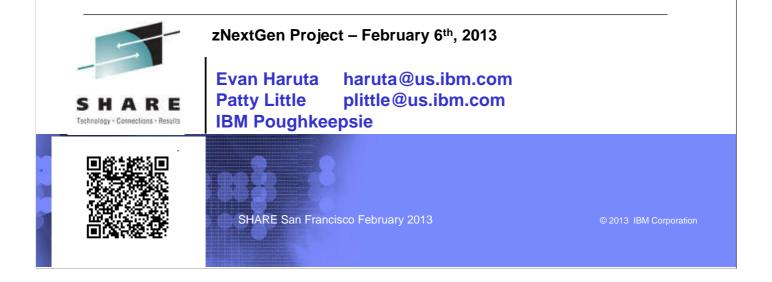


Session 12842 z/OS Debugging: Old Tricks for New Dogs



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

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.

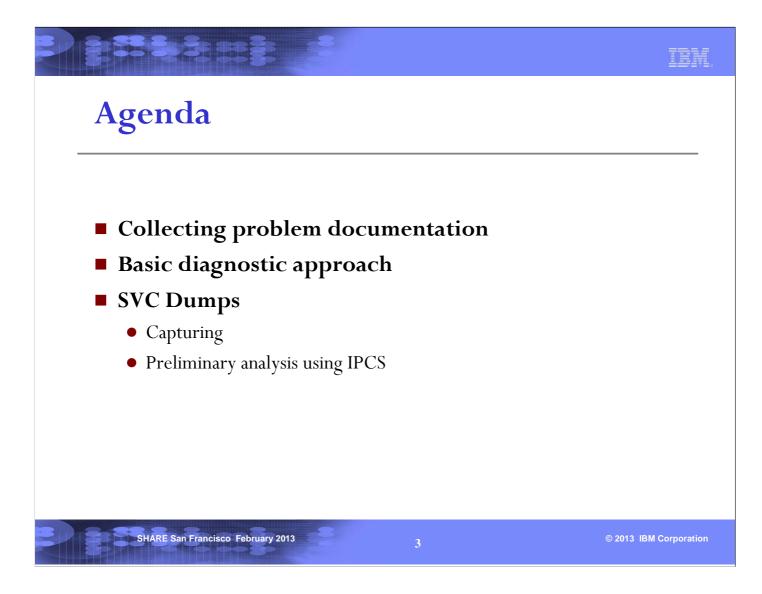
Notes

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 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 claracteristics will vary depending on individual ustomer configurations and conditions. This publication was produed in the United Status. IBM may not filter 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 a 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

And assessments tragenting forms funded includes also rates are adopted or transformed or matchares or the probability or any other claims related to non-Information adopted or non-BM products is obtained from the manufactures of these 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 these products.

SHARE San Francisco February 2013



This presentation will discuss:

-the various kinds of problem documentation available in z/OS debugging

-basic diagnostic approaches

-the preliminary steps of diagnosing an SVC dump.

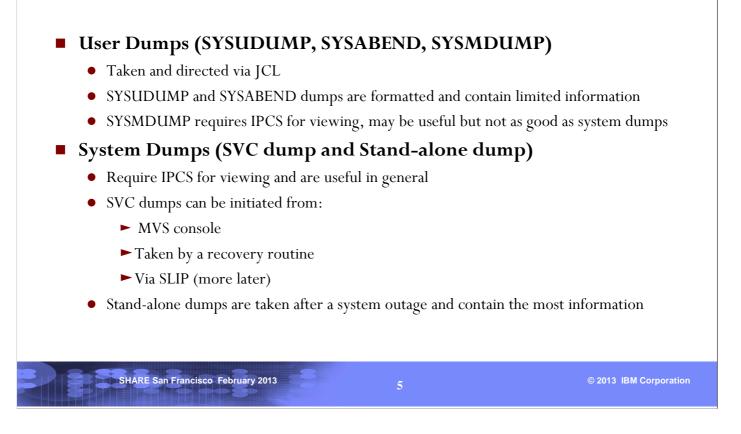
Types of problem documentation

The objective of collecting and reviewing problem documentation is to solve the problem Do you need to collect and review all kinds of documentation for every problem?
No. Only what is needed to solve the problem. You start with the basic problem documentation. If the problem is still not solved you investigate further with additional problem documentation
Basic problem documentation
 Data that would give you a picture of the failing program/function and the system at the time of the problem
Historical problem documentation
 Data that would give you a history of the events or errors leading up to the problem (and are usually available)
Specific problem documentation
 Specific data unique to a particular problem that would provide additional clues (usually obtained via a recreate of the problem)
SHARE San Francisco February 2013 4 © 2013 IBM Corporation

Even though there are different kinds of problem documentation, the debugger may not need to review all of them to resolve a problem. The key thing to do is to follow a logical diagnostic approach and continue to review the documentation until the problem is resolved. Start with the basic problem documentation and then analyze additional documentation if necessary.

Sometimes the basic problem documentation may not be available and the debugger would have to use ways such as SLIP to gather them.

Basic problem documentation - dumps



The basic problem documentation is a dump. Dumps give a picture of the failing program and the system at the time of the problem. We do not usually find user dumps very useful in debugging z/OS problems, so make sure you collect an SVC dump for a problem.

Historical problem documentation

System Trace

- Not available in an external media
- IPCS SYSTRACE formats the system trace table in a dump (short duration a few seconds usually)

LOGREC

- Provides a history of errors prior to the problem
- IPCS VERBX LOGDATA formats the in-core LOGREC buffer records in a dump

SYSLOG / OPERLOG

- Provides a history of messages prior to the problem
- IPCS VERBX MTRACE formats the most recently issued messages in a dump

SMF records and RMF reports

- Used in performance analysis and generally not used for common failures
- May be useful when debugging a performance related problem

SHARE San Francisco February 2013 6 © 2013 IBM Corporation

The historical problem documentation shows a history of errors or events leading up to the problem. They are usually available in a z/OS system as they are turned on and generated continuously.

Note that SMF records and RMF reports are generally not useful for a common failure such as an abend. We would usually review the system trace first, followed by the LOGREC and SYSLOG.

Specific problem documentation

GTF Trace

- Provides a history of system events in an external dataset (much longer duration than System Trace) can be formatted via IPCS GTFTRACE
- Activated via START GTF command with options that are helpful for the problem

Component Trace (CTRACE)

- Provides a history of events in a system component that exploits CTRACE
- Some components have minimal CTRACE running, others do not run at all
- CTRACE can be configured via the TRACE command
- A dump containing the CTRACE buffers needs to be taken (formatted via IPCS CTRACE)

Output of MVS commands

SHARE San Francisco February 2013

- Output from commands that display specific data related to the problem
- Data collected by SLIP
 - Dump or GTF trace collected via a SLIP trap on specific conditions

Specific problem documentation is usually collected via a recreate of the problem. The debugger should plan and test how to gather the specific problem documentation and the information that should be collected. Every attempt should be made to avoid having to recreate the problem over and over again.

IEM

Basic diagnostic approach

Use the available documentation to answer questions such as:
How did the problem occur (when, where, what happened)?
Why did the error occur?
Was the failing program/function at fault? Or was it a victim?
If it was a victim, how can the culprit be identified?
Why is this problem happening now? What was changed?
Use the manuals for reference (such as <u>z/Architecture POPs</u> or <u>MVS</u>

- System Codes) when applicable, don't ignore the details
- Don't give up. Keep asking questions!
- Problem recreate is the last resort

SHARE San Francisco February 2013

• Plan and test how specific problem documentation can be obtained from the recreate

The basic diagnostic technique is to find out all the facts about the problem and then use a logical approach to solve the problem. Do not ignore the details or make assumptions without facts. Assume recreate is impossible and you have this one chance to solve the problem. Exhaust all avenues and turn over every stone.

IEM

Capturing SVC Dumps

- SVC dumps can be taken by system/recovery routines via the SDUMP macro after an error or an abnormal event (without any action on your part)
- If you need to capture an SVC dump, you can:
 - Issue the MVS operator command DUMP to dump a job or address space (or several of them)
 OR
 - Use a **SLIP trap** with ACTION=SVCD to get a dump when a specific event occurs
- The SDATA parameter can be used in all of the above methods to identify which areas of storage to dump

There are 3 ways that SVC dumps can be taken:

SHARE San Francisco February 2013

•The SDUMP macro can be used by a recovery routine to take a dump. These dumps are usually the first sign of s problem.

•The DUMP command can be used to take a picture of an address space or a job, usually after they are hung.

•SLIP traps are very powerful in capturing a dump when an event occurs.

Complete or Partial SVC dump? SVC dumps can be partial for many reasons Partial dumps have missing storage or data Message IEA611I (dynamic dump dataset) or IEA911E (pre-allocated dump dataset) indicates whether a dump is complete or partial • IEA611I {Complete | Partial} DUMP ON dsname • IEA911E {Complete | Partial} DUMP ON SYS1.DUMPxx • If dump is partial, message will include **SDRSN** bits that explain why • Check for these messages in SYSLOG (you will not find them in the VERBX MTRACE output) What to do if the dump is partial? Do not discard it, analyze the dump to get the most out of it • If 'missing storage or data' is an inhibiting factor, investigate the reasons for a partial dump and resolve them so that you can get a complete dump next time SHARE San Francisco February 2013 © 2013 IBM Corporation

An SVC dump can be complete or partial. A partial dump will have missing storage or data, but it can still be useful. It is important to check if the dump is partial if you find that you are unable to display some storage or data from the dump.

The SDRSN is mapped in MVS Data Areas. The most common reasons for a partial dump are:

•MAXSPACE limit has been reached

•Address space being dumped is terminating or is not releasing its local lock

For MAXSPACE issues, consider whether the MAXSPACE specification needs to be increased. This is controlled through the CHNGDUMP command.

A partial dump due to an address space terminating or not releasing its local lock is environmental and cannot be prevented. In either case the partial dump should still be examined.

In the case of the address space terminating, there may have been earlier dumps produced. If a recreate is required, it might be possible to identify an earlier event that could trapped and dumped prior to the address space entering termination.

SVC Dump - preliminary analysis

<section-header><list-item><list-item><list-item><list-item><list-item><list-item><list-item><list-item><section-header>

The first thing to do when debugging an SVC dump is to find out what it contains and the reason for the dump. Then collect the error information and see if you can solve the problem. Some problems are easy, some are hard, and some are known problems!

IEM

TRM **SVC Dump - further analysis** How do I find out more about a 31-bit storage address? • Was this storage dumped? Is this storage in Private or Common? • If this storage contains code, what module is in it? How do I investigate errors or activity prior to the problem? • From system messages and logrec • From the System Trace • From the TCB/RB structure The above topics have been included in the appendix of this presentation but will not be discussed due to time constraints SHARE San Francisco February 2013 © 2013 IBM Corporation

This presentation includes an Appendix section containing additional diagnostic techniques of a SVC dump.

What type of SVC 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



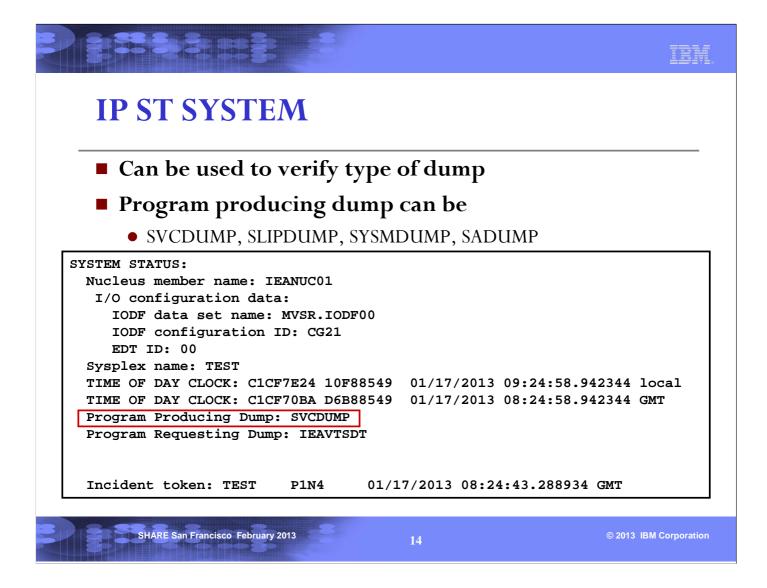
The most common type of dump is the SVC Dump. SYSMDUMP is very much like an SVC Dump but with less content. A standalone dump represents a system outage and is more complicated to work with. You may hear the term Transaction Dump (TDUMP). This is similar in content to a SYSMDUMP.

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.



The time provided 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).

A SLIP dump is actually a type of SVC dump as well, but the ST SYSTEM command makes the distinction for us between whether it is an SVC dump generated by SLIP versus one generated by a recovery routine or operator console dump request.

Use this comm	and to det	ermine the	e type of SVC
dump, as well	as the reaso	on for the	dump
 Recovery initiat system related i 	1 ,1	cally have a CO	OMPID= and other
• Console dumps the dump title	have a title of v	whatever the u	aser puts in COMM= as
 Dumps taken as title 	a result of a SL	.IP trap have '	SLIP DUMP ID=xxx' in

The dump title can provide a clue as to why the dump was taken. See the examples on the next page.

IBM.

■] _	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	
L	ITLE IST 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	1

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.

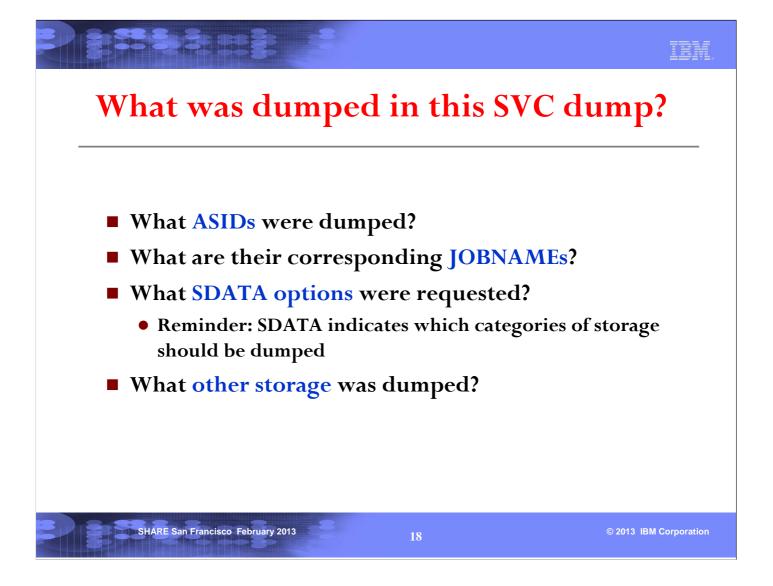
IP LIST SLIPTRAP (L SLIPTRAP)

 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:

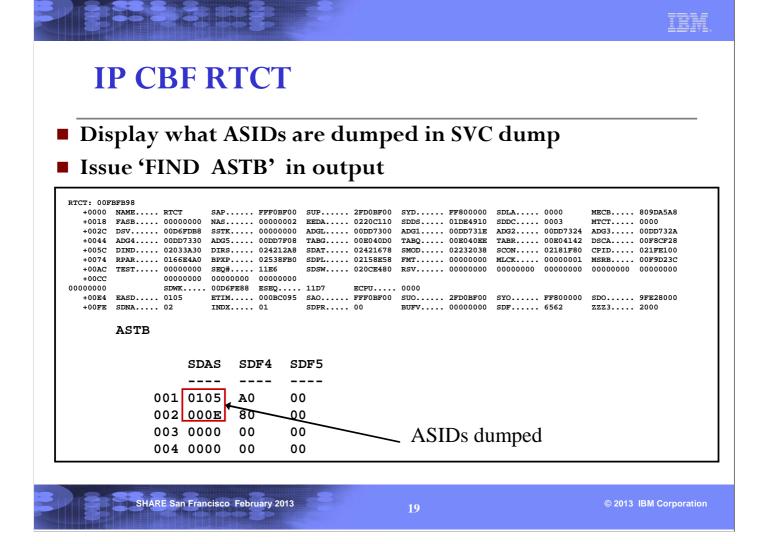
	SLIPTRAP			7
	LIST 00. LITERAL LENGTH(X'22') CI	HARACTER		
	00000000 SLIP SET,C=9C6,ID=\$WK	5,A=SVCD,ML=1		
RE	SHARE San Francisco February 2013	17	© 2013 IBM Corpora	ation

In this example, the dump was taken as a result of a SLIP on an ABEND9C6. The ID of the SLIP is \$WK5.



An SVC dump usually contains 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 IPCS commands do not always default to what you would expect.)

The SDATA options describe what areas of storage were requested to be in the dump.



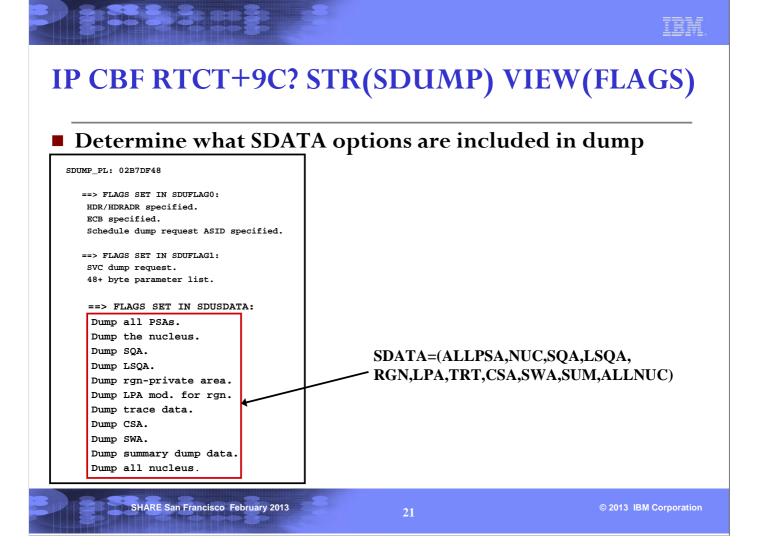
Knowing which address spaces are dumped is useful for determining what address space storage you can expect to find in the dump. It may give you a clue about what address spaces are involved in the problem.

ID	CEI	ECT /	ΑΤΤ	
ASI	D/JO	BNAM	E translation	
ASID J	OBNAME	ASCBADDR	SELECTION CRITERIA	
		00FD3900		
		00F4DE80		
		00F4DD00		
		00F4DB80		
0005 1	DUMPSRV	00F50980	ALL	
0006 2	XCFAS	00F50800	ALL	
0007 0	GRS	00F50680	ALL	
0008 \$	SMSPDSE	00F4EF80	ALL	
0009 0	CONSOLE	00F4EE00	ALL	
000A V	MLM	00F4EC80	ALL	
000B A	ANTMAIN	00FC6400	ALL	
000C #	ANTAS000	00FC6280	ALL	
000D I	DEVMAN	00FC6100	ALL	
	OMVS	00FA2800	ALL	

SHARE San Francisco February 2013

This report associates an ASID with a JOBNAME. From the CBF RTCT example on the previous page, we can see that asid(x'E') corresponds to a jobname of OMVS.

TDM



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:

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

NOTE: Typically the 'source' is a dump, but it could be a trace.

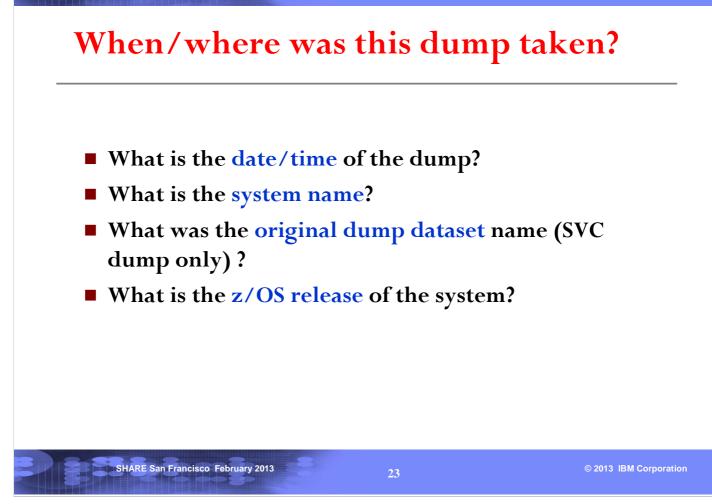
The IPCS inventory panel allows the user to view all the dumps in his IPCS dump directory. The IPCS inventory panel shows all the dumps the user has initialized under IPCS and is currently working with.

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

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

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

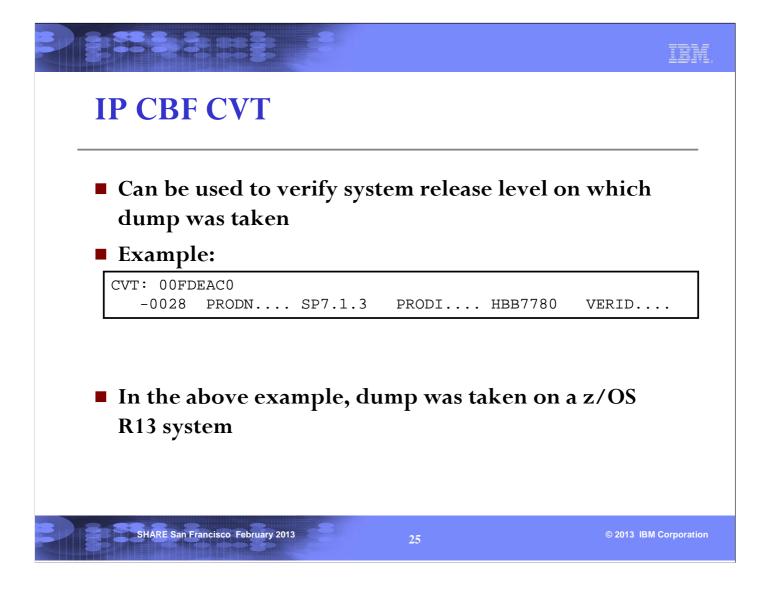
LZ gives detailed and rather cryptic information about the address spaces, dataspaces and storage included in a dump.



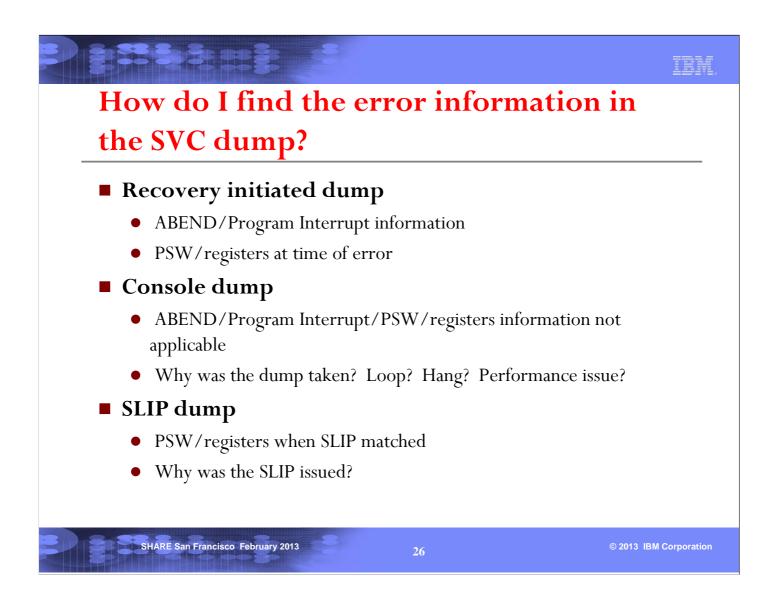
It is important to find out when and where the dump was taken. You may need to correlate this dump with other events occurring at the same time on this system or other systems.

	lom.
IP ST WORKSHEET (ST W)	
MVS Diagnostic Worksheet	
Dump Title: TESTLOOP DUMP	
CPU Model 2097 Version 00 Serial no. 026CC4 Address 02 Date: 09/28/2012 Time: 23:00:11.245557 Local	
Original dump dataset: D53DUMP.DYNSRV.ST5.D090928.T230007.SV00001	
Information at time of entry to SVCDUMP: Original SV dataset	VC dump
HASID 014B PASID 014B SASID 014B PSW 070C0000 A9A01482	
CML ASCB address 00000000 Trace Table Control Header address 7ED85000	
Dump ID: 001 Error ID: N/A	
SDWA address N/A	
System Name	
SYSTEM RELATED DATA	
CVT SNAME (154) SP5 VERID (-18)	
SHARE San Francisco February 2013 24 © 2013 I	BM Corporation

This is an example of a SVC dump taken on system SP5 on 9/28/12. The time is actually the time of the end of the global data capture phase of SVC dump processing.



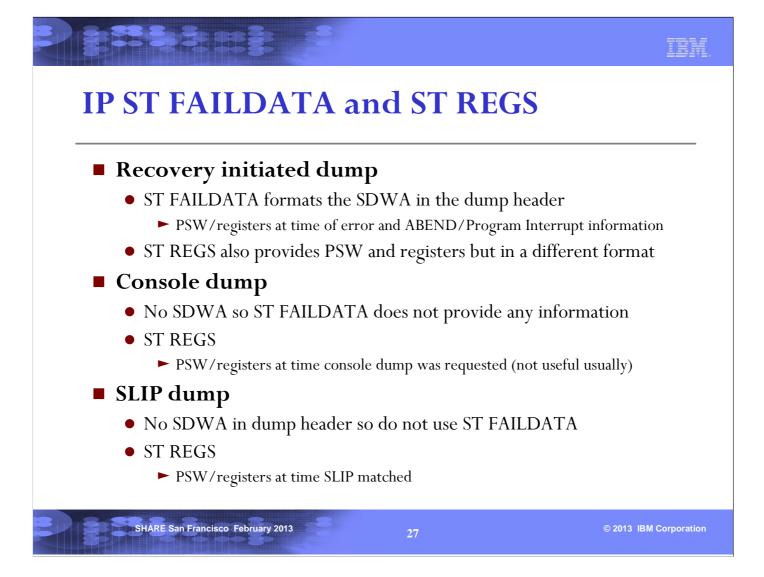
This is important because you should use the same level of IPCS as the z/OS release of the system on which the dump was taken. IPCS will put out a message if you use a level of IPCS that is different than the level of the z/OS system on which the dump was taken.



There is always a set of PSW/register 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/register set 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.



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
- •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.

IP ST FAI		1
SEARCH ARGUMENT ABS	DS/IEFW21SD#L RIDS/IEFAB4A2 AB/S00C4 PRCS/00000004	
Symptom PIDS/5752SC1B4 RIDS/IEFW21SD#L RIDS/IEFAB4A2 AB/S00C4 PRCS/00000004 REGS/0E064 REGS/0CA88 RIDS/IEFDB402#R	Description Program id: 5752SC1B4 Load module name: IEFW21SD Csect name: IEFAB4A2 System abend code: 00C4 Abend reason code: 00000004 Register/PSW difference for ROE: 064 Register/PSW difference for ROC: A88 Recovery routine csect name: IEFDB402	ABEND code and Reason code

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.

IP ST FAILDATA... Failing instruction = 947F8004 (NI) Protection Time of Error Information Exception PSW: 071C3000 81DA5AAA Instruction/length: 04 Interrupt code: 0004 Failing instruction text: CAB2947F 800441A0 315E50A0 Registers 0-7 GR: 008DFFD0 008E2EEC 008FA934 01DA6021 008E3003 008DFDD8 008DFDB8 Registers 8-15 GR: 0000000 01DA621C 0000000 008E2EA0 81DA5022 008E2EA0 81DA5A46 Home ASID: 0017 Primary ASID: 0017 Secondary ASID: 0017 PKM: 8000 AX: 0000 EAX: 0000 RTM was entered because of a program check interrupt. The error occurred while an enabled RB was in control. No locks were held. No super bits were set. The failing instruction is attempting to write into location 4 since R8 contains 0's. That is why the PIC 4 occurred. SHARE San Francisco February 2013 © 2013 IBM Corporation

In the case of a program check, 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).

In the machine instruction 947F8004, the base register position holds an 8. This means that this instruction, an aNd Immediate, is attempting to alter storage pointed to by register 8. Register 8 contains zero, so this instruction is trying to update low core location zero. Thankfully, there is extra protection on this critical area of system storage, so an ABEND0C4 PIC4 results, preventing an overlay. Note that in this example, the PSW execution key is 1, so even without special low core protection, the NI operation would have failed, since the low core storage is key0, which does not match the execution key.

CPU STATU		222 (DIRD	TNO IN DO	MARY, KEY 1, AMODE 31,		Additional
	BLED FOR PE		ING IN PRI	MARI, KEI I, AMODE SI,	DAI ON)	information
ASID(X')017') 01DA	5AAA. IEFW	21SD+0AAA	IN EXTENDED PLPA		on PSW
	at F85380,	• •	•			011 F 5 W
				block is dispatched		
HOME AS:	D: 0017 PR	IMARY ASID	: 0017 SEC	ONDARY ASID: 0017		
GPR VAL	JES					
0-3	008DFFD0	008E2EEC	008FA934	01DA6021		
4-7	008E3003	008DFDD8	008DFDB8	008E02C8		
8-11	L 00000000	01DA621C	00000000	008E2EA0		
12-1	5 81DA5022	008E2EA0	81DA5A46	81DA62E0		
ACCESS 1	REGISTER VA	LUES				
	008FB01F	-	00000000	0000000		
	00000000			0000000		
/						
	L 00000000	01DA621C	00000000	008E2EA0	1	

This report is not dependent on an SDWA (as is ST FAILDATA), and is useful for both SLIP dumps and dumps generated by a recovery routine. It provides more information about 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).

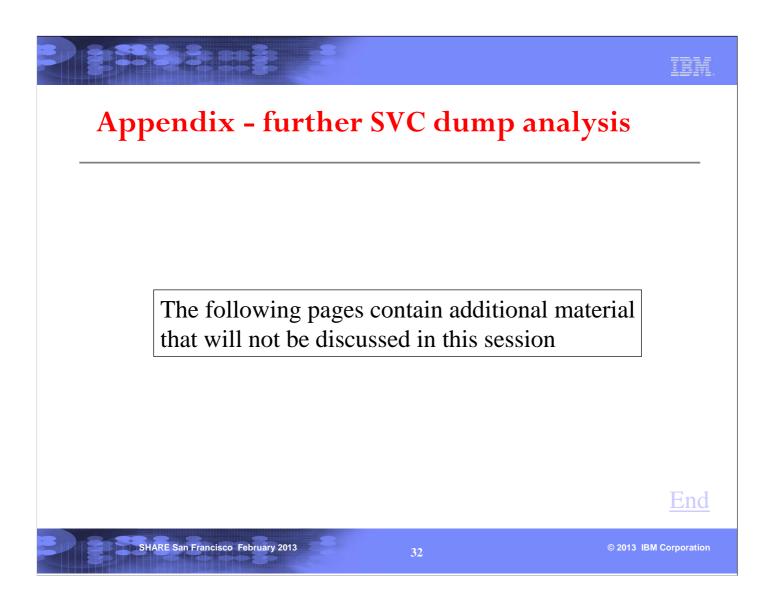
Reference Information

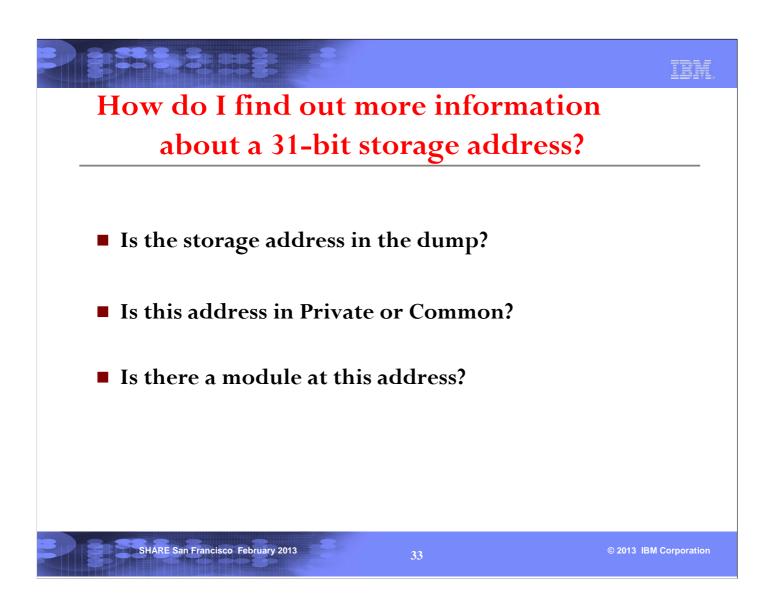
Manuals

- <u>z/OS MVS IPCS Commands</u>
- z/OS MVS IPCS Customization
- <u>z/OS MVS IPCS User's Guide</u>
- z/OS MVS Diagnosis: Reference
- z/OS MVS Diagnosis: Tools and Service Aids
- <u>z/OS MVS System Codes</u>
- <u>z/OS MVS Data Areas</u>

SHARE San Francisco February 2013

31





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 ===> 07228000.:075AEFFFStorage not available	с-
075AF000 A7F40014 0000000 C9C5C5D4 C2F8F0F4 x4IEEMB804	
 Storage not available' can be due to: storage is not dumped, or storage is paged out, or storage address is invalid 	
SHARE San Francisco February 2013 34 © 2013 IBM Corpora	tion

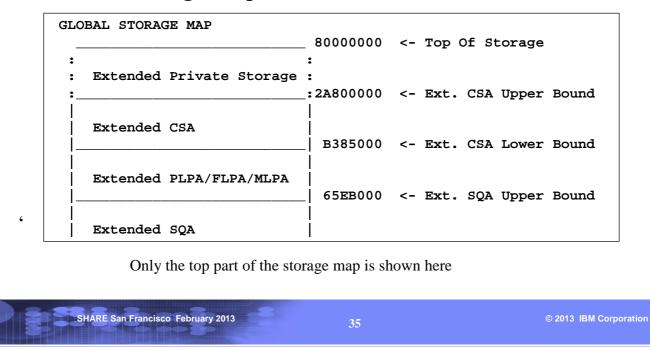
There are several reasons for 'storage not available', and it also depends on the kind of dump.

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

In an SVC dump or SYSMDUMP, some common areas (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'

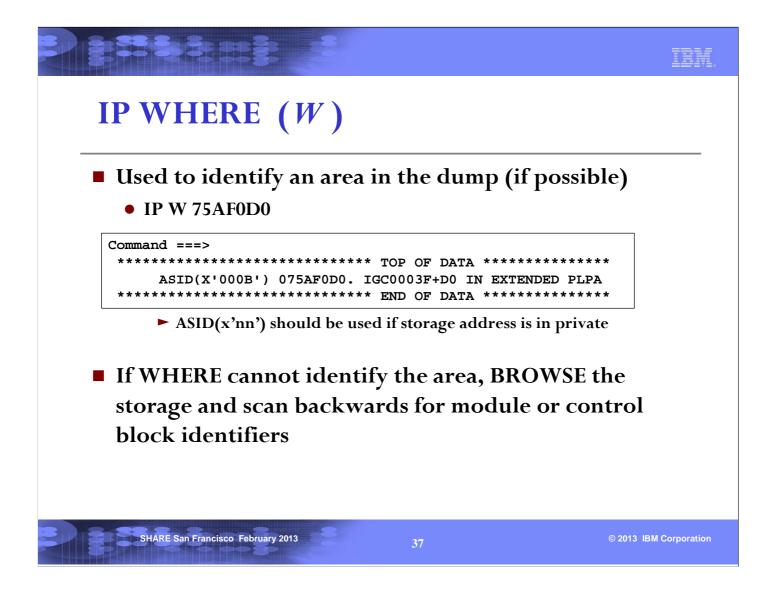
Scroll to the bottom and then page back to find the Global Storage Map



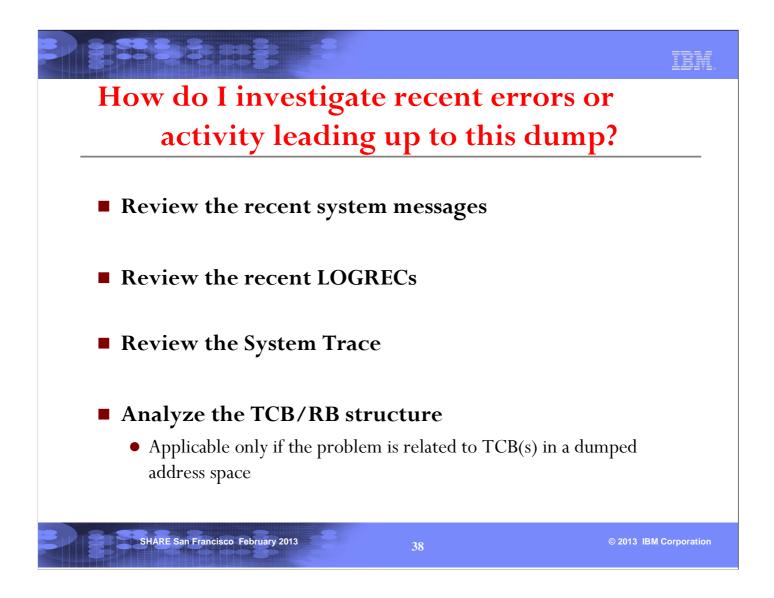
VERBX VSMDATA with a parameter of NOASIDS provides a global storage map that shows the boundaries of different area of storage for every address space. This will allow you to figure out whether the address at hand is in private or common. To see the details of the private storage of an address space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

IP VERBX VSA	ADATA 'ASID(nn) SUMM'
(nn in deci	
	om and then page back to find the
Local Storage Ma	Р
LOCAL STORAGE MAP	
	80000000 <- Top of Ext. Private
Extended	Ì
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
:	:
	t of the storage map is shown here
Only, the ten non	

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.



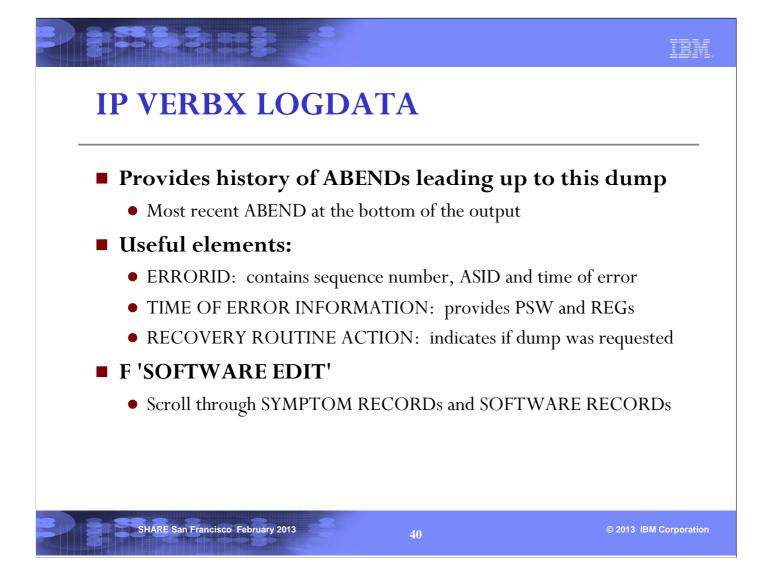
For an address that may point to a module or a common control block, the IPCS WHERE subcommand can be used. Note that the ASID parameter should be used if the address is in private storage.



It is always a good practice to find out 'what led up to the problem'. There are a few areas in the dump that can help.

<section-header><list-item><list-item><list-item><list-item><list-item><list-item><list-item>

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.



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.

			LBM.
IP VE	RBX LOGDAT	'A: Example	
TYPE: SOFTWARE RECO (SVC 13) FORMATTED BY: IEAVTE		EPORT DAY.YEAR REPORT DATE: 205.12 ERROR DATE: 203.12 HH:MM:SS.TH TIME: 12:38:51.76	
SEARCH ARGUMENT ABST	SYSTEM NAME: PS01 CPU=0000 ASID=00D4 TIME=12:38:5 RACT S/IKJEFT01#L RIDS/IKJEFTSC AB/S05		
SYMPTOM	DESCRIPTION		
PIDS/####28502 RIDS/IKJEFT01#L RIDS/IKJEFTSC AB/S0522 REGS/0EC3C REGS/0DCB0 RIDS/IKJEFT05#R	PROGRAM ID: ####28502 LOAD MODULE NAME: IKJEFT01 CSECT NAME: IKJEFTSC SYSTEM ABEND CODE: 0522 REGISTER/PSW DIFFERENCE FOR ROI REGISTER/PSW DIFFERENCE FOR ROI RECOVERY ROUTINE CSECT NAME: IM	D: CB0	
SHARE Sa	n Francisco February 2013	41	© 2013 IBM Corporation

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

31 **32 31 31** 31 31 31

TIME: Time the error occurred

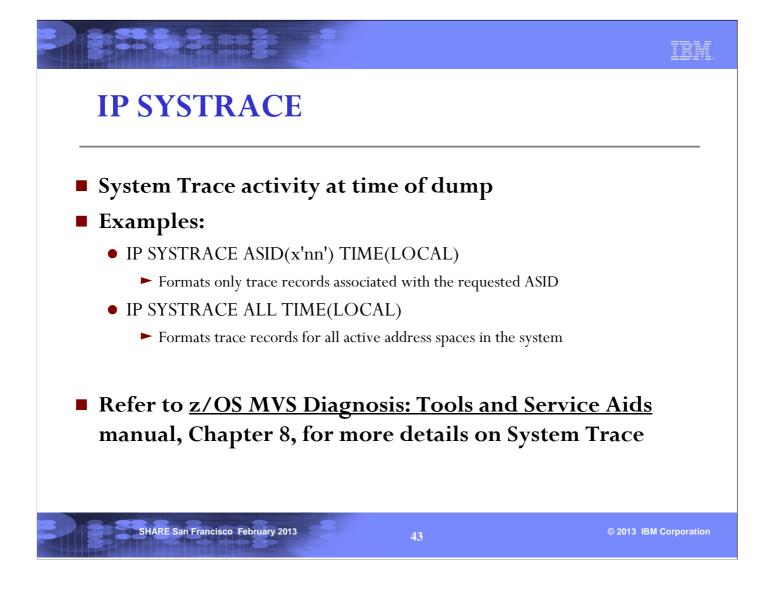
This example is of an ABEND522 that occurred in TSO csect IKJEFTSC in asid(x'D4') jobname PALNSMY at 12:38:51 on Julian day 203 in the year 2012.

IP VERBX LOGDATA: Example (cont) TIME OF ERROR INFORMATION PSW: 070D1000 00007848 INSTRUCTION LENGTH: 02 INTERRUPT CODE: 0001 FAILING INSTRUCTION TEXT: B06C1311 0A01 4100 00061B11 REGISTERS 0-7 GR: 00000001 FFFF93FC 00005FF8 00006EB0 008F2848 008FDE28 008C5FF8 FD000000 REGISTERS 8-15 GR: 008FD214 00006C78 008F35D8 00006B98 4000771E 00006B98 00006C0C 808FD040 PRIMARY ASID: 00D4 HOME ASID: 00D4 SECONDARY ASID: 00D4 PKM: 8040 AX: 0000 EAX: 0000 RTM WAS ENTERED BECAUSE ABTERM PROCESSING FORCED THE TASK TO TERMINATE. THE ERROR OCCURRED WHILE AN ENABLED RB WAS IN CONTROL. RECOVERY ROUTINE ACTION THE RECOVERY ROUTINE REQUESTED THAT TERMINATION PROCESSING CONTINUE. AN SVC DUMP WAS NOT REQUESTED. NO LOCKS WERE REQUESTED TO BE FREED. SHARE San Francisco February 2013 © 2013 IBM Corporation

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.

IBM.



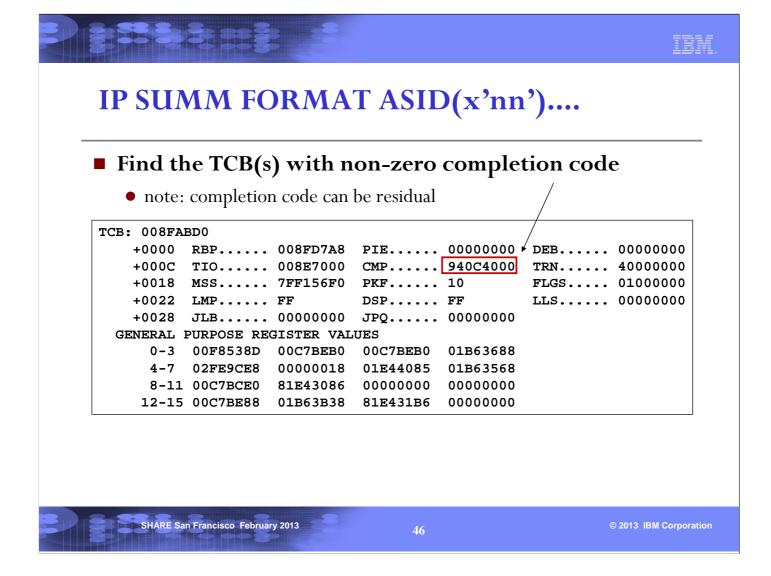
The TIME(LOCAL) parameter converts the time in SYSTRACE to local time. The default is raw hex timestamps.

			LZ and	d low	ver				
01	010D	006F0858	DSP		070C2000	BF5125F0	00000000	BF6C275A	0009в968
		006F0858 006F0858		_	078C0000 078C1000	BF6968A0 81452E60		00000000 00000000 00000000	3F781288
01	01BD	006E0758	*RCVY	PROG			940C4000	00000004	00000000
	In	z/OS R	13						
01	00E4	009FF800	DSP		-	_0AF2F598 80000000	00000000	E5000800	72F65720
		009FF800	-	010		_0AF8D180 80000000		072FF800	
01	00E4	009FF800	*RCVY	PROG			940C4000	00000010	0000000

The system trace table consists of many different kinds of entries. Prior to z/OS V1 R13, all PSWs in the system trace entries are 64-bit (scrunched). z/OS V1 R13 is the first release to support program execution above the 2G bar, as long as the program does not invoke system services. Since the instruction address can now be greater than 31 bits, some of the system trace entries have been changed to contain 128-bit PSWs. Note that the 128-bit PSW is displayed in 2 lines, but not in the order of bit 0 to bit 127. The instruction address is displayed first, then followed by the first half of the PSW in the next line.

	IP SUMM FORMAT ASID(x'nn')	
_	Formats the key control blocks of an address space	_
	 First scroll down to bottom to see the TCB summary 	
[* * * * TCB SUMMARY * * * *	
	JOB CONSOLE ASID 000B ASCB 00FA8080 FWDP 00000000 BWDP 00000000	
	0000008	
	TCB AT CMP NTC OTC LTC TCB BACK	
	008FE050 00000000 0000000 0000000 008FF890 008FD0D0 00000000	
	008FD0D0 00000000 0000000 008FE050 0000000 008FF890 008FE050	
	008FF890 00000000 008FD0D0 008FE050 008F0938 008FF2A0 008FD0D0	
	008FF2A0 00000000 0000000 008FF890 008F92F0 008F9E88 008FF890	
ies	008F9E88 00000000 008FF2A0 008FF890 00000000 008F9CF0 008FF2A0	
itted	••••	
re	•••	
	008FABD0 940C4000 0000000 008F5170 00000000 00000000 00647E88	
L		
	non-zero completion code (ABEND0C4)	
	non-zero completion code (ABENDUC4)	

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.

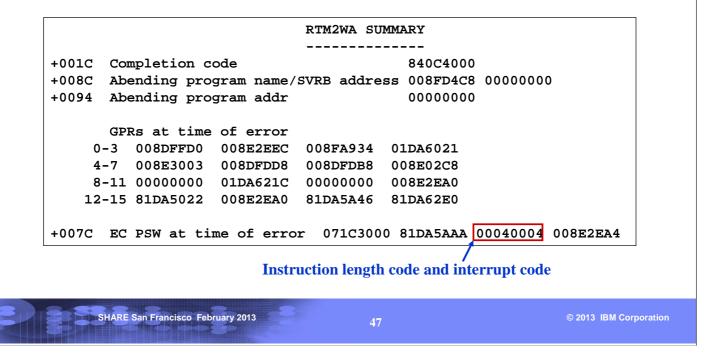


Once you found a TCB with non-zero completion code, you can issue a FIND of 'TCB: 00xxxxx' from the top of the output to find the TCB. The error under this TCB may be the one causing the dump to be taken. Or it can be a residual completion code.

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)



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.

IP SU	MM FORMA	T ASII	D(x'nr	n')	
There	are RBs under e	each TCB			
• RBs	are used to save status	after an int	errupt (usu	ally an SV	C)
ACTIVE RBS			1 1	,	,
PRB: 008F				SVC 1A	
-0020	XSB 008FAB38	FLAGS2	00000080	RTPSW1	0000000
-0014	0000000	RTPSW2			
-0008	FLAGS1 02000002	WLIC	0002001A	SZSTAB	00110082
+000C	FLCDE 00D22490	OPSW	071C1000	81F0E8BA	
+0018	SQE 00000000	LINK	008FABD0		► PSW
•••••	•••				that issued
••••					SVC 1A
SVRB: 008	FD358				
egsat -0020	XSB 008FD428	FLAGS2	00000000	RTPSW1	00000000
e time -0014	0000000	RTPSW2	00000000	01FF8000	
-0008	FLAGS1 02000000	WLIC	00020063	SZSTAB	001ED022
as issued +000C	FLCDE 00000000	OPSW	070C1000	81FF850A	
+0018	Q 00000000	LINK	008FAAD0		
+0020	GPR0-3 7F7238E4	7F725358	7F71E000	02F83E78	
SHARE S	an Francisco February 2013	48			© 2013 IBM Corporation

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).

	RBs a	re in chi	ronolog	ical orde	r		
	1125 0		8		-		
sv	RB: 008	FD4C8			Program int	errupt code 4	4
	-0020 -0014	XSB	008FD598 81DA5AAA	FLAGS2 RTPSW2			071C3000
	-0008	FLAGS1	02000000	WLIC	00040004	SZSTAB	001ED022
		Q		LINK		or Driving and	PSW
	 RB: 008	••					at time of error
50	-0020	XSB		FLAGS2		RTPSW1	0000000
egs at e time				WLIC	0002000C	00000000 SZSTAB	001ED022
error	+000C +0018	FLCDE Q		OPSW LINK		81ED9880	
	+0020	GPR0-3	008DFFD0	008E2EEC	008FA934	01DA6021	

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.

RB in IPCS SUMM FORMAT ASID(x'nn')

```
In z/OS R13
```

```
      SVRB: 009FD548

      -0020 XSB..... 7FFF9338 FLAGS2... 00
      RTPSW1... 070C0001

      -0014
      89F8D186 RTPSW2... 0006003B
      7FFFFBD0

      -0008 FLAGS1... 0200000 WLIC.... 0006003B
      3ZSTAB... 001ED022

      +0000 RSV..... 00000000 OPSW.... 070C0001
      89F8D186

      XSB: 7FFF9338
      LINK.... 00000000 XLIDR... 00000000

      +0000 XSB..... XSB
      LINK.... 00000000 XLIDR... 00000000
```

+0014 XLAS.... 00000000 TKN.... 0000 ASD.... 0000 Lines omitted here ... +00D4 AX..... 0000 PASID.... 00D4 +00D8 BEA..... 00000000 09F8BF52 +00E0 PSW16.... 07040001 80000000 00000000 09F8D186

- RBOPSW still exists but it is not relied upon
- **XSBOPSW16** contains the official 128-bit PSW

	SHARE San Francisco February 2013	50	© 2013 IBM Corporation
--	-----------------------------------	----	------------------------

The RB contains a field called RBOPSW, which is used by z/OS to save the interrupted PSW of a TCB. This PSW is 64-bit and will still exist in z/OS V1 R13. It will be maintained by z/OS, but it will not be not used by z/OS to re-dispatch the TCB. The official 128-bit PSW is now in the XSB. In a dump, you should usually find that the RBOPSW and XSBOPSW16 are similar.



Session 12842 z/OS Debugging: Old Tricks for New Dogs

Thank You Your comments will be greatly appreciated



© 2012 IBM Corporation