Finding debugging clues in LE dumps

Thomas Petrolino
tapetro@us.ibm.com
IBM Poughkeepsie
Session 17688
Trademarks

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

- CICS®
- DB2®
- Language Environment®
- OS/390®
- z/OS®

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:
Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
Agenda

- CSI Training
- The Crime is committed
- Collecting the Clues
- Analyzing the Crime Scene
- Back in the Lab
- Forcing LE to Spill the Beans
- Sources of Additional Information
- Appendix
  - Programs
  - Really Cool Lab Equipment
Investigator training
Major clues to look for (ABENDs)

- **U4038**  A severe (unhandled) error occurred.
- **U4039**  A severe (unhandled) error occurred AND LE took a system dump.
- **U4083**  Save area back chain error
- **U4087**  Error during condition processing
- **U4093**  Error during initialization
- **U4094**  Error during termination
Major clues to look for (ABENDs)

- U4038/U4039
  - A severity 2 or greater condition has gone unhandled
  - Application will terminate (gracefully)
  - Clues will be gathered and collected if requested
    - TERMTHDACT controls amount and type of clues
    - CEEDUMP and/or DYNDUMP can be collected
Major clues to look for (ABENDs)

- **U40xx**
  - Application will terminate immediately
  - TERMTHDACT does NOT control what clues are gathered
  - No CEEDUMP is generated
  - DYNDUMP can be used (3rd suboption) to collect clues
Major clues to look for (messages)

- Message (and module) prefixes
  - CEE CEL (but may be reporting errors elsewhere)
  - IGZ COBOL
  - IBM PL/I
  - FOR Fortran (also AFH)
  - EDC C/C++

See z/OS Language Environment Run-Time Messages
Some DNA mapping

- Condition Token (Feedback Code)
  - Example: 00030C89 59C3C5C5 xxxxxxxxx
    - 0003 | 0C89 | 59 | C3C5C5 | xxxxxxxxx
      - 0003 Severity
        - 0000 Informational (I)
        - 0001 Warning (W)
      - 0002 Error (E)
      - 0003 Severe (S)
      - 0004 Critical (C)
Some DNA mapping

- Condition Token (Feedback Code)
  - Example: 00030C89 59C3C5C5 xxxxxxxxx
    - 0003 | 0C89 | 59 | C3C5C5 | xxxxxxxxx
      - 0003  Severity (S)
      - 0C89  Hex message number
        (3209)
      - 59  Flags (ignore)
      - C3C5C5  Hex (EBCDIC) facility ID
        (message prefix) (CEE)
      - xxxxxxxxx  Instance specific info (internal)
  - This token represents message CEE3209S
Timeline of a crime

1. An application begins to run.
2. Out of nowhere there is a gun shot (an 0C9 occurs).
3. LE Paramedics are quickly dispatched to help.
4. Can the patient survive?
   - Yes: The application continues on its journey.
   - No: The app tried to leave clues about the killer (TERMTHD).
5. The CSI team can look at CEEDUMPs or SYSTEM dumps (U4039).
6. The CSI team can look at CEEDUMPs or SYSTEM dumps (U4039).
7. The CSI team can look at CEEDUMPs or SYSTEM dumps (U4039).
8. Call the janitors, time to clean up the scene.
9. How do we publicize the cause of death? (ABTERM)
   - Yes: Issue Original ABEND or LE ABEND Job Terminates.
The crime is committed
The criminal application is written

- A COBOL program (COBOLED1) calls
- A 2nd COBOL program (COBOLED2) which calls
- A C program (CPROG3)
- CPROG3 divides by zero!

See appendix for these programs
Collecting the clues
Major Sources of Evidence

- CEEDUMP
- LEDATA IPCS Verb Exit
- Other Language Environment-produced reports
  - Options Report
  - Storage Report
  - Heap Storage Diagnostics Report
Major Sources of Evidence

- Getting useful information
  - Use Language Environment run-time option TERMTHDACT() to request Language Environment take a dump
    - DUMP CEEDUMP with storage
    - TRACE CEEDUMP with traceback only
  - **UADUMP** CEEDUMP, system dump via U4039
    - UAONLY No CEEDUMP, system dump via U4039
    - UATRACE CEEDUMP (traceback) and system dump via U4039

  - (UAIMM) System dump via original error (only for debug purposes), also TRAP(ON,NOSPIE)
CEEDUMP (the picture)

- Formatted dump produced by Language Environment
  - Failure information, traceback, control blocks, heaps, run-time options report
- Written to CEEDUMP data set:
  - CEEDUMP DD if allocated
  - Dynamically allocated if not available based on CEEDUMP run-time option
    - CEEDUMP(60, SYSOUT = *, FREE = END, SPIN = UNALLOC)
  - CICS written to CESE Transient Data Queue
    - With TERMTHDACT(, CICSDDS) written as part of the CICS dump data set (CICS transaction dump)
CEEDUMP (the picture)

- Advantages
  - Immediately available and readable
  - Can provide a lot of information (with the right set of compile and run-time options)

- Disadvantages
  - Snapshot of the crime scene
    - May not contain all clues necessary to solve the crime
IPCS readable dump (the body)

- IPCS support to format and analyze data in a system dump
- Options to generate numerous reports
  - CEE3ABD information and more
- System dump generated:
  - For an unhandled condition of severity 2 or greater
    - TERMTHDACITY(UADUMP/UATRACE/UASONLY)
      - with SYSMDUMP DD
      - DYNDUMP(hlq,DYNAMIC,TDUMP)
  - When CEE3ABD is called by the application
  - Using system mechanisms (SLIP, Console Dump)
IPCS readable dump (the body)

- **DYNDUMP** run-time option
  - **DYNDUMP**(hlq,U4039-ABEND,U40xx-ABEND)
    - hlq
      - *USERID or *USERID.hlq
      - *TSOPREFIX or *TSOPRE
        - (also *TSOPREFIX.hlq or *TSOPRE.hlq)
      - Up to 26 characters of an MVS data set name
IPCS readable dump (the body)

- **DYNDUMP** run-time option
  - **DYNDUMP**(hlq,U4039-ABEND,U40xx-ABEND)
    - **U4039-ABEND**
      - **NODYNAMIC** (default)
        - DYNDUMP turned off for U4039 ABENDs
      - **DYNAMIC**
        - DYNDUMP active for U4039 ABENDs if no SYSMDUMP, SYSUDUMP or SYSABEND DD.
    - **FORCE**
      - DYNDUMP active for U4039 ABENDs even with above DDs allocated
    - **BOTH**
      - You want it all!!!
IPCS readable dump (the body)

- **DYNDUMP** run-time option
  - **DYNDUMP**(*hlq*,U4039-ABEND,U40xx-ABEND)
    - U4039-ABEND
      - **TERMTHDACT** MUST be set to **UADUMP**, **UATRACE** or **UAONLY** to generate a U4039.
      - U4038 does not produce a dump!
      - Example: **DYNDUMP**(JMONTI,FORCE,TDUMP)

+CEE3798I ATTEMPTING TO TAKE A DUMP FOR ABEND U4039 TO DATA SET: JMONTI.D201.T1336225.JMONTI@B

IEA822I COMPLETE TRANSACTION DUMP WRITTEN TO JMONTI.D201.T1336225.JMONTI@B
+CEE3797I LANGUAGE ENVIRONMENT HAS DYNAMICALLY CREATED A DUMP.
IPCS readable dump (the body)

- **DYNDUMP run-time option**
  - **DYNDUMP**(hlq,U4039-ABEND,U40xx-ABEND)
    - U40xx-ABEND
      - TDUMP (Default)
        - DYNDUMP is active for all U40xx ABENDs (other than U4039) which request a dump.
    - NoTDUMP
      - DYNDUMP is not active for U40xx ABENDs
IPCS readable dump (the body)

- DYNDUMP run-time option
  - Not honored for CICS
  - Use CEMT to request system dumps in CICS
    - CEMT SET TRD(40xx) SYS ADD
IPCS readable dump (the body)

- Advantages
  - More complete picture of the crime scene
  - IPCS tools available for debugging

- Disadvantages
  - Additional skills required for analysis
  - Application programmers may not have access to system dumps and/or IPCS
  - Sometimes have to get your hands dirty (with bits and bytes)
False Leads!
Don’t be fooled!

- Don’t SLIP on Language Environment reissued ABEND (eg, 0C4)
  - Remember this is taking a picture after the crime scene has been cleaned up!
Analyzing the Crime Scene
Messages are the first clue

- Examine the Language Environment message file (usually SYSOUT)

CEE3209S The system detected a fixed-point divide exception (System Completion Code=0C9).
  From compile unit POSIX.RTL.UX29.SRC(CPROG3) at entry point CPROG3 at statement 8 at compile unit offset +0000008C at entry offset +0000008C at address 219C60FC.

- Message tells us what the crime was and where it was committed!
  - A divide by zero occurred at offset x’8C’ in CPROG3 (statement 8)!
Getting a better picture

The header section of the CEEDUMP

CEE3DMP V1 R12.0 Condition processing resulted in the unhandled condition 07/20/10 1:36:25 PM
ASID: 0180   Job ID: JOB15785   Job name: JMONTI@B   Step name: GO   UserID: JMONTI

CEE3845I CEEDUMP Processing started.
The traceback section of the CEEDUMP

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEEHDSP</td>
<td>+000041FA</td>
<td></td>
<td>CEEPLPKA</td>
<td>CEEHDSP</td>
<td>HLE7770</td>
<td>Call</td>
</tr>
<tr>
<td>2</td>
<td>CPROG3</td>
<td>+0000008C</td>
<td>8</td>
<td>CPROG3</td>
<td>CPROG3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>@@FECB</td>
<td>+003122B7C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>@@DB2C</td>
<td>+00000102</td>
<td></td>
<td>CEEEV003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>IGZCFCC</td>
<td>+000002C0</td>
<td></td>
<td>IGZCPAC</td>
<td>IGZCFCC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>COBOLED2</td>
<td>+000004C0</td>
<td>28</td>
<td>COBOLSHR</td>
<td>COBOLED2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>COBOLED1</td>
<td>+0000037A</td>
<td>10</td>
<td>COBOLSHR</td>
<td>COBOLED1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSA Addr</th>
<th>E Addr</th>
<th>PU Addr</th>
<th>PU Offset</th>
<th>Comp Date</th>
<th>Compile Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2199D768</td>
<td>217C8180</td>
<td>217C8180</td>
<td>+000041FA</td>
<td>20100319</td>
<td>CEL</td>
</tr>
<tr>
<td>2199D6D8</td>
<td>219C6070</td>
<td>219C6070</td>
<td>+0000008C</td>
<td>20070212</td>
<td>C/C++</td>
</tr>
<tr>
<td>2199D5E8</td>
<td>219BDE40</td>
<td>219BDE40</td>
<td>+003122B7C</td>
<td></td>
<td>LIBRARY</td>
</tr>
<tr>
<td>2199D56D</td>
<td>21C504F0</td>
<td>21C504F0</td>
<td>+00000102</td>
<td></td>
<td>LIBRARY</td>
</tr>
<tr>
<td>2199D380</td>
<td>21922548</td>
<td>21922548</td>
<td>+000002C0</td>
<td>20100316</td>
<td>LIBRARY</td>
</tr>
<tr>
<td>2199D1D8</td>
<td>21702220</td>
<td>21702220</td>
<td>+000004C0</td>
<td>20100720</td>
<td>COBOL</td>
</tr>
<tr>
<td>2199D030</td>
<td>21700000</td>
<td>21700000</td>
<td>+0000037A</td>
<td>20100720</td>
<td>COBOL</td>
</tr>
</tbody>
</table>

Fully Qualified Names

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>Program Unit</th>
<th>Load Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>CPROG3</td>
<td>POSIX.RTL.UT29.SRC(CPROG3)</td>
<td>CPROG3</td>
</tr>
</tbody>
</table>
Getting a better picture

The condition information section of the CEEDUMP

Condition Information for Active Routines
Condition Information for POSIX.RTL.UT29.SRC(CPROG3) (DSA address 2199D6D8)

CIB Address: 2199E060
Current Condition:
CEE0198S The termination of a thread was signaled due to an unhandled condition.

Original Condition:
CEE3209S The system detected a fixed-point divide exception (System Completion Code=0C9).

Location:
Program Unit: POSIX.RTL.UT29.SRC(CPROG3)
Entry: CPROG3 Statement: 8 Offset: +0000008C

Machine State:
ILC..... 0002 Interruption Code..... 0009
PSW..... 078D2400 A19C60FE
GPR0..... 00000000_00000000 GPR1..... 00000000_0000000A GPR2..... 00000000_A1CD09BC
GPR3..... 00000000_219C60B8 GPR4..... 00000000_2199D2D8 GPR5..... 00000000_21F91A00 GPR6..... 00000000_21F92AC8
GPR7..... 00000000_219BDE40 GPR8..... 00000000_A19C63A8 GPR9..... 00000000_21F93368 GPR10..... 00000000_A19C6070
GPR11.... 00000000_A19C60A0 GPR12.... 00000000_21713B58 GPR13.... 00000000_2199D6D8 GPR14.... 00000000_00000000
GPR15.... 00000000_00000006

Storage dump near condition, beginning at location: 219C60EC
+000000 219C60EC D0848910 0011B01 41E00006 8EE00020 1DE0180F 5000D080 4400C1AC 47F03058 ...
Getting a better picture

The storage around regs section of the CEEDUMP

Storage around GPR15(21922548)

-0020 21922528  F0F3F1F6  F2F0F4F4  F0F0F0F3  F0C3F0F0  00074040 4040
+0000 21922548  47F0F028  00C3C5C5  000001E0  00000D78  47F0F001 0000
+0020 21922568  00000000  0000D500  90ECD00C  18BF1841  58A09058 58C0

Local Variables:

18  01 WS-VARS          AN-GR
19  02 WS-COMP1         S9999 COMP       +00010
20  02 WS-COMP2         S9999 COMP       +00000
21  02 WS-COMP3         S9999 COMP       +00032
22  01 DYN-NAME         X(8) DISP        'CPROG3 ' 

• To obtain Local Variables you must use the TEST compiler option
Even more clues!

- CEEDUMP also contains
  - Run-time options report
  - COBOL working storage
  - Language Environment control blocks
  - Heap storage
  - Language specific information
  - Any much more

- But it is only a picture of the crime
Examining the body

- IPCS Readable dumps
  - A bigger better picture of the crime
  - Use VERBX LE DATA
    - ‘Summary’ general info and run-time options
    - ‘CEEDUMP’ traceback similar to CEEDUMP
    - ‘CM’ Condition management
    - ‘SM’ Storage Management (Stacks/Heaps)
    - ‘HEAP’ Heap data
    - ‘STACK’ Stack data
    - ‘NTHREAD’ traceback for all threads
    - ‘ALL’ All the output
Examining the body

- IP VERBX LEDATA ‘CEEDUMP’

Traceback:

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEEHSDMP</td>
<td>+000000DA</td>
<td>CEEPLPKA</td>
<td>CEEHSDMP</td>
<td>HLE7770</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CEEHDSPI</td>
<td>+000003EBA</td>
<td>CEEPLPKA</td>
<td>CEEHDSPI</td>
<td>HLE7770</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CPROG3</td>
<td>+0000008C</td>
<td>CPROG3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>@@FECB</td>
<td>+00312B7A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>@@DB2C</td>
<td>+00000102</td>
<td>CEEEV003</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>IGZCFCC</td>
<td>+000002C0</td>
<td>IGZCPAC</td>
<td>IGZCFCC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>COBOLED2</td>
<td>+000004C0</td>
<td>COBOLSHR</td>
<td>COBOLED2</td>
<td>.....%</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>COBOLED1</td>
<td>+0000037A</td>
<td>COBOLSHR</td>
<td>COBOLED1</td>
<td>.....%</td>
<td>Call</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSA Addr</th>
<th>PU Addr</th>
<th>PU Offset</th>
<th>Comp Date</th>
<th>Compile Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>219A0860</td>
<td>217D8038</td>
<td>+000000DA</td>
<td>20100319</td>
<td>CEL</td>
</tr>
<tr>
<td>2199D768</td>
<td>217C8180</td>
<td>+000003EBA</td>
<td>20100319</td>
<td>CEL</td>
</tr>
<tr>
<td>2199D6D8</td>
<td>219C6070</td>
<td>+0000008C</td>
<td>20070212</td>
<td>C/C++</td>
</tr>
<tr>
<td>2199D5E8</td>
<td>219BDE40</td>
<td>+00312B7A</td>
<td>20100316</td>
<td>LIBRARY</td>
</tr>
<tr>
<td>2199D560</td>
<td>21C504F0</td>
<td>+00000102</td>
<td>20100720</td>
<td>COBOL</td>
</tr>
<tr>
<td>2199D380</td>
<td>21922548</td>
<td>+000002C0</td>
<td>20100720</td>
<td>COBOL</td>
</tr>
<tr>
<td>2199D1D8</td>
<td>21702220</td>
<td>+000004C0</td>
<td>20100720</td>
<td>COBOL</td>
</tr>
<tr>
<td>2199D030</td>
<td>21700000</td>
<td>+0000037A</td>
<td>20100720</td>
<td>COBOL</td>
</tr>
</tbody>
</table>
Examining the body

- **IP VERBX LEDATA ‘CM’**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBH</td>
<td>2170F410</td>
</tr>
<tr>
<td>EYE:CIBH</td>
<td>000000000000</td>
</tr>
<tr>
<td>BACK:00000000</td>
<td>FRWD:2199E570</td>
</tr>
<tr>
<td>PTR_CIB:2199E060</td>
<td></td>
</tr>
<tr>
<td>FLAG1:C5</td>
<td>ERROR_LOCATION_FLAGS:</td>
</tr>
<tr>
<td>HDLQ:00000000</td>
<td>STATE:00000000</td>
</tr>
<tr>
<td>PRM_DESC:00000000</td>
<td></td>
</tr>
<tr>
<td>PRM_PREFIX:00000000</td>
<td></td>
</tr>
<tr>
<td>PRM_LIST:2199E078</td>
<td>2199E140 2199E14C 2170FA5C</td>
</tr>
<tr>
<td>PARM_DESC:00000000</td>
<td>PARM_PREFIX:00000000</td>
</tr>
<tr>
<td>PARM_LIST:2199E13C</td>
<td>2199E060 2199E14C 2170FA5C</td>
</tr>
<tr>
<td>CIB_SIZ:010C</td>
<td>CIB_VER:0004</td>
</tr>
<tr>
<td>FLG_5:48</td>
<td>FLG_6:23</td>
</tr>
<tr>
<td>FLG_7:04</td>
<td>FLG_8:00</td>
</tr>
<tr>
<td>FLG_1:00</td>
<td>FLG_2:00</td>
</tr>
<tr>
<td>FLG_3:00</td>
<td>FLG_4:05</td>
</tr>
<tr>
<td>OLD_COND_64:00030C89</td>
<td>59C3C5C5 (CEE3209S)</td>
</tr>
<tr>
<td>OLD_MIB:00000001</td>
<td>COND_64:00030C89 59C3C5C5 (CEE3209S)</td>
</tr>
<tr>
<td>MIB:00000001</td>
<td>PL:219C6018</td>
</tr>
<tr>
<td>SV1:2199D6D8</td>
<td>SV2:2199D6D8</td>
</tr>
<tr>
<td>SV1:2199D6D8</td>
<td>INT:219C60FC</td>
</tr>
<tr>
<td>MID:00000003</td>
<td></td>
</tr>
<tr>
<td>HDL_SF:21714600</td>
<td>HDL_EPT:A1918000</td>
</tr>
<tr>
<td>HDL_RST:00000000</td>
<td></td>
</tr>
<tr>
<td>RSM_SF:2199D6D8</td>
<td>RSM_POINT:219C60FE</td>
</tr>
<tr>
<td>RSM_MACHINE:2170F858</td>
<td></td>
</tr>
<tr>
<td>COND_DEFAULT:00000003</td>
<td>Q_DATA_TOKEN:2170F548</td>
</tr>
<tr>
<td>FDBK:00000000</td>
<td></td>
</tr>
<tr>
<td>ABNAME:.........</td>
<td>BBRANCH_OFFSET:00000000</td>
</tr>
<tr>
<td>BBRANCH_STMTID:.......</td>
<td>BBRANCH_STMTLEN:0000</td>
</tr>
</tbody>
</table>
Examining the body

- IP VERBX LEDATA ‘CM’
  - PSW and Regs at time of condition

Machine State
+000248  MCH_EYE:ZMCH
+000250  **GPR00:00000000**  GPR01:0000000A
+000258  GPR02:A1CD09BC  GPR03:219C60B8
+000260  GPR04:2199D2D8  GPR05:21F91A00
+000268  GPR06:21F92AC8  GPR07:219BDE40
+000270  GPR08:A19C63A8  GPR09:21F93368
+000278  GPR10:A19C6070  GPR11:A19C60A0
+000280  GPR12:21713B58  GPR13:2199D6D8
+000288  **GPR14:00000000**  GPR15:00000006
+000290  **PSW:078D2400**  A19C60FE
+000298  ILC:0002  IC1:00  IC2:09  PFT:00000000
Back in the lab
Who Took the Wild Branch…

The Clue: The Breaking Event Address

- Available on z/Architecture machines with PER-3 facility
- 64-bit CPU register that is updated with the address of any instruction that causes a break in sequential instruction execution
- BEA register contents saved when interrupt occurs
- BEA value saved in the SDWA and EPIE
- LE uses this value to identify possible wild branch location
  - LE may also attempt to use linkage register (i.e., R14)
Who Took the Wild Branch...

CEEDUMP Output

Traceback:

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEEH DSP</td>
<td>+000040EC</td>
<td></td>
<td>CEEPLPKA</td>
<td>CEEH DSP</td>
<td>UK28165</td>
<td>Call</td>
</tr>
<tr>
<td>2</td>
<td>OFF CLIFF</td>
<td>-20904128</td>
<td></td>
<td>CSICB BEA</td>
<td>OFF CLIFF</td>
<td></td>
<td>Exception</td>
</tr>
<tr>
<td>3</td>
<td>LEAP</td>
<td>+0000031E</td>
<td>21</td>
<td>CSICB BEA</td>
<td>LEAP</td>
<td></td>
<td>Call</td>
</tr>
<tr>
<td>4</td>
<td>WILD</td>
<td>+0000031E</td>
<td>9</td>
<td>CSICB BEA</td>
<td>WILD</td>
<td></td>
<td>Call</td>
</tr>
</tbody>
</table>

Condition Information for Active Routines

Condition Information for OFF CLIFF (DSA address 20998370)

CIB Address: 20998E30

Current Condition:
CEE0198S The termination of a thread was signaled due to an unhandled condition.

Original Condition:
CEE3201S The system detected an operation exception (System Completion Code=0C1).

Location:
Program Unit: OFF CLIFF Entry: OFF CLIFF Statement:  Offset: -20904128

Possible Bad Branch: Statement: 35 Offset: +00000342

Machine State:
ILC..... 0002 Interruption Code..... 0001
PSW..... 078D0000  80000002
GPR0..... 209B8214 GPR1..... 00000000 GPR2..... 209B8190 GPR3..... 20904400
GPR4..... 20904160 GPR5..... 20994100 GPR6..... 00000000 GPR7..... 00FD8640
GPR8..... 209B8210 GPR9..... 20994C78 GPR10.... 20904250 GPR11.... 20904324
GPR12.... 209139C0 GPR13.... 20998370 GPR14.... A090446C GPR15.... 00000000
Who Took the Wild Branch...

LEDATA Output (‘CM’ or ‘ALL’)

CIBH: 209102A8
+EYE:CIBH BACK:00000000 FRWD:20999340

+000070 OLD_MIB:00000000

+0000BC ABNAME:........

+000220 BBRANCH_STMTID:35

Machine State

MCH_EYE:ZMCH

+000250 GPR00:209B8214 GPR01:00000000
+000258 GPR02:209B8190 GPR03:20904400
+000260 GPR04:20904160 GPR05:20994100
+000268 GPR06:00000000 GPR07:00FD8640
+000270 GPR08:209B8210 GPR09:20994C78
+000278 GPR10:20904250 GPR11:20904324
+000280 GPR12:209139C0 GPR13:20998370
+000288 GPR14:A090446C GPR15:00000000
+000290 PSW:078D0000 80000002

+0002EC INT_SF:00000000 FLAGS:40 EXT:00000000 BEA:2090446A
Forcing LE to Spill the Beans
Coercing Evidence from C-RTL

`perror()`: Prints string with `errno` message to stderr

`strerror()`: Returns message string for a given `errno` value

Environment variables to tell these functions to provide more information

- `_EDC_ERRNO_DIAG`
- `_EDC_ADD_ERRNO2`
Coercing Evidence from C-RTL...

_EDC_ERRNO_DIAG = “x,y”

**x**
- 0: Do not generate additional diagnostic information (default)
- 1: Call ctrace() – writes a traceback to CEEDUMP
- 2: Call csnap() – writes a condensed dump to CEEDUMP
- 3: Call cdump() – writes a CEEDUMP; sends snap dump to CEESNAP

**y**
- List of errnos for which action “x” is to be taken, separated by commas
- If not specified, defaults to all errno values
Coercing Evidence from C-RTL…

▫️ _EDC_ADD_ERRNO2

▫️ Tells perror() and strerror() whether to append errno2 information to their output

▫️ For strerror(121):
  ▫️ _EDC_ADD_ERRNO2 = “0” or unset:
  ▫️ EDC5121I Invalid argument
  ▫️ With _EDC_ADD_ERRNO2 = “1”:
    ▫️ EDC5121I Invalid argument. (errno2=0x0C0F8402)

▫️ Default perror() behavior for z/OS V1.9 is to include errno2 value

▫️ Interpret errno2 value using:
  ▫️ BPXMTEXT TSO command (handles both USS and LE errno2 values)
  ▫️ z/OS UNIX System Services Messages and Codes
Sources of Additional Information

- z/OS Language Environment Debugging Guide
- z/OS Language Environment Programming Reference
- z/OS MVS IPCS Commands
- z/OS UNIX System Services Messages and Codes
- Web site
Appendix
IDENTIFICATION DIVISION.
PROGRAM-ID. COBOLED1.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-VARS.
  05 WS-COMP1 PIC S9(4) COMP-4 VALUE 1234.
PROCEDURE DIVISION.
MAIN-PROG.
  CALL "COBOLED2".
  STOP RUN.
END PROGRAM COBOLED1.
IDENTIFICATION DIVISION.
PROGRAM-ID. COBOLED2.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-VARS.
   05 WS-COMP1 PIC S9(4) COMP-4 VALUE ZEROES.
   05 WS-COMP2 PIC S9(4) COMP-4 VALUE ZEROES.
   05 WS-COMP3 PIC S9(4) COMP-4 VALUE ZEROES.
   01 DYN-NAME PIC X(8).
PROCEDURE DIVISION.
MAIN-PROG.
   MOVE 32 TO WS-COMP3.
   MOVE 10 TO WS-COMP1.
   MOVE "CPROG3" TO DYN-NAME.
   CALL DYN-NAME.
   STOP RUN.
END PROGRAM COBOLED2.
Program CPROG3

```c
#pragma linkage(CPROG3,COBOL)
#include <stdio.h>
void CPROG3()
{
    int i,j,k;
    /* Compiler needs to be coerced to */
    /* divide by zero. */
    j = 5;
    k = 10;
    i = 6 / (k - j*2);
    return;
}
```
Program WILD

IDENTIFICATION DIVISION.
PROGRAM-ID. WILD.

DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
PROCEDURE DIVISION.
MAIN-PROG.
  CALL "LEAP".
  GOBACK.
END PROGRAM "WILD".
Program LEAP

IDENTIFICATION DIVISION.
PROGRAM-ID. LEAP.

DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
PROCEDURE DIVISION.
MAIN-PROG.
   CALL "OFFCLIFF".
   GOBACK.
END PROGRAM "LEAP".
IDENTIFICATION DIVISION.
PROGRAM-ID. OFFCLIFF.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 FP USAGE IS PROCEDURE-POINTER.
01 OBVIOUS PIC X(4) VALUE "OBVS".

LINKAGE SECTION.
PROCEDURE DIVISION.
MAIN-PROG.
   SET FP TO NULL.
   CALL FP.
   DISPLAY "SURVIVED THE JUMP!"
   GOBACK.
END PROGRAM "OFFCLIFF".
Really cool lab equipment
Reconstructing a Vandalized Stack

The Crime: Stack control data is damaged, unable to get a complete Traceback:

Registers and PSW:

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPR0</td>
<td>84000000</td>
</tr>
<tr>
<td>GPR1</td>
<td>84000FF3</td>
</tr>
<tr>
<td>GPR2</td>
<td>00010078</td>
</tr>
<tr>
<td>GPR3</td>
<td>20EE0DF0</td>
</tr>
<tr>
<td>GPR4</td>
<td>20F21018</td>
</tr>
<tr>
<td>GPR5</td>
<td>20EE2530</td>
</tr>
<tr>
<td>GPR6</td>
<td>20EE149C</td>
</tr>
<tr>
<td>GPR7</td>
<td>20F016B0</td>
</tr>
<tr>
<td>GPR8</td>
<td>20EE22E8</td>
</tr>
<tr>
<td>GPR9</td>
<td>00000004</td>
</tr>
<tr>
<td>GPR10</td>
<td>20ED6378</td>
</tr>
<tr>
<td>GPR11</td>
<td>2181018C</td>
</tr>
<tr>
<td>GPR12</td>
<td>20EE5A20</td>
</tr>
<tr>
<td>GPR13</td>
<td>20EE13F0</td>
</tr>
<tr>
<td>GPR14</td>
<td>A180FFA8</td>
</tr>
<tr>
<td>GPR15</td>
<td>00000004</td>
</tr>
<tr>
<td>PSW</td>
<td>078D1000</td>
</tr>
<tr>
<td>PSW</td>
<td>A180FFBA</td>
</tr>
</tbody>
</table>

Traceback:

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E</th>
<th>Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
</table>

WARNING An invalid DSA pointer was found on traceback while processing

DSA: 20EE13F0 and CAA: 20EE5A20
Reconstructing a Vandalized Stack...

The Clues:

- **CEECAADDSDA, “Dummy DSA”** – First Stack Frame on the stack

- **Backchain Pointer** – Location in stack frame that points to previous stack frame, offset X’04’

- **Next Available Byte, “NAB”** – Location in stack frame where next stack frame is to be built, offset X’4C’

The Tool: IPCS **RUNCHAIN** command

- Processes a chain of control blocks

- Input: Starting address, “link” offset
Reconstructing a Vandalized Stack...

LEDATA Output

```
CEEDAA:  20EE5A20
+000000  FLAG0:00  LANGP:08  BOS:20F01018  EOS:20F21018
+000044  TORC:00000000  TOVF:80071660  ATTN:20EE0E38
+0002CC  DMC:00000000  ABCODE:00000000  RSNCODE:00000000
+0002D8  ERR:20EE2DF0  GETSX:80070B20  DDSA:20EE63C0
+0002E4  SECTSIZ:00000000  PARTSUM:00000000
+0002EC  SSEXPNT:00000000  EDB:20EE4698  PCB:20EE41E8
```
Reconstructing a Vandalized Stack…

ip runchain address(20ee63c0) display length(100) link(76) name(dsa)

: 

DSA005
LIST 20F01510 ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F01510.) KEY(88)
20F01510. 00104001 20F01370 00000000 A0ED44E0
20F01520. 20ED6250 20EFDE90 00000000 20F21230 20ED4478 20ED41F8 20EFD100 00000000
20F01540. 00FD8640 00058A80 20FDE60 20ED42E8 20ED43A0 20EE5A20 00000004 20F016B0
20F01560. 20F01410 20ED1F18 20F01510 20EFDE60 20EFD100

DSA006
LIST 20F016B0 ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F016B0.) KEY(88)
20F016B0. 00104001 20F01510 20F018D0 A0ED8730
20F016C0. 80072020 A0ED85C0 20EFE048 00000000 20ED4478 20ED6288 20EFD100 00000000
20F016E0. 00FD8640 00058A80 000581BC 20ED42E8 20ED83C0 20EE5A20 00000004 20F01850
20F01700. 20F015B0 20ED3FA8 20F016B0 20EFE048 20EFD100

DSA007
LIST 20F01850 ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F01850.) KEY(88)
20F01850. 00002001 20F016D0 20F01A70 A18F2934
20F01860. A18E2588 00000000 00000000 00000095 00102001 20F01610 20F019F0 A198CE12
20F01880. 20ED61C8 00000001 20F21230 20EFD6D0 20ED6064 00000010 00000004 20ED41F8
20F018A0. 20F01750 20ED6038 20EFDE60 20ED62D4 20EFD100

:
Reconstructing a Vandalized Stack...

LEDATA ‘DSA(20F016B0) CEEDUMP' Output

Traceback:

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LOOK</td>
<td>+000024DE</td>
<td>CSICBLDP</td>
<td>LOOK</td>
<td>.....%.</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FORTHE</td>
<td>+0000031E</td>
<td>CSICBLDP</td>
<td>FORTHE</td>
<td>.....%.</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>WEAPON</td>
<td>+0000031E</td>
<td>CSICBLDP</td>
<td>WEAPON</td>
<td>.....%.</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+20ED03BE</td>
<td>Call</td>
<td></td>
</tr>
</tbody>
</table>

WARNING An invalid DSA pointer was found on traceback while processing

DSA: BADBADBAD and CAA: 20EE5A20

<table>
<thead>
<tr>
<th>DSA</th>
<th>DSA Addr</th>
<th>E Addr</th>
<th>PU Addr</th>
<th>PU Offset</th>
<th>Comp Date</th>
<th>Compile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20F016B0</td>
<td>20ED6250</td>
<td>20ED6250</td>
<td>+000024DE</td>
<td>20080128</td>
<td>COBOL</td>
</tr>
<tr>
<td>2</td>
<td>20F01510</td>
<td>20ED41C0</td>
<td>20ED41C0</td>
<td>+0000031E</td>
<td>20080128</td>
<td>COBOL</td>
</tr>
<tr>
<td>3</td>
<td>20F01370</td>
<td>20ED2130</td>
<td>20ED2130</td>
<td>+0000031E</td>
<td>20080128</td>
<td>COBOL</td>
</tr>
<tr>
<td>4</td>
<td>20F011D0</td>
<td>00000000</td>
<td>00000000</td>
<td>+20ED03BE</td>
<td>**********</td>
<td></td>
</tr>
</tbody>
</table>
Reconstructing a Vandalized Stack...

ip runchain address(20ee63c0) display length(100) link(76) name(dsa)

DSA001
LIST 20EE63C0. ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20EE63C0.) KEY(88)
20EE63C0. 00000000 000D008 00000000 0006AB30 20ECE000 00000000 20EE47B8 00000000
20EE63E0 LENGTH(X'20')==>All bytes contain X'00'
20EE6400. 00000000 20EE5A20 00000000 20F01030 20F01030 00000000 00000000 00000000
20EE6420. 00000000

DSA002
LIST 20F01030. ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F01030.) KEY(88)
20F01030. 00104001 20EE63C0 20F014A8 A0ECE324
20F01040. 20ED00A0 20EFD578 00000000 20F21050 20ECE2BC 20ECE038 20EE47B8 00000000
20F01060. 00000000 00058A80 20EFD448 20ECE128 20ECE1E4 20EE5A20 00000000 20F011D0
20F01080. 00000000 00000000 20F011D0 20EFD448 00000000

DSA003
LIST 20F011D0. ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F011D0.) KEY(88)
20F011D0. 00104001 BADBADBD 00000000 A0ED03C0
20F011E0. 20ED2130 20EFDBC0 00000000 20F210F0 20ED0358 20ED00D8 20EFD100 00000000
20F01200. 00FD8640 00058A80 20EFDA90 20ED01C8 20ED0280 20EE5A20 00000000 20F01370
20F01220. 00000000 00000000 20F011D0 20EFDA90 20EE47B8
Reconstructing a Vandalized Stack...

LEDATA ‘DSA(20F01030) CEEDUMP’ Output

Traceback:

<table>
<thead>
<tr>
<th>DSA Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+00000322</td>
<td></td>
<td>CSICBLDP</td>
<td>DUMPSTER</td>
<td>.....%.  Call</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSA</th>
<th>DSA Addr</th>
<th>E Addr</th>
<th>PU Addr</th>
<th>PU Offset</th>
<th>Comp Date</th>
<th>Compile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20F01030</td>
<td>20ECE000</td>
<td>20ECE000</td>
<td>+00000322</td>
<td>20080128</td>
<td>COBOL</td>
</tr>
</tbody>
</table>
Reconstructing a Vandalized Stack...

What about the corrupted stack frame?

```
ip runchain address(20ee63c0) display length(100) link(76) name(dsa)

DSA002
LIST 20F01030. ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F01030.) KEY(88)
 20F01030. 00104001 20EE63C0 20F014A8 A0ECE324
 20F01040. 20ED00A0 20EFD578 00000000 20F21050 20ECE2BC 20ECE038 20EE47B8 00000000
 20F01060. 00000000 00058A80 20EFD448 20ECE128 20ECE1E4 20EE5A20 00000000 20F011D0
 20F01080. 00000000 00000000 20F01030 20EFD448 00000000

DSA003
LIST 20F011D0. ASID(X'01CE') LENGTH(X'64') AREA
ASID(X'01CE') ADDRESS(20F011D0.) KEY(88)
 20F011D0. 00104001 BADBADBD 00000000 A0ED03C0
 20F011E0. 20ED2130 20EFDBC0 00000000 20F210F0 20ED0358 20ED00D8 20EFD100 00000000
 20F01200. 00FD8640 00058A80 20EFDA90 20ED01C8 20ED0280 20EE5A20 00000000 20F01370
 20F01220. 00000000 00000000 20F011D0 20EFDA90 20EE47B8
```

R15 slot of the previous stack frame provides yet another clue!
Reconstructing a Vandalized Stack...

**IPCS Browse Panel**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>20ED00A0</td>
<td>47F0F028</td>
<td>00C3C5C5</td>
<td>000001A0</td>
<td>00000014</td>
<td>( \cdot \cdot \cdot )C( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED00B0</td>
<td>47F0F001</td>
<td>98CEAC00</td>
<td>20ED0156</td>
<td>00000000</td>
<td>( \cdot \cdot \cdot )q( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED00C0</td>
<td>00000000</td>
<td>00000000</td>
<td>90ECD00C</td>
<td>4110F038</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED00D0</td>
<td>98EFF04C</td>
<td>07FF0000</td>
<td>20ED00A0</td>
<td>00000000</td>
<td>q( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED00E0</td>
<td>20ED1F18</td>
<td>20ED014E</td>
<td>20ED00A0</td>
<td>20ED02CE</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED00F0</td>
<td>20ED83C0</td>
<td>20ED016A</td>
<td>00104001</td>
<td>00000008</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0100</td>
<td>C9D5E3C8</td>
<td>C5404040</td>
<td>F2F0F0F8</td>
<td>F0F1F2F8</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0110</td>
<td>F1F7F4F0</td>
<td>F3F5F0F3</td>
<td>F0F3F0F0</td>
<td>04740000</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0120</td>
<td>0000076C</td>
<td>A0C87CCC</td>
<td>20000000</td>
<td>10000000</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0130</td>
<td>00000000</td>
<td>08000000</td>
<td>08000000</td>
<td>00000000</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0140</td>
<td>00000002</td>
<td>0000F000</td>
<td>40404040</td>
<td>0005C9D5</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0150</td>
<td>E3C8C540</td>
<td>40400500</td>
<td>000120ED</td>
<td>83100000</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
<tr>
<td>20ED0160</td>
<td>0000FFFF</td>
<td>FFB220ED</td>
<td>00A020ED</td>
<td>00D80000</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
<td>( \cdot \cdot \cdot )</td>
</tr>
</tbody>
</table>
Reconstructing a Vandalized Stack...

Complications

- More substantial damage to the stack can make this task much more difficult
- NAB is only useful for current stack segment
  - Use “LEDATA ‘STACK’” to identify additional stack segments
  - Use RUNCHAIN command on first stack frame in the segment (Starts at offset X’18’ into stack segment)
    OR
  - Increase initial stack segment size with STACK run-time option so that NAB is always in current stack segment
Interpreting Raw Data

Manually mapping storage is tedious:

```
+04 = backchain
+10 = Called routine
+4C = NAB

There has to be a better way!
```
Interpreting Raw Data…

IPCS CBFORMAT Command

- Formats a control block

- Syntax: `CBF(ORMAT) <cbaddr> STR(<cbname>)`
  
  - `<cbaddr>` can be address or symbol
  
  - `STRUCTURE` support provided by various components
    
    - IPCS Commands, Appendix D for MVS control blocks
    
    - Language Environment Debugging Guide for LE control blocks
      
      - CEExxx for AMODE 24/31 (CEECAA, CEECIBH, CEEDSA, CEEEDB, CEEHANC, CEESTKH, etc.)
      
      - CELxxxx for AMODE 64 (CELCIBH, CELDSA, CELEDB, CELAA, CELCA, CELSANC, etc.)
Interpreting Raw Data...

CBF 20F016B0 str(ceedsa)

or

CBF DSA006 str(ceedsa)

DSA: 20F016B0

+000000  FLAGS:0010  MEMD:4001  BKC:20F01510  FWC:20F018D0
+00000C  R14:A0ED8730  R15:80072020  R0:A0ED85C0
+000018  R1:20EFE048  R2:00000000  R3:20ED4478
+000024  R4:20ED6288  R5:20EFD100  R6:00000000
+000030  R7:00FD8640  R8:00058A80  R9:000581BC
+00003C  R10:20ED42E8  R11:20ED83C0  R12:20EE5A20
+000048  LWS:00000004  NAB:20F01850  PNAB:20F015B0
+000064  RENT:20ED41C0  CILC:20EFDE08  MODE:00058108
+000078  RMR:20ED8EB8
Interpreting Raw Data...

CBF can be combined with RUNCHAIN command:

```
ip runchain address(20f01030) link(76) name(dsa) exec((cbf x str(ceedsa)))
```

DSA001

```
LIST 20F01030. ASID(X'01CE') LENGTH(X'04') AREA
ASID(X'01CE') ADDRESS(20F01030.) KEY(88)

DSA:  20F01030
+000000  FLAGS:0010  MEMD:4001  BKC:20EE63C0  FWC:20F014A8
+00000C  R14:A0ECE324  R15:20ED00A0  R0:20EFD578
+000018  R1:00000000  R2:20F21050  R3:20ECE2BC
+000024  R4:20ECE038  R5:20EE47B8  R6:00000000
+000030  R7:00000000  R8:00058A80  R9:20EFD448
+00003C  R10:20ECE128  R11:20ECE1E4  R12:20EE5A20
+000048  LWS:00000000  NAB:20F011D0  PNAB:00000000
+000064  RENT:00000000  CILC:00000000  MODE:00000000
+000078  RMR:00000000
```

DSA002

```
LIST 20F011D0. ASID(X'01CE') LENGTH(X'04') AREA
ASID(X'01CE') ADDRESS(20F011D0.) KEY(88)

DSA:  20F011D0
+000000  FLAGS:0010  MEMD:4001  BKC:BADBADBD  FWC:00000000
+00000C  R14:A0ED03C0  R15:20ED2130  R0:20EFDBC0
+000018  R1:00000000  R2:20F210F0  R3:20EDE358
+000024  R4:20EED0D8  R5:20EFD100  R6:00000000
+000030  R7:00ED8640  R8:00058A80  R9:20EFDAA90
+00003C  R10:20EED01C8  R11:20ED0280  R12:20EE5A20
+000048  LWS:00000000  NAB:20F01370  PNAB:00000000
+000064  RENT:20ECE000  CILC:20EFD3F0  MODE:00058108
+000078  RMR:20ED8EB8