z/OS Basics: Q & A on a Dump using IPCS

MVS Core Technologies Project - August 5th 2010

Jerry Ng
IBM Poughkeepsie
jerryng@us.ibm.com
Trademarks

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

- MVS
- OS/390®
- z/Architecture®
- z/OS®
- z9

* 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 or claims relating to 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

- Purpose
- A quick IPCS review
- Q & A’s on a Dump using IPCS
- Index
Purpose

- You are a beginner in using IPCS
- You were given a dump to review
- How do you start to investigate the problem?
- How do you answer common questions on the dump using IPCS?

The purpose of this presentation is not to provide you with a comprehensive procedure for diagnosing problems, but rather to familiarize you with some of the IPCS commands that can be used on a dump.
IPCS - Basic Navigation

- Main menu after invoking IPCS dialog from ISPF
- Customizable like other ISPF dialogs
  - Panels shipped by default in SYS1.SBLSPNL0 pointed to by ISPPLIB

```
---------------------- z/OS 01.11.00 IPCS PRIMARY OPTION MENU
OPTION   ===>
0  DEFAULTS   - Specify default dump and options
1  BROWSE     - Browse dump data set
2  ANALYSIS   - Analyze dump contents
3  UTILITY    - Perform utility functions
4  INVENTORY  - Inventory of problem data
5  SUBMIT     - Submit problem analysis job to batch
6  COMMAND    - Enter subcommand, CLIST or REXX exec
T  TUTORIAL   - Learn how to use the IPCS dialog
X  EXIT       - Terminate using log and list defaults
```

This is the main menu that’s presented when IPCS is invoked in ISPF Dialog Mode. We’ll discuss some of the options during this presentation. Note that the z/OS release level is at the top of the menu. You should use the same level of IPCS as the dump.

You can jump to an OPTION from any IPCS command line by entering ‘=n’, where ‘n’ is the OPTION #. For example, entering ‘=0’ on the IPCS command line will jump to IPCS OPTION 0 - DEFAULTS.
IPCS Commands

There are 2 basic types of IPCS commands:

- **IPCS Subcommands**
  - used to analyze and format data from the dump
  - can be issued from IPCS Primary Option 6, or from the command line (prefixed by IP)

- **IPCS Dialog Primary Commands**
  - used to process the data from the current IPCS output
  - can be issued from the command line (with no IP prefix required)

Instead of navigating through some of the IPCS panels, one can invoke IPCS commands to accomplish the same tasks on a dump.

There are 2 basic types of IPCS commands: subcommands and primary commands. Subcommands work on the dump and primary commands work on the current IPCS output.
IPCS Option 6: COMMAND Panel

- Lists many of the IPCS subcommands available

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

==> 
```

```
-------------------- IPCS Subcommands and Abbreviations --------------------

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

- IPCS subcommands can also be entered from any IPCS Command line with prefix of IP

IPCS subcommands can be entered from IPCS Option 6 (IPCS Subcommand Entry), or they can be issued on any command line with the prefix of IP (short form of IPCS). Details about these IPCS subcommands can be found in the z/OS MVS IPCS Commands manual.
Stacking IPCS Subcommands

To be able to stack IPCS subcommands, SYS1.SBLSTBL0 must be concatenated to ISPTLIB, and ISPF keyword NEWAPPL(BLSG) must be used when invoking IPCS initially.

In this example, the IPCS LIST subcommand is issued while viewing the output of a command. The output of the LIST command will be displayed next. Hitting PF3 will return IPCS to the previous output screen.
Specifying the *data-descr* in IPCS Subcommands

- There are 3 basic attributes of a data-descriptor (data-descr):
  - Virtual Address in Hex (required)
    - address that begins with an alphabet must end with a period
  - Length (optional) – default is 4 bytes
  - ASID (optional) – default depends on type of dump and what is dumped
  - For example,
    - IP LIST 123458  ASID(X’20’) LEN(100)
    - IP WHERE FE1268.

If you look into the IPCS Commands manual, you will see that some of the IPCS subcommands accept ‘data-descr’ as a parameter. Basically this is the virtual address that needed to be supplied to the command. There are many attributes of the ‘data-descr’. The most common ones are the virtual address, length and ASID.

If you don’t specify the length, the default is 4 bytes. If you don’t specify ASID, IPCS will use the one in its default table. To see the default values issue the IPCS subcommand SETDEF (or IP SETD) and look at the bottom section for Local Defaults. The default ASID is in the last line.

You can change the default ASID by using the SETDEF subcommand. For example, IP SETD ASID(x’nn’).
Using a IPCS Primary command

**Issue Primary command on command line**

<table>
<thead>
<tr>
<th>IPCS OUTPUT STREAM</th>
<th>Command ===&gt; FIND 8FDB90</th>
<th>SCROLL ===&gt; CSR</th>
</tr>
</thead>
</table>

**Time of Error Information**

PSW: 070C0000 87A19A94  Instruction length: 04  Interrupt code: 0004  
Failing instruction text: B0509620 F01945E0 C6AC947F

Registers 0-7
GR: 00000004 00005FD0 008FDB90 008E6E90 87A19A8C 87A327F8 87A19580 008E6E90  
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Registers 8-15
GR: 00000000 008FDB90 00000000 008E6E90 87A19580 008E6E90 87A19A8C 00000000  
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

FIND is a great example of an IPCS Primary command. Note that no IP prefix is required when a primary command is issued.
(1) What type of dump is it?

The following unformatted dumps require IPCS for viewing:

- **SVC dump**
  - Recovery initiated dump
  - Console dump
  - SLIP dump
- **Standalone dump**
- **SYSMDUMP**

Knowing the type of dump will give you an idea of why the dump was taken and how you should start your investigation.

The most common type of dump is the SVC Dump. SYSMDUMP is very much like an SVC Dump but with less contents. Standalone dumps represents a system outage and is more complicated to work with. If you hear of a Transaction Dump, it is similar in contents to a SYSMDUMP.

We will focus more on SVC dumps in this session.

A recovery initiated dump is taken due to an error. So one would need to find out what is the error and why it occurred.

Console dumps are taken via the DUMP command on the operator’s console. It is usually due to a problem with a job or address space. One would need to analyze the units of work in the address space. This will require reviewing the TCBs and RBs in the address space. We will touch on this near the end of the session.

SLIP dumps are taken for a purpose. One would need to find out the SLIP trap that caused the dump to be taken, then get the PSW/registers information and investigate accordingly.
IP ST SYSTEM

- Can be used to verify type of dump
- Program producing dump can be
  - SVCDUMP, SLIPDUMP, SYSMDUMP, SADUMP

SYSTEM STATUS:
- Nucleus member name: IEANUC01
- I/O configuration data:
  - IODF data set name: MVSRIODF00
  - IODF configuration ID: CG21
  - EDT ID: 00
- Sysplex name: TEST
- TIME OF DAY CLOCK: C1CF7E24 10F88549 01/17/2008 09:24:58.942344 local
- TIME OF DAY CLOCK: C1CF70BA D6B88549 01/17/2008 08:24:58.942344 GMT
- Program Producing Dump: SVCDUMP
- Program Requesting Dump: IEAVTSHT

Incident token: TEST P1N4 01/17/2008 08:24:43.288934 GMT

The time of the dump is the time that global data capture completed. This is an example of an SVC dump.

Note that SVCDUMP in the ST SYSTEM output can mean a recovery initiated SVC dump or a Console dump. To distinguish these types of dumps you need to look at the dump title (see next page).
IP LIST TITLE  \((L\ TITLE)\)

- **Use this command to determine the type of SVC dump**
  - Recovery initiated dumps typically have a COMPID= and other system related information
  - Console dumps have a title of whatever the user puts in COMM= as the dump title
  - Dumps taken as a result of a SLIP trap have ‘SLIP DUMP ID=xxx’ in title

The dump title can provide a clue as to why the dump was taken. See the examples on next page.
Examples of SVC Dump Title

- **Recovery initiated dump**

  ```
  TITLE
  LIST 00. LITERAL LENGTH(X'58') CHARACTER
  00000000 | COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPXPRSRB+????,ABE |
  00000040 | ND=S00C6,REASON=00000006 |
  ```

- **Console dump**

  ```
  TITLE
  LIST 00. LITERAL LENGTH(X'19') CHARACTER
  00000000 | JOB PAYROLL IS HUNG |
  ```

- **SLIP dump**

  ```
  TITLE
  LIST 00. LITERAL LENGTH(X'17') CHARACTER
  00000000 | SLIP DUMP ID=0001 |
  ```

Recovery initiated dumps have dump titles that are pre-coded in the recovery routines. They usually contain technical information about the component or product that experienced the error.

Console dump titles are supplied by the user via the DUMP command on the console. They are usually less technical and more human-like.

SLIP dump titles are system generated and contain the words ‘SLIP DUMP ID=‘ in it.
IP LIST SLIPTRAP  \( (L \text{ SLIPTRAP}) \)

- If the SVC dump is a SLIP dump, use this command to find out the SLIP trap that matched (and caused this dump to be taken)
- Example:

```
SLIPTRAP
LIST 00. LITERAL LENGTH(X'22') CHARACTER
00000000 | SLIP SET,C=9C6,ID=$WK5,A=SVCD,ML=1   |
```

In this example, the dump was taken as a result of a SLIP on an abend9C6. The ID of the SLIP is $WK5.
(2) When/where was this dump taken?

- Date/time of the dump
- System name
- Original dump dataset name (SVC dump only)
- z/OS release of the system

It is important to find out when and where the dump was taken. You may need to correlate this dump with other events occurring on the same system at around the same time.
This is an example of a SVC dump taken on system SP5 on 9/28/09. The time is actually the time of the end of the global data capture phase of SVC dump.
IP CBF CVT

- Can be used to verify system release level on which dump was taken

**Example:**

```
CVT: 00FDEAC0
-0028 PRODN.... SP7.1.1 PRODI.... HBB7760 VERID....
```

- In the above example, dump was taken on a z/OS R11 system

This is important because you should use the same level of IPCS as the z/OS release of the system on which the dump was taken.
(3) What was dumped in this SVC dump?

- ASIDs that were dumped
- Names of these Address Spaces
- SDATA options included in dump
- Other storage that was dumped

An SVC dump usually contain one or more address spaces. You will need to know the ASID numbers so that you can specify the right ones when using IPCS subcommands.

The SDATA options describe what area of storage is included in the dump.
### IP CBF RTCT

- **Display what ASIDs are dumped in SVC dump**
- **Issue ‘FIND  ASTB’ in output**

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>ASID</th>
<th>SVC</th>
<th>SYD</th>
<th>SDLA</th>
<th>MECB</th>
<th>ASID dumped</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0000</td>
<td>NAME</td>
<td>RTCT</td>
<td>SAP</td>
<td>SUP</td>
<td>SYD</td>
<td>SDLA</td>
<td>00105</td>
</tr>
<tr>
<td>+0018</td>
<td>PNAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00080</td>
</tr>
<tr>
<td>+002C</td>
<td>DEV</td>
<td>0004</td>
<td>STH</td>
<td></td>
<td></td>
<td></td>
<td>00020</td>
</tr>
<tr>
<td>+0044</td>
<td>ADG4</td>
<td>0037</td>
<td>STMT</td>
<td></td>
<td></td>
<td></td>
<td>00030</td>
</tr>
<tr>
<td>+005C</td>
<td>DIMD</td>
<td>0203</td>
<td>DRS</td>
<td></td>
<td></td>
<td></td>
<td>00040</td>
</tr>
<tr>
<td>+0074</td>
<td>BPXR</td>
<td>0166</td>
<td>BFXP</td>
<td></td>
<td></td>
<td></td>
<td>00050</td>
</tr>
<tr>
<td>+00AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00060</td>
</tr>
<tr>
<td>+00CC</td>
<td>SDWK</td>
<td>00D6</td>
<td>FFXP</td>
<td></td>
<td></td>
<td></td>
<td>00070</td>
</tr>
<tr>
<td>+000000</td>
<td>SDXX</td>
<td>0006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00080</td>
</tr>
<tr>
<td>+000E</td>
<td>KASD</td>
<td>0105</td>
<td>ETIM</td>
<td></td>
<td></td>
<td></td>
<td>00090</td>
</tr>
<tr>
<td>+0010</td>
<td>SDXX</td>
<td>0105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000A0</td>
</tr>
</tbody>
</table>

#### ASIDs dumped

Knowing which address spaces are dumped is useful for determining what address space storage you can expect to find in the dump. It may give you a clue about what address spaces are involved in the problem.
IP SELECT ALL

ASID/JOBNAME translation

<table>
<thead>
<tr>
<th>ASID</th>
<th>JOBNAME</th>
<th>ASCBADDR</th>
<th>SELECTION CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td><em>MASTER</em></td>
<td>00FD3900</td>
<td>ALL</td>
</tr>
<tr>
<td>0002</td>
<td>PCAUTH</td>
<td>00F4DE80</td>
<td>ALL</td>
</tr>
<tr>
<td>0003</td>
<td>RASP</td>
<td>00F4DD00</td>
<td>ALL</td>
</tr>
<tr>
<td>0004</td>
<td>TRACE</td>
<td>00F4DB80</td>
<td>ALL</td>
</tr>
<tr>
<td>0005</td>
<td>DUMPSRV</td>
<td>00F50980</td>
<td>ALL</td>
</tr>
<tr>
<td>0006</td>
<td>XCFAS</td>
<td>00F50800</td>
<td>ALL</td>
</tr>
<tr>
<td>0007</td>
<td>GRS</td>
<td>00F50680</td>
<td>ALL</td>
</tr>
<tr>
<td>0008</td>
<td>SMSPDSE</td>
<td>00F4EF80</td>
<td>ALL</td>
</tr>
<tr>
<td>0009</td>
<td>CONSOLE</td>
<td>00F4EE00</td>
<td>ALL</td>
</tr>
<tr>
<td>000A</td>
<td>WLM</td>
<td>00F4EC80</td>
<td>ALL</td>
</tr>
<tr>
<td>000B</td>
<td>ANTMAIN</td>
<td>00FC6400</td>
<td>ALL</td>
</tr>
<tr>
<td>000C</td>
<td>ANTAS000</td>
<td>00FC6280</td>
<td>ALL</td>
</tr>
<tr>
<td>000D</td>
<td>DEVMAN</td>
<td>00FC6100</td>
<td>ALL</td>
</tr>
<tr>
<td>000E</td>
<td>OMVS</td>
<td>00FA2800</td>
<td>ALL</td>
</tr>
</tbody>
</table>

Or IP SELECT ASID(x'nn') / IP SELECT JOB(jobname)

This report associates an ASID with a JOBNAME. From the CBF RTCT example in last page, we can see that asid(x'E') corresponds to a jobname of OMVS.
Determine what SDATA options are included in dump

SDUMP_PL: 02B7DF48

==> FLAGS SET IN SDUFLAGS:
  HDR/HDRADR specified.
  ECB specified.
  Schedule dump request ASID specified.

==> FLAGS SET IN SDUFLAG1:
  SVC dump request.
  48+ byte parameter list.

==> FLAGS SET IN SDUSDATA:
  Dump all PSAs.
  Dump the nucleus.
  Dump SQA.
  Dump LSQA.
  Dump rgn-private area.
  Dump LPA mod. for rgn.
  Dump trace data.
  Dump CSA.
  Dump SWA.
  Dump summary dump data.
  Dump all nucleus.

SDATA=(ALLPSA,NUC,SQA,LSQA,
RGN,LPA,TRT,CSA,SWA,SUM,ALLNUC)

Knowing the dump options that were requested can help you determine why the storage or report that you're browsing is not available. This is because the option that would have included the relevant storage was not requested. For example, if you're trying to browse private storage of an address space, and the storage is not available, it may be due to RGN not being specified in SDATA.
IPCS Option 4: Inventory

- **IPCS Inventory**: IPCS Option 4 (=4 on any IPCS command line) brings up the following panel:

```
IPCS INVENTORY - SMTUSR1.DUMPZ11.DEBUG --------------------------------------
Command ===>                                                  SCROLL ===> CSR

AC Dump Source                                                 Status
__ DSNAME('SMTUSR1.TEST1.DUMP')  . . . . . . . . . . . . . . . . . . OPEN
   Title=COMPON=CMND-ESTAE,COMPID=SC1B8,ISSUER=IEECB860,FAILURE IN COMMAND K
   Psym=RIDS/IEE8203D#L RIDS/IEE8203D PIDS/5752SC1B8 AB/S00C4 RIDS/IEECB860#
__ DSNAME('SMTUSR1.TEST2.DUMP')  . . . . . . . . . . . . . . . . . . CLOSED
   Title="TO BE OR NOT TO BE"
```

Useful line commands in the AC field:
- **DD**  Delete the source and optionally, the source dataset
- **SD**  Establish the source as the IPCS local and global default
- **LZ**  List dump description with storage summary

The IPCS inventory panel allows the user to view all the dumps in his IPCS dump directory. The IPCS inventory panel shows all the dumps the user has initialized under IPCS and is currently working with.

One can enter a line command in the column AC next to a dump, and then hit <enter>.

SD selects a dump to be used as the current source for IPCS.

DD deletes information of a dump from the IPCS directory. This is needed when the user does not need to review the dump anymore. The dump will disappear from the IPCS inventory.

LZ gives information about the address spaces, dataspaces and storage dumped in the dump.
(4) How do I find the PSW/registers information in the dump?

- **SVC dump**
  - Recovery initiated dump
    - PSW/registers at time of error
  - Console dump
    - PSW/registers not applicable
  - SLIP dump
    - PSW/registers when SLIP matched

- **Standalone dump**
  - PSW/registers of each CPU when dump was taken

There is always a set of PSW/registers information crucial for debugging a recovery initiated dump or SLIP dump.

For a console dump, one needs to investigate the state of an address space, and there is no PSW/registers of interest initially.

For a standalone dump, one needs to investigate the state of the whole system. The PSW/registers of each CPU is a good place to start, but there are many other pieces of information to be reviewed. Details of how to investigate a standalone dump will not be covered in this session.
IP ST FAILDATA and ST REGS

- **Recovery initiated dump**
  - ST FAILDATA formats the SDWA in the dump header
    - PSW/registers at time of error and other helpful information
  - ST REGS also provides PSW and registers but in a different format

- **Console dump**
  - No SDWA so ST FAILDATA does not provide any information
  - ST REGS
    - PSW/registers at time console dump was requested.
      - Not normally very useful

- **SLIP dump**
  - No SDWA in dump header so do not use ST FAILDATA
  - ST REGS
    - PSW/registers at time SLIP matched

---

An SVC dump may be captured as the result of a program recognizing that an error has occurred and invoking recovery, as a result of a SLIP trap, or because the operator/system programmer requests a dump. The most basic and crucial piece of debugging information is typically the PSW and register information at the time of error or failure.

The availability of this information will depend on how the dump was taken. ST FAILDATA typically gives the most information about a problem, but is only available if a populated SDWA (System Diagnostic Work Area) is available. This will only be the case if a recovery routine requests that the dump be captured. In a console dump, the registers found in ST REGS show what was happening at the time the dump was requested, which means the program running at the time is not likely the one of interest, and the PSW and registers are less useful. In a SLIP generated dump, there is no SDWA, but the PSW/registers may be of value since they are captured at the time the SLIP trap springs.
The first page of the ST FAILDATA output contains information about the program involved with the error, as well as the ABEND code and reason code.
Find out the PIC (Program Interrupt Code) from the Interrupt code. In this example, it is a PIC4 (Protection Exception).

For PIC 10,11,38,39,3A,3B the PSW at time of error points at the failing instruction.

For PIC 4, the PSW points after the failing instruction.

When reviewing the Failing instruction text, start in the middle (6 bytes into text), and backup the number of bytes (if necessary) specified by the Instruction length. For a PIC 4, we back up by the Instruction length (4 bytes in this case).
IP ST REGS

CPU STATUS:

PSW=071C3000 81DA5AAA (RUNNING IN PRIMARY, KEY 1, AMODE 31, DAT ON) DISABLED FOR PER
ASID(X'0017') 01DA5AAA. IEFW21SD+0AAA IN EXTENDED PLPA
ASCB23 at F85380, JOB(JERRY), for the home ASID
ASXB23 at 8FDF00 for the home ASID. No block is dispatched
HOME ASID: 0017 PRIMARY ASID: 0017 SECONDARY ASID: 0017

GPR VALUES
0-3 008DFFD0 008E2EEC 008FA934 01DA6021
4-7 008E3003 008DFDD8 008DFDB8 008E02C8
8-11 00000000 01DA621C 00000000 008E2EA0
12-15 81DA5022 008E2EA0 81DA5A46 81DA62E0

ACCESS REGISTER VALUES
0-3 008FB01F 00000001 00000000 00000000
4-7 00000000 00000000 00000000 00000000
8-11 00000000 01DA621C 00000000 008E2EA0
12-15 81DA5022 008E2EA0 81DA5A46 81DA62E0

This report is not dependent on an SDWA (as ST FAILDATA), and is useful for both SLIP dumps and dumps generated by a recovery routine. It provides more information on the PSW, but the Failing Instruction Text, Instruction Length, and Interrupt Code are not provided (as with ST FAILDATA).

It also provides contents of Access registers and Control registers (not shown above).
IP ST CPU REGS

Display registers for each CPU in a SADUMP

CPU(X’01’) STATUS:
PSW=00020000 80000000 00000000 00004084
(Running in PRIMARY, key 0, AMODE 31, DAT OFF)
Disabled for PER I/O MCH
ASCB25 at FA2380, JOB(KILLER), for the home ASID
ASXB25 at 8FDD00 and TCB25D at 8FF2A0 for the home ASID
HOME ASID: 0019 PRIMARY ASID: 0019 SECONDARY ASID: 0019
CLTE: 0245AF80
+0000 BLSD...... 00000000 XDS...... 00000000 XRES..... 00000000 X
+0018 IXSH...... 00FCF7E0 IXDS...... 00000000 IXLL..... 00000000 U
+0030 WLMQ...... 00FCF7F0 REGS..... 00000000 CNTX..... 00FCF800 S
HOLDING LOCK(S): CPU
CURRENT FRR STACK IS: PROGRAM
Unable to complete FRR stack analysis, unknown stack pointer

General purpose register values
0-1 00000000_00000000 0000001C_00000000
2-3 20202020_00F51450 0C0C0C0C_00000003
4-5 073D9136_023F7C68 073D916A_012C6738
6-7 073D91D4_02450900 073D9288_00FD5100
8-9 04280000_022E0548 24082C30_00000000
10-11 00000000_00FD57A0 00000000_01FAC3EC
12-13 04280000_811EB3F0 00000000_023F8488
14-15 00000000_40000000 00000000_00000000

This command is particularly useful in a standalone dump. In this example, the PSW for CPU1 is indicative of a WAIT084 RC04 (FRR stack corruption). The registers at the time of the Wait State follow for CPU1.
IP ST FAILDATA – ABEND0C1 Example

Address of last instruction causing a break in sequential execution prior to program check

| PSW: 07850000 80000000 00000000 00007FF8 |
| Instruction length: 02  Interrupt code: 0001 |
| Breaking instruction text: 00000000 00000000 00000000 |
| Breaking event address: 00000000 00007F20 |

Registers 0-7
GR: 00000000 00007EF8 00000040 007D5D84 007D5D60 007FF448 007C7FE0 FD000000
AR: 00000000 00000000 00000000 00000000

Registers 8-15
GR: 00007F22 007FF708 00000000 007FF448 965AB8B2 00006F60 80007F22 000007FF6
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Possibly a BALR using R14 and R15 was issued

The Interrupt code of x’0001’ indicates a PIC 1 (Operation Exception). The failing instruction was 2 bytes in length. Note from the Failing instruction text that we tried to execute zeros, hence the PIC 1. There was likely a BALR R14,R15 from address x’7F20’ to address x’7FF6’. It was at x’7FF6’ that we then tried to execute the zeros. The PSW instruction address was incremented by the machine by 2 bytes and so it now contains x’7FF8’.

BEAR = Breaking Event Address Register, that is, the address of the last instruction causing a break in sequential execution saved in a 64-bit hardware register (1 register per CP). Examples of such instructions are BALR, BASR and LPSW.

BEAR contents are provided in the output of ST FAILDATA, VERBX LOGDATA, or in the RTM2WA (see later).
(5) How do I display storage in a dump?

■ Quick Method:
  - IP L aaaaaaa LEN(X’nn’) ASID(X‘nn’)
    ▶ L is the short form of the LIST subcommand
    ▶ LEN is length and its short form is also L
    ▶ ASID should be used if it is private storage (else IPCS will use a default ASID)

■ Scrollable Method:
  - ‘=1’ on any command line in IPCS takes you to the BROWSE menu (IPCS Option 1) that can be used to browse storage

Described are two ways of displaying storage, each with their own benefits.

The IPCS LIST subcommand is useful if you’re examining an IPCS report and you just want to know where an address points. After seeing the output from LIST you can then ‘PF3’ back to the report. On the other hand, if you want to look at a PSW address and scan backwards in storage to look for module eyecatchers, then IPCS Option 1 will allow you to easily scroll through storage.
In our example, the address we’re interested in is x’07208CE0’ in ASID(x’65’).

In the first example, LIST can be used to display x’60’ bytes of the data.

In the second example, adding a data attribute of I (instruction) changes the output to display the data as if they are instructions.
IPCS Option 1: BROWSE Panel

- **Browsing Storage:** IPCS Option 1 (=1 on any IPCS command line) brings up the following panel:

```
------------------------ IPCS - ENTRY PANEL  ---------------------------------
CURRENT DEFAULTS:
  Source ==> DSNAME('SHARE.S2822.DUMP1A')
  Address space ==> ASID(X'0065')

OVERRIDE DEFAULTS:                          (defaults used for blank fields)
  Source ==> DSNAME('SHARE.S2822.DUMP1A')
  Address space ==> (blank to display pointer stack)
  Password      ==> (optional text)

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

- Enter ASID and storage address here and then hit <ENTER>, or just hit <ENTER> to get to the Pointer Stack on next page.

An “Address” and “Address space” can be filled in on this screen to jump right to the scrollable storage at the specified address. However, hitting <ENTER> on this screen without filling in an “Address” under the “POINTER” will bring up a POINTER STACK containing a list of pointers that have been defined for this dump.
### IPCS Option 1: Browse Panel

**Pointer Stack**

<table>
<thead>
<tr>
<th>DSNAME ('SHARE.S2822.DUMP1A') POINTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASID(X'0065') is the default address space</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PTR</th>
<th>Address</th>
<th>Address space</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>00000200</td>
<td>ASID(X'0065')</td>
<td>AREA</td>
</tr>
</tbody>
</table>

Remarks:

*************** END OF POINTER STACK ***************

Enter address and ASID

Enter line commands in the PTR field:
- **D** Delete the pointer from the stack
- **E** Edit the pointer, the address, address space, data type, and remark
- **F** Format and display the structure described by the pointer
- **I** Insert a pointer after this pointer
- **R** Repeat this pointer immediately following this pointer
- **S** Select the storage addressed by the pointer for display

In our example, the address we’re interested in browsing is X’200’. Thus, we can enter 00000200 in the “Address” field for pointer (PTR) 0001. The “Address space” value of ASID(X’0065’) is sufficient since this is PSA storage. An ‘s’ can then be placed next to the PTR 0001 that we just defined for address X’200’, followed by hitting <ENTER>. We are then taken to scrollable storage at address X’200’, which does indeed contain the ‘PSA’ eyecatcher (see next page).
IPCS Option 1: Browse Panel

Storage Display

To display a different storage location while in Browse:

- Use the LOCATE Primary command (short form is also L), or
- Put an indirect storage pointer on the left of a storage address and then hit enter

% = 24-bit   ? = 31-bit   ! = 64-bit

While viewing the storage in BROWSE, a LOCATE primary command can be used to change the display to a different storage location.

The command “L E00F04. asid(x'65')” in the above example refers to the IPCS Primary command ‘LOCATE’, not the IPCS LIST Subcommand. The ‘.’ after the address x’E00F04’ tells IPCS that the LOCATE is being done on an address, rather than an IPCS symbol with the same name. This is particularly important when using an address that begins with a letter (i.e., A,B,C,D,E,F).

One can also put an indirect storage pointer on the left of any address on the BROWSE display and then hit <ENTER>. BROWSE will then display the storage at that address.
(6) How do I find out more information about a 31-bit storage address?

- Is the storage address in the dump?
- Is this address in Private or Common?
- Is there a module at this address?

When you are debugging a problem with a set of PSW and registers, it is important that you know where the PSW and registers point to. This section will answer some of the questions you may have on an address.

If the address is in common storage, an ASID parameter is not needed when displaying the storage.
Use IP LIST or BROWSE to display storage address

- If storage is not in dump, you will receive ‘storage not available’ message
  - IP L 72345678 ASID(X’B’)

```
LIST 72345678. ASID(X'000B') LENGTH(X'04') AREA 72345678. LENGTH(X'04')==>Storage not available
```

- In BROWSE

```
ASID(X'000B') ADDRESS(07228000.) STORAGE -------------------------------
Command ===> SCROLL ===> C
07228000.:075AEFFF.--Storage not available
075AF000 A7F40014 00000000 C9C5C5D4 C2F8F0F4 | x4......IEEMB804 |
```

‘Storage not available’ can be due to:
- storage is not dumped, or
- storage is paged out, or
- storage address is invalid

There are several reasons for ‘storage not available’, and it also depends on the kind of dump.

Since everything should be dumped in a standalone dump, ‘storage not available’ most likely means that the storage address is invalid.

In a SVC dump or SYSMDUMP, some common area (for example, LPA) are usually not dumped. So it is important to know which area the storage address belongs to (see next page).
IP VERBX VSMDATA ‘NOASIDS SUMM’

- Scroll to the bottom and then page back to find the Global Storage Map

<table>
<thead>
<tr>
<th>GLOBAL STORAGE MAP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>80000000 &lt;- Top Of Storage</td>
</tr>
<tr>
<td>:</td>
<td>: Extended Private Storage</td>
</tr>
<tr>
<td>:</td>
<td>: 2A8000000 &lt;- Ext. CSA Upper Bound</td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>: B3850000 &lt;- Ext. CSA Lower Bound</td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>: 65EB0000 &lt;- Ext. SQA Upper Bound</td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
</tbody>
</table>

Only the top part of the storage map is shown here

VERBX VSMDATA with a parameter of NOASIDS provides a global storage map that shows the boundaries of different area of storage for every address space. This will allow you to figure out whether the address at hand is in private or common. To see the details of the private storage of an address space see next page.
To see the details of the layout of private storage in a particular address space, use VERBX VSMDATA with an ASID parameter. This is useful after you have determined that the address at hand is in the private region of an address space.

**IP VERBX VSMDATA ‘ASID(nn) SUMM’ (nn in decimal)**

- Scroll to the bottom and then page back to find the Local Storage Map

```
LOCAL STORAGE MAP

| Extended                                                                 | 80000000 <- Top of Ext. Private |
| LSQA/SWA/229/230                                                        | 80000000 <- Max Ext. User Region Address |
|                                                                         | 7EB73000 <- ELSQA Bottom |
| (Free Extended Storage)                                                 | 2A9C9000 <- Ext. User Region Top |
| Extended User Region                                                    | 2A800000 <- Ext. User Region Start |
```

Only the top part of the storage map is shown here
IP WHERE (W)

- Used to identify an area in the dump (if possible)
  - IP W 75AF0D0

```
Command ===>  
****************************** TOP OF DATA *********************  
  ASID(X'000B') 075AF0D0. IGC0003F+D0 IN EXTENDED PLPA  
****************************** END OF DATA *********************  
```

- ASID(x’nn’) should be used if storage address is in private

- If WHERE cannot identify the area, BROWSE the storage and scan backwards for module or control block identifiers

For an address that may point to a module or a common control block, the IPCS WHERE subcommand can be used. Note that the ASID parameter should be used if the address is in private.
(7) How do I find the recent errors or activity leading up to this dump?

- Review the recent system messages
- Review the recent LOGRECs
- Review the System Trace
- Analyze the TCB/RBs structure
  - Applicable only if the problem is related to a TCB(s) in an address space

It is always a good practice to find out ‘what led up to the problem’. There are a few area in the dump that can help.
IP VERBX MTRACE

- Provides a snapshot of what is taking place in the system log just before the dump
- Useful to see if a job was started, a message was issued, or a command was issued just prior to the problem
- May see messages on delayed issue queues that are not shown in SYSLOG (e.g., waitstate messages)

- Refer to z/OS MVS Diagnosis: Tools and Service Aids manual, Chapter 9, for more details on Master Trace

VERBX MTRACE displays the last messages that were issued to SYSLOG leading up to the dump, and may also give an indication of what jobs started just prior to the problem.
IP VERBX LOGDATA

- Provides history of ABENDs leading up to this dump
  - Most recent ABEND at the bottom of the output
- Useful elements:
  - ERRORID: contains sequence number, ASID and time of error
  - TIME OF ERROR INFORMATION: provides PSW and REGs
  - RECOVERY ROUTINE ACTION: indicates if dump was requested
- F 'SOFTWARE EDIT'
  - Scroll through SYMPTOM RECORDs and SOFTWARE RECORDs

VERBX LOGDATA is useful for reviewing the most recent ABENDs that occurred prior to the dump. The SOFTWARE RECORDs and SYMPTOM RECORDs are often of most value.
IP VERBXL LOGDATA: Example

SEQ (sequence number): if sequence numbers are the same for multiple ABEND records, it indicates that this is the same ABEND being recorded by different recovery routines

ASID: ASID that encountered the error

TIME: Time the error occurred

This example is of an abend522 that occurred in TSO csect IKJEFTSC in asid(x‘D4’) PALNSMY at 12:38:51 on Julian day 203 in the year 2002.
The failing instruction was an SVC 1 (x’0A01’) Wait that was issued by PSW address x’7846’. Note that the PSW address x’7848’ points after the failing instruction, so we had to back up by the INSTRUCTION LENGTH (02) to find the failing PSW address x’7846’ and failing instruction (x’0A01’) in the FAILING INSTRUCTION TEXT.

The registers displayed are the registers at the time of the failure (abend522). An SVC dump was not requested by recovery for this abend522.
IP SYSTRACE

- System Trace activity at time of dump

- Examples:
  - IP SYSTRACE ASID(x'nn') TIME(LOCAL)
    - Formats only trace records associated with the requested ASID
  - IP SYSTRACE ALL TIME(LOCAL)
    - Formats trace records for all active address spaces in the system

- Refer to z/OS MVS Diagnosis: Tools and Service Aids manual, Chapter 8, for more details on System Trace

The TIME(LOCAL) parameter converts the time in SYSTRACE to local time. The default is raw timestamps.
IP SYSTRACE: Example

<table>
<thead>
<tr>
<th>PR</th>
<th>ASID</th>
<th>WU-ADDR</th>
<th>IDENT</th>
<th>CD/D</th>
<th>PSW-</th>
<th>ADDRESS-</th>
<th>UNIQUE-1</th>
<th>UNIQUE-2</th>
<th>UNIQUE-4</th>
<th>UNIQUE-5</th>
<th>UNI</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>SVCR</td>
<td>38</td>
<td>075C3000</td>
<td>81FF774C</td>
<td>00000000</td>
<td>00000000</td>
<td>800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>SVC</td>
<td>63</td>
<td>070C1000</td>
<td>81FF850A</td>
<td>00000008</td>
<td>00000000</td>
<td>008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>PGM</td>
<td>011</td>
<td>071C3000</td>
<td>81DA506E</td>
<td>00040011</td>
<td>008E2EA4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>SVC</td>
<td>78</td>
<td>071C2000</td>
<td>81DA5346</td>
<td>0000E512</td>
<td>0000027A</td>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>SVCR</td>
<td>78</td>
<td>071C2000</td>
<td>81DA5346</td>
<td>00000000</td>
<td>00000280</td>
<td>008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>SVC</td>
<td>77</td>
<td>071C2000</td>
<td>81DA5588</td>
<td>00000000</td>
<td>FFF00000</td>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>SVCR</td>
<td>77</td>
<td>071C2000</td>
<td>81DA5588</td>
<td>00000000</td>
<td>FFF00000</td>
<td>800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>PGM</td>
<td>004</td>
<td>071C3000</td>
<td>81DA5AAA</td>
<td>00040004</td>
<td>008E2EA4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0017</td>
<td>008FABD0</td>
<td>*RCVY PROG</td>
<td></td>
<td></td>
<td></td>
<td>940C4000</td>
<td>00000004</td>
<td>000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An error occurred due to a program check

A PGM entry means a program interrupt occurred and a RCVY entry means RTM (Recovery Termination Manager) was entered to process the error. It is always a good practice to look for RCVY entries in the trace table.
The SUMMARY subcommand can be used to format the control blocks of an address space. The TCBs in the address space should be investigated for any recent errors. At the bottom of the SUMM FORMAT output is the TCB summary. Note any TCBs with non-zero completion code under the CMP field.
IP SUMM FORMAT ASID(x’nn’). . . .

Find the TCB(s) with non-zero completion code

- note: completion code can be residual

```
TCB: 008FABD0
   +0000  RBP...... 008FD7A8  PIE...... 00000000  DEB...... 00000000
   +000C  TIO...... 008E7000  CMP...... 940C4000  TRN...... 40000000
   +0018  MSS...... 7FF156F0  PKF...... 10  FLGS...... 01000000
   +0022  LMP...... FF  DSP...... FF  LLS...... 00000000
   +0028  JLB...... 00000000  JPQ...... 00000000

GENERAL PURPOSE REGISTER VALUES
  0-3  00F8538D  00C7BEB0  00C7BEB0  01B63688
  4-7  02FE9CE8  00000018  01E44085  01B63568
  8-11 00C7BCE0  81E43086  00000000  00000000
 12-15 00C7BE88  01B63B38  81E431B6  00000000
```

Once you found a TCB with non-zero completion code, you can issue a FIND of ‘TCB: 00xxxxxx’ from the top of the output to find the TCB. The error under this TCB may be the one causing the dump to be taken. Or it can be a residual completion code.
IP SUMM FORMAT ASID(x’nn’). . .

- There can be RTM2WA’s under the TCB
  - if the error was very recent (or the dump was produced due to it)

<table>
<thead>
<tr>
<th>RTM2WA SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>+001C Completion code</td>
</tr>
<tr>
<td>+008C Abending program name/SVRB address</td>
</tr>
<tr>
<td>+0094 Abending program addr</td>
</tr>
</tbody>
</table>

  GPRs at time of error
  0-3 008DFFD0 008E2EEC 008FA934 01DA6021
  4-7 008E3003 008DFDD8 008DFDB8 008E02C8
  8-11 00000000 01DA621C 00000000 008E2EA0
  12-15 81DA5022 008E2EA0 81DA5A46 81DA62E0
| +007C EC PSW at time of error | 071C3000 81DA5AAA 00040000 008E2EA4 |

Instruction length code and interrupt code

If the TCB is going through recovery processing for the error in the completion code, you will find a RTM2WA (RTM2 Work Area) after the TCB. The above shows the RTM2WA Summary which contains the PSW at time of error and the registers, as well as the instruction length code, interrupt code, and the translation exception address (not applicable in this case since the interrupt code represents a protection exception).

The Translation Exception Address is the address that caused a PIC 10,11,38,39,3A or 3B.
There are RBs under each TCB

- RBs are used to save status after an interrupt (usually an SVC)

The RBs are used to save status (PSW and Registers) after an interrupt, and can be used to show the recent activity of the TCB.

In the SUMM FORMAT output, RBs are listed in chronological order: oldest RB at the top, most recent RB at the bottom. Note that each TCB has RBs under it, so make sure that you are looking at the right ones. Check RBLINK field (offset x’1C’) of the first RB, it contains the TCB address. RBLINK in subsequent RBs points backwards to the previous RB.

In this example, the first RB indicates that an SVC x’1A’ (LOCATE SVC) was issued at 1F0E8BA. The WLIC field contains the instruction length code and the interrupt code, and an SVC instruction has a length of 2 bytes. The registers at the time of the SVC x’1A’ were saved in the next RB (an SVRB). Then an SVC x’63’ (DYNAMIC ALLOCATION SVC) was issued from 1FF850A. The registers at the time of the SVC x’63’ were saved in the next SVRB (see next page).
Then under the processing of the SVC x’63’, a program check occurred (protection exception) at 1DA5AAA. The WLIC field has 0004004. This does not represent an SVC interrupt because the instruction length is 4. It was a program interrupt. The registers at the time of error were saved in the next SVRB. The second SVRB in the picture represents RTM processing after the PIC 4.

Note that eventually you will reach the bottom RB (the most recent RB). This most recent RB is called the Top RB and is pointed to by TCBRBP (sorry, it is confusing, the Top RB is at the bottom). The registers of the Top RB are saved in the TCB.
Reference Information

- Manuals
  - z/OS MVS IPCS Commands
  - z/OS MVS IPCS Customization
  - z/OS MVS IPCS User’s Guide
  - z/OS MVS Diagnosis: Reference
  - z/OS MVS Diagnosis: Tools and Service Aids
  - z/OS MVS System Codes
## Index

<table>
<thead>
<tr>
<th>Question</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) What type of dump is it?</td>
<td>11</td>
</tr>
<tr>
<td>(2) When/where was this dump taken?</td>
<td>16</td>
</tr>
<tr>
<td>(3) What was dumped in this SVC Dump?</td>
<td>19</td>
</tr>
<tr>
<td>(4) How do I find the PSW/registers information?</td>
<td>24</td>
</tr>
<tr>
<td>(5) How do I display storage?</td>
<td>31</td>
</tr>
<tr>
<td>(6) How do I find out more about a 31-bit address?</td>
<td>36</td>
</tr>
<tr>
<td>(7) How do I find the recent errors or activity?</td>
<td>41</td>
</tr>
</tbody>
</table>