Understanding the z/OS System Trace

April 18th, 2018

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

Trademarks

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

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

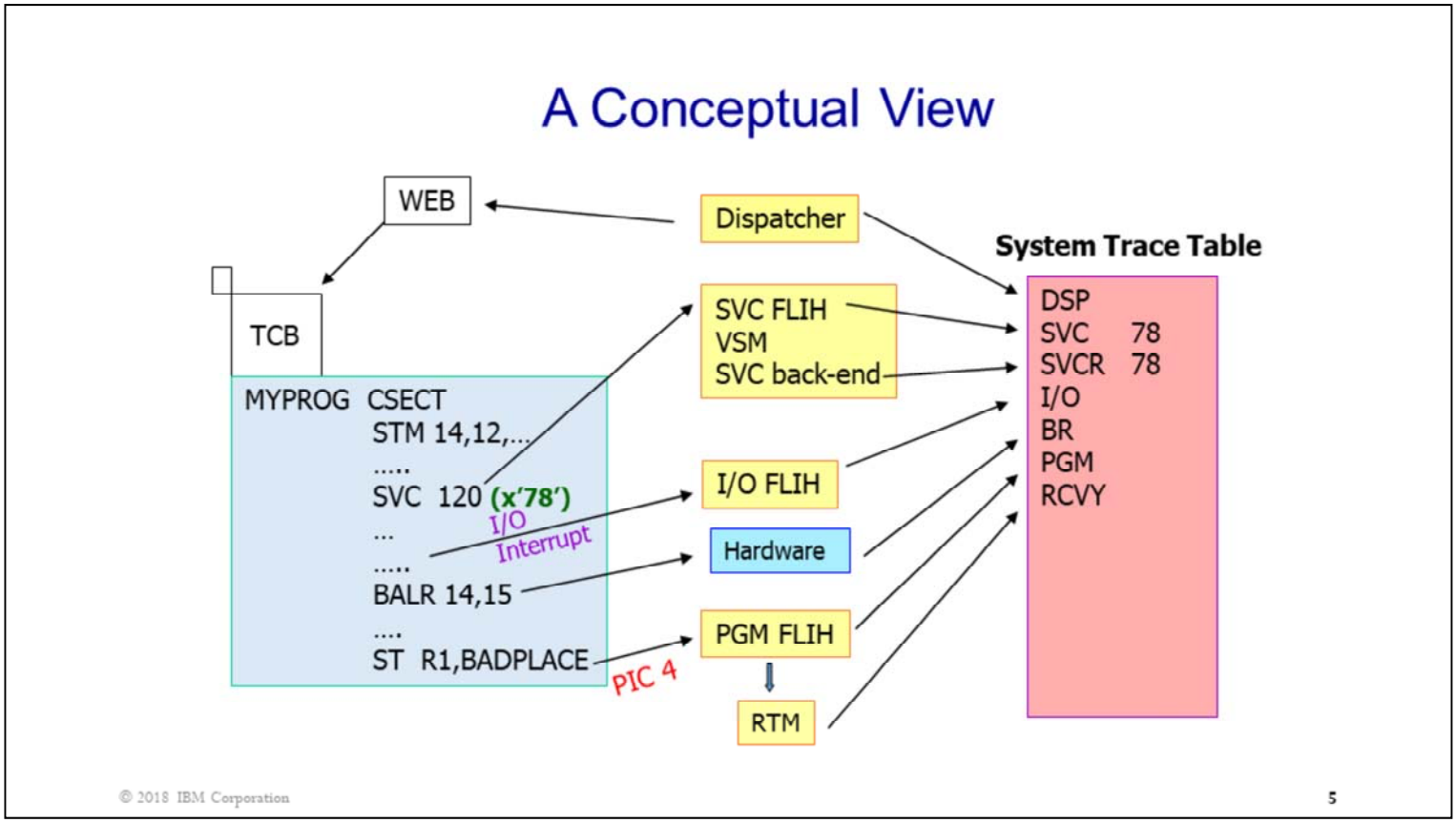
* Registered trademarks of IBM Corporation

Table of Contents

The Basics

The z/OS System Trace holds a wrap-around history of significant system events such as dispatching of work, interrupts, and errors

- Kept in-core in page-fixed CPU-related trace buffers
  - Buffers default to 1Meg in size
  - Typically holds .1 to 1 second of trace history
- Found in various types of dumps
  - Included in SVC dump via SDATA TRT option
  - Included in user dumps by default
  - Included in SAdumps
- Formatted chronologically via **IPCS SYSTRACE**
  - Activity on various CPs get merged by timestamp



The Dispatcher searches the Work Unit Queue to look for a WEB (Work Element Block) to dispatch. In this example, a TCB WEB is found and a program MYPROG under this TCB is dispatched. A DSP trace entry is generated by the Dispatcher prior to it giving control to MYPROG.

The program MYGROG starts to execute and issues an SVC 78. This results in an SVC Interrupt and the SVC FLIH receives control. The SVC FLIH traces the SVC 78 (getmain or freemain), then routes control to VSM to handle the request. After the request is processed, the SVC back-end clean up routine receives control and traces SVCR which signifies the completion of the SVC. MYPROG then receives control at the point after the SVC 78.

MYPROG continues to run and then takes an I/O interrupt. The I/O FLIH receives control and traces the I/O interrupt.

After the I/O interrupt has been processed, MYPROG continues to run and then issue a BALR 14,15 to branch to a subroutine. The execution of the BALR instruction causes the hardware to insert a BR trace entry in the system trace table. (Branch tracing is not active by default but can be turned on with the TRACE system command.)

Later MYPROG attempts to store the contents of R1 into a variable BADPLACE but the storage address of BADPLACE is bad. This results in a PIC 4 (protection exception). The Program FLIH receives control and traces the program check. Then the program FLIH passes control to RTM, and RTM traces its activity.

From this example, you can see that the system trace table shows some (but not all) of the activity under program MYPROG.

System Trace Reference Material

- MVS IPCS Commands
  - SYSTRACE command
- MVS Diagnosis: Tools and Service Aids
  - Section "System Trace" describes trace entries
- MVS Diagnosis: Reference
  - Chapter 4 defines SVC interface registers
- MVS Data Areas
  - Defines Unique fields for certain types of SSRV entries
- MVS System Commands
  - TRACE ST command to adjust System Trace buffer size, turn on/off branch tracing

Reading SYSTRACE output

```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------
Command ===>
 ********************************************************* TOP OF DATA **********************************

 ------------------------------------------------ SYSTEM TRACE TABLE ------------------------------------
 --                                               -----------------
 --                                               -----------------
 PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIME
                                                 UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE

 00 0021 009FE030  DSP        00000000_014C9DDE  00000000 814C9DDE 00FB3788  00000000 00000000 0021 0021 23:00:07.958350
                              07041000 80000000
 00 0021 009FE030  SVC     4F 00000000_014CA396  009FDCB0 00000011 FFFFFFEF  Status   Start SRBs only    23:00:07.958354
                              07041000 80000000
 02 0165 009FFB00  DSP        00000000_0137DA1C  00800000 00000001 05882000  00000000 00000000 0049 0165 23:00:07.958355
                              07044000 80000000
 00 0021 009FE030  SVCR    4F 00000000_014CA396  00000000 00000000 05616F00                             23:00:07.958365
                              07040000 80000000
 02 0165 009FFB00  PC      ...   0      095701CE             01F01
 02 0165 009FFB00  PR      ...   0      095701CE 29A06B5A                                    0165
```

- Entries are presented in chronological order
  - Oldest entry at the TOP
  - Newest entry at the BOTTOM
  - Entries are inter-mixed across CPs
- Entries typically 1-2 lines each

7

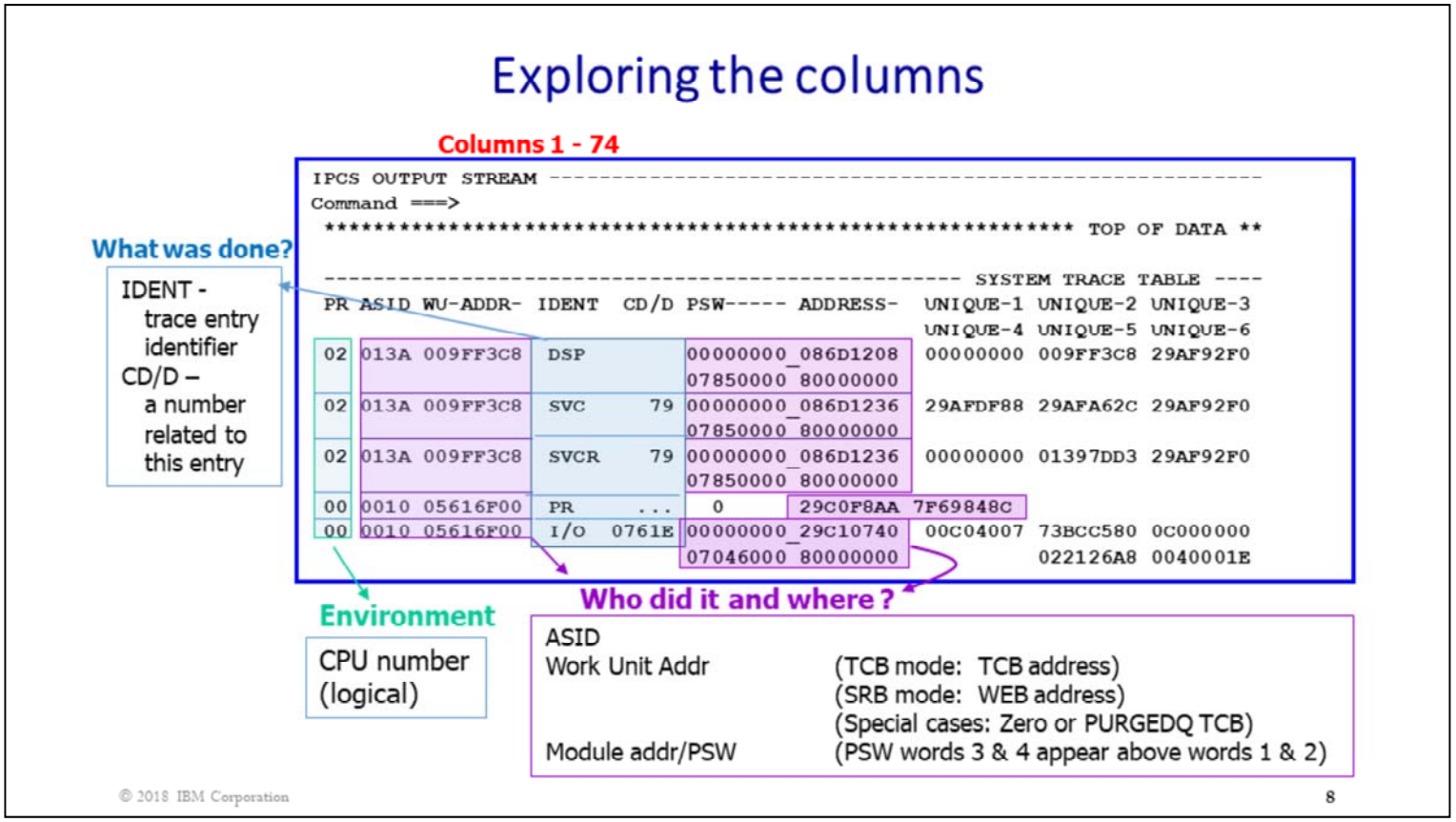
When the system trace table is displayed by IPCS SYSTRACE, the oldest entry is at the top and the newest entry is at the bottom. For SYSTRACE and any large IPCS report, scroll max to the bottom of the output (and back) before viewing the entries as this primes the IPCS buffers, causing FINDs to be much faster.

This trace excerpt shows a TCB at address 9FE030 in ASID X'21' running on CP0 at the same time that a TCB at address 9FFB00 in ASID X'165' is running on CP2.

Note that the system trace formatter tells us that the SVC 4F is invoking the STATUS system service, requesting a STATUS START of SRBs.

Note that the SVCR 4F PSW matches the SVC 4F PSW.

Note that the PR PSW address matches that of the PC 1F01. PC 1F01 is a user PC so cannot be identified by the SYSTRACE formatter.



The IDENT column identifies the type of system activity.

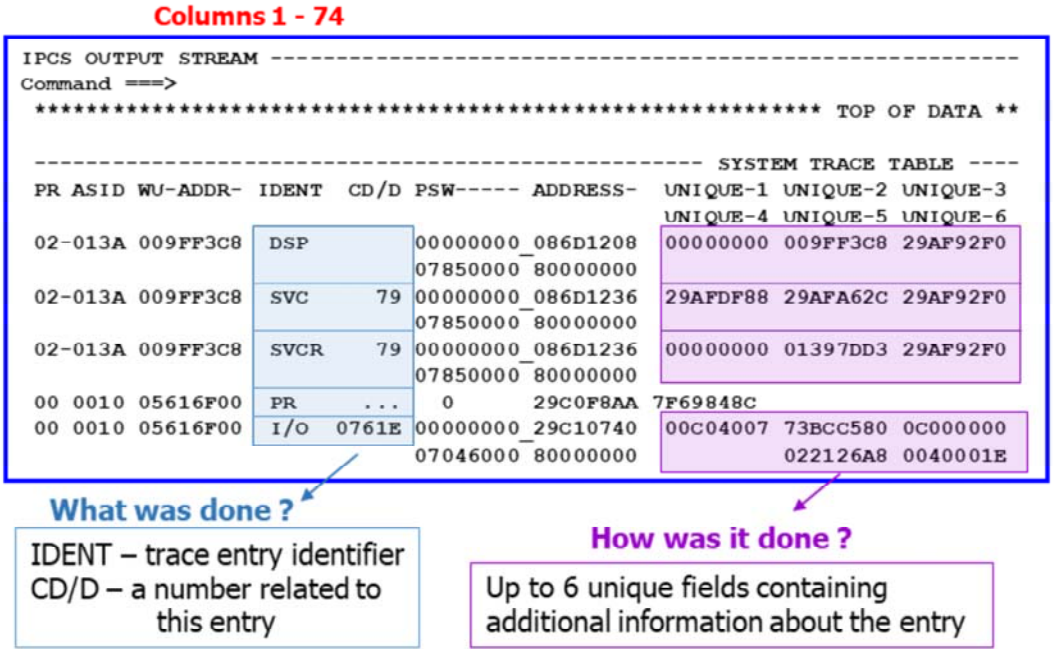
The CD/D column contains a number related to the system activity (for example, an SVC number, an interrupt code, or a device number).

The ASID and WU-ADDR columns identify the work unit under which the system activity was traced. Note that usually the WU Address will be the WEB address if the entry is for activity under an SRB, or the TCB address if the entry is for activity under a TCB. Occasionally you will see zeros in the WU-ADDR column. This occurs for WAIT trace events (CPU entering an enabled “no work” WAIT), entries traced as a CP is coming out of an enabled WAIT, and sometimes for I/O subchannel events. In the case of a SRB SUSP entry, the WU-ADDR column will contain the PURGEDQ TCB address.

The PSW ADDRESS column identifies where the activity occurred.

The PR (processor) column identifies the logical CPU that the work unit is running on.

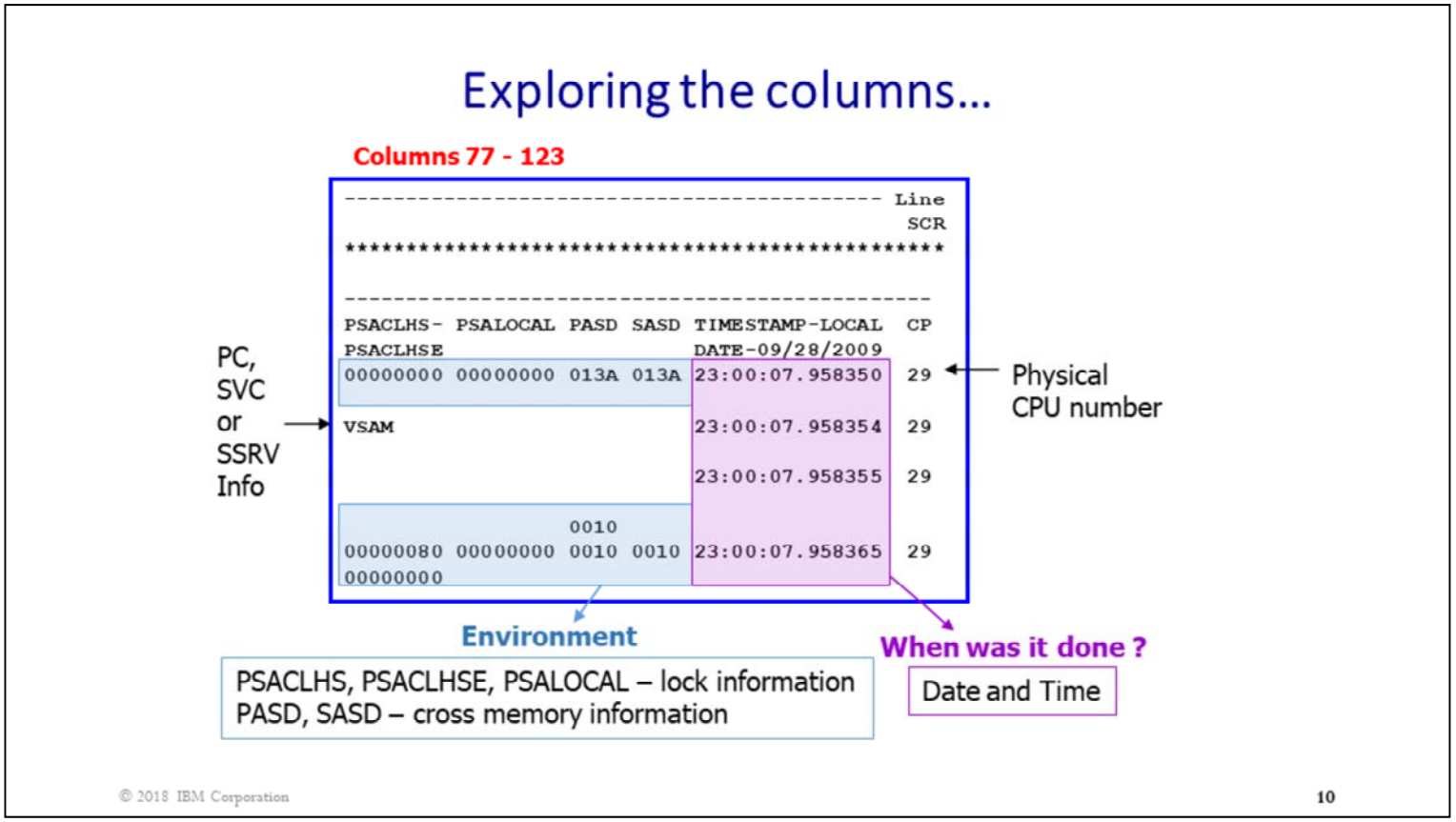
Exploring the columns...



The IDENT column identifies the type of system activity.

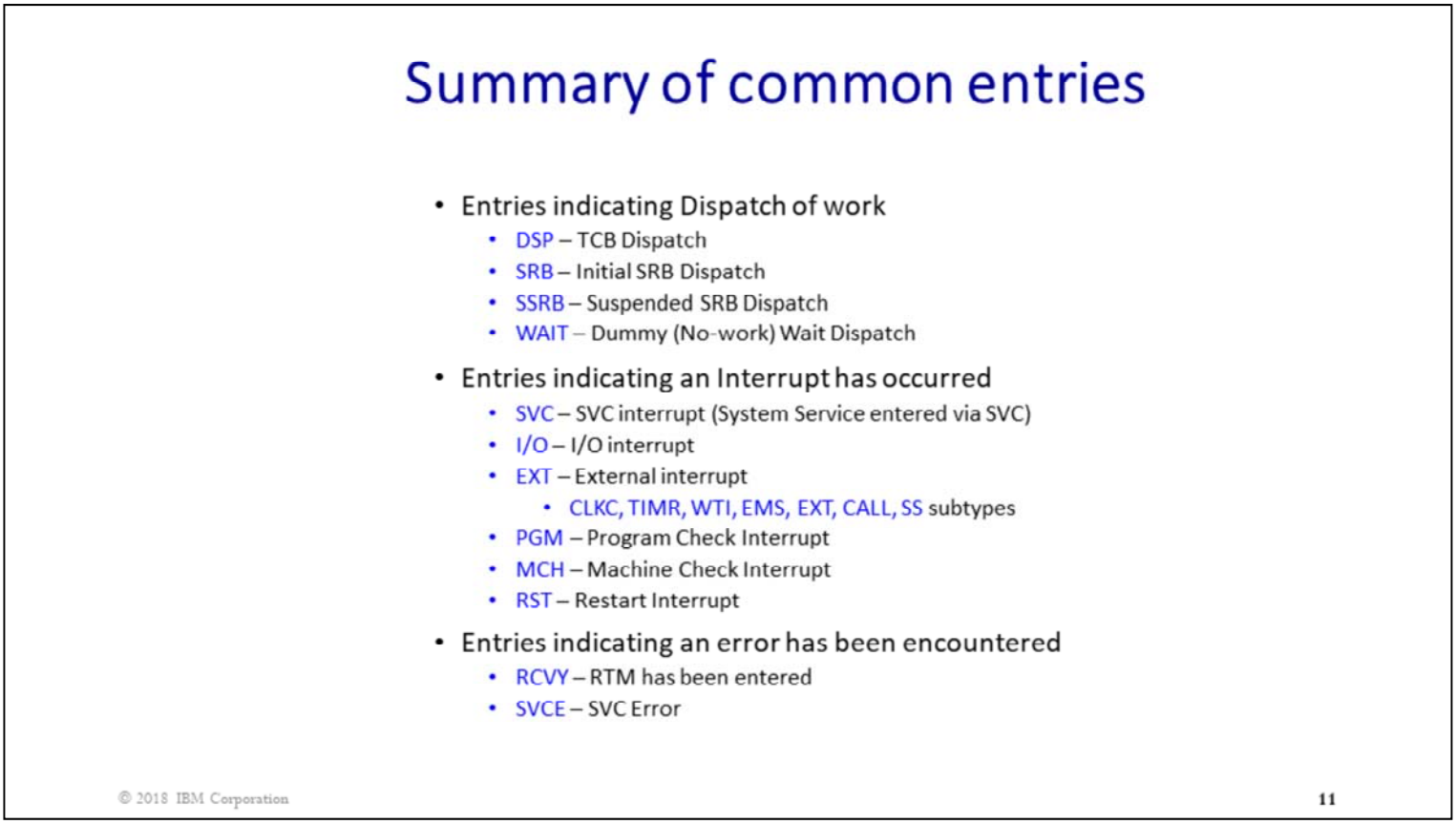
The CD/D column contains a number related to the system activity (for example, an SVC number, an interrupt code, or a device number).

The UNIQUE fields contains further information about the entry. An entry can have up to 6 unique fields.



The PSACLHS/PSACLHSE and PSALOCAL columns provide local lock information.

For PC, SVC or SSRV entries, there is additional information about what the system service is. This information is also under the PSACLHS/PSACLHSE column.



The common system trace entries can be classified into the above groups.

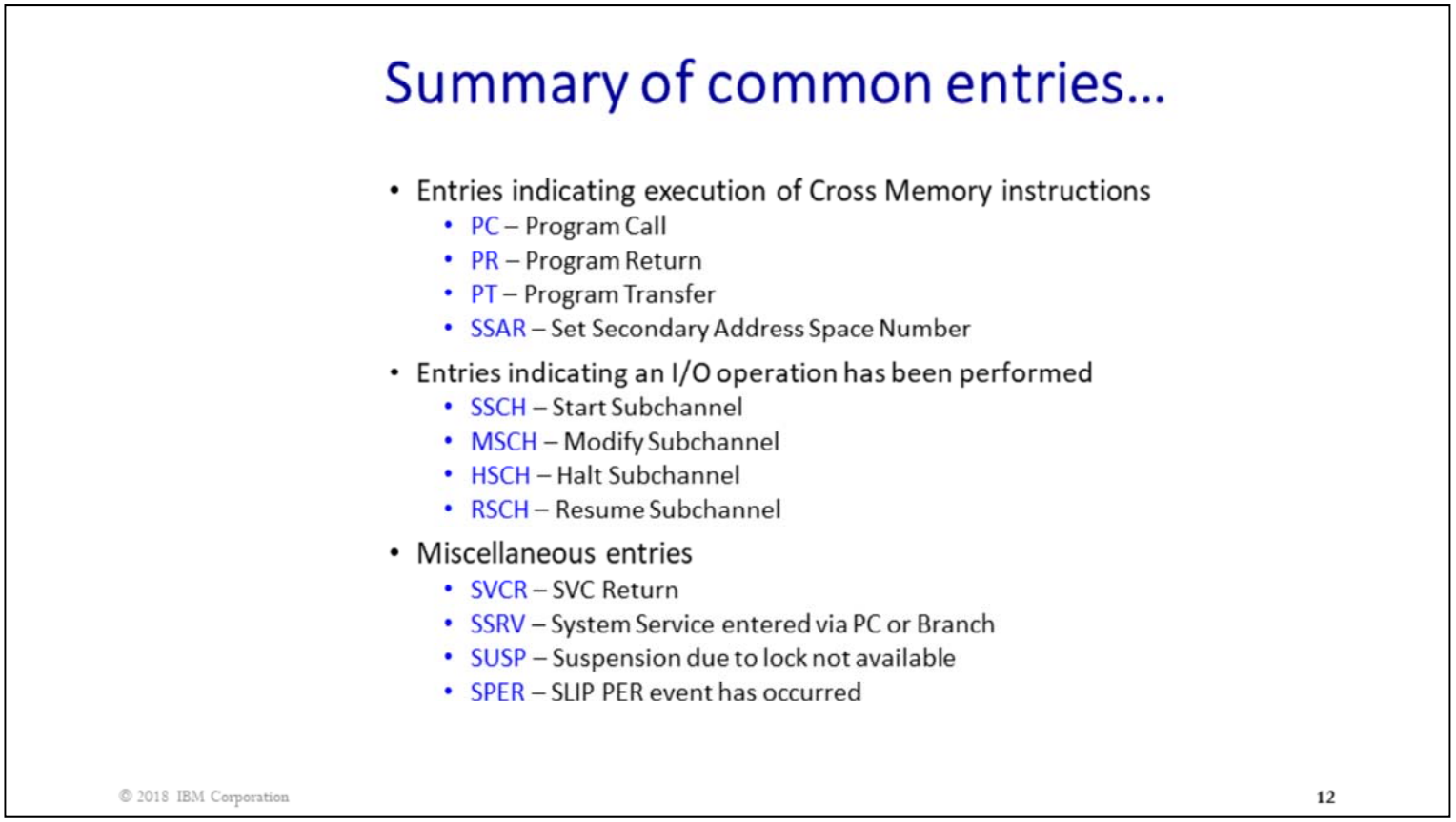
Dispatch entries are generated by the Dispatcher.

Interrupt entries are generated by the Interrupt Handlers.

SVC interrupts are generated when an SVC is issued to invoke a system service.

RCVY entries are generated by RTM.

SVCE indicates that an SVC has been issued in an invalid environment. This is almost always for an SVC D ABEND request. The fact that an ABEND occurred is much more interesting to us than the fact that the SVC was issued under an invalid environment (which is quite common/normal for ABENDs).



SVC, SVCR, and SSRV entries are written when a system service has been invoked.

Cross Memory instruction trace entries are created by hardware.

I/O operation trace entries are generated by I/O Supervisor routines.

SUSP entries are created by the Lock Manager when a work unit is suspended for a lock.

Common *RCVY* entries

- RCVY PROG* – RTM1 is being entered for a program check interruption
- RCVY FRR* – RTM1 is invoking a functional recovery routine (FRR)
- RCVY RTRY* – Retry from an FRR
- RCVY PERC* – RTM1 FRRs did not retry; control "percolates" to RTM2
- RCVY ESTA** – RTM2 is invoking an ESTAE-type recovery routine
- RCVY ESTR** – Retry from an ESTAE-type recovery routine
- RCVY ABT* – Request for abnormal end of a TCB via CALLRTM macro
- RCVY MEM* – Request for abnormal end of an address space via CALLRTM macro

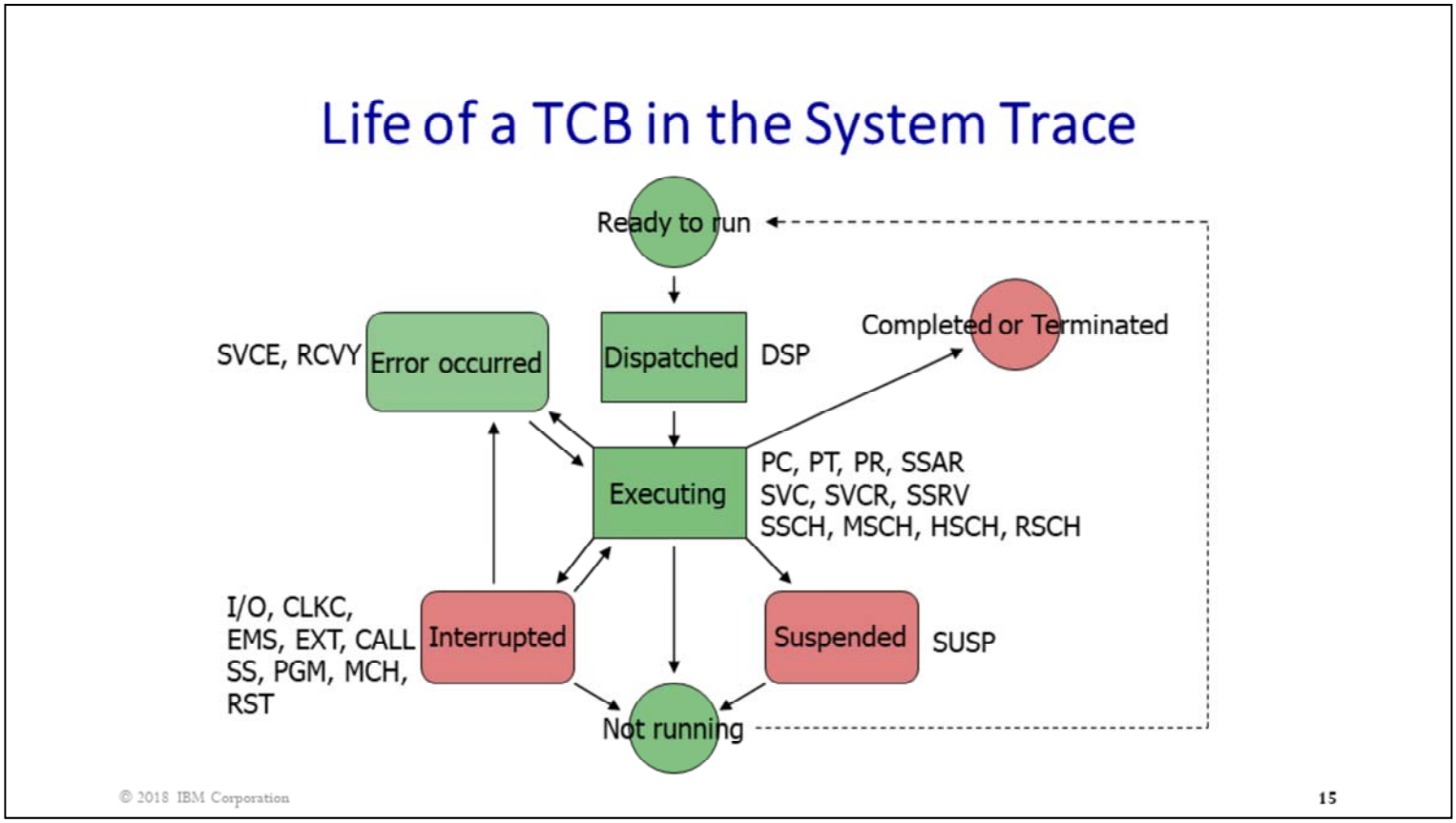
* *UNIQUE1 = Completion Code UNIQUE2 = Reason Code*

** *UNIQUE1 = SDWA address UNIQUE2 = 64-bit parm ptr (available as of z/OS R2.1)*

The above are various RCVY trace entries. They are described in the chapter on System Trace in z/OS MVS Diagnosis: Tools and Service Aids. The above list is not comprehensive so if you see a RCVY entry not listed above, check out the manual.

Interpreting system trace entries

- **All system trace entry types** are described in the "System Trace" chapter of MVS Diagnosis: Tools and Service Aids
- **SVC routine interface registers (15, 0, and 1)** are defined in chapter 4 of MVS Diagnosis: Reference
  - UNIQUE 1/2/3 fields on SVC entry map to registers 15/0/1, respectively
- **Unique trace data for some SSRV events** can be obtained from the MVS Data Areas manuals as follows:
  - PC Auth – PCTRC data area
  - Supervisor Control – SPTRC data area
  - Task Management – TMTRC data area
- **Unique trace data for other components' SSRV events such as VSM, RSM, and GRS** is described under the SSRV section of the "System Trace" chapter in MVS Diagnosis: Tools and Service Aids.



The above flowchart gives a summary of the trace entries that can be generated under a TCB.

A DSP entry is created when the TCB is dispatched. While it is executing, various trace entries can be generated from different activity under the TCB. If the TCB takes an interrupt, a FLIH will generate the appropriate trace entry representing the interrupt.

After the interrupt is handled, there are 3 possible cases:

(1) The TCB continues to execute, generating more trace entries.

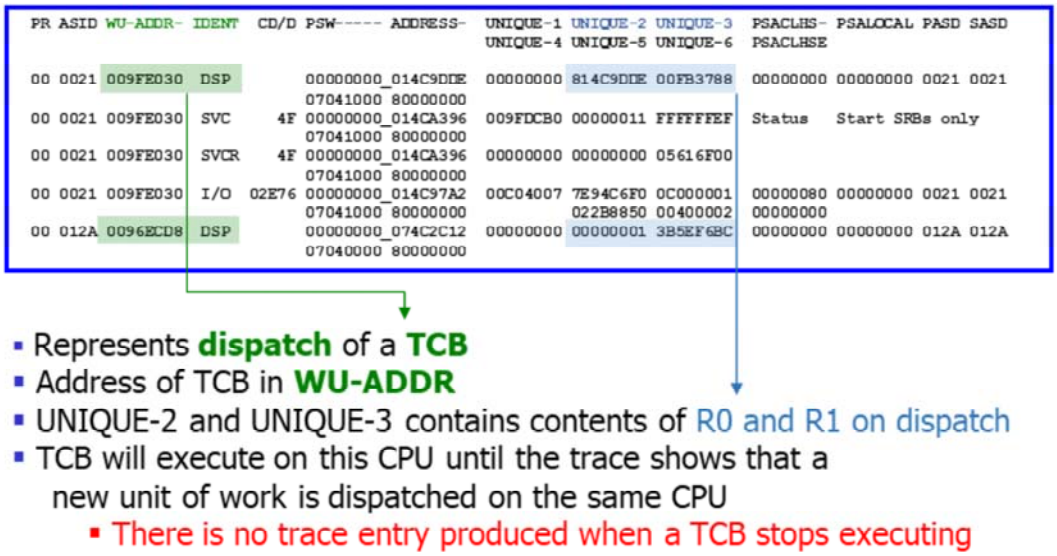
(2) The TCB is preempted or it is not dispatchable anymore. It will then stop running. Note that no trace entry is generated when the TCB is put into a 'not running' state.

(3) An error condition is detected by the FLIH and RTM is invoked. RTM will then execute under the TCB, generating trace entries from its activity.

While executing, the TCB can be suspended for a lock and then enter the 'not running' state.

While executing, the TCB can also go through normal or abnormal termination.

TCB Dispatch - *DSP*



```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE
00 0021 009FE030 DSP         00000000_014C9DDE 00000000 814C9DDE 00FB3788  00000000 00000000 0021 0021
                             07041000 80000000
00 0021 009FE030 SVC     4F  00000000_014CA396 009FDCB0 00000011 FFFFFFEF  Status   Start SRBs only
                             07041000 80000000
00 0021 009FE030 SVCR    4F  00000000_014CA396 00000000 00000000 05616F00
                             07041000 80000000
00 0021 009FE030 I/O  02E76  00000000_014C97A2 00C04007 7E94C6F0 0C000001  00000080 00000000 0021 0021
                             07041000 80000000          022B8850 00400002  00000000
00 012A 0096ECD8 DSP         00000000_074C2C12 00000000 00000001 3B5EF6BC  00000000 00000000 012A 012A
                             07040000 80000000
```

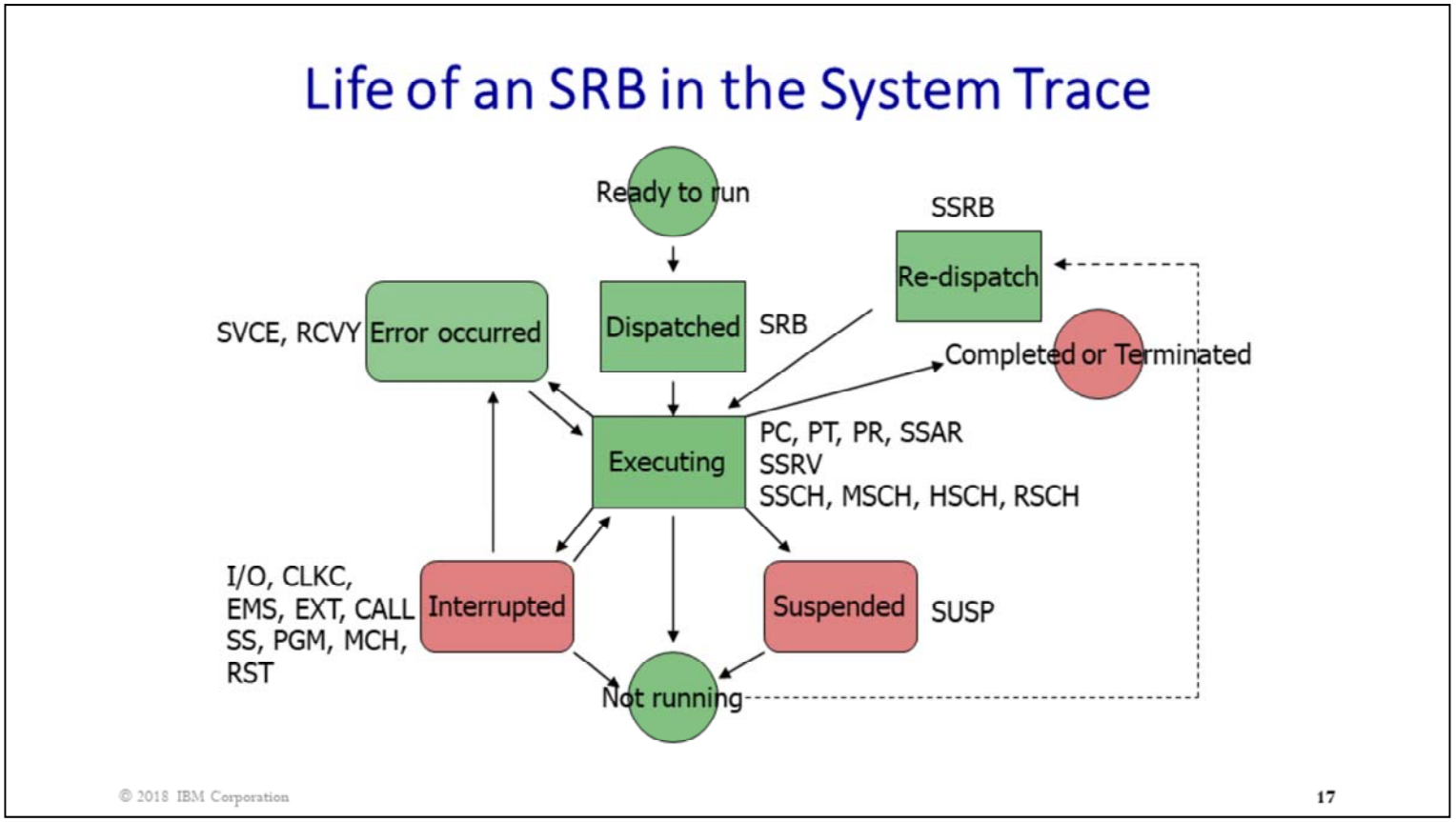
- Represents **dispatch** of a **TCB**
- Address of TCB in **WU-ADDR**
- UNIQUE-2 and UNIQUE-3 contains contents of R0 and R1 on dispatch
- TCB will execute on this CPU until the trace shows that a new unit of work is dispatched on the same CPU
  - There is no trace entry produced when a TCB stops executing

16

The DSP entry indicates that a TCB is dispatched on this CPU. It also signifies that the previous unit of work has completed.

Note that there is no trace entry produced when a TCB stops executing. In the above example, TCB 9FE030 in ASID 21 took an I/O interrupt on CPU 0. Then the next trace entry on the same CPU is a dispatch of another TCB 96ECD8 in ASID 12A. This indicates that the TCB 9FE030 stops running (or it is preempted).

When TCB 9FE030 gets redispatched following the I/O interrupt, the dispatch PSW will be the same as that of the I/O interrupt. This makes sense since the work unit is resuming where it left off when the I/O interrupt occurred. The TCB will not necessarily get redispatched on CP0.



The above flowchart gives a summary of the trace entries that can be generated under a SRB.

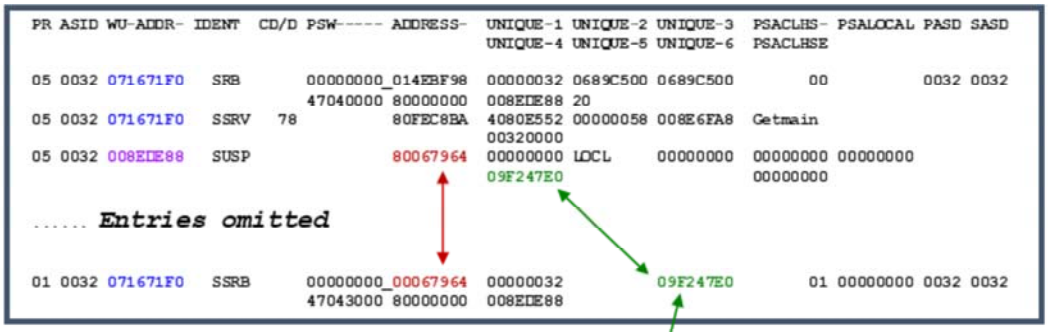
The flowchart is similar to that of a TCB, except that:

(1) A SRB entry is traced when an SRB is dispatched (instead of DSP entry for TCB)

(2) SRB cannot issue SVCs, so there are no SVC or SVCR entries generated under an SRB

(3) A SSRB entry is traced when an SRB is re-dispatched (instead of DSP entry when a TCB is re-dispatched)

SRB Dispatch/Suspend/Redispatch



```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE

05 0032 071671F0  SRB        00000000_014EBF98 00000032 0689C500 0689C500        00           0032 0032
                             47040000 80000000 008EDE88 20
05 0032 071671F0  SSRV   78           80FEC8BA 4080E552 00000058 008E6FA8  Getmain
                                                00320000
05 0032 008EDE88  SUSP                80067964 00000000 LOCL     00000000  00000000 00000000
                                                09F247E0                    00000000

..... Entries omitted

01 0032 071671F0  SSRB       00000000_00067964 00000032          09F247E0        01 00000000 0032 0032
                             47043000 80000000 008EDE88
```

- WEB address in WU-ADDR except...
  - PURGEDQ TCB address traced in WU ADDR field of SRB SUSP
- SSRB entry's UNIQUE-3 field contains SSRB address
- PSW matches original PSW at point when SRB was suspended
  (or, in the case of a preemptable SRB, interrupted off the CP)

18

When a SRB is suspended or preempted, an SSRB is used to save status of the SRB. When this unit of work is re-dispatched, an SSRB entry is traced. The PSW of the SSRB entry should match the original PSW when the SRB was suspended or interrupted.

A PURGEDQ TCB is a TCB that is associated with the SRB and who may get abended should the SRB suffer an abend. If a TCB terminates, all SRBs who have that TCB as their PURGE TCB also get driven through termination.

Where do I begin in System Trace?

It depends on the type of dump!

- Recovery initiated dump
  - Find the error event and review previous activity
- SLIP dump
  - Find where the SLIP matched and review previous activity
- Console dump
  - No particular milestone
  - Need to review overall activity of address space(s) dumped
- Standalone dump
  - Go to end of system trace and review most recent activity

The system trace table in a dump contains many entries. Depending on the type of dump, your focus area in the system trace is different. In most cases the debugger will scroll max to the bottom first, them scroll max to the top before any search in system trace.

For a recovery-initiated dump, you want to find the trace entries representing the error event and then review previous activity. This would typically be either an SVC D/SVCE D entry, or a RCVY entry.

For a SLIP dump, you want to find the trace entry indicating the SLIP matched and then review previous activity. For a dump generated by a PER trap, you would want to locate the SPER trace entry. For a SLIP dump triggered for an abend, you would want to locate the SVC D or SVCE D entry for the abend.

For a console dump there is no particular milestone in the system trace. Often a console dump is taken for hang which means the system trace table may not be too useful.

For a standalone dump you want to review the most recent system activity found at the end of the trace.

SYSTRACE Filters & Formatting Options

Filters

- Address space — ASID, JOBNAME
- Work unit — TCB, WEB [for SRBs]
- CP — CPU, CPUTYPE, CPUMASK
- No filter — ALL

Other options

- Time — TIME(LOCAL/GMT/**HEX**)
- Statistics — PERFDATA, STATUS

REPORT VIEW on SYSTRACE report command line

- Provides advanced ISPF-like editing capability

Some filters can be combined.

SYSTRACE default is to filter by “current ASIDs”. For an SVC dump of an error, or a SLIP dump, the current ASIDs are generally the cross memory environment at the time of the event. For a console dump, the current ASID is ASID 1 (not the dumped ASID!). For a SAdump, the current ASIDs are those which owned work active on CPs at the time the system was stopped. Issue IPCS SELECT CURRENT if you want to see what IPCS considers to be the “current” address spaces in a dump. Remember, you can always explicitly specify what address spaces you want to see formatted in the system trace table.

Timestamps in system trace can be formatted in hex format, local time, or GMT time. Default is hex so usually you will want to put TIME(LOCAL) [or TI(LO) for short] on your SYSTRACE command.

The PERFDATA option totals and summarizes time dispatched per CP, per address space, and per work unit. It also summarizes time spent for I/O to various devices. For more information about SYSTRACE PERFDATA, please see SHARE presentation “**z/OS Debugging:** *Old Dogs and New Tricks”* (Anaheim 2012).

STATUS gives a summary of the time range spanned by the entries for each CP.

REPORT VIEW is not an IPCS command, nor is it a filter specific to SYSTRACE. Type it on the command line of any IPCS report to enter a mode that gives you ISPF-like edit capability. You can exclude lines, delete lines, FIND ALL, SORT, etc.

SYSTRACE examples

1) SYSTRACE JOBNAME(TEST1)
2) SYSTRACE ASID(X'1B',X'20') TI(LO)
3) SYSTRACE ASID(59) WEB(X'05311280')
4) SYSTRACE ASID(X'3B') TCB(X'987658')
5) SYSTRACE CPU(X'12') ALL
6) SYSTRACE CPU(0:11) ALL TI(LO)
7) SYSTRACE CPUMASK(FFF) TI(LO)
8) SYSTRACE CPUTYPE(STANDARD) ASID(8)
9) SYSTRACE CPUTYPE(ZIIP) ALL
10) SYSTRACE ALL TI(LO)
11) SYSTRACE PERFDATA(DOWHERE)
12) SYSTRACE STATUS TI(LO)

1) Format all entries for job TEST1.
2) Format all entries for ASIDs X'1B' and X'20'. They will be sorted chronologically. The timestamp will be formatted as local time.
3) Format the activity under WEB 5311280 in ASID 59. The WEB may be associated with a TCB or an SRB. If it is associated with the TCB, the TCB address could have been used instead as is demonstrated in example 4.
4) Format the activity under TCB 987758 in ASID X'3B' = ASID 59.
5) Format all trace entries for CP X'12'. Note that the default is the current ASID, not ALL!
6) Format all entries for CPs 0 thru 11.
7) This command is the same as example 6 but specifies CPs to be formatted using a mask rather than a range.
8) Format all the activity for ASID 8 on standard CPs only.
9) Format all ZIIP activity only.
10) Format all trace entries.
11) Format statistics related to time used by CP, ASID, work unit, and I/O processing. This also maps PSWs where SRB dispatches and CLKC interrupts have occurred to module and offset.
12) Format time ranges covered by each CP, displaying as local time.

Examples

PGM Interrupts: The good, the bad, & the RCVY

- There are 3 possibilities after a program interrupt (aka program check)
  - The Good – interrupt is resolvable
    - Interrupt is synchronously resolved and program continues running
    - Unit of work is suspended and gets redispatched after the interrupt is asynchronously resolved
  - The Bad – interrupt is unresolvable
    - The RCVY PROG entry is written and RTM is entered
- Absence of a RCVY entry after the PGM means that the program interrupt was successfully resolved.

If the request is for a frame to back a page on first reference, the page fault will be resolved synchronously and all you will see is a PGM 11.

If the request is for a frame to back a page that is out on DASD, the page fault will be resolved asynchronously since the system must do I/O to bring the page content in from DASD. In the trace table you will see the PGM interrupt followed by a DSP or SSRB entry. The intervening I/O interrupt may not be apparent unless a SYSTRACE ALL is issued.

If the page reference is invalid (perhaps the address is bogus) or cannot be resolved, the PGM entry will be followed by a RCVY PROG entry.

The good PGM

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE

00 0136 009FFB00 PC     ...   0      08BE587C             0030B           Storage  Obtain
00 0136 009FFB00 SSRV   132          00000000  4000E600 00002200 7FFCBE00  Storage  Obtain
                                               01360000
00 0136 009FFB00 PR     ...   0      08BE587C 0159A552                                    0136

00 0136 009FFB00 PGM    011 00000000_08BE590A  00060011 00000000           00000000 00000000 0136
                              07041000 80000000           7FFCC000           00000000
00 0136 009FFB00 PC     ...   0      08BDB964             00C04           DFSMS
```

- No RCVY trace entry after PGM
  - The interrupted unit continues to run after the interrupt is resolved synchronously (above example),
    OR
  - The interrupted unit is suspended and will be re-dispatched after the interrupt is resolved asynchronously
    - PGM, then next entry for same work unit is a DSP

24

The above is an example of a TCB taking a page fault (PIC 11) which was resolved synchronously. The TCB continued to run after the page fault had been handled.

In the case where the unit of work must get suspended while the page fault is resolved, there is no trace entry written for page fault suspension. Therefore what you would see in the trace is the PGM entry for the page fault, then the next entry for the same unit of work would be a DSP. Note that there could be a significant number of trace entries for other units of work between the PGM and the DSP, since it might take a little time for the page fault to get resolved.

The bad PGM

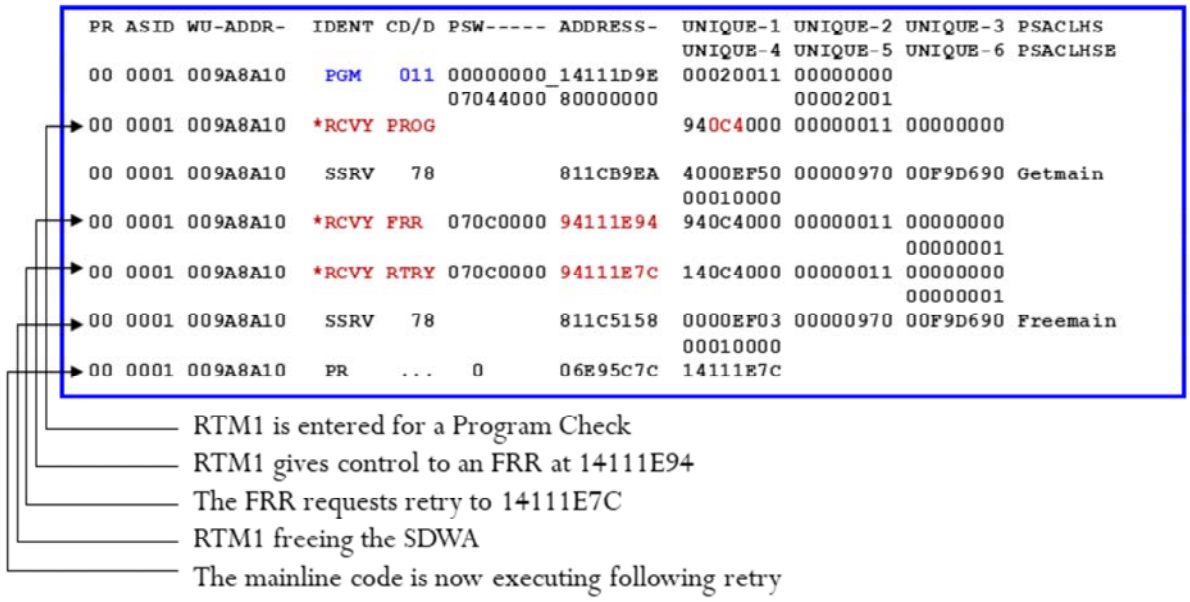
```
PR ASID WU-ADDR- IDENT  CD/D PSW-----  ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD
                                                 UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE

0B-05AE 006D4B58  SSRV    78          812B301A  4000EF50 00000998 00EB9668  Getmain
                                                 00010000
0B-05AE 006D4B58  SSRV   112          819715BA  08EA35A8 00FDC800 8BBEB508  Schedule
                                                 00000000
0B-05AE 006D4B58  PGM    011 00000000_0C137114  00040011 00000000           00000000 00000000 02CA 02CA
                             07046000 80000000           00013001           00000000
0B-05AE 006D4B58 *RCVY   PROG                    940C4000 00000011 00000000  00000000 00000000 02CA 02CA
                                                                             00000000
0B-05AE 006D4B58 *RCVY   FRR  070C4000 8C136F10  400C4000 00000011 00000000  00000000 00000000 02CA 02CA
                                                                 00000001  00000000
```

- RCVY PROG trace entry after PGM
  - The program interrupt is not resolvable and RTM is entered

The above is an example of an unresolvable page fault (PIC 11). The 'RCVY PROG' entry indicates that RTM was entered to terminate the TCB with a system completion code of 0C4 (see UNIQUE 1 field). The 'RCVY FRR' entry indicates that an FRR received control.

RTM1 Trace Pattern: FRR Retry



```
PR ASID WU-ADDR-  IDENT CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3 PSACLHS
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6 PSACLHSE
00 0001 009A8A10   PGM   011 00000000_14111D9E 00020011 00000000
                             07044000 80000000           00002001
00 0001 009A8A10  *RCVY PROG                    940C4000 00000011 00000000

00 0001 009A8A10   SSRV   78          811CB9EA  4000EF50 00000970 00F9D690 Getmain
                                                00010000
00 0001 009A8A10  *RCVY FRR  070C0000 94111E94  940C4000 00000011 00000000
                                                                  00000001
00 0001 009A8A10  *RCVY RTRY 070C0000 94111E7C  140C4000 00000011 00000000
                                                                  00000001
00 0001 009A8A10   SSRV   78          811C5158  0000EF03 00000970 00F9D690 Freemain
                                                00010000
00 0001 009A8A10   PR     ...   0     06E95C7C  14111E7C
```

RTM1 is entered for a Program Check
RTM1 gives control to an FRR at 14111E94
The FRR requests retry to 14111E7C
RTM1 freeing the SDWA
The mainline code is now executing following retry

Here we see a case where a program check causes entry into RTM1, and at this time there is one FRR established. RTM1 gives the FRR control, and it elects to retry the error.

RTM1 Trace Pattern: No FRR retry, RTM1 percolates

```
PR ASID WU-ADDR-  IDENT CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3 PSACLHS
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6 PSACLHSE
00 0054 0097BE88  PGM    004 00000000_014EEB6E  00060004 00000000
                             47041000 80000000           00000000
00 0054 0097BE88 *RCVY PROG                     940C4000 00000004 00000000

00 0054 0097BE88  SSRV    78          810F41E2  4000EF50 00000998 00FC8668 Getmain
                                                00010000
00 0054 0097BE88 *RCVY FRR   470C0000 813D1318  940C4000 00000004 00000000
                                                                  00000001
00 0054 0097BE88 *RCVY PERC                     94206000 000000C8
                                                                  00000000
00 0054 0097BE88  SSRV   12D          8154B464  0097BE88 000C8000 FF3A0000 Status
                                                00000000
00 0054 0097BE88  SSRV   12D          8154B480  0097BE88 000B8000 00000000 Status
                                                00000000
00 0054 0097BE88  SSRV    78          812C0AA0  0000EF03 00000998 00FC8668 Freemain
                                                00010000
00 0054 0097BE88  DSP        00000000_012C0C38  00000000 00000658 7F43D9A8
                             47542000 80000000
00 0054 0097BE88  SVC      D 00000000_012C0C3A  00000000 00000658 7F43D9A8
                             47542000 80000000
```

The only FRR did not retry so RTM1 will "percolate" to RTM2.

RTM1 passes control to RTM2 by forcing an SVC D.

27

Here we see a case where a program check causes entry into RTM1, and at this time there is one FRR established. RTM1 gives the FRR control, and it elects to percolate (i.e. not to retry). RTM1 then enters RTM2 via an SVC D.

The SSRV 12D entries in the system trace table are due to RTM1 setting and resetting non-dispatchability bits as it sets the TCB up to issue the SVC D for RTM2 entry.

RTM1 Trace Pattern: No FRR defined

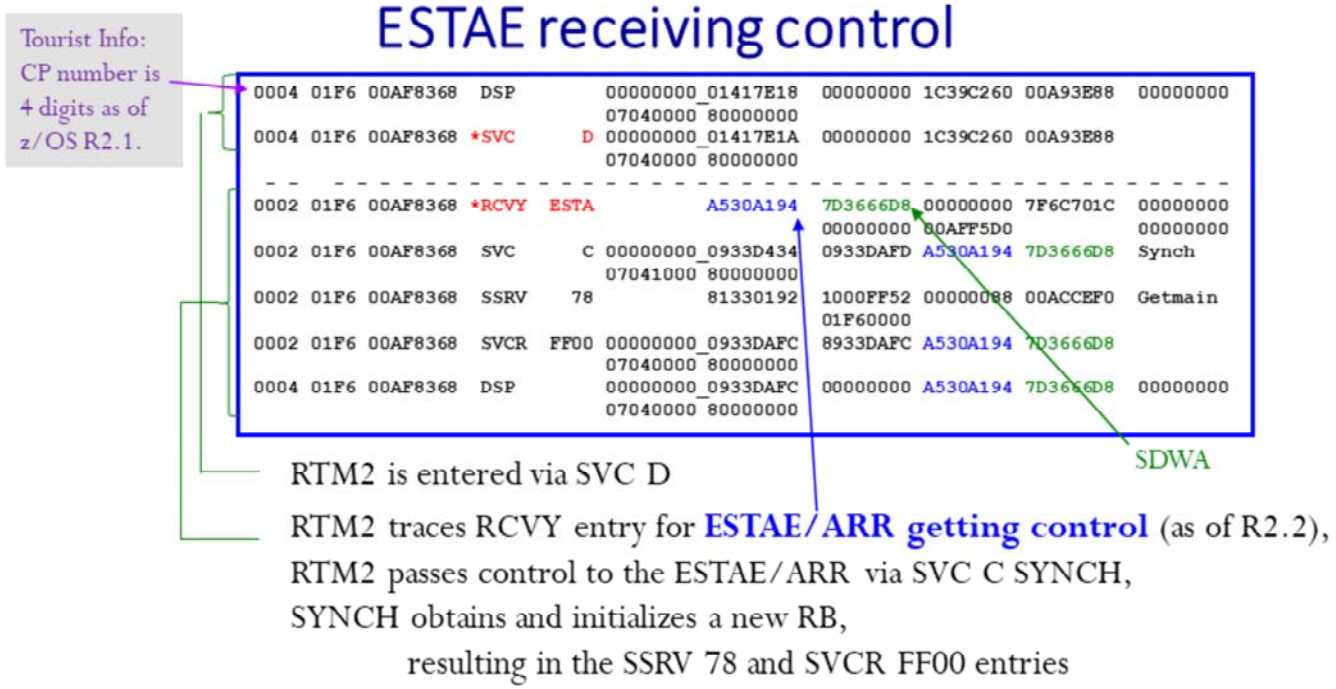
```
PR ASID WU-ADDR-  IDENT CD/D PSW-----  ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3 PSACL
                                                 UNIQUE-4 UNIQUE-5 UNIQUE-6 PSACL
01-001C 008DBC48   PGM   004 00000000_24600C16  00040004 00000000
                             07850000 80000000            00000000
01-001C 008DBC48  *RCVY PROG                     940C4000 00000004 00000000

01-001C 008DBC48   SSRV  12D          8153CF34  008DBC48 000C8000 FF3A0000 Status
                                                 00000000
01-001C 008DBC48   SSRV  12D          8153CF50  008DBC48 000B8000 00000000 Status
                                                 00000000
01-001C 008DBC48   DSP        00000000_01299712 00000000 00000000 24600BDC
                              07850000 80000000
01-001C 008DBC48  *SVC     D  00000000_01299714 00000000 00000000 24600BDC
                              07850000 80000000
01-001C 008DBC48   SSRV   78          833614AE  0000FF50 000000C8 008CFEB0 Getmain
                                                 001C0000
01-001C 008DBC48   SSRV   78          833614E4  0000FF70 00001220 7F711DE0 Getmain
                                                 001C0000
```

No RCVY FRR entry between RCVY PROG and SVC D

Here we see a case where a program check causes entry into RTM1, but this time there is no FRR established. RTM1 then enters RTM2 via an SVC D.

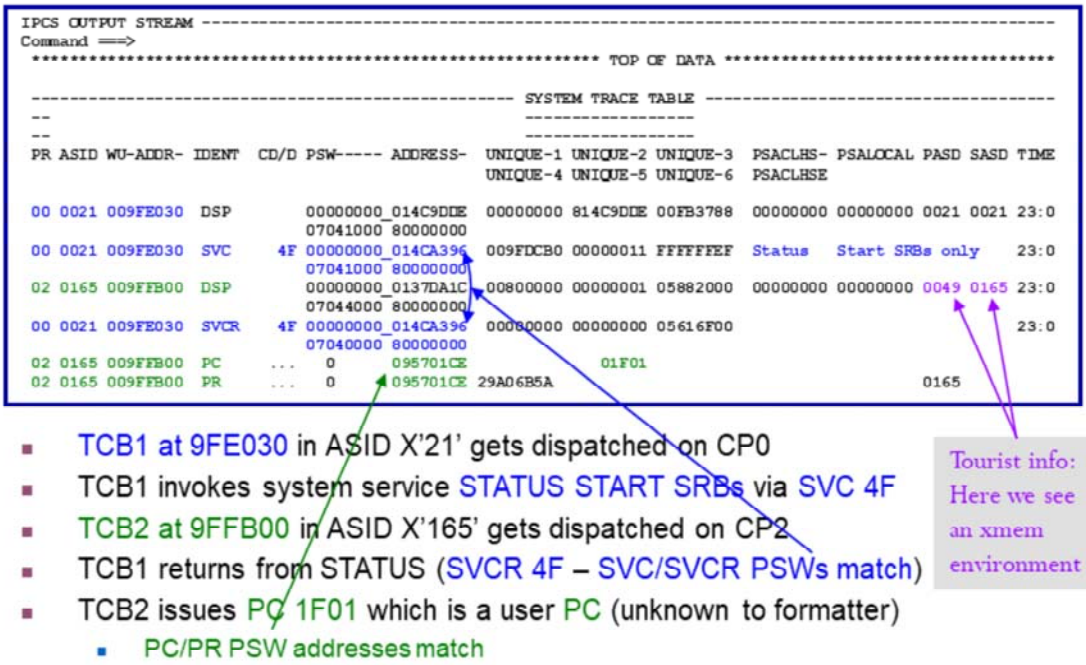
RTM2 Trace Pattern: ESTAE receiving control



The only way to enter RTM2 is via an SVC D. When RTM1 wants to pass control to RTM2, it sets up the abending TCB so that its PSW points to an SVC D instruction embedded within RTM1 code, then RTM1 forces the TCB to be redispatched.

RTM2 passes control to an ESTAE-type routine via a SYNCH macro/service. SYNCH processing results in the creation of a new RB (a PRB), and the ESTAE-type recovery routine will be driven under this RB. The SSRV 78 entry is the obtaining of the storage for the new RB. The SVCR FF00 is tricky to explain, but can be thought of as an indicator that a new RB is now set up to receive control. The DSP is the dispatch of that new RB. The PSW address on the SVC C, SVCR FF00, and DSP entries is the same and actually points into the RTM2 load module IGC0101C. This entry point in the RTM2 load module will branch enter the recovery routine. The recovery routine address is found in the PSW address of the RCVY ESTA trace entry, as well as in the Unique 2 field of the SVC C SYNCH trace entry.

An example of parallel activity



```
IPCS OUTPUT STREAM ----------------------------------------------------------------------------------------
Command ===>
 ********************************************************* TOP OF DATA **********************************

 ------------------------------------------------ SYSTEM TRACE TABLE ------------------------------------
 --                                               -----------------
 --                                               -----------------
 PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIME
                                                 UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE

 00 0021 009FE030 DSP        00000000_014C9DDE  00000000 814C9DDE 00FB3788  00000000 00000000 0021 0021 23:0
                             07041000 80000000
 00 0021 009FE030 SVC     4F 00000000_014CA396  009FDCB0 00000011 FFFFFFEF  Status   Start SRBs only    23:0
                             07041000 80000000
 02 0165 009FFB00 DSP        00000000_0137DA1C  00800000 00000001 05882000  00000000 00000000 0049 0165 23:0
                             07044000 80000000
 00 0021 009FE030 SVCR    4F 00000000_014CA396  00080000 00000000 05616F00                            23:0
                             07040000 80000000
 02 0165 009FFB00 PC      ...   0      095701CE            01F01
 02 0165 009FFB00 PR      ...   0      095701CE  29A06B5A                                      0165
```

- TCB1 at 9FE030 in ASID X'21' gets dispatched on CP0
- TCB1 invokes system service STATUS START SRBs via SVC 4F
- TCB2 at 9FFB00 in ASID X'165' gets dispatched on CP2
- TCB1 returns from STATUS (SVCR 4F – SVC/SVCR PSWs match)
- TCB2 issues PC 1F01 which is a user PC (unknown to formatter)
  - PC/PR PSW addresses match

Tourist info: Here we see an xmem environment

30

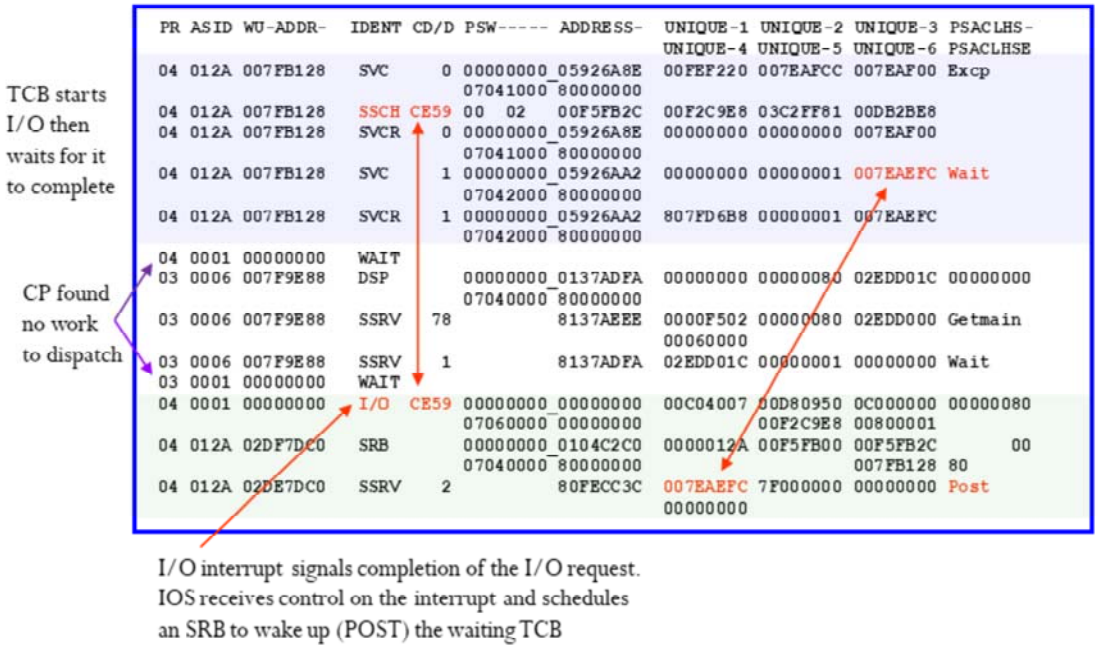
This trace excerpt shows a TCB at address 9FE030 in ASID X'21' running on CP0 at the same time that a TCB at address 9FFB00 in ASID X'165' is running on CP2.

Note that the system trace formatter tells us that the SVC 4F is invoking the STATUS system service, requesting a STATUS START of SRBs.

Note that the SVCR 4F PSW matches the SVC 4F PSW.

Note that the PR PSW address matches that of the PC 1F01. PC 1F01 is a user PC so cannot be identified by the SYSTRACE formatter.

A general example (continued on next slide)



PR	ASID	WU-ADDR-	IDENT	CD/D	PSW-----	ADDRESS-	UNIQUE-1	UNIQUE-2	UNIQUE-3	PSACLHS-
							UNIQUE-4	UNIQUE-5	UNIQUE-6	PSACLHSE
04	012A	007FB128	SVC	0	00000000_05926A8E		00FEF220	007EAFCC	007EAF00	Excp
					07041000_80000000					
04	012A	007FB128	SSCH	CE59	00 02	00F5FB2C	00F2C9E8	03C2FF81	00DB2BE8	
04	012A	007FB128	SVCR	0	00000000_05926A8E		00000000	00000000	007EAF00	
					07041000_80000000					
04	012A	007FB128	SVC	1	00000000_05926AA2		00000000	00000001	007EAEFC	Wait
					07042000_80000000					
04	012A	007FB128	SVCR	1	00000000_05926AA2		807FD6B8	00000001	007EAEFC	
					07042000_80000000					
04	0001	00000000	WAIT							
03	0006	007F9E88	DSP		00000000_0137ADFA		00000000	00000080	02EDD01C	00000000
					07040000_80000000					
03	0006	007F9E88	SSRV	78		8137AEEE	0000F502	00000080	02EDD000	Getmain
							00060000			
03	0006	007F9E88	SSRV	1		8137ADFA	02EDD01C	00000001	00000000	Wait
03	0001	00000000	WAIT							
04	0001	00000000	I/O	CE59	00000000_00000000		00C04007	00D80950	0C000000	00000080
					07060000_00000000			00F2C9E8	00800001	
04	012A	02DF7DC0	SRB		00000000_0104C2C0		0000012A	00F5FB00	00F5FB2C	00
					07040000_80000000				007FB128	80
04	012A	02DE7DC0	SSRV	2		80FECC3C	007EAEFC	7F000000	00000000	Post
							00000000			

TCB 7FB128 in ASID 12A issued an SVC 0 (EXCP) on CPU4. This caused a SSCH to be issued to device CE59. After the SVC 0 completed the TCB issued a WAIT with wait count of 1 and ECB address of 7EAEFC. Then this CPU entered a no-work wait.

TCB 7F9E88 in ASID 6 was dispatched on CPU 3. It issued a getmain for 80 bytes in subpool 245 and obtained the storage at 2EDD000. It then issued an SVC 1 Wait with wait count of 1 and ECB address of 2EDD01C. Then this CPU entered a no-work wait (WAIT trace entry).

On CPU 4, an I/O interrupt from device CE59 occurred. Then an SRB was dispatched to run in I/O POST STATUS routine. It issued a POST with ECB 7EAEFC. This woke up the TCB 7FB128 in ASID 12A

A general example (continued)

PR	ASID	WU-ADDR-	IDENT	CD/D	PSW-----	ADDRESS-	UNIQUE-1	UNIQUE-2	UNIQUE-3	PSACLHS-
							UNIQUE-4	UNIQUE-5	UNIQUE-6	PSACLHSE
04	012A	007FB128	DSP		00000000_05926AA2		00000000	00000001	007EAEFC	00000000
					07042000_80000000					
04	012A	007FB128	SSRV	78		8591F94C	4050E603	00000148	007EAEB8	Freemain
							012A0000			
04	012A	007FB128	SVC	10	00000000_00D29A32		00000000	000065B0	007D5FB8	Purge
					07840000_00000000					
04	012A	007FB128	SSRV	A		812E8442	FFFFFFFF	FE0000F8	007D73D8	Freemain
							012A00FF			
04	012A	007FB128	*SVCE	D	00000000_0116E06E		00000010	84000000	8430A000	00000001
					07041000_80000000		00400004			00000000

TCB gets redispatched now that its I/O has completed.
However, a little while later it suffers an ABEND30A when trying to freemain storage in LSQA SP254.

TCB 7FB128 in ASID 12A was then dispatched on CPU 4. It issued a freemain for 148 bytes in subpool 230, starting from the address of 7EAEB8. Then it issued a SVC 10 (I/O purge). Then the PURGE SVC routine suffered an ABEND30A RC10 while trying to free storage in LSQA SP254. This abend will cause RTM to receive control. When the trace entry is an SVCE D, this indicates that there is something special about the error environment (it is not Enabled Unlocked Task mode, or an EUT FRR exists) and so RTM1 receives control first. If the trace entry is SVC D, then control goes directly into RTM2.

So let's debug with SYSTRACE!

The Mystery of The Disappearing Workarea

IP SYSTRACE ASID(X'1A') TCB(X'5D8728') TI(LO)

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE
00 001A 005D8728  DSP         00000000_08102A00 00000000 00000001 08103D00  00000000
                             07850000 80000000
00 001A 005D8728  PGM    011  00000000_08102A20 00060011 00000000           00000000
                             07851000 80000000          0810B800           00000000
00 001A 005D8728 *RCVY  PROG                    940C4000 00000011 00000000  00000000
                                                                            00000000
00 001A 005D8728  SSRV   12D          814C0AF6  005D8728 000C8000 FF3A0000  Status
                                                00000000
00 001A 005D8728  SSRV   12D          814C0B12  005D8728 000B8000 00000000  Status
                                                00000000
00 001A 005D8728  DSP         00000000_01187B18 00000000 00000001 08103D00  00000000
                             07851000 80000000
00 001A 005D8728  SVC       D 00000000_01187B1A 00000010 00000001 08103D00
                             07851000 80000000
```

Here we see the now-familiar picture of a program check occurring, resulting in an ABEND0C4 PIC11. RTM1 is entered for the ABEND0C4. There are no FRRs so the error is percolated from RTM1 to RTM2.

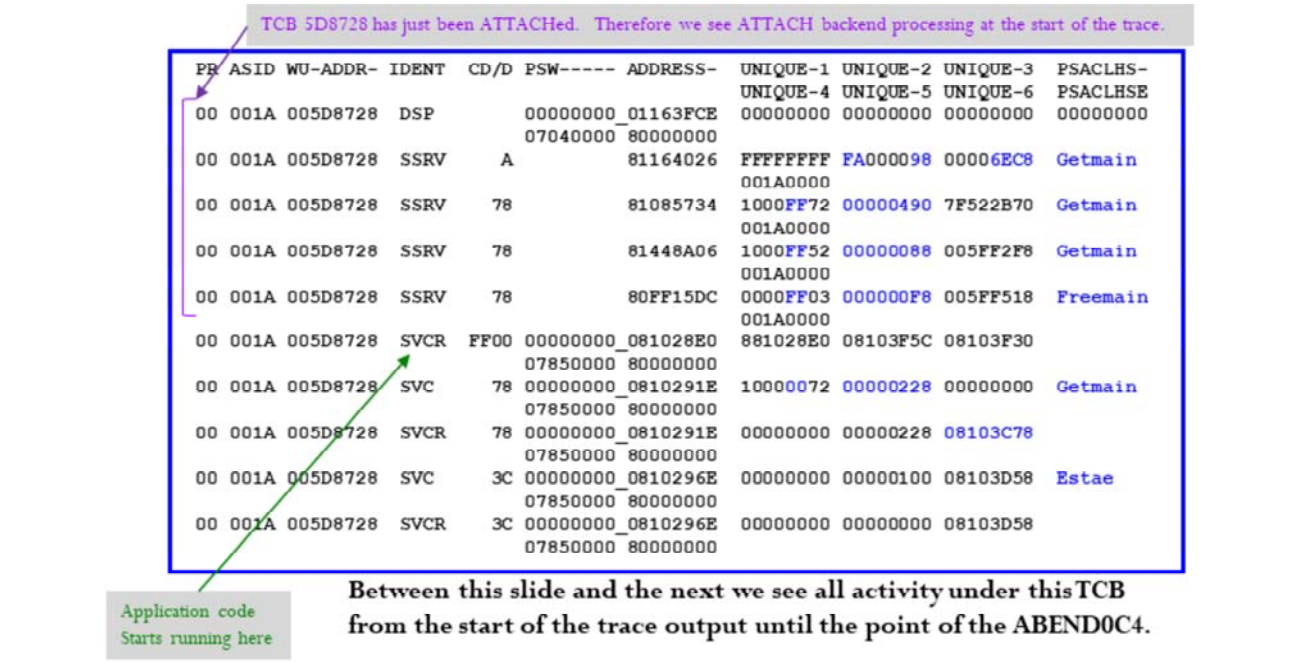
34

We know we have a dump of an ABEND0C4 PIC11 under TCB 5D8728 in ASID 1A. (Perhaps we got this information from IP ST REGS output.)

We filter the system trace output to include only entries for TCB 5D8728 in ASID 1A.

We locate the ABEND0C4 by searching on *RCVY. Just before it is the PGM 11 entry. The Unique-2 and Unique-5 fields together give us the page address (Translation Exception Address = TEA) that the code tried to touch but could not. Note that it is page address 0810B000 (content of last 3 digits is irrelevant). This seems like a reasonable address. Why could the code not touch this page?

The Mystery of The Disappearing Workarea



DSP: The TCB is dispatched. A WHERE on the PSW address would show that this is in module IGC042 in the back end of ATTACH processing. Our TCB has just been ATTACHed.

SSRV A: This GETMAIN is happening under the back end of ATTACH processing. ATTACH is obtaining a work area of length X'98' bytes from SP X'FA' = SP250. The storage is obtained at address 6EC8.

SSRV 78 (twice): Two GETMAINs for SP255 LSQA storage. One GETMAIN is for length X'490' and the other is for length X'88'. The GETMAIN of length X'88' is the RB being obtained for this newly ATTACHed TCB.

SSRV 78: FREEMAIN from SP255 for length X'F8' by operating system code.

SVCR FF00: Operating system is giving control to the new RB. At this point application code now begins to run.

SVC 78/SVCR 78: Application GETMAINs storage in SP0 for length X'228' bytes. Assigned storage address is 8103C78.

SVC 3C/SVCR 3C: Application sets up ESTAE recovery.

The Mystery of The Disappearing Workarea

PR	ASID	WU-ADDR-	IDENT	CD/D	PSW-----	ADDRESS-	UNIQUE-1	UNIQUE-2	UNIQUE-3	PSACLHS-
							UNIQUE-4	UNIQUE-5	UNIQUE-6	PSACLHSE
00	001A	005D8728	SVC	78	00000000	08102998	30000072	00007000	00000000	Getmain
					07850000	80000000				
00	001A	005D8728	SVCR	78	00000000	08102998	00000000	00007000	08104000	
					07850000	80000000				
00	001A	005D8728	SVC	78	00000000	081029BA	30000072	00001000	00000000	Getmain
					07850000	80000000				
00	001A	005D8728	SVCR	78	00000000	081029BA	00000000	00001000	0810B000	
					07850000	80000000				
00	001A	005D8728	SVC	78	00000000	081029EE	00000003	00007000	08104000	Freemain
					07850000	80000000				
00	001A	005D8728	SVCR	78	00000000	081029EE	00000000	00007000	08104000	
					07850000	80000000				
00	001A	005D8728	SVC	2	00000000	081029F6	00000000	00000000	08103F58	Post
					07850000	80000000				
00	001A	005D8728	SVCR	2	00000000	081029F6	00C100FF	00000002	00C100FF	
					07850000	80000000				
00	001A	005D8728	SVC	1	00000000	08102A00	00C100FF	00000001	08103D00	Wait
					07850000	80000000				
00	001A	005D8728	SVCR	1	00000000	08102A00	805FF318	00000001	08103D00	
					07850000	80000000				
00	001A	005D8728	DSP		00000000	08102A00	00000000	00000001	08103D00	00000000
					07850000	80000000				
00	001A	005D8728	PGM	011	00000000	08102A20	00060011	00000000		00000000
					07851000	80000000		0810B800		00000000
00	001A	005D8728	*RCVY	PROG			940C4000	00000011	00000000	00000000

SVC 78/SVCR 78: The application does a GETMAIN for X'7000' bytes from SP0 and is given storage at address 8104000 thru 810AFFF.

SVC 78/SVCR 78: The application does a GETMAIN for X'1000' bytes from SP0 and is given storage at address 810B000 thru 810BFFF. Note that page 810B000 is the page for which we suffered the ABEND0C4.

SVC 78/SVCR 78: The application does a FREEMAIN for the 7 pages of storage from 8104000 thru 810AFFF, which it obtained previously.

SVC 2/SVCR 2: The application does a POST of the ECB at 8103F58.

SVC 1/SVCR 1: The application goes into a WAIT on the ECB at 8103D00.

DSP: At some point the application TCB has gotten POSTed because now we see the TCB getting dispatched. (If we were to reformat SYSTRACE to show entries beyond our TCB, we would be able to see the POST occurring.)

PGM 011: Shortly after the point of dispatch, our ABEND0C4 PIC11 occurs while trying to touch the page at 810B000. We just saw this storage get GETMAINed a few entries back. We should be able to touch it! What happened to it??

The Mystery of The Disappearing Workarea

IP SYSTRACE ASID(X'1A') TI(LO)

```
PR ASID WU-ADDR- IDENT  CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE
00 001A 005D8728  DSP          00000000_01163FCE 00000000 00000000 00000000 00000000
                               07040000_80000000
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00 001A 005D8728  SVCR  FF00 00000000_081028E0 881028E0 08103F5C 08103F30
                               07850000_80000000
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00 001A 005D8728  SVC     78 00000000_081029BA 30000072 00001000 00000000 Getmain
                               07850000_80000000
00 001A 005D8728  SVCR    78 00000000_081029BA 00000000 00001000 0810B000
                               07850000_80000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00 001A 005D8728  SVC      1 00000000_08102A00 00C100FF 00000001 08103D00 Wait
                               07850000_80000000
00 001A 005D8728  SVCR     1 00000000_08102A00 805FF318 00000001 08103D00
                               07850000_80000000
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00 001A 005D8590  SVC     78 00000000_08102EF8 00000003 00001000 0810B000 Freemain
                               07850000_80000000
00 001A 005D8590  SVCR    78 00000000_08102EF8 00000000 00001000 0810B000
                               07850000_80000000
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
00 001A 005D8728  DSP          00000000_08102A00 00000000 00000001 08103D00 00000000
                               07850000_80000000
00 001A 005D8728  PGM     011 00000000_08102A20 00060011 00000000          00000000
                               07851000_80000000           0810B800       00000000
00 001A 005D8728 *RCVY  PROG                     940C4000 00000011 00000000 00000000
                                                                            00000000
```

TCB 5D8728 GETMAINs a page of storage at 810B000.

TCB 5D8590 freed the storage that TCB 5D8728 had GETMAINed and was trying to use!

TCB 5D8728 abends trying to touch the storage it obtained.

37

If we know that storage has been GETMAINed, and then when someone tries to touch it they suffer a translation exception such as a PGM 11 (meaning the storage is not available), this implies that someone has freed the storage between the time of the GETMAIN and the time of the abend. In this case, 810B000 was a local storage address. (It was GETMAINed from SP0 which is a private storage subpool.) Therefore, we look for someone within the address space [IP SYSTRACE ASID(X'1A')] doing the freeing of the storage. Had the freed area been in global (common) storage, then we would have needed to look at all address spaces on the system (IP SYSTRACE ALL) to try to find who freed the storage.

The Smoking Gun

IP LIST TITLE

```
TITLE
LIST 00. LITERAL LENGTH(X'4D') CHARACTER
00000000 | COMPON=TASK MGMT,COMPID=SC1CL,ISSUER=IEAVEPST,POST FAILED -- UNE |
00000040 | XPECTED ERROR                                                    |
```

Problem: A dump was produced with a title indicating a failure in the POST service module IEAVEPST.

The Smoking Gun

IP WHERE 1265720 shows PSW points to POST module IEAVEPST+FB8

IP SYSTRACE ASID(X'3F') TI(LO)

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address-  Unique-1 Unique-2 Unique-3  PSACLHS- PSALOCAL PASD SASD
                                                  Unique-4 Unique-5 Unique-6  PSACLHSE
0000 003F 04464C80  SRB         00000000_127CF7A0 0000003F 124C9084 124C9010        00          003F 003F
                                07040000_80000000 008D5988 20
0000 003F 04464C80  PC      ...   6   1  14D405BE           0018A500
0000 003F 04464C80  PC      ...   0            00_14F079A4     0030E           Post
0000 003F 04464C80  PGM     011 00000000_01265720 00040011 00000000           00000001 00DBC580 0099 0099
                                07042000_80000000           00F0F800           00000000
0000 003F 04464C80 *RCVY    PROG                  940C4000 00000011 00000000  00000001 00DBC580 0099 0099
                                                                              00000000
```

- An SRB entered POST via a PC 30E.
- While running under POST processing, a PGM 11 occurred trying to touch storage on the page at 00F0F000 in ASID 99.
- The PGM 11 was not resolvable, resulting in an ABEND0C4 PIC11
- What happened??

39

A translation exception address (TEA) indicates the page that could not be accessed. You cannot tell from the TEA which byte on the page was being accessed. The last 3 digits of the TEA are flags, not part of the address. Therefore, in the above example, we know that the POST code was trying to touch storage somewhere on page F0F000.

The Smoking Gun

- Inspection of POST module IEAVEPST's code shows the following instructions leading up to time of error:

```
ICM     R5,X'7',X'00'(R6)
AHI     R5,-X'40'
SLR     R4,R4
ICM     R4,X'3',X'3A'(R5)   [PGM 11 occurred on this instruction]
```

- At time of error:
  - Reg6=61305FF4 and Reg5=00F0F0B0
  - Content of storage at 61305FF4 is F0F0F0F0 (consistent with above)
  - In fact, the entire page at 61305000 contains X'F0' throughout
  - Other pages around 61305000 contain X'F0' throughout
- Theory: Pages at and around 61305000 in ASID X'99' have been overlaid.

When someone overlays a large quantity of storage, they often take page faults along the way as they touch storage that they shouldn't. Often they even program check when they eventually come to a page of storage that is not getmained.

The Smoking Gun

Gotcha!!

Reminder: our address of interest is 61305000.

IP SYSTRACE ASID(X'99') TI(LO)

```
PR   ASID WU-Addr- Ident  CD/D PSW----- Address-  Unique-1 Unique-2 Unique-3  PSACLHS- PSALOCAL PASD SASD
                                                   Unique-4 Unique-5 Unique-6  PSACLHSE
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612D3400          00000000
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612D4400          00000000
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612D7400          00000000
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612D8400          00000000
0004 0099 007A2E00  EXT    TIMR 00000000_34D4D782  00001005                  00000000 00000000 0099 0099
                               07040400_80000000                             00000000
0004 0099 007A2E00  DSP         00000000_34D4D782  00000000 00000243 0000000C 00000000 00000000 0099 0099
                               07040400_80000000
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612D9400          00000000
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612DA400          00000000
0004 0099 007A2E00  PGM     011 00000000_34D4D79E  00060011 00000000          00000000 00000000 0099 0099
                               07040400_80000000           612DB400          00000000
```


Since the overlaid storage was in private of ASID 99, we look at work running in ASID 99 to see if we see anything suspicious.

Since we saw that page 61305000 was overlaid, we search the trace for nearby addresses. A couple searches we might try are: FIND ' 613' and FIND ' 612' . We would also search for ABEND0C4's.

We find someone page faulting their way through storage as they overlay page after page. Someone has a MVC instruction that has run amok. The owners of the code at PSW address 34D4D79E in ASID 99 need to take a look at this problem.

Appendix

What's new?

Recent changes

- RCVY ESTA and RCVY ESTR show ESTAE entry/retry (R2.2)
- CP numbers now 4 digits rather than 2 (R2.1)
- Friendlier formatting of external interrupts (R2.1)
- New EXT WTI (Warning Track Interrupt) (R2.1)

R13

```
02 00D5 00AA5288  EXT   1005 00000000_00849A32  00001005
                             07851400 80000000
03 0001 00000000  CALL       00000000_00000000  00001202 00000000
                             07060000 00000000
00 00D5 00AC1E88  CLKC       00000000_00FF2B34  00001004 00000000      0000
                             07040000 80000000
01 00D5 00AA57A0  EMS        00000000_0084A78C  00001201 40800000 1870EF28
                             07851400 80000000  815BC1B8
```

R2.1

```
0008 00EC 00000000  EXT   TIMR 00000000_253725E8  00001005
                               07846000 80000000
0008 0001 00000000  EXT   CALL 00000000_00000000  00061202 00000000
                               07060000 00000000
0004 008E 00AFE990  EXT   CLKC 00000000_00CE59CA  00001004 06AD7030      0000
                               07850000 00000000
0008 0001 00000000  EXT    EMS 00000000_00000000  00021201 40800000 0715BA50
                               07060000 00000000  81197E20
0008 00EC 056B3280  EXT    WTI 00000000_20D184D8  00001007
                               07042000 80000000
```


RCVY ESTA was demonstrated in the main part of the presentation.

What is a Warning Track Interrupt? This is an external interrupt triggered by the LPAR to signal that it is about to take away the physical CP that is "backing" this logical CP that is receiving the interrupt. The WTI external interrupt interrupts the unit of work executing on the CP that is about to be stolen, and allows the operating system to "undispatch" the unit of work. This allows the unit of work to be redispatched on another CP. Without the WTI, the unit of work would lose the physical CP that it was executing on, and would be in limbo, unable to execute until the logical CP was matched to a new physical CP. WTI's can only be presented to enabled units of work. WTI's can only be honored if the executing unit of work is preemptable, meaning it can be undispatched from the logical CP on which it is executing and put back into the dispatcher work search queue. (TCBs and some SRBs are preemptable; other SRBs are non-preemptable.)

Hyphens and gaps

'-' 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

- See next slide for explanation of why this is seen in any system trace table.

Explanation of "hyphenated" entries

- Trace buffers are CPU-based
- At any given time, some CPUs may be driving work that writes many trace entries, while other CPUs may be driving work that writes few trace entries
- CPUs driving work with much trace activity will fill up and wrap faster and thus hold a shorter history
- Consider this simplistic example:
  - Work on CP0 is writing 10000 entries in .1 sec
  - Work on CP1 is writing 20000 entries in .1 sec
  - If each trace buffer holds 20000 entries, then:
    - CP0's buffer holds .2 seconds worth of history
    - CP1's buffer holds .1 seconds worth of history
  - SYSTRACE formatter merges entries by time, so CP0 will have a .1 second range of entries that does not exist in CP1

zIIPs, Parked CPs, and Hyphens (oh my!)

- A zIIP is a specialty engine
  - zIIPs often drive less workload than general CPs, therefore writing fewer trace entries, so wrapping their trace buffer less frequently
- A parked CP is a discretionary (vertical low) CP that WLM has decided is not needed under the present workload volume
  - Discretionary CPs may be frequently parked/unparked
  - Parked CPs are not creating trace entries
  - Discretionary CPs will typically have a longer history in their trace buffers than general CPs due to trace inactivity while parked
- It is common for zIIPs and/or discretionary CPs to have a much longer trace history than general CPs

zIIPs, Parked CPs, and Gaps

- Sometimes a zIIP may go several seconds without any work to do
- Sometimes a discretionary CP will be parked for several seconds and so not be executing any work
- Such periods of inactivity can lead to "gaps" in the system trace table:
  - The formatter reports gaps of roughly 2 seconds or greater

From: **IP SYSTRACE CPU(3) ALL TI(LO)**

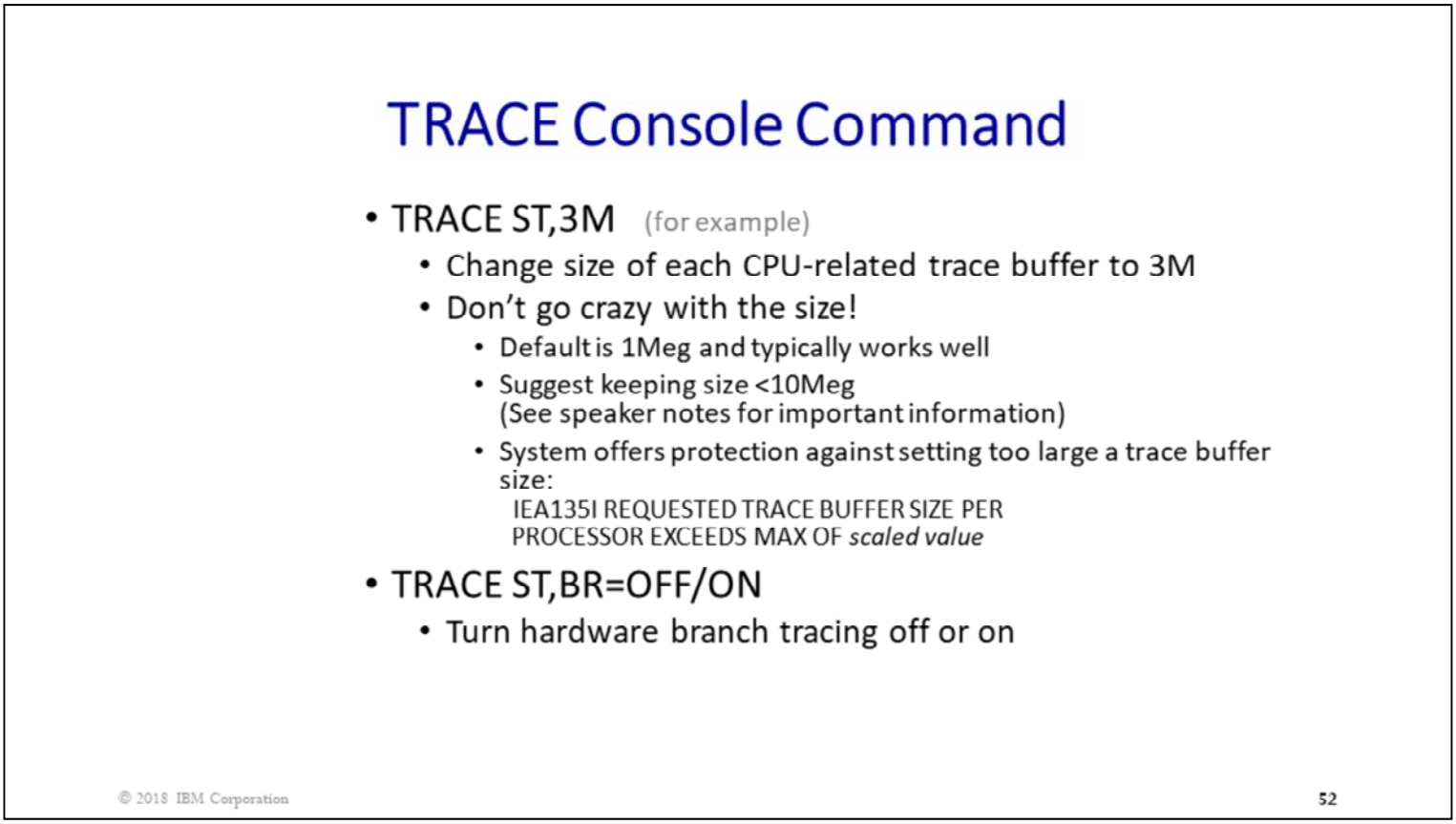
```
0003-0001 00000000  WAIT                                                                    03:16:58.45044950
0003 ******** Time-gap over 00000002 secs. Previous timestamp= 03:16:58.45044950, Current timestamp= 03:17:00.63006119.
0003-0001 00000000  CALL       00000000_00000000  00011202 00000000          00000000 00000000 0001 0001 03:17:00.63006119
                               07060000 00000000                             00000000
```

Hyphens and gaps – 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

- A TCB can get interrupted off of one CP and dispatched on another, so you may not be able to successfully read TCB flow in a hyphenated section of trace
- If the problem can be related to or caused by events on other CPUs, you should try to limit your investigation to the section of the trace table with no '-' entries

Handy commands



Remember: Supply a reasonable value for the system trace buffer size. Consider the available central storage and the actual storage required for system trace. (Size specified is per buffer, and trace buffers are pagefixed.) Supplying a large buffer size value could cause a shortage of pageable storage in the system.

Note that if an unreasonable trace buffer size is entered on the TRACE command, the operating system will issue IEA135I and deny the request. However this is a safety net. The first resource to rely on is your good judgement based on awareness of your system storage needs!

If you omit the nnnM or nG parameter, the system assumes 1M for each processor, or the size established by the last TRACE command that specified a table size during the IPL.

Captured system trace in SAdump

- SYSTRACE TTCH(LIST) TI(LO)
  - Provides the TTCH address and timestamp of system trace snapshots that existed for errors in flight when the system was stopped
- SYSTRACE TTCH(X'yyyyyyyy') plus other trace filters as desired
  - Formats the captured trace represented by a specific TTCH
  - Looks just like a regular system trace report
  - Example on next slide

You cannot find captured system traces in an SVC dump, only in a Sadump. More often then not, there are no captured traces in a Sadump, but sometimes you will find one or two, sometimes dozens. It depends on part whether the system went down rather suddenly versus with a flurry of abends. It is always worth looking for a captured trace in a complex problem since, if available, they offer a peek into the past and sometimes prove to be a real treasure trove of information. See the example on the next page.

Note that a captured trace, when formatted via SYSTRACE, looks identical to any "normal" system trace. In fact it is the same thing, namely trace buffers snapshotted at a particular point in time – it's just a different and earler set than that which the Sadump is displaying via the regular SYSTRACE command without TTCH specified.

Example: captured system trace

IP SYSTRACE TTCH(LIST) TI(LO) [against SAdump only]

Asterisk denotes a mini-trace, not full-sized

```
 TTCH       ASID  TCB       TIME
*7F543000   0038  009CB308  11/20/2015 16:42:46.743560
*7F559000   000C  009FACD8  11/20/2015 16:42:46.740580
*7E666000   0005  009CFE88  11/20/2015 16:42:46.714808
 7E684000   0005  009CFE88  11/20/2015 16:42:46.693258
 7EC58000   0038  009CB308  11/20/2015 16:42:46.443832
 7EFD2000   0001  009BCA40  11/20/2015 16:42:46.345454
```

IP SYSTRACE TTCH(X'7EC58000') TI(LO) ALL

```
---------------------------------------------- System Trace Table ----------------------------------------------
--                                              ------------------
--                                              ------------------
PR   ASID WU-Addr- Ident  CD/D PSW----- Address-  Unique-1 Unique-2 Unique-3  PSACLHS- PSALOCAL PASD SASD Time Local--------
                                                 Unique-4 Unique-5 Unique-6  PSACLHSE PSALOCAL PASD SASD Date-11/18/2015

002C-0001 009FC350  PTI    ...   0      015900FC           0001
002C-0001 009FC350  SSIR   ...     0001                                                          0001
002C-0001 00000000  EXT    EMS 00000000_0123E7D4  00001201 40800000 02FD7958  00000000 00000000 0001 0001 17:34:41.919299338
                               05041000 80000000  8ACA51AE                    00000000
 . . . . . Etc . . . . .
```


The system will snap a mini-trace instead of a full-sized trace once it hits a certain number of “in-flight” system traces. This is done for performance reasons, as frequent simultaneous snapping of full-size system traces by RTM2 can lead to local lock contention issues in the TRACE address space.

Mini system traces are 64K in size and therefore hold significantly less history than full-sized system traces.

Questions?