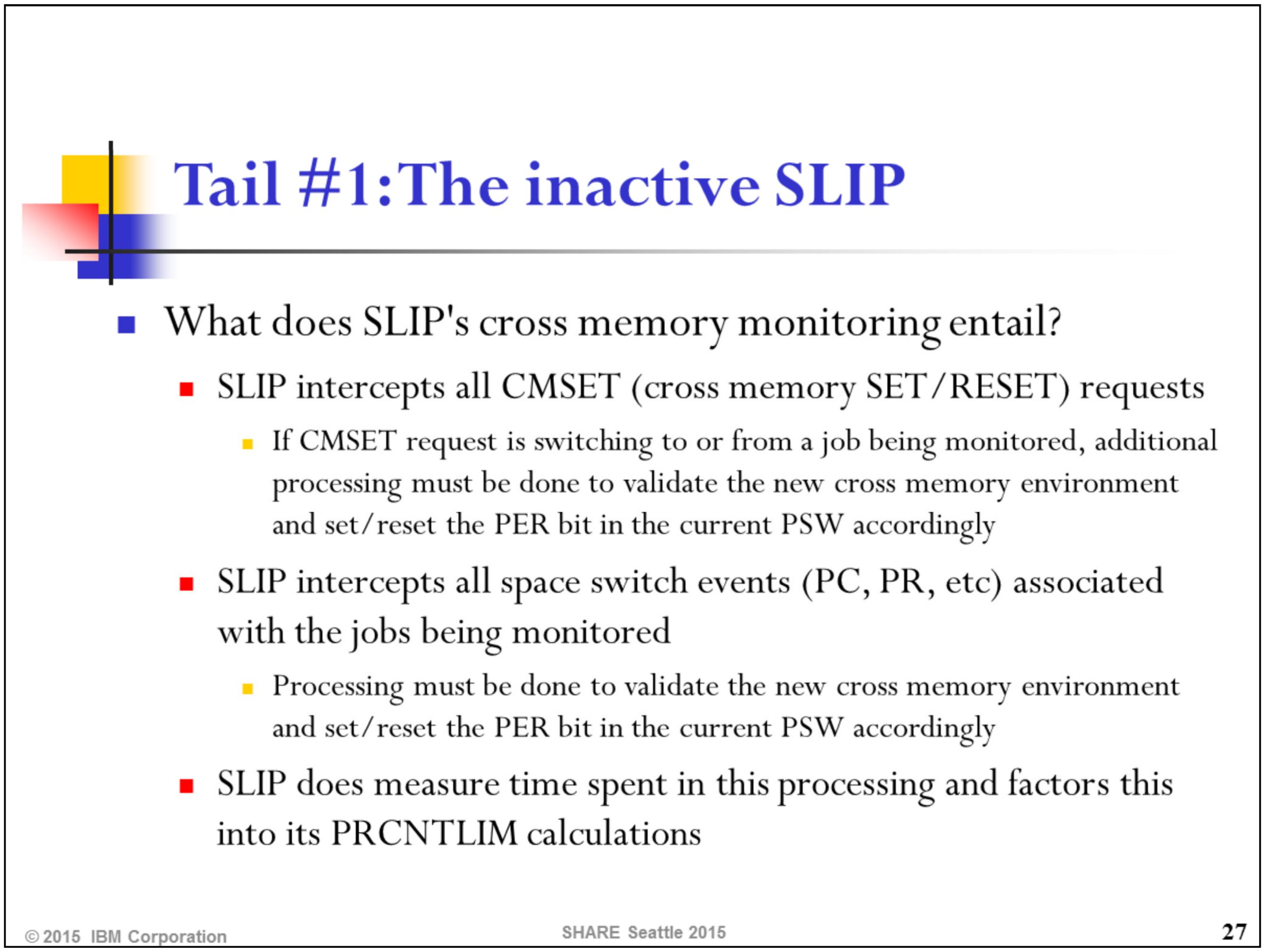
Whenever possible, JOBNAME=,MODE=HOME should be coded on an IF or SBT PVTMOD PER trap.



Tail #1:The inactive SLIP

- Because PVTMOD xyz had not yet been loaded, the SLIP state was "ENABLED, INACTIVE"
  - Customer experienced significant performance impact which cleared up when the SLIP was removed
- How can an inactive SLIP cause performance impact?
  - When a SLIP is enabled, some environmental set up must be done in anticipation of the SLIP becoming active
  - MODE=HOME tells SLIP that the trap should only match when executing in a Primary=Home environment
    - SLIP must monitor cross memory activity for the specified jobs
    - If no jobs are specified, SLIP monitors cross memory activity for **ALL** jobs

MODE=HOME means that the PRIMARY address space (the space in which execution is occurring) is the same as the HOME address space (the space where the job originated). When SLIP is monitoring for MODE=HOME and an an event occurs that could affect the cross memory status, SLIP must get control to see if it needs to adjust the PER bit in the PSW of the executing unit of work.



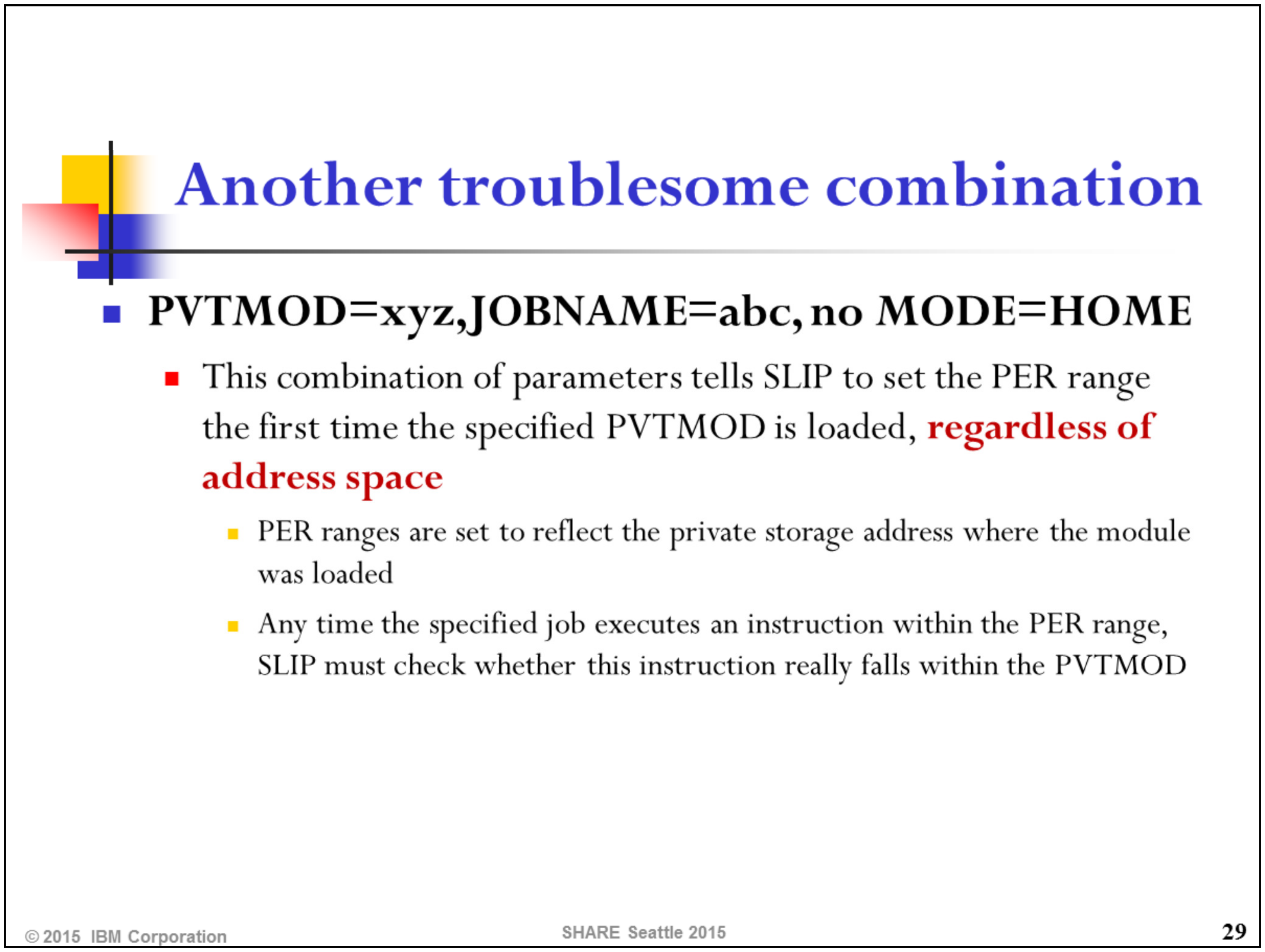
Tail #1:The inactive SLIP

- What does SLIP's cross memory monitoring entail?
  - SLIP intercepts all CMSET (cross memory SET/RESET) requests
    - If CMSET request is switching to or from a job being monitored, additional processing must be done to validate the new cross memory environment and set/reset the PER bit in the current PSW accordingly
  - SLIP intercepts all space switch events (PC, PR, etc) associated with the jobs being monitored
    - Processing must be done to validate the new cross memory environment and set/reset the PER bit in the current PSW accordingly
  - SLIP does measure time spent in this processing and factors this into its PRCNTLIM calculations

SLIP cross memory monitoring is an **extremely** performance-sensitive path. It is also a path that may be executed with a high frequency for jobs that do a great deal of cross memory activity. Therefore it is important to limit monitoring of cross memory activity to a small number of address spaces. Otherwise, system performance problems can result. If PER interrupts are occurring, PRCNTLIM processing will offer some protection. However, if PER interrupts are not occurring, PRCNTLIM processing will not be able to provide protection.

Tail #1:The inactive SLIP

- So why didn't PRCNTLIM protect the system?
  - PRCNTLIM checks are only made when a PER interrupt occurs
  - When a SLIP is inactive, PER ranges have not been set up in the PER control registers yet, so no PER interrupts take place



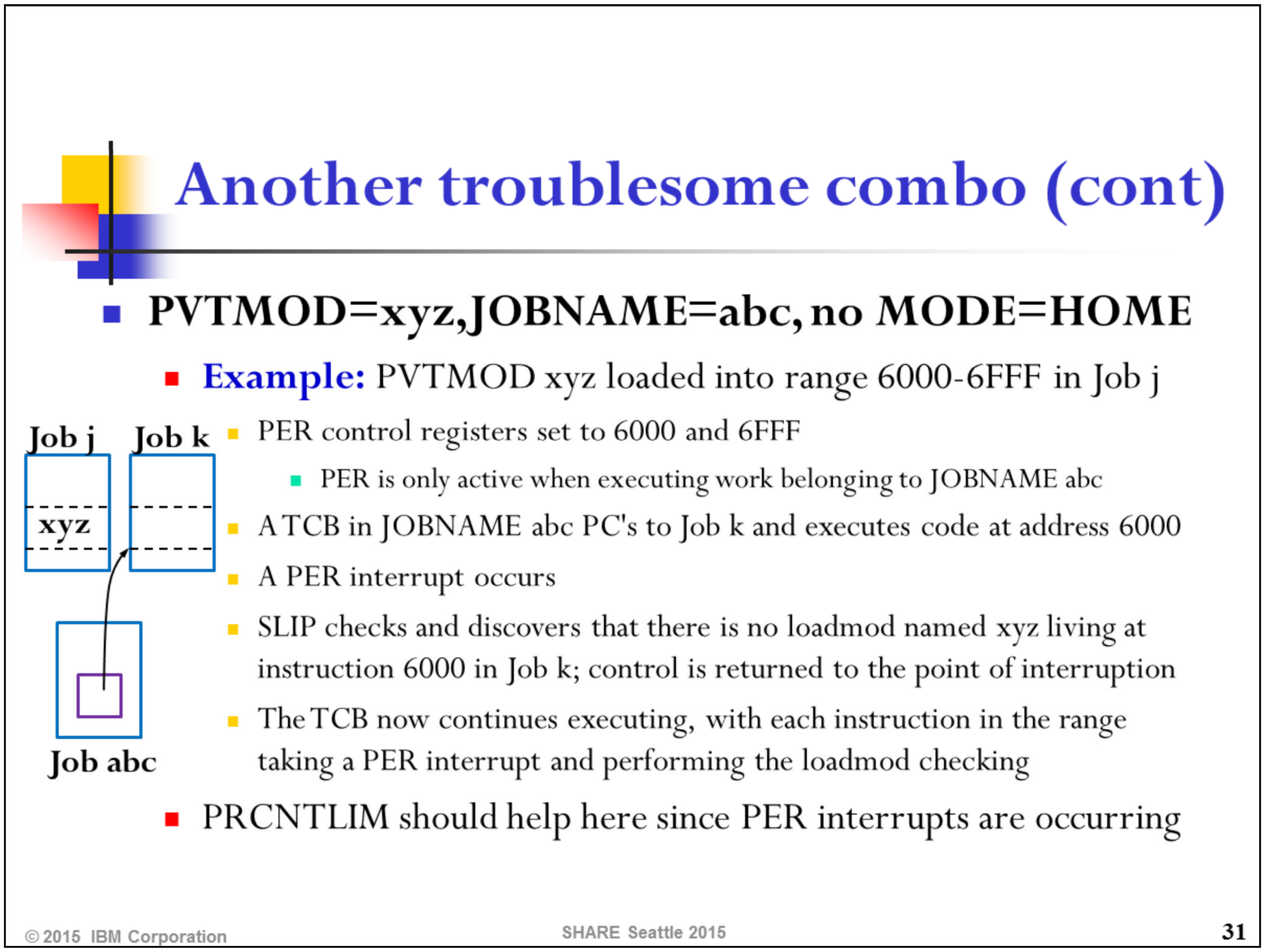
Another troublesome combination

- **PVTMOD=xyz,JOBNAME=abc, no MODE=HOME**
  - This combination of parameters tells SLIP to set the PER range the first time the specified PVTMOD is loaded, **regardless of address space**
    - PER ranges are set to reflect the private storage address where the module was loaded
    - Any time the specified job executes an instruction within the PER range, SLIP must check whether this instruction really falls within the PVTMOD

Specifying JOBNAME without MODE=HOME on a PVTMOD SLIP can result in significant performance impact. The following slide details why.

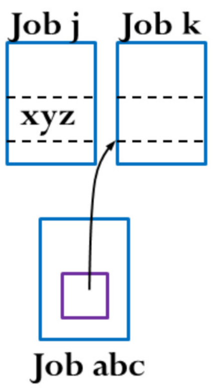
Another troublesome combo (cont)

- **PVTMOD=,JOBNAME=,MODE=HOME**
  - The dual role of the MODE=HOME parameter on a PVTMOD PER trap
    - On SLIP enablement:
      - MODE=HOME used in conjunction with JOBNAME enforces that the trap is made ACTIVE only when the specified PVTMOD is loaded into the specified JOB
    - During normal system execution (as seen previously):
      - MODE=HOME enforces that SLIP is only monitoring for the PER event when the event occurs in a cross memory environment where the primary and home ASIDs are equal, that is, PASID=HASID

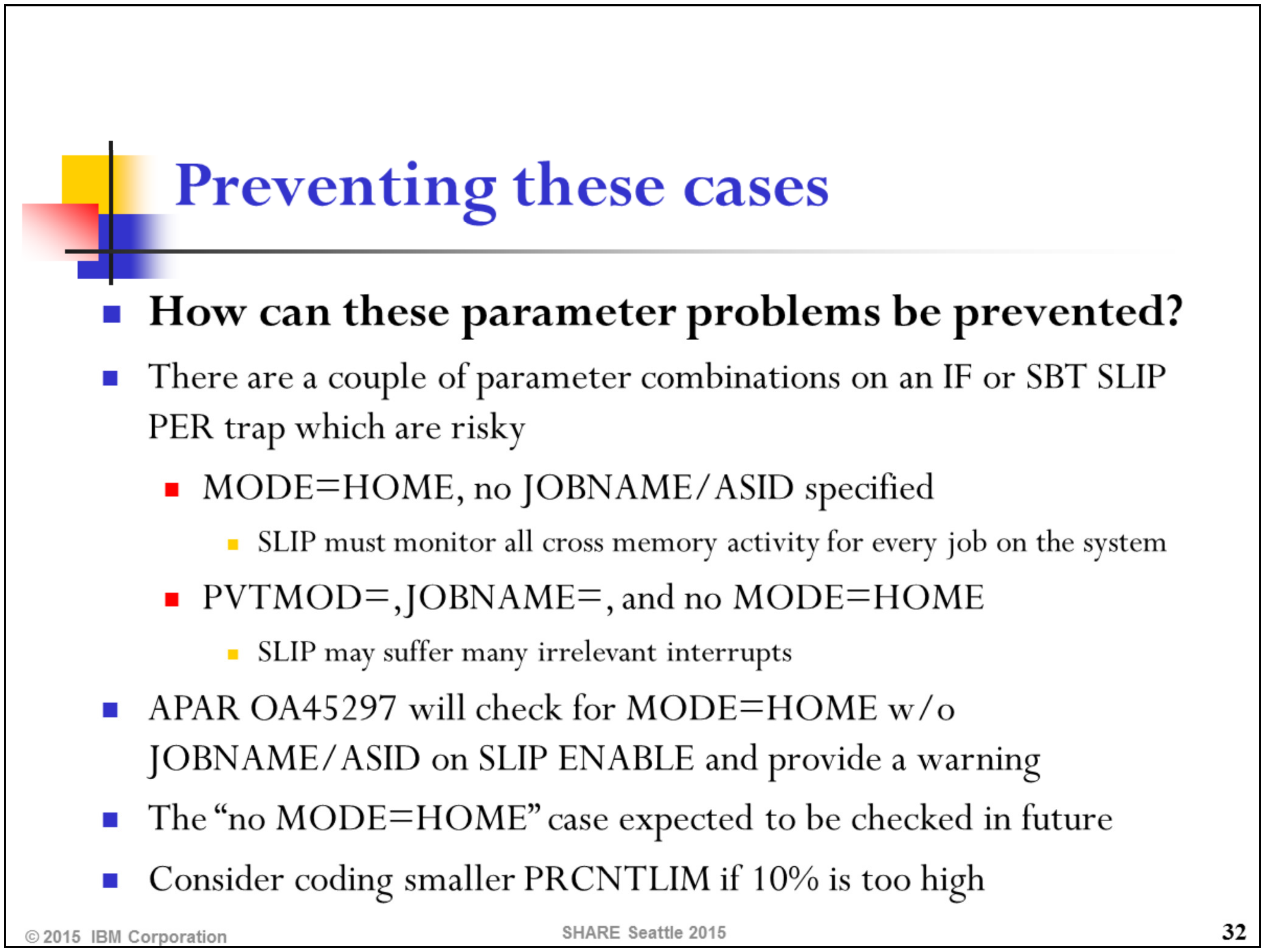


Another troublesome combo (cont)

- PVTMOD=xyz,JOBNAME=abc, no MODE=HOME
  - **Example:** PVTMOD xyz loaded into range 6000-6FFF in Job j
    - PER control registers set to 6000 and 6FFF
      - PER is only active when executing work belonging to JOBNAME abc
    - A TCB in JOBNAME abc PC's to Job k and executes code at address 6000
    - A PER interrupt occurs
    - SLIP checks and discovers that there is no loadmod named xyz living at instruction 6000 in Job k; control is returned to the point of interruption
    - The TCB now continues executing, with each instruction in the range taking a PER interrupt and performing the loadmod checking
  - PRCNTLIM should help here since PER interrupts are occurring

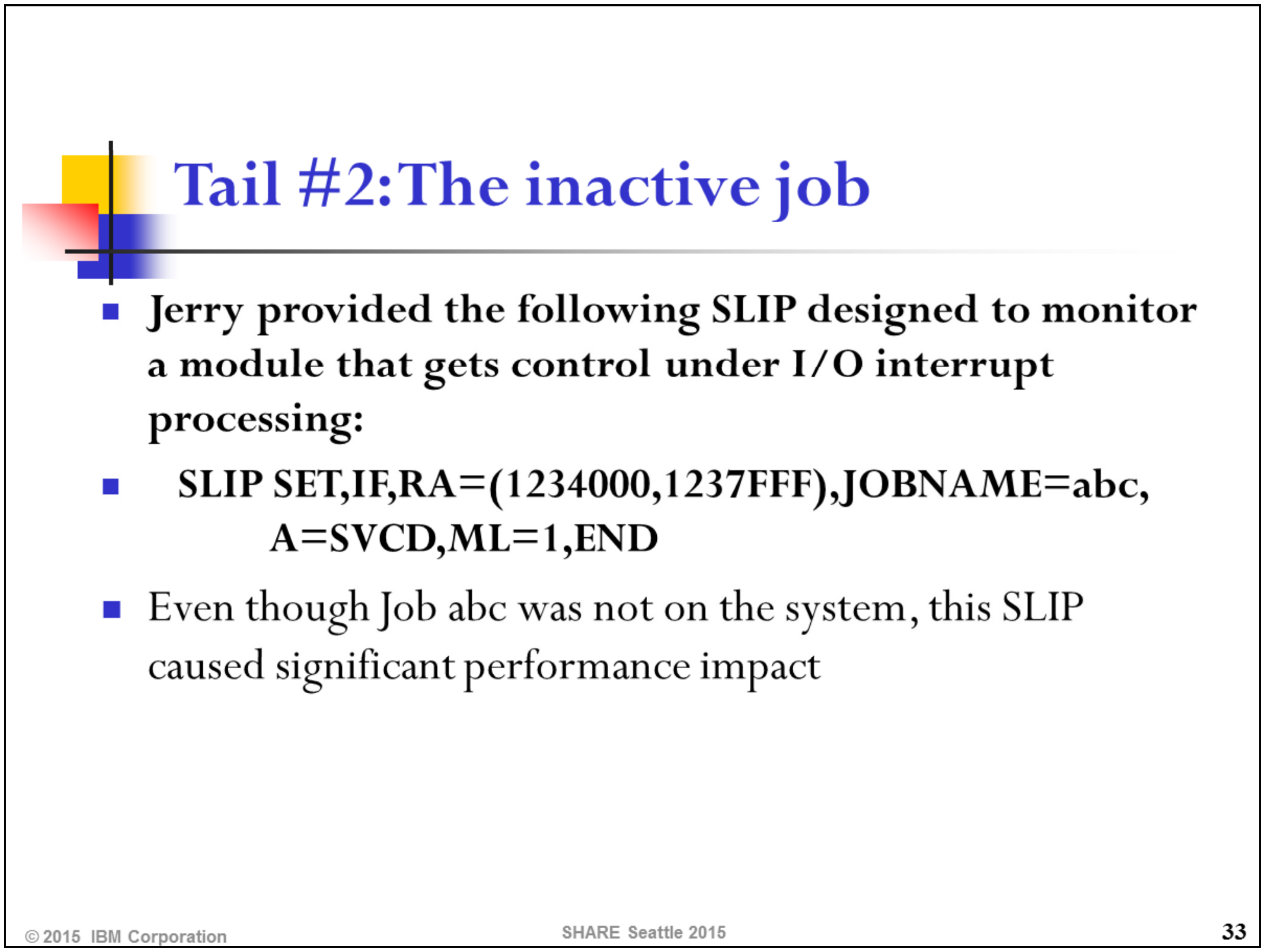


Whether the above combination of SLIP parameters causes a problem is really a function of whether the PVTMOD could be loaded in multiple jobs, and if so, where it gets placed in storage. If no other jobs load the PVTMOD, you could probably get away without coding MODE=HOME. However, the point of these slides is to be conservative with SLIP and make no assumptions. Therefore, it is best to explicitly specify the desired environment to the maximum degree possible.



Preventing these cases

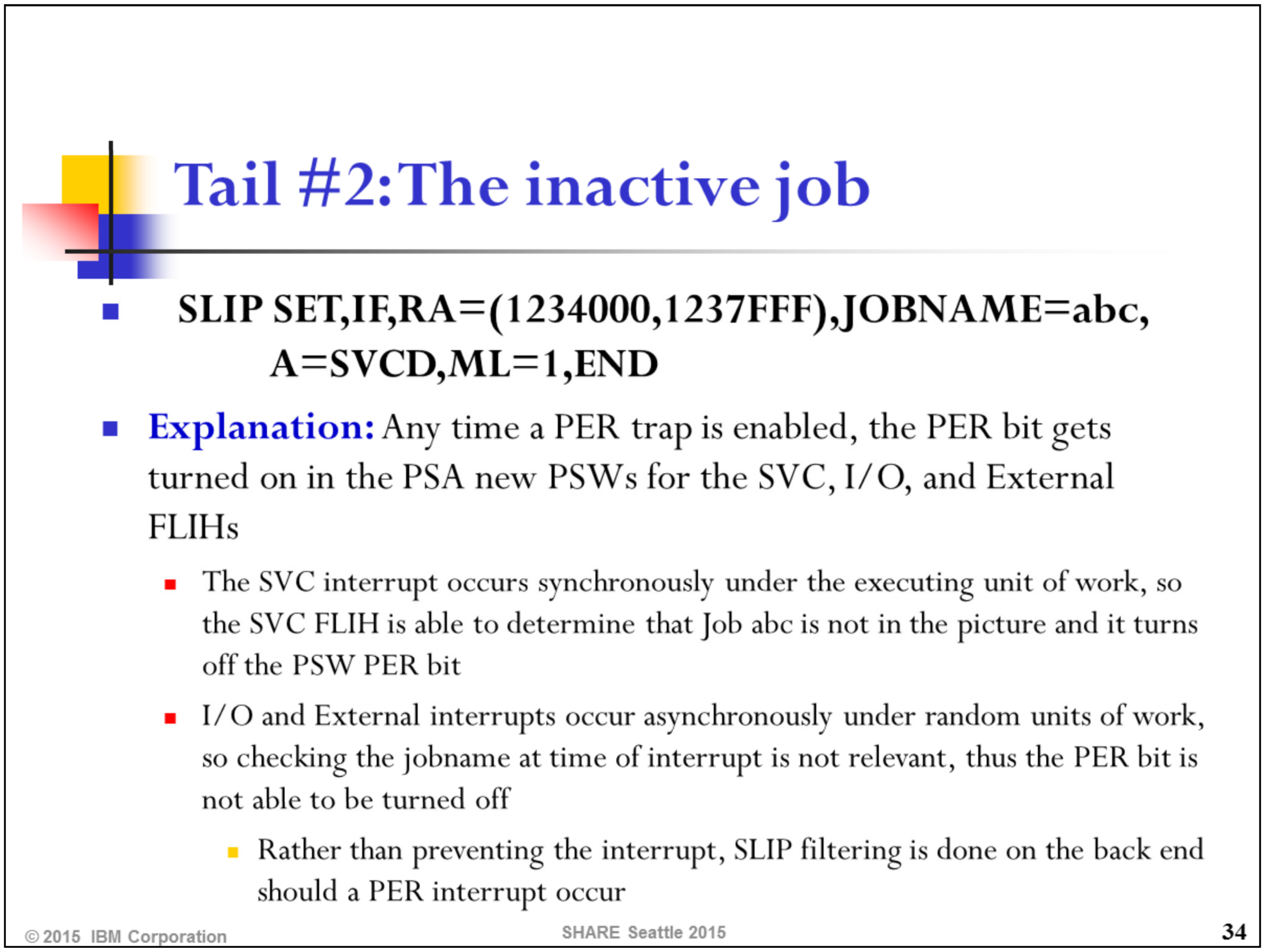
- **How can these parameter problems be prevented?**
- There are a couple of parameter combinations on an IF or SBT SLIP PER trap which are risky
  - MODE=HOME, no JOBNAME/ASID specified
    - SLIP must monitor all cross memory activity for every job on the system
  - PVTMOD=,JOBNAME=, and no MODE=HOME
    - SLIP may suffer many irrelevant interrupts
- APAR OA45297 will check for MODE=HOME w/o JOBNAME/ASID on SLIP ENABLE and provide a warning
- The “no MODE=HOME” case expected to be checked in future
- Consider coding smaller PRCNTLIM if 10% is too high



Tail #2:The inactive job

- **Jerry provided the following SLIP designed to monitor a module that gets control under I/O interrupt processing:**
- **SLIP SET,IF,RA=(1234000,1237FFF),JOBNAME=abc,**
  A=SVCD,ML=1,END
- Even though Job abc was not on the system, this SLIP caused significant performance impact

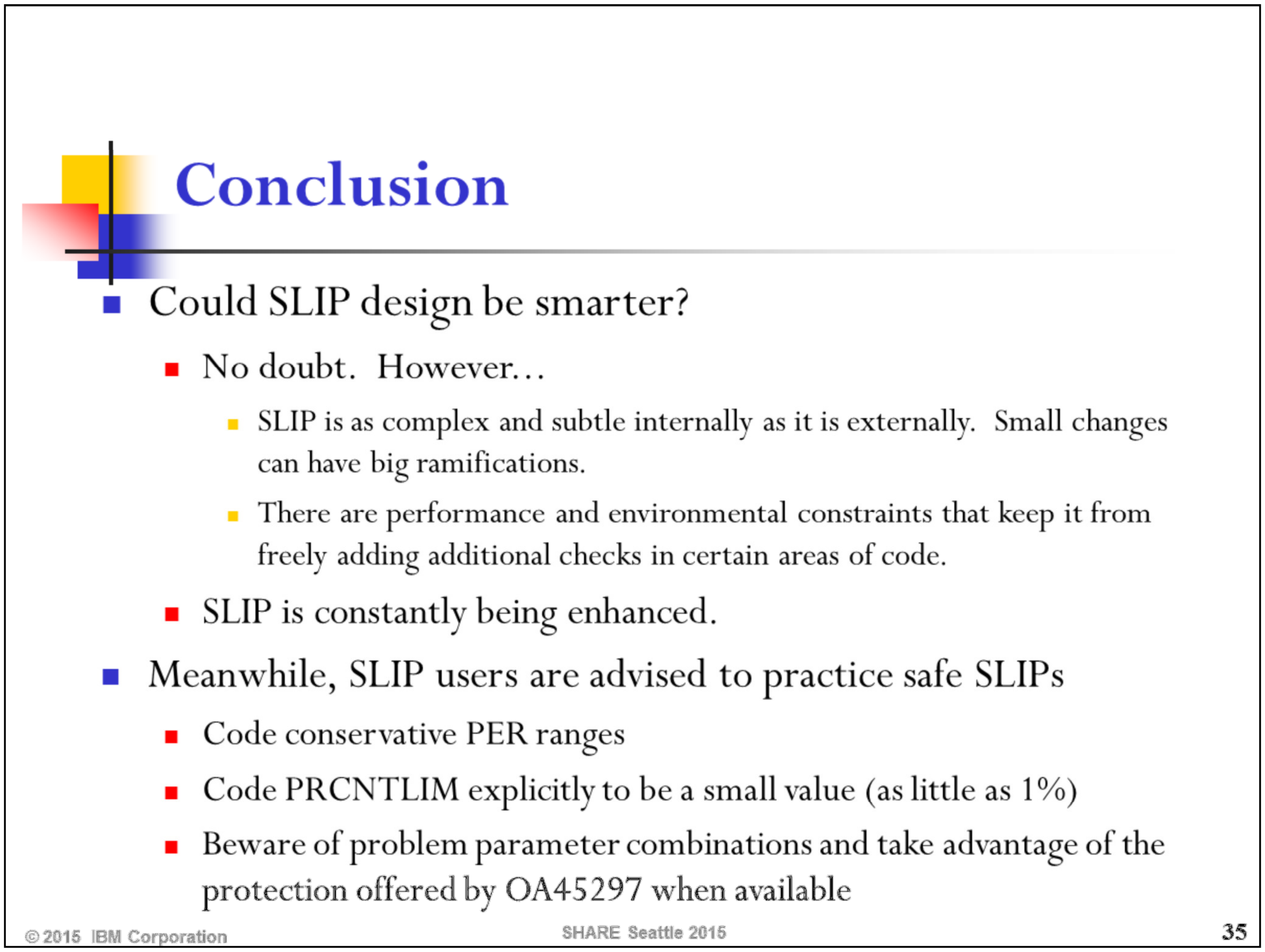
This code targeted by the SLIP was media manager code that ran as an extension of the I/O interrupt. It was surprising to see that this SLIP caused overhead even when the jobname specified on the SLIP was not on the system.



Tail #2:The inactive job

- **SLIP SET,IF,RA=(1234000,1237FFF),JOBNAME=abc, A=SVCD,ML=1,END**
- **Explanation:** Any time a PER trap is enabled, the PER bit gets turned on in the PSA new PSWs for the SVC, I/O, and External FLIHs
  - The SVC interrupt occurs synchronously under the executing unit of work, so the SVC FLIH is able to determine that Job abc is not in the picture and it turns off the PSW PER bit
  - I/O and External interrupts occur asynchronously under random units of work, so checking the jobname at time of interrupt is not relevant, thus the PER bit is not able to be turned off
    - Rather than preventing the interrupt, SLIP filtering is done on the back end should a PER interrupt occur

SLIP's goal is to prevent as many unnecessary PER interrupts as possible through careful control of the PER bit in the PSW of active units of work. Setting the PER bit only for units of work associated with the specified jobname is one way that SLIP accomplishes this. However, as this example points out, there are some environments where SLIP cannot exert this level of control in preventing the interrupt, and instead must take the interrupt and then filter it via SLIP software.



Conclusion

- Could SLIP design be smarter?
  - No doubt. However…
    - SLIP is as complex and subtle internally as it is externally. Small changes can have big ramifications.
    - There are performance and environmental constraints that keep it from freely adding additional checks in certain areas of code.
  - SLIP is constantly being enhanced.
- Meanwhile, SLIP users are advised to practice safe SLIPs
  - Code conservative PER ranges
  - Code PRCNTLIM explicitly to be a small value (as little as 1%)
  - Beware of problem parameter combinations and take advantage of the protection offered by OA45297 when available

z/OS Tools & Toys REXX Exec's

Do you know?

z/OS Tools & Toys – Diagnostic Tools

- There are 'Tools & Toys' REXX Exec's available in IPCS for dump diagnosis. They are as-is, not documented, not supported and not warranted
- These Exec's can be invoked by:
  - Issuing IP exec_name <input>, or
  - Selecting from the IPCS 2.6i panel (=2.6i on command line)
- Use IP exec-name HELP to get more information
- See http://www-03.ibm.com/systems/z/os/zos/features/unix/tools/ and then click on Code Samples

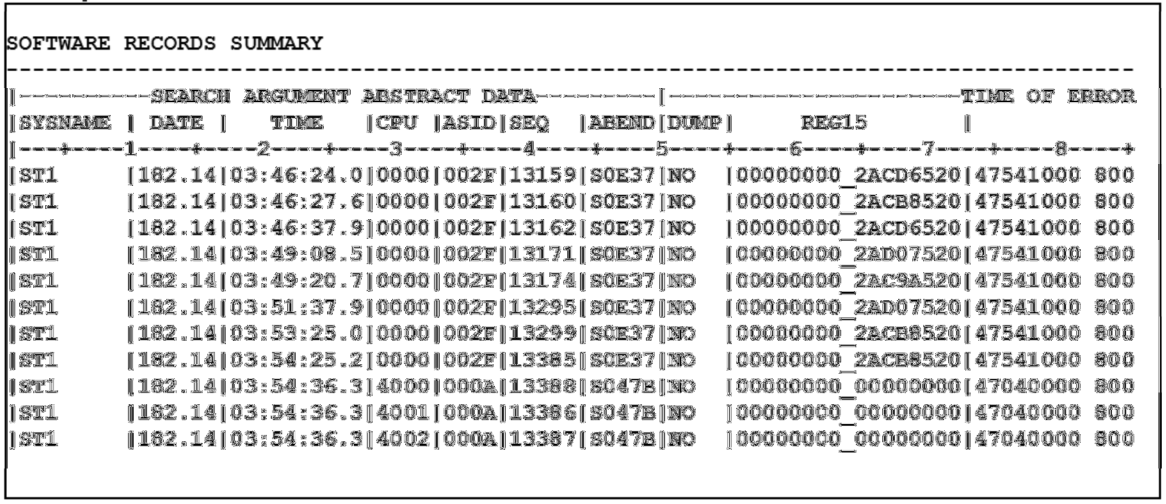
IP IEAVLOGD

- A z/OS 'Tools and Toys' REXX Exec – as-is, not documented, not supported, not warranted
- Provides one line summary of software/symptom/MIH records in:
  - input formatted LOGREC dataset, or
  - VERBX LOGDATA output from the current dump

```
===> IP IEAVLOGD
```

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

IP IEAVLOGD example

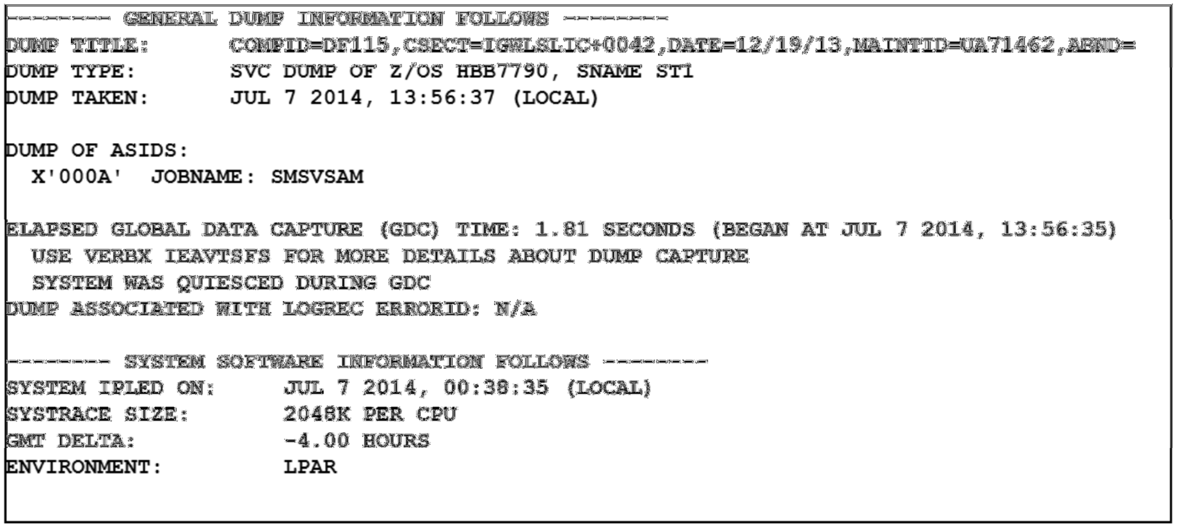


```
SOFTWARE RECORDS SUMMARY
-----------------------------------------------------------------------------------------
|-----------SEARCH ARGUMENT ABSTRACT DATA----------[-----------------------TIME OF ERROR
|SYSNAME | DATE |    TIME    |CPU |ASID|SEQ  |ABEND|DUMP|     REG15         |
|---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+
|ST1      |182.14|03:46:24.0|0000|002F|13159|S0E37|NO   |00000000_2ACD6520|47541000 800
|ST1      |182.14|03:46:27.6|0000|002F|13160|S0E37|NO   |00000000_2ACB8520|47541000 800
|ST1      |182.14|03:46:37.9|0000|002F|13162|S0E37|NO   |00000000_2ACD6520|47541000 800
|ST1      |182.14|03:49:08.5|0000|002F|13171|S0E37|NO   |00000000_2AD07520|47541000 800
|ST1      |182.14|03:49:20.7|0000|002F|13174|S0E37|NO   |00000000_2AC9A520|47541000 800
|ST1      |182.14|03:51:37.9|0000|002F|13295|S0E37|NO   |00000000_2AD07520|47541000 800
|ST1      |182.14|03:53:25.0|0000|002F|13299|S0E37|NO   |00000000_2ACB8520|47541000 800
|ST1      |182.14|03:54:25.2|0000|002F|13385|S0E37|NO   |00000000_2ACB8520|47541000 800
|ST1      |182.14|03:54:36.3|4000|000A|13388|S047B|NO   |00000000_00000000|47040000 800
|ST1      |182.14|03:54:36.3|4001|000A|13386|S047B|NO   |00000000_00000000|47040000 800
|ST1      |182.14|03:54:36.3|4002|000A|13387|S047B|NO   |00000000_00000000|47040000 800
```

IP IEAVDUMP

- A z/OS 'Tools and Toys' REXX Exec – as-is, not documented, not supported, not warranted
- Displays general and environmental information about the the dump
- Time of dump provided may not be accurate depending on local offset – double check with IP ST SYSTEM output

IP IEAVDUMP example

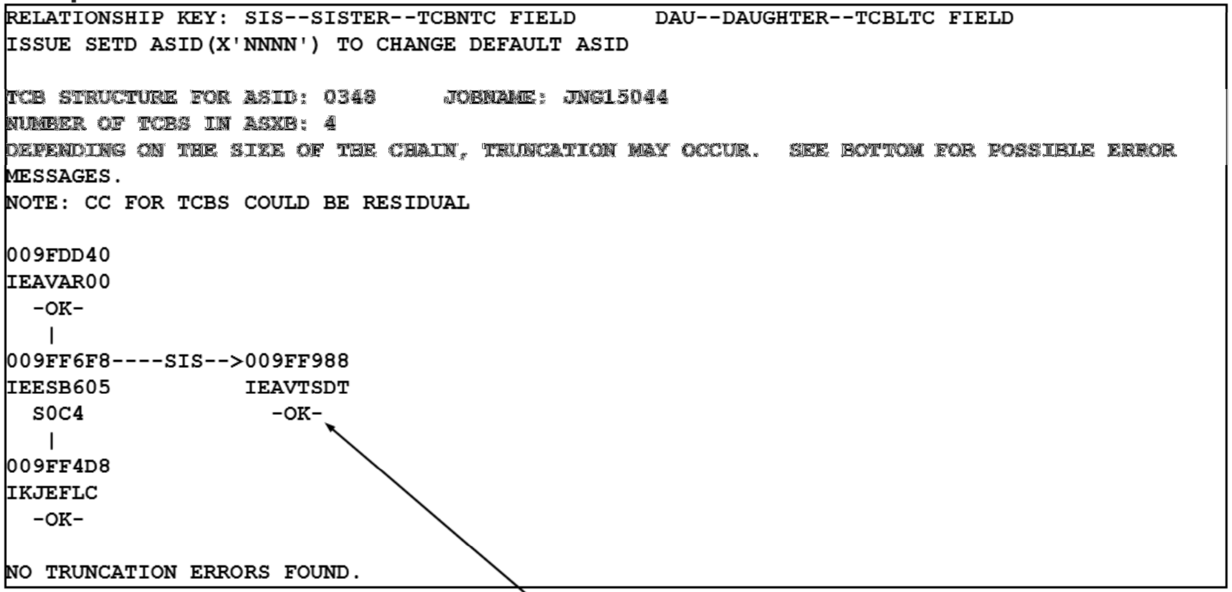


```
-------- GENERAL DUMP INFORMATION FOLLOWS --------
DUMP TITLE:       COMPID=DF115,CSECT=IGWLSLIC+0042,DATE=12/19/13,MAINTID=UA71462,ABND=
DUMP TYPE:        SVC DUMP OF Z/OS HBB7790, SNAME STI
DUMP TAKEN:       JUL 7 2014, 13:56:37 (LOCAL)

DUMP OF ASIDS:
  X'000A'  JOBNAME: SMSVSAM

ELAPSED GLOBAL DATA CAPTURE (GDC) TIME: 1.81 SECONDS (BEGAN AT JUL 7 2014, 13:56:35)
  USE VERBX IEAVTSFS FOR MORE DETAILS ABOUT DUMP CAPTURE
  SYSTEM WAS QUIESCED DURING GDC
DUMP ASSOCIATED WITH LOGREC ERRORID: N/A

-------- SYSTEM SOFTWARE INFORMATION FOLLOWS --------
SYSTEM IPLED ON:      JUL 7 2014, 00:38:35 (LOCAL)
SYSTRACE SIZE:        2048K PER CPU
GMT DELTA:            -4.00 HOURS
ENVIRONMENT:          LPAR
```

IP IEAVTCBM

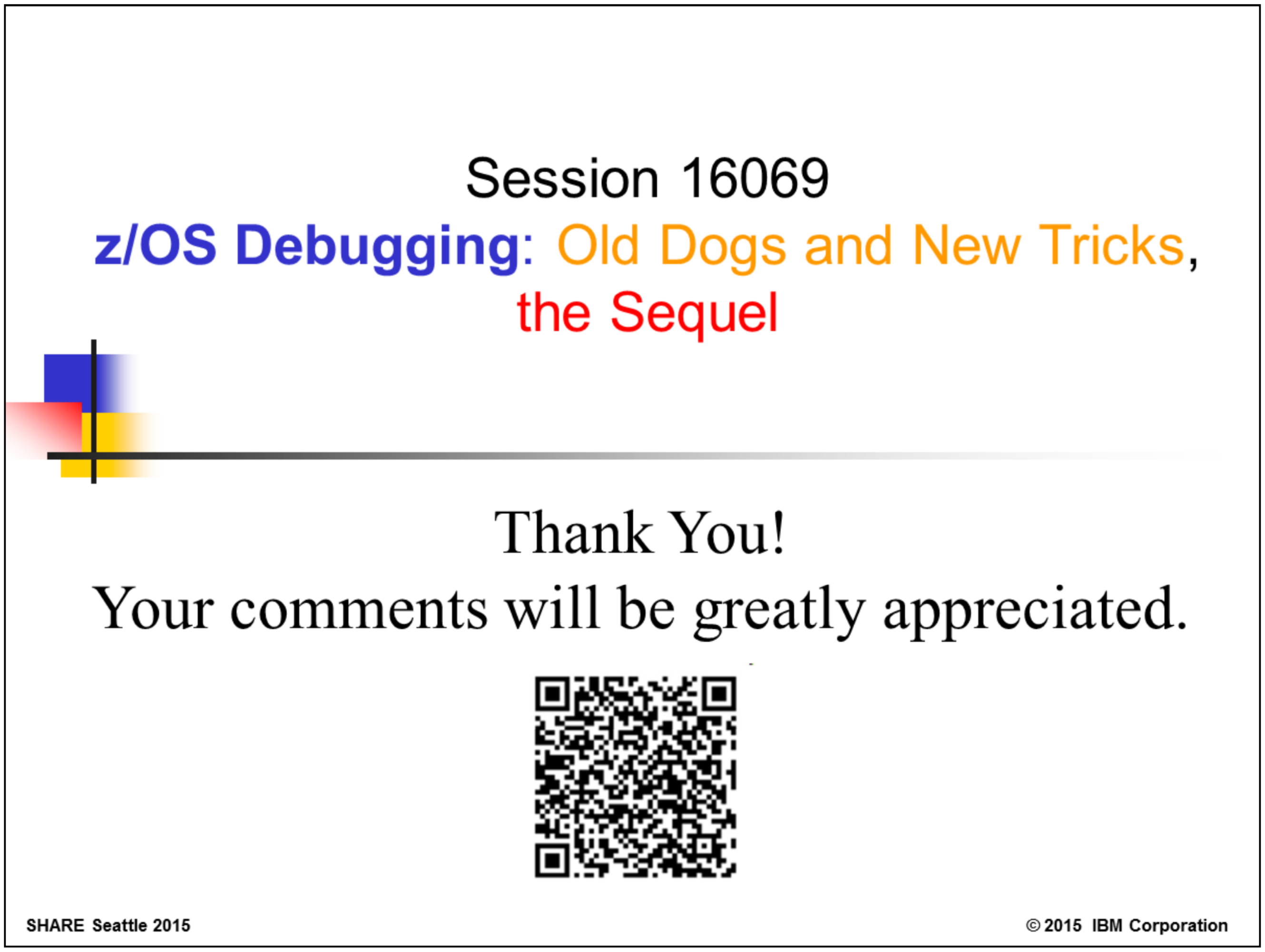
- A z/OS 'Tools and Toys' REXX Exec – as-is, not documented, not supported, not warranted
- Displays TCB family structure of the default ASID (mother/daughter/sister)
- Output also includes TCB completion code
- Output may be truncated if screen is not wide enough

IP IEAVTCBM example



```
RELATIONSHIP KEY: SIS--SISTER--TCBNTC FIELD      DAU--DAUGHTER--TCBLTC FIELD
ISSUE SETD ASID(X'NNNN') TO CHANGE DEFAULT ASID

TCB STRUCTURE FOR ASID: 0348     JOBNAME: JNG15044
NUMBER OF TCBS IN ASXB: 4
DEPENDING ON THE SIZE OF THE CHAIN, TRUNCATION MAY OCCUR.  SEE BOTTOM FOR POSSIBLE ERROR
MESSAGES.
NOTE: CC FOR TCBS COULD BE RESIDUAL

009FDD40
IEAVAR00
  -OK-
   |
009FF6F8----SIS-->009FF988
IEESB605          IEAVTSDT
  S0C4              -OK-
   |
009FF4D8
IKJEFLC
  -OK-

NO TRUNCATION ERRORS FOUND.
```

Completion code is zero



Session 16069

z/OS Debugging: Old Dogs and New Tricks, the Sequel

Thank You!

Your comments will be greatly appreciated.

SHARE Seattle 2015