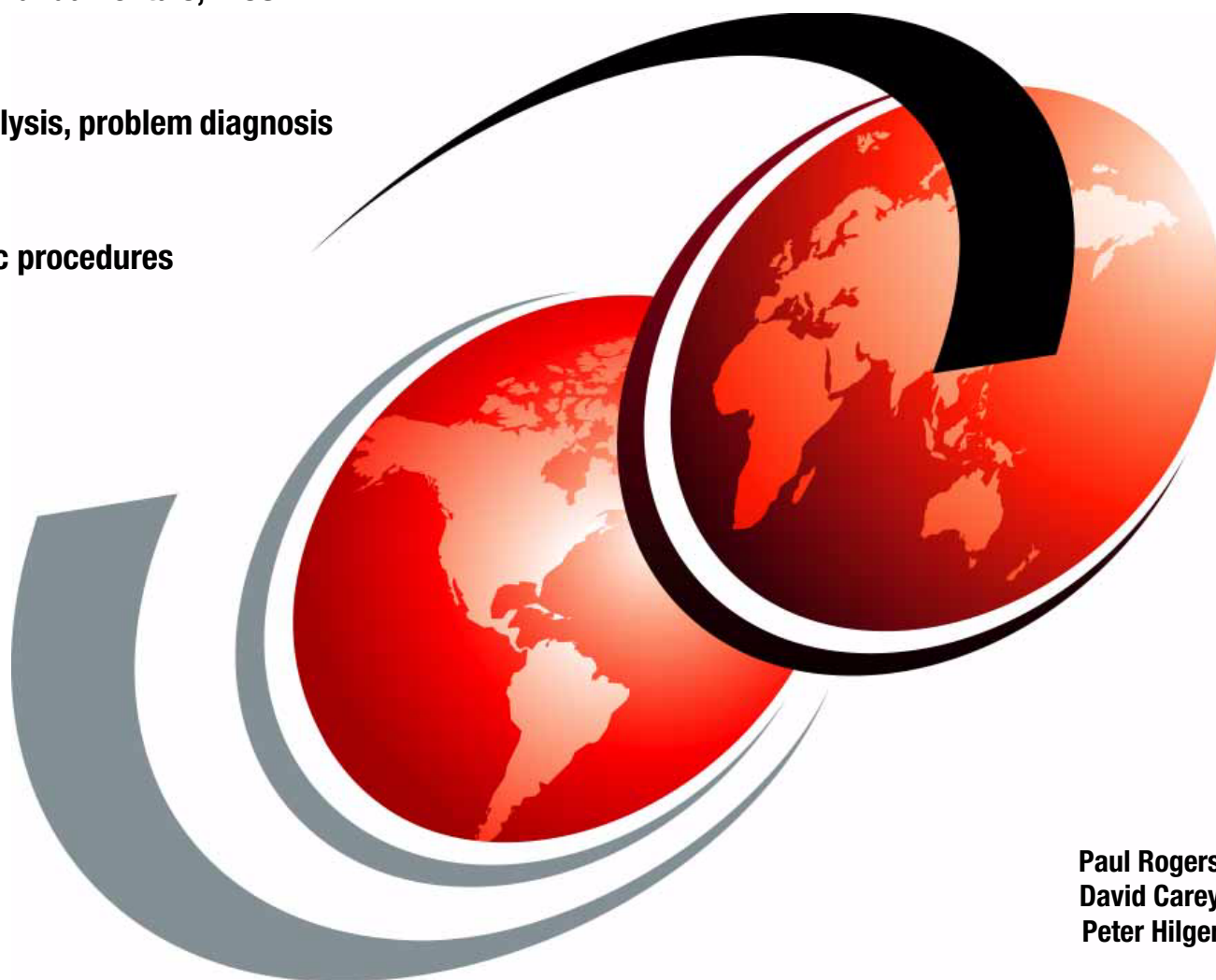


ABCs of z/OS System Programming Volume 8

Diagnosis fundamentals, IPCS

Dump analysis, problem diagnosis

Diagnostic procedures



Paul Rogers
David Carey
Peter Hilger

Redbooks



International Technical Support Organization

ABCs of z/OS System Programming Volume 8

May 2007

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (May 2007)

This edition applies to Version 1 Release 8 of z/OS (5694-A01), Version 1 Release 8 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
 Preface	ix
The team that wrote this book	x
Become a published author	x
Comments welcome	x
 Chapter 1. z/OS problem diagnosis fundamentals	1
1.1 Problem identification	2
1.2 What version or release is running	4
1.3 Waits, system hangs, and abends	6
1.4 Logging messages	8
1.5 Dumps and traces	10
1.6 Tools and service aids	12
1.7 Tools and service aids	15
1.8 Problem analysis with IPCS	17
1.9 SMP/E and maintenance	18
1.10 Using SMP/E and dumps	20
1.11 SDSF and RMF	23
 Chapter 2. Problem resolution steps	25
2.1 Identifying a problem	26
2.2 Prioritize problem resolution	28
2.3 Problem severity	30
2.4 Analyze a problem - ask for assistance	32
2.5 Gather Messages and Logrec	34
2.6 SYSLOG processing	36
2.7 SYSLOG messages	37
2.8 OPERLOG (operations log)	39
2.9 Job error logs	40
2.10 Logrec data set	43
2.11 Analyzing EREP reports	45
2.12 Using EREP	46
2.13 EREP reports	48
2.14 EREP parameter and control statements	50
2.15 Copy logs to tape	52
2.16 Implement a resolution	53
 Chapter 3. Common problem types	55
3.1 Common problem types	56
3.2 Stand-alone dumps	58
3.3 Symptom dump output	59
3.4 Waits, hangs, and loops	61
3.5 SLIP command	63
3.6 Storage overlays	65
3.7 Storage overlay during IPL	66
3.8 Storage overlay in a production system	68
3.9 SLIP to catch the overlayer	69

Chapter 4. Dump processing	71
4.1 Getting or requesting dumps	72
4.2 Slip commands	74
4.3 SLIP dumps	76
4.4 SNAP dumps	79
4.5 Stand-alone dumps	81
4.6 The SADMP program	84
4.7 Using stand-alone dumps	86
4.8 SADMP processing	88
4.9 SVC dumps	90
4.10 Allocating SYS1.DUMPxx data sets	92
4.11 Automatic allocation of SVC dump data sets	94
4.12 Dumping multiple address spaces in a sysplex	96
4.13 Managing taking a dump	99
4.14 Customizing dumps using SDATA options	101
4.15 Dump options and considerations	103
4.16 Catalog address space (CAS) dumps	104
 Chapter 5. z/OS trace processing	 107
5.1 z/OS trace facilities	108
5.2 GTF trace definitions	110
5.3 Implementing GTF trace	112
5.4 Component trace (CTRACE)	116
5.5 Implementing component trace	118
5.6 Component trace for System Logger	120
5.7 Master trace	122
5.8 GFS trace	124
5.9 System trace	126
5.10 SMS tracing	128
5.11 Trace data using an external writer	130
 Chapter 6. IPCS dump debugging	 133
6.1 IPCS dump debugging	135
6.2 IPCS command processing	137
6.3 IPCS dump debug example	139
6.4 IPCS support of large data sets	141
6.5 Setting the IPCS defaults	143
6.6 IPCS utility menu	145
6.7 SADMP dump data set utility	146
6.8 Using IPCS subcommands	147
6.9 SADMP analysis and COPYDUMP	149
6.10 IPCS COPYDUMP	151
6.11 Using subcommands	152
6.12 Analyzing dumps	154
6.13 IPCS trace commands - MTRACE	156
6.14 SYSTRACE command	158
6.15 IPCS SUMMARY subcommand	160
6.16 What is VERBX	162
6.17 IPCS VERBX LOGDATA command	164
6.18 Using the SYS1.LOGREC	166
6.19 IPCS virtual storage commands	168
6.20 Using IPCS to browse storage	172
6.21 Using IPCS to find the failing instruction	174
6.22 Analyzing for resource contention	176

6.23 Searching IBM problem databases	178
Chapter 7. z/OS Language Environment.	181
7.1 Language Environment ABEND and CEEDUMP handling	182
7.2 Common Language Environment messages	184
7.3 Language Environment message abend prefixes.	185
7.4 Collecting debug documentation.	187
7.5 Language Environment and CICS debugging.	189
7.6 Language Environment and UNIX System Services dumps.	191
7.7 Understanding CEEDUMP	193
7.8 ZMCH control block.	196
7.9 IPCS and Language Environment.	198
Chapter 8. Debug and maintenance tools.	201
8.1 Using SMP/E.	202
8.2 Find a load module	204
8.3 AMBLIST job to get LMOD and source information	207
8.4 IEAABD00, IEADMP00 and IEADMR00 members	210
8.5 PDATA options (only valid for IEADMP00)	212
8.6 SDATA and PDATA recommendations.	213
Chapter 9. SDSF and RMF	217
9.1 System Display and Search Facility (SDSF).	218
9.2 Using the SYSLOG for debugging	221
9.3 RMF Resource Measurement Facility.	223
9.4 RMF Monitor I data gathering	225
9.5 Monitor II data gathering	227
9.6 RMF Monitor III data gathering	229
Chapter 10. z/Architecture and addressing	231
10.1 Program status word (PSW)	232
10.2 Program-status word (PSW)	233
10.3 64-bit addressing.	236
10.4 Next sequential instruction	238
10.5 64-bit address space.	240
Appendix A. IPCS tools and lab exercises	243
A.1 IPCS lab exercise agenda	244
A.2 IPCS lab setup instructions.	248
A.3 Commands to analyze dumps	249
A.4 The RTCT control block	253
A.5 The IP ST REGS command	255
A.6 Browsing storage	257
A.7 IPCS SYSTRACE subcommand	260
A.8 IPCS VERBX MTRACE subcommand	264
A.9 IP SUMMARY FORMAT subcommand	266
A.10 The IP ANALYZE RESOURCE subcommand	268
A.11 Diagnosing excessive CPU time.	270
A.12 TSO user hung	271
A.13 Job IBMUSER3 hung (contention problem?)	272
A.14 A standalone dump example	273
A.15 LIST TITLE and LIST SLIPTRAP - Answers	275
A.16 IP ST WORKSHEET - Answers	275
A.17 Using the RTCT control block - Answers	275

A.18	Information from IP ST REGS - Answers	276
A.19	IP SYSTRACE - Answers	276
A.20	IP VERBX MTRACE - Answers	277
A.21	SUMMARY FORMAT - Answers	277
A.22	ANALYZE RESOURCE - Answers	277
A.23	Diagnosing excessive CPU time - Answers	278
A.24	TSO user hung - Answers	278
A.25	Job IBMUSER3 hung (contention problem?) - Answers	278
A.26	A standalone dump example - Answers	279
A.27	Diagnosing loops and hangs	281
Appendix B. Using IPCS to diagnose abends		283
B.1	Lab exercises	284
	Diagnosing an ABEND0C1 dump	289
	Diagnosing an ABEND0C4	291
	Diagnosing ABEND138 errors	293
	Diagnosing storage problems - ABEND878	295
	Diagnosing local storage shortage	299
	Lab exercise #1 - Answers IP ST REGS	302
	Lab exercise #1 - Answers IP SYSTRACE	302
	Lab exercise #1 - Answers Summary Format	302
	Lab exercise #2 - Answers diagnosing an ABEND0C1	303
	Lab exercise #3 - Answers diagnosing an ABEND0C4	303
	Lab exercise #4 - Answers diagnosing an ABEND138	304
	Lab exercise #5 - Answers diagnosing storage - ABEND878	305
	Lab exercise #5 - Answers ABEND878 - Analyzing storage use	306
	Lab exercise #5 - Answers ABEND878 - CSA/SQA tracker	306
	Lab exercise #6 - Answers diagnosing local storage shortages	307
Appendix C. z/OS trace processing data		309
C.1	GFS trace information	310
C.1.1	DIAGxx parmlib member syntax	310
C.1.2	GFS trace data	310
C.1.3	IPCS MVS dump component data analysis panel	311
C.1.4	SUMMARY subcommand parameters	312
C.1.5	VERBEXIT subcommand	313
C.1.6	VERBX VSMDATA subcommand	314
C.1.7	STATUS FAILDATA subcommand	314
Appendix D. IPCS commands		317
D.1	IPCS commands	318
Related publications		323
	IBM Redbooks	323
	Other publications	323
	How to get Redbooks	324
	Help from IBM	324

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of 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.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	IMS™	REXX™
DB2®	Language Environment®	RMF™
DFSMS™	MQSeries®	System/360™
Enterprise Systems	MVS™	System/370™
Architecture/390®	MVS/XA™	VisualAge®
Geographically Dispersed Parallel	OS/390®	VTAM®
Sysplex™	Parallel Sysplex®	WebSphere®
GDPS®	Redbooks®	z/Architecture®
Infoprint®	Redbooks (logo)  ®	z/OS®
IBM®	RACF®	zSeries®

The following terms are trademarks of other companies:

Java, RSM, Virtual Storage Manager, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Other company, product, or service names may be trademarks or service marks of others.

Preface

The ABCs of z/OS® System Programming is an 11-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information you need to start your research into z/OS and related subjects. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection serves as a powerful technical tool.

This publication, Volume 8, shows you how to:

- ▶ Adopt a systematic and thorough approach to dealing with problems and identifying the different types of problems
- ▶ Determine where to look for diagnostic information and how to obtain it
- ▶ Interpret and analyze the diagnostic data collected
- ▶ Escalate problems to the IBM® Support Center when necessary
- ▶ Collect and analyze diagnostic data—a dynamic and complex process
- ▶ Identify and document problems, collect and analyze pertinent diagnostic data and obtain help as needed, to speed you on your way to problem resolution

The content of the volumes is as follows:

Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation

Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKST, authorized libraries, SMP/E, Language Environment®

Volume 3: Introduction to DFSMS™, data set basics storage management hardware and software, catalogs, and DFSMSvts

Volume 4: Communication Server, TCP/IP, and VTAM®

Volume 5: Base and Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex™ (GDPS®)

Volume 6: Introduction to security, RACF®, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries® firewall technologies, LDAP, and Enterprise identity mapping (EIM)

Volume 7: Printing in a z/OS environment, Infoprint® Server and Infoprint Central

Volume 8: An introduction to z/OS problem diagnosis

Volume 9: z/OS UNIX® System Services

Volume 10: Introduction to z/Architecture®, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC

Volume 11: Capacity planning, performance management, WLM, RMF™, and SMF

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Paul Rogers is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS, z/OS UNIX, JES3, and Infoprint Server. Before joining the ITSO 19 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England providing OS/390® and JES support for IBM EMEA and the Washington Systems Center. He has worked for IBM for 39 years.

David Carey is a Senior IT Advisory Specialist with the IBM Support Center in Sydney, Australia, where he provides defect and non-defect support for CICS®, CICSplex/SM, the WebSphere® MQ family of products, and z/OS. David has been working in the IT industry for 25 years and has written extensively about diagnostic processes for the ITSO.

Peter Hilger is an IT Specialist at the ITS Technical Support Center, Mainz, Germany, working with defect-related support for customers.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks® in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an e-mail to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



z/OS problem diagnosis fundamentals

There should be a staff of people who diagnose software problems that occur while running the operating system. These people are usually system programmers for the installation.

If an installation does not wish to debug the problem or does not have the source code involved in the problem, use a diagnostic procedure to collect the problem data needed for reporting the problem to IBM. IBM will debug the problem and provide a fix.

If an installation wishes to debug the problem and has the source code, use a diagnostic procedure to collect problem data. The installation's diagnostician can use this data to debug the problem. If the problem is in IBM code, the diagnostician should report the problem to IBM.

To perform problem determination in a z/OS system address space, it may become necessary to determine the cause of the problem by searching problem databases, and, if necessary, reporting the problem to the IBM support center. This applies to a customer support person who can troubleshoot problems, such as the system programmer or system administrator, an experienced security administrator, or an experienced storage administrator.

The steps taken to investigate and analyze a problem are as follows:

- ▶ Problem determination
- ▶ Determining system problem indications
- ▶ Analyzing logs and error information
- ▶ Looking at dumps and traces

1.1 Problem identification

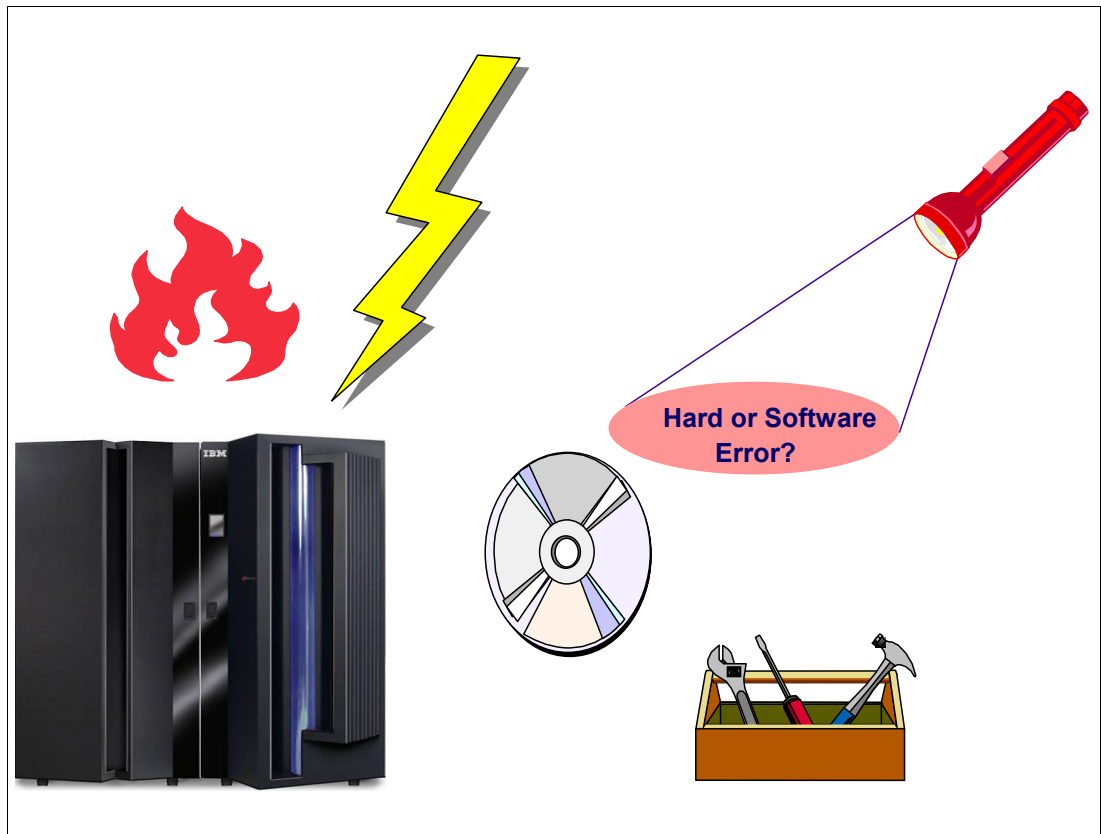


Figure 1-1 Problem determination

Identifying the problem

A system problem can be described as any problem on your system that causes work to be stopped or degraded. The steps involved in diagnosing these problems are different for each type of problem. Before you can begin to diagnose a system problem, however, you have to know what kind of problem you have. It may be either a hardware or software error.

Problem identification is often not a straightforward process, but an investigative exercise that requires a structured method that will enable the correct initial assessment to be made. This initial phase is important because decisions you make now relating to diagnostic data collection will influence the speed of the resolution.

In an ideal world, the programs you write would run perfectly, and never encounter an error, either software or hardware. In the real world, programs do encounter errors that can result in the premature end of the program's processing. These errors could be caused by something your program does, or they could be beyond your program's control.

Software errors

MVS™ allows you to provide something called *recovery* for your programs; that means you can anticipate and possibly recover from software errors that could prematurely end a program. To recover from these errors, you must have one or more user-written routines called recovery routines. The general idea is that, when something goes wrong, you have a recovery routine standing by to take over, fix the problem, and return control to your program so that processing can complete normally; if the problem cannot be fixed, the recovery

routine would provide information about what went wrong. If correctly set up, your recovery should, at the very least, provide you with more information about what went wrong with your program than you would have had otherwise.

Hard errors

If, in a multiprocessing system, a failure occurs in one central processor, the system invokes alternate central processor recovery (ACR) on another central processor. The system records the error as a hard failure that does not cause the processor to end.

1.2 What version or release is running

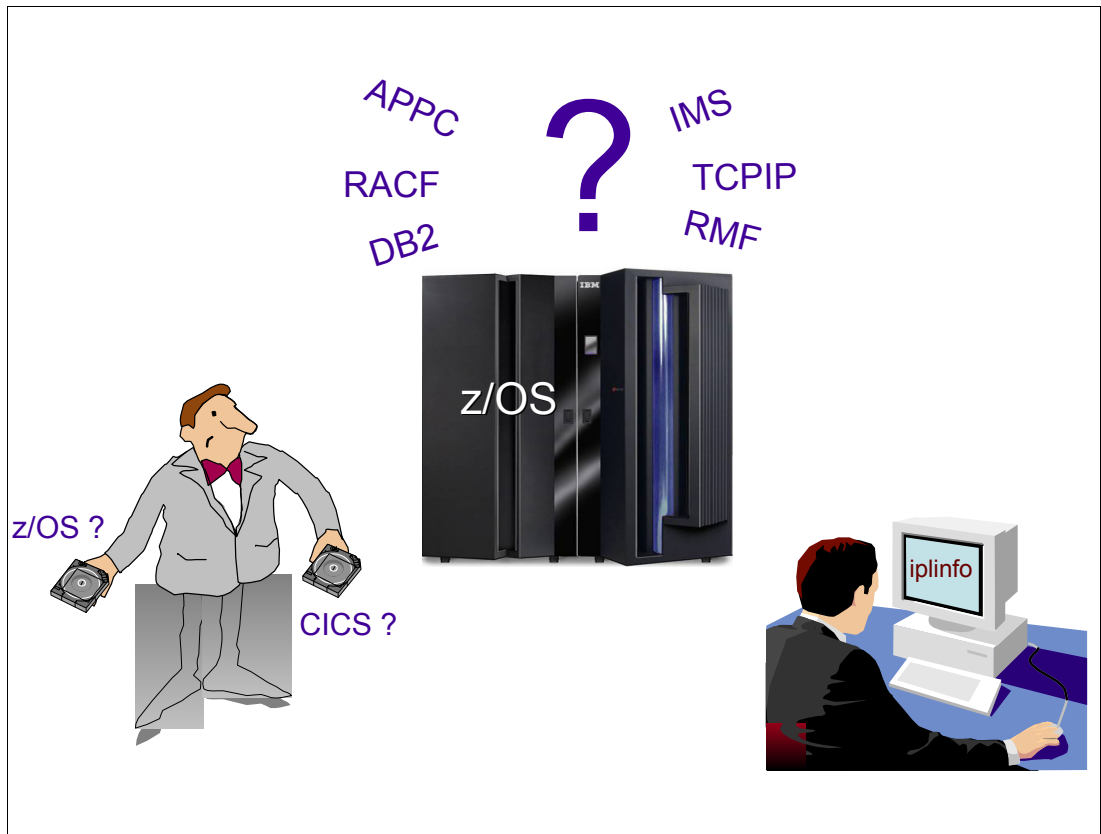


Figure 1-2 What version or release am I running

What release is running

Different platforms use different commands to show you product information. With many environments now comprising combinations of different platforms, operating systems and products all interact with the z/OS operating system in a distributed topology.

This information is vital to ensure that during problem analysis, we know exactly what system and product level we are dealing with and what maintenance has been applied to the product or module that is failing.

The sources of this information vary from the most obvious source, the system and job logs, to far more detailed interrogation using SMP/E and dump interrogation via IPCS.

In this chapter we discuss how to locate this important information.

Note: Do not overlook the most obvious source of release information that is often recorded in the console or job log messages generated during startup of the operating system or product.

How to get version or release information

In z/OS, the job log often shows release information generated during the start sequence for a product. Figure 1-3 shows an example of the CICS startup message written to the CICS job log.

```
DFHSI1500 SCSCPAA1 CICS startup is in progress for CICS Transaction Server Ver-
sion 3.1.0
```

Figure 1-3 CICS startup message

IPLINFO command

Very useful is the DISPLAY IPLINFO console command. It will show the following:

```
RESPONSE=MCEVS1
IEE254I 19.41.34 IPLINFO DISPLAY 911
SYSTEM IPLED AT 16.01.08 ON 03/09/2005
RELEASE z/OS 01.04.00 LICENSE = z/OS
USED LOAD00 IN SYS1.IPLPARM ON 8120
ARCHLVL = 2 MTLSHARE = N
IEASYM LIST = 00
IEASYS LIST = 00
IODF DEVICE 8120
IPL DEVICE 8101 VOLUME VS14R1
```

ABEND symptom string

The ABEND symptom string that is written to the master console and system log shows relevant release and maintenance information. Figure 1-4 shows an example of a CICS abend message in the MVS SYSLOG.

```
15.12.09 STC05964 +DFHME0116 CBNZPF00
(Module:DFHMEME) CICS symptom string for message
DFHFC0002 is PIDS/565514700 LVLS/530 MS/DFHFC0002 RIDS/DFHFCDN
PTFS/UQ56477 PRCS/00000445
```

Figure 1-4 CICS abend message in MVS syslog

This indicates that the CICS release in this case is R530 (known as CICS/TS 1.3) and the module where the failure was detected, DFHFCDN, has PTF UQ56477 applied. The PIDS field identifies the product compid.

WebSphere MQ and IMS releases

Figure 1-5 shows how WebSphere MQSeries® for z/OS displays the release level in the MQ MSTR joblog.

```
CSQY000I +MQT1 IBM WebSphere MQ for z/OS V5.3.1
```

Figure 1-5 WebSphere MQ for z/OS version information

Figure 1-6 displays the IMS™ release information that is written to the IMS CTL joblog.

```
DFSA0E00 - IMSID IMS VERSION SMPLEVEL GEN DATE GEN TYPE 869
          IMST      610      34C      031112 MODBLK
```

Figure 1-6 IMS Version information written to the joblog

1.3 Waits, system hangs, and abends

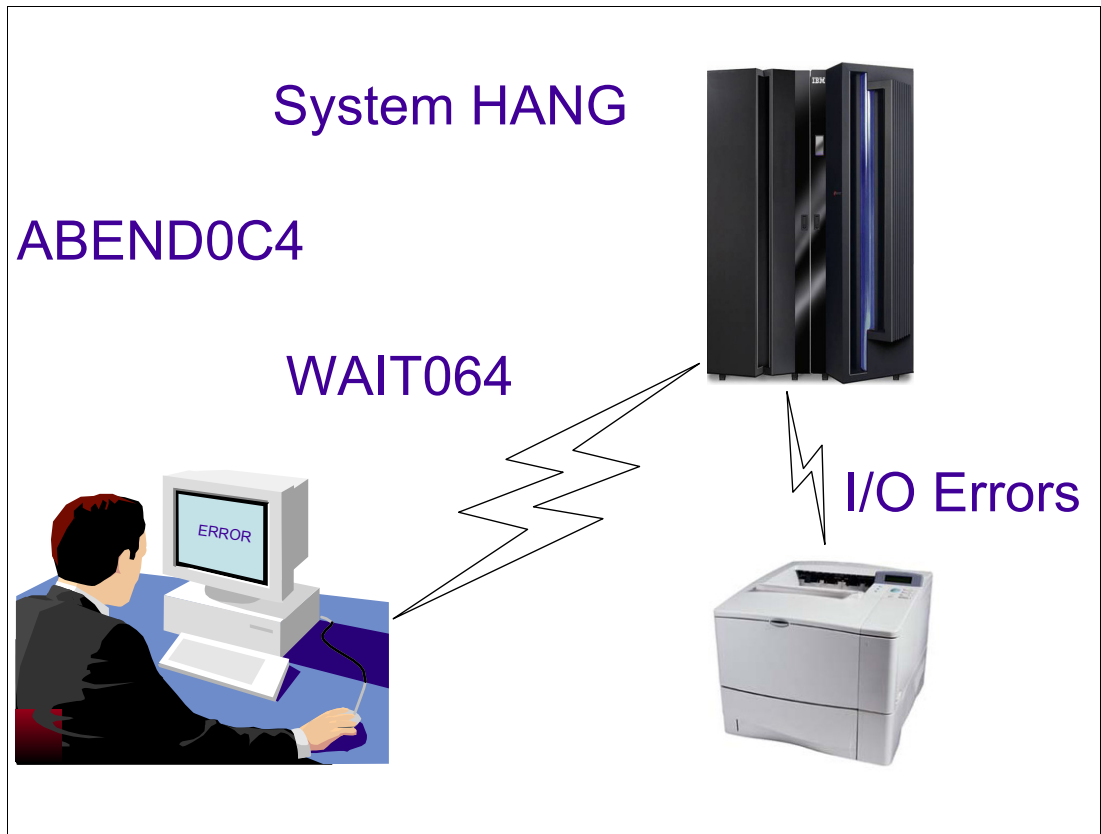


Figure 1-7 Problem determination

System problem indication

For the failure of an application program or program product, the program requests a SYSMDUMP dump.

If the system waits, hangs, or enters a loop, the operator requests a stand-alone dump.

Stand-alone dumps

The stand-alone dump program produces a high-speed, unformatted dump of main storage and parts of paged-out virtual storage on a tape device or a direct access storage device (DASD). The stand-alone dump program, which you create, must reside on a storage device that can be used to IPL.

Produce a stand-alone dump when the failure symptom is a wait state with a wait state code, a wait state with no processing, an instruction loop, or slow processing.

Use a stand-alone dump when:

- ▶ The system stops processing.
- ▶ The system enters a wait state with or without a wait state code.
- ▶ The system enters an instruction loop.
- ▶ The system is processing slowly.

These dumps show main storage and some paged-out virtual storage occupied by the system or stand-alone dump program that failed. Stand-alone dumps can be analyzed using IPCS.

Note: For additional information, see Appendix A-28, “Flowchart for loops and hangs” on page 281.

Abends

The term that is used most often here in relation to system or application problems is “abend”, which stands for abnormal end. Later we will discuss the different types of abends and also some other key factors that can affect system and application performance. We will also discuss some of the tools that can assist with determining what is occurring at a given point in time of the system. The following shows the different problem areas:

- ▶ Application program abends
- ▶ System program abends
- ▶ I/O errors
- ▶ System wait states
- ▶ System, subsystem, and application hangs
- ▶ System, subsystem, and application loops

1.4 Logging messages

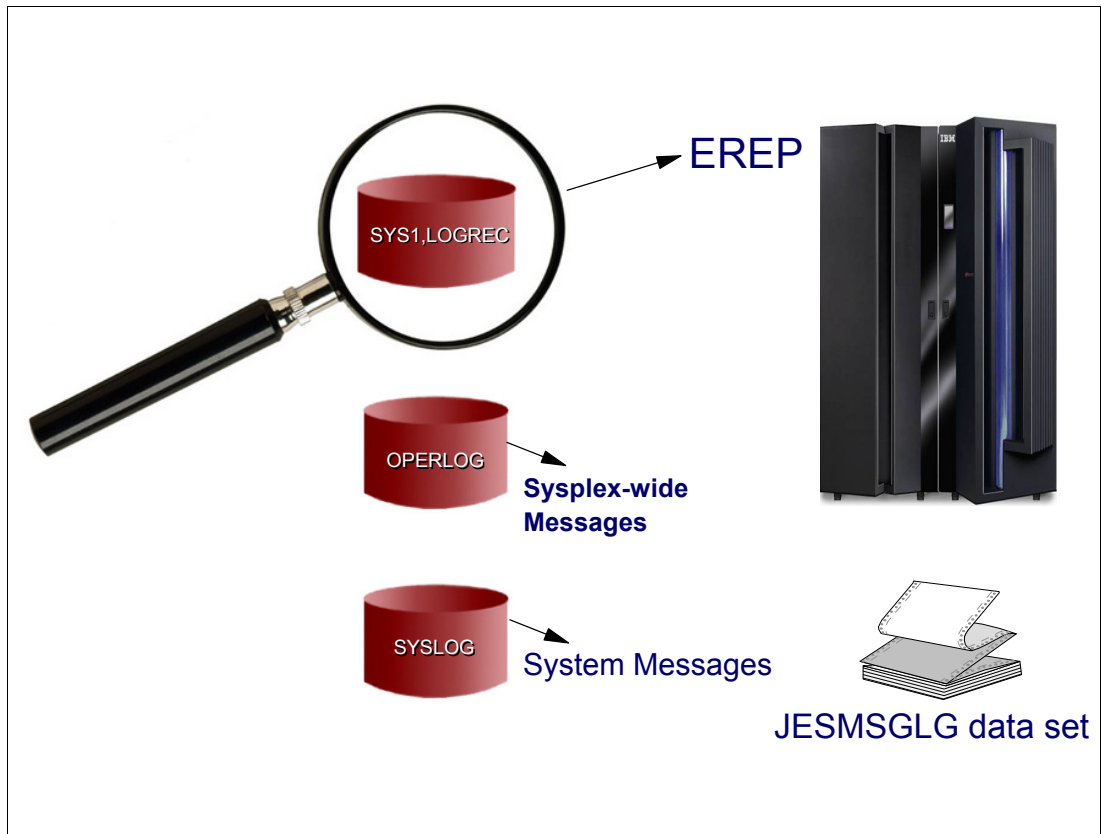


Figure 1-8 Problem determination

Logging messages and error information

On z/OS there are multiple choices to log messages and error related information. It depends on the installation settings and job-related options. The different log scenarios are shown in the following sections.

System log (SYSLOG) or console log

The system log (SYSLOG) is a data set residing in the primary job entry subsystem's spool space. It can be used by application and system programmers to record communications about problem programs and system functions. The operator can use the LOG command to add an entry to the system log.

Operations log (OPERLOG)

The operations log (OPERLOG) is a log stream that uses the system logger to record and merge communications about programs and system functions from each system in a sysplex. Only the systems in a sysplex that have specified and activated the operations log will have their records sent to OPERLOG. For example, if a sysplex has three systems, SYS A, SYS B, and SYS C, but only SYS A and SYS B activate the operations log, then only SYS A and SYS B will have their information recorded in the operations log.

JESMSGLOG output data set

The JESMSGLOG output data set for each job in the system contains system messages related to that job.

Error log (logrec)

When an error occurs, the system records information about the error in the logrec data set or the logrec log stream. The information provides you with a history of all hardware failures, selected software errors, and selected system conditions. Use the Environmental Record, Editing, and Printing program (EREP):

- ▶ To print reports about the system records
- ▶ To determine the history of the system
- ▶ To learn about a particular error

Use the records in the logrec data set or the logrec log stream as additional information when a dump is produced. The information in the records will point you in the right direction while supplying you with symptom data about the failure.

You clear the logrec data set when it is full or nearly full. To initialize or reinitialize it, use the service aid program IFCDIP00. To clear a full logrec data set, use EREP. IFCDIP00 creates a header record and a time stamp record for the logrec data set.

Note: The logrec data set is an unmovable data set. If you attempt to move it after IPL using a program, such as a defragmentation program, your system will experience difficulty both reading from and writing to the data set.

1.5 Dumps and traces

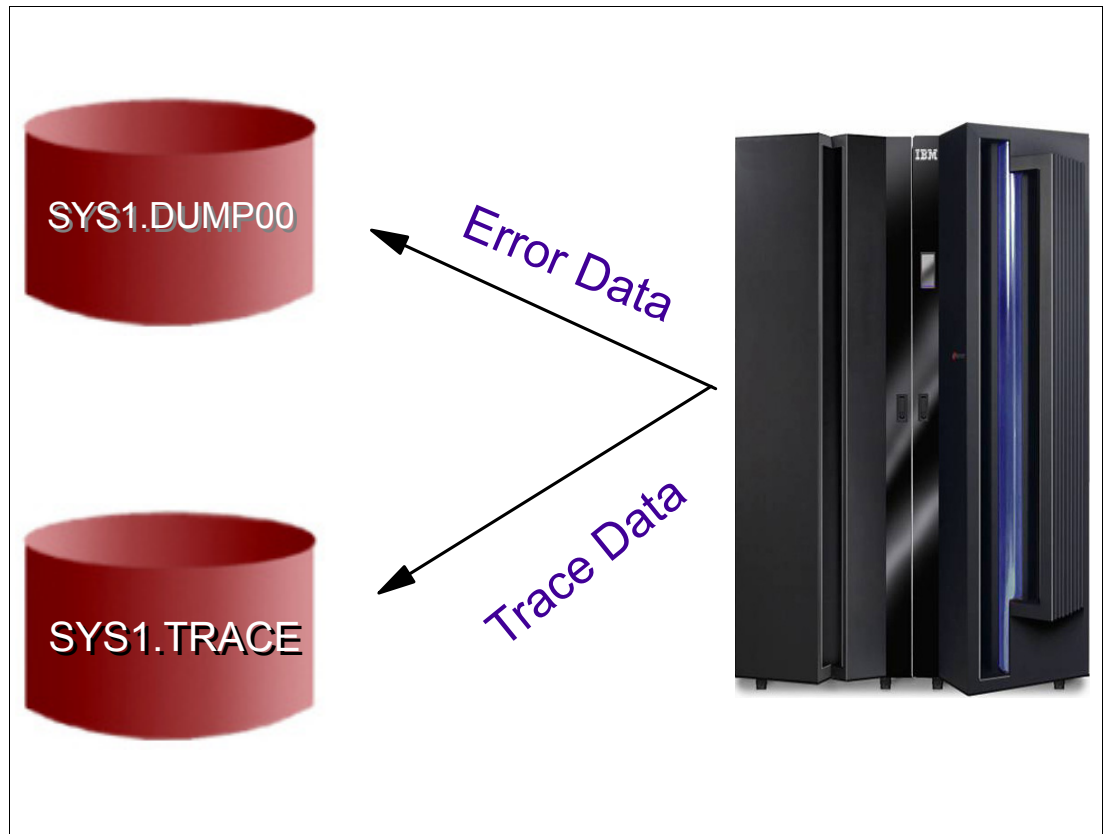


Figure 1-9 Problem determination

System dumps

A system generates a system dump when a severe error occurs if the dump is not suppressed by dump analysis and elimination (DAE). System dumps can also be user-initiated. A system dump creates a picture of an address space memory at the time of an error or after entering the dump command. A stand-alone dump creates a picture of all activities in the system. The following dumps can be initiated or requested by definition:

- ▶ Abend dumps
- ▶ SNAP dumps
- ▶ Stand-alone dumps
- ▶ SVC dumps
- ▶ Dumps triggered by an SLIP (serviceability level indication processing)

Traces

Another useful source of diagnostic data is the trace. Tracing collects information that identifies ongoing events that occur over a period of time during system initialization and operation. Some traces are running all the time so that trace data will be available in the event of a failure. Other traces must be explicitly started to trace a defined event.

- ▶ Component trace (CTRACE)
- ▶ Master trace (MTRACE)

- ▶ System trace (SYSTRACE)
- ▶ Getmain/Freemain trace (GFS)
- ▶ SMS trace

There are more traces that can be activated related to different components such as VIT VTAM internal trace. Normally the traces are written into a storage buffer, but if you would like to trace a longer time period you may use Generalized Trace Facility (GTF). GTF collects the trace data and stores it on a DASD volume.

1.6 Tools and service aids

Dumps

- ABEND dump - SNAP dump - Stand-Alone dump - SVC dump

Traces

- Component trace - GFS trace - GTF trace - Master trace - System trace

Service aids

- AMBLIST - Common storage tracking - DAE - IPCS - Logrec data set - SLIP traps - SPZAP

Figure 1-10 Tools and service aids

Tools and service aids

The following tools and service aids are provided by MVS for problem diagnosis.

ABEND dump Use an ABEND dump when ending an authorized program or problem program because of an uncorrectable error. The dump shows:

- ▶ The virtual storage for the program requesting the dump
- ▶ System data associated with the program

The system can produce three types of ABEND dumps, SYSABEND, SYSMDUMP, and SYSUDUMP. Each dumps different areas. Select the dump that gives the areas needed for diagnosing your problem. The IBM-supplied defaults for each dump are:

- ▶ SYSABEND dump - The largest of the ABEND dumps, containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program.
- ▶ SYSMDUMP dump - Contains a summary dump for the failing program, plus some system data for the failing task. SYSMDUMP dumps are the only ABEND dumps that you can format with IPCS.
- ▶ SYSUDUMP dump - The smallest of the ABEND dumps, containing data and areas only about the failing program.

SNAP dump Use a SNAP dump when testing a problem program. A SNAP dump shows one or more areas of virtual storage that a program, while running,

requests the system to dump. A series of SNAP dumps can show an area at different stages in order to picture a program's processing, dumping one or more fields repeatedly to let the programmer check intermediate steps in calculations. SNAP dumps are preformatted; you cannot use IPCS to format them.

Stand-alone dump Use a stand-alone dump when:

- ▶ The system stops processing.
- ▶ The system enters a wait state with or without a wait state code.
- ▶ The system enters an instruction loop.
- ▶ The system is processing slowly.

These dumps show main storage and some paged-out virtual storage occupied by the system or stand-alone dump program that failed. Stand-alone dumps can be analyzed using IPCS.

SVC dump SVC dumps can be used in two different ways:

- ▶ Most commonly, a system component requests an SVC dump when an unexpected system error occurs, but the system can continue processing.
- ▶ An authorized program or the operator can also request an SVC dump when they need diagnostic data to solve a problem.

SVC dumps contain a summary dump, control blocks, and other system code, but the exact areas dumped depend on whether the dump was requested by a macro, command, or SLIP trap. SVC dumps can be analyzed using IPCS.

Component trace Use a component trace when you need trace data to report an MVS component problem to the IBM Support Center. Component tracing shows processing within an MVS component. Typically, you might use component tracing while recreating a problem. The installation, with advice from the IBM Support Center, controls which events are traced for a component.

GFS trace Use GFS trace to collect information about requests for virtual storage through the GETMAIN, FREEMAIN, and STORAGE macros.

GTF trace Use a GTF trace to show system processing occurring in the system over time. The installation controls which events are traced. GTF tracing uses more resources and processor time than a system trace. Use GTF when you are familiar enough with the problem to pinpoint the one or two events required to diagnose your system problem. GTF can be read to an external data set as well as a buffer.

Master trace Use the master trace to show the messages to and from the master console. Master trace is useful because it provides a log of the most recently issued messages. These can be more pertinent to your problem than the messages accompanying the dump itself.

System trace Use system trace to see system processing occurring in the system over time. System tracing is activated at initialization and, typically, runs continuously. It records many system events, with minimal detail about each. The events traced are predetermined, except for branch tracing. This trace uses fewer resources and is faster than a GTF trace.

AMBLIST Use AMBLIST when you need information about the content of load modules and program objects or when you have a problem related to the modules on your system. AMBLIST is a program that provides lots of data about modules in the system, such as a listing of the load modules,

map of the CSECTs in a load module or program object, list of modifications in a CSECT, map of modules in the LPA (link pack area), and a map of the contents of the DAT-on nucleus.

- Common storage** Use common storage tracking to collect data about requests to obtain or free storage in CSA, ECSA, SQA, and ESQA. This is useful to identify jobs or address spaces using an excessive amount of common storage or ending without freeing storage. Use RMF or the IPCS VERBEXIT VSMDATA subcommand to display common storage tracking data.
- DAE** Use *dump analysis and elimination* (DAE) to eliminate duplicate or unneeded dumps. This can help save system resources and improve system performance.
- IPCS** Use IPCS to format and analyze dumps, traces, and other data. IPCS produces reports that can help in diagnosing a problem. Some dumps, such as SNAP and SYSABEND and SYSUDUMP ABEND dumps, are preformatted—they are not formatted using IPCS.
- Logrec data set** Use the logrec data set as a starting point for problem determination. The system records hardware errors, selected software errors, and selected system conditions in the logrec data set. Logrec information gives you an idea of where to look for a problem, supplies symptom data about the failure, and shows the order in which the errors occurred.
- SLIP traps** Use *serviceability level indication processing* (SLIP) to set a trap to catch problem data. SLIP can intercept program event recording (PER) or error events. When an event that matches a trap occurs, SLIP performs the problem determination action that you specify:
- ▶ Requesting or suppressing a dump.
 - ▶ Writing a trace or a logrec data set record.
 - ▶ Giving control to a recovery routine.
 - ▶ Putting the system in a wait state.
- SPZAP** Use the SPZAP service aid to dynamically update and maintain programs and data sets. For problem determination, you can use SPZAP to:
- ▶ Fix program errors by replacing a few instructions in a load module or member of a partitioned data set (PDS).
 - ▶ Insert an incorrect instruction into a program to force an abend or make a SLIP trap work.
 - ▶ Alter instructions in a load module to start component trace.
 - ▶ Replace data directly on a direct access device to reconstruct a volume table of contents (VTOC) or data records that were damaged by an input/output (I/O) error or program error.

1.7 Tools and service aids

- ☐ SPZAP
- ☐ SADMP
- ☐ SDUMP
- ☐ System trace
- ☐ External traces (GTF and CTRACE)
- ☐ SLIP
- ☐ IPCS

Figure 1-11 Diagnostic tools and service aids enhanced in z/OS V1R7

Tools and service aids

Tools include dumps and traces, while service aids include the other facilities provided for diagnosis.

For z/OS V1R7, the following enhancements have been made to the tools and service aids:

- SPZAP** SPZAP is a service aid program that operates in problem state. It allows you to dynamically update and maintain programs and data sets. SPZAP can be used to apply fixes to modules or programs that need to be at current levels of the operating system.
- SPZAP has been enhanced to support DSNTYPE=LARGE data sets. DSNTYPE=LARGE data sets are like conventional sequential data sets except for the fact that they may span more than 64K tracks per volume.
- SADMP** You need to make several decisions when planning for a stand-alone dump. You implement most of these decisions when you create the stand-alone dump program, either when you code the AMDSADMP macro, when you assemble the macro, or when you use the SADMP option on the IPCS Dialog.
- SADMP is the most fundamental diagnostic tool. The focus in z/OS V1R7 is to get SADMPs captured quickly and effectively when they are needed. Installations that are enlarging the sizes of their LPARs should consider the effect on SADMP production and analysis in their planning.

SDUMP	<p>An SVC dump provides a representation of the virtual storage for the system when an error occurs. Typically, a system component requests the dump from a recovery routine when an unexpected error occurs. However, an authorized program, or the operator, can also request an SVC dump when diagnostic dump data is needed to solve a problem.</p> <p>SDUMP is the preferred dumping tool in MVS via its many faces: DUMP command, SYSMDUMP, and transaction dump. SDUMP is improved in a number of areas and also focused on better analysis aids, partly to help the traditional audience of system programmers and vendor support personnel and partly to help traditional users of formatted dumping tools who are migrating to unformatted dumping at an increasing rate in the last several years.</p>
External trace	<p>Transaction trace supports the use of an external writer for processing transaction trace records. An external writer is specified on the initial command that activates transaction trace or is specified standalone while transaction trace is active.</p> <p>The changes for external trace writing support increased system speed, complexity, and size.</p>
SLIP	<p>The SLIP command controls SLIP (serviceability level indication processing), a diagnostic aid that intercepts or traps certain system events and specifies what action to take. Using the SLIP command, you can set, modify, and delete SLIP traps.</p> <p>For SLIP, improvements have been included in z/OS V1R7 to make it easier to trap circumstances where dumping, tracing, or related actions need to be taken.</p>
IPCS	<p>The interactive problem control system (IPCS) is a tool provided in the MVS system to aid in diagnosing software failures. IPCS provides formatting and analysis support for dumps and traces produced by MVS, other program products, and applications that run on MVS.</p> <p>For IPCS, enhancements in z/OS V1R7 include support for large block sizes, compression, and striping. You can limit the scope of analysis with the PROFLE command, and report handling is enhanced to enable you to focus only on pertinent information.</p>

1.8 Problem analysis with IPCS

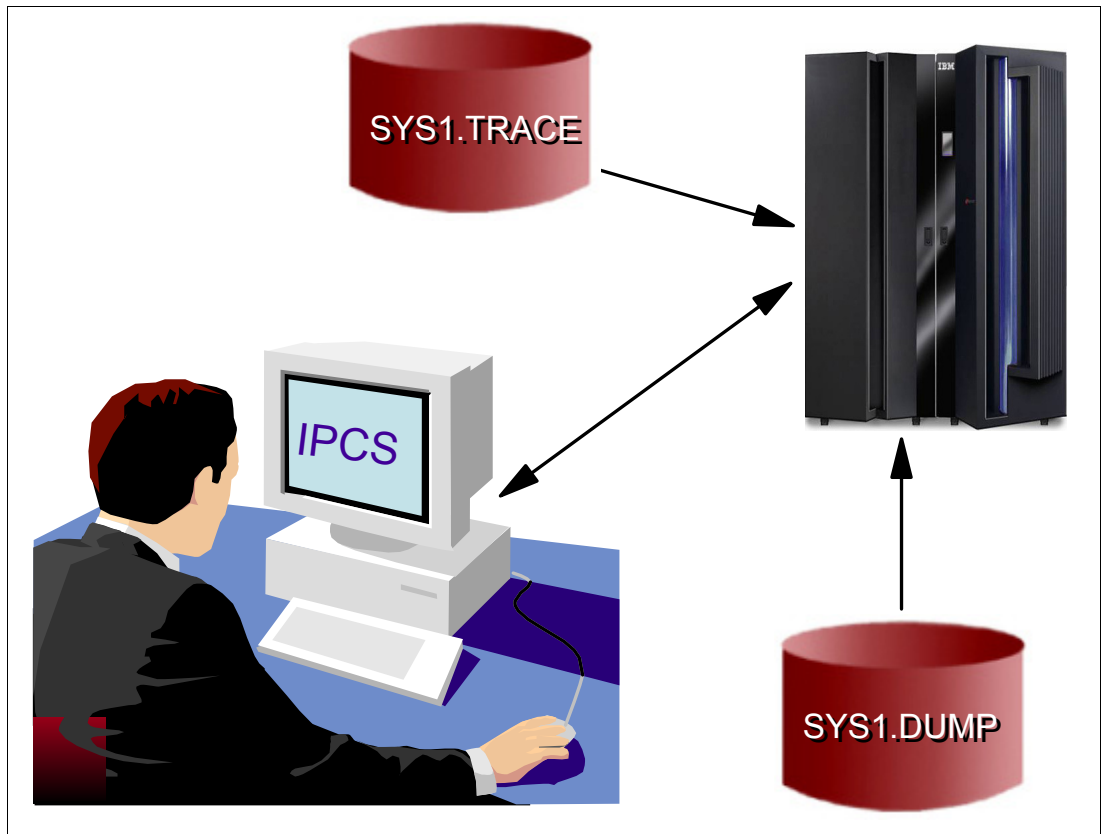


Figure 1-12 Problem determination with IPCS

Dump debug tool IPCS

The most powerful diagnostic tool at your disposal is Interactive Program Control System (IPCS). IPCS is a tool provided in the MVS system to aid in diagnosing software failures. IPCS provides formatting and analysis support for dumps and traces produced by MVS, other program products, and applications that run on MVS. There is an easy way to use IPCS to get search arguments that can be used to look for already known problems.

SVC dumps, stand-alone dumps, and some traces are unformatted and need to be formatted before any analysis can begin. IPCS provides the tools to format dumps and traces in both an online and batch environment. It provides you with commands that will let you interrogate specific components of the operating system, and enables you to review storage locations associated with an individual task or control block. IPCS allows you to quickly review and isolate key information that will assist with your problem determination process.

Some dumps such as CEEDUMP are in a readable format. To debug these dumps you have to browse them.

1.9 SMP/E and maintenance

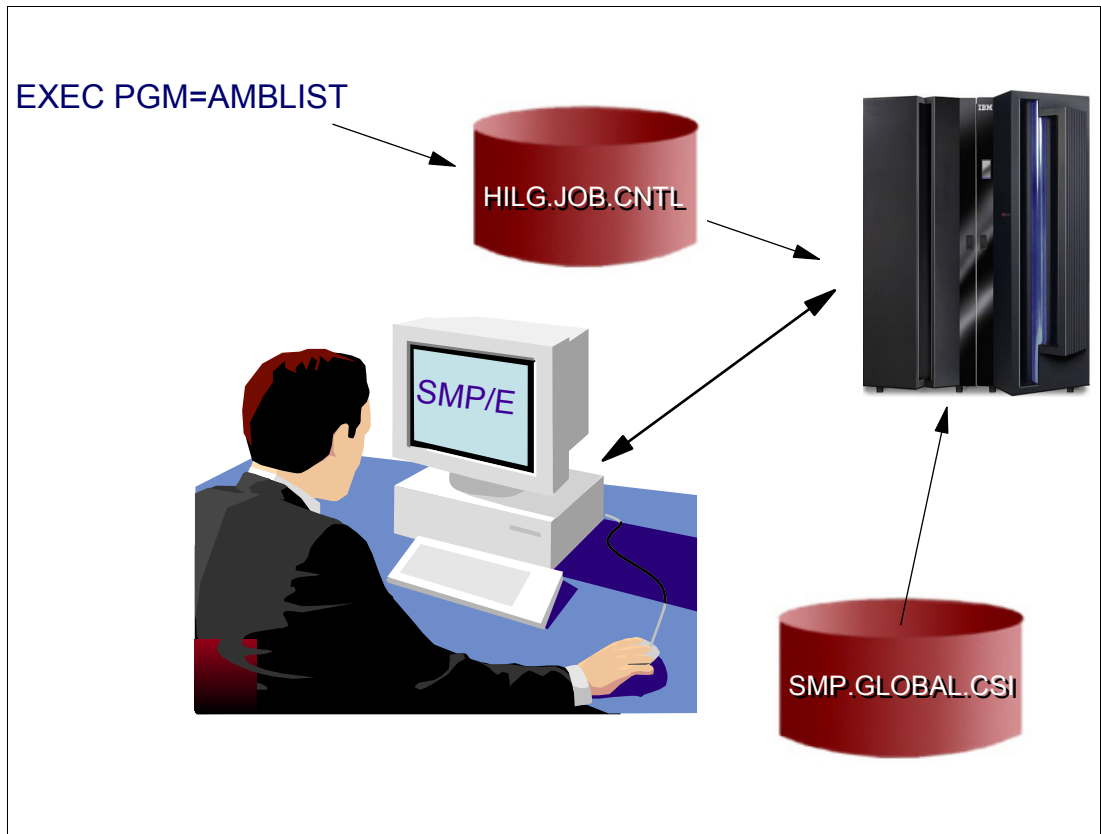


Figure 1-13 Problem determination with SMP/E

Helpful tool and program to get maintenance information

Analyzing a dump you may find that you need a maintenance level for a module you found in the storage area where the problem occurred.

SMP/E is a tool designed for managing the installation of software products on your z/OS system and to track the modifications you make to those products. Usually, it is the system programmer's responsibility to ensure that all software products and their modifications are properly installed on the system. Using SMP/E you can check which maintenance has been installed for different components.

Dump information does not always provide the module name. Instead, it provides the Load Module Name (LMOD). LMOD is a group of modules linked together. To find the module name you are interested in, you need to run the JCL for PGM=AMBLIST. The output can list either the modules or modules and the source. This selection depends on what you are looking for.

PGM=AMBLIST

The AMBLIST service aid prints formatted listings of modules to aid in problem diagnosis. Use it to list the CSECTs in the load module. Use the offset into the load module to identify the CSECT containing the failing instruction. Then subtract the starting address of the CSECT from the instruction address to obtain the offset into the CSECT.

AMBLIST can be used to provide listings showing:

- ▶ The attributes of program modules
- ▶ The contents of the various classes of data contained in a program module, including SYM records, IDR records, external symbols (ESD entries), text, relocation entries (RLD entries), and ADATA
- ▶ A module map or cross-reference for a program module
- ▶ The aliases of a program module, including the attributes of the aliases

AMBLIST problem data

AMBLIST provides the following problem data:

- ▶ Formatted listing of an object module
- ▶ Map of the control sections (CSECTs) in a load module or program object
- ▶ List of modifications to the code in a CSECT
- ▶ Map of all modules in the link pack areas (LPAs)
- ▶ Map of the contents of the DAT-on nucleus (The map no longer represents the IPL version and message AMB129I will be issued.)

1.10 Using SMP/E and dumps

- ❑ Use SMP/E to verify product and PTF levels
 - Use the SMP/E CSI GZONE query
 - Displays FMIDs in global zone data sets
 - CROSS-ZONE QUERY
 - Maintenance levels for load modules
- ❑ IPCS can be used for operating system releases
 - Format the CVT with the CBFORMAT command
 - Release levels and FMIDs
 - Product maintenance levels
 - CICS - VERBX DFHPD530 'LD=1'
 - DB2 - run DIAGNOSE DISPLAY MEPL utility

Figure 1-14 Using SMP/E and dumps for release and product information

Using SMP/E

In z/OS, SMP/E can be used to verify product and PTF levels. SMP/E is used to manage and maintain information related to system and product installation and maintenance. With SMP/E you can interrogate what has been installed into the product libraries, but this does not necessarily reflect what has been migrated to a production environment. So take care when assuming that the maintenance that is supposed to have fixed a problem, has actually been moved into the production data sets.

SMP/E does not manage the migration of upgrades. Figure 1-15 shows the result of an SMP/E CSI GZONE query. This displays the Function Modification Identifiers (FMIDs), or, more specifically, product components that have been received into the global zone data sets. This is the first installation level. The next is to APPLY the product or maintenance into the TARGET libraries, then finally ACCEPT the product or maintenance into the DLIBs, or distribution libraries.

Entry Type:	GZONE	Zone Name:	GLOBAL
Entry Name:	GLOBAL	Zone Type:	GLOBAL
Default OPTIONS:	CICSOPT	Related Zone:	

ZONES	CIC22DZ	CIC22TZ	
SRELS	C150		
FMIDS	DELCIPM	HBDD110	HCCV320
	HCP2200	HOB5110	HOB7110
	JCI620D	JCI6201	JCI6202
	JCMZ230	JCP2202	
		HCI6200	HCMZ100
		HOZ2110	H24D120
		JCMZ111	JCMZ130
		JCMZ201	JCMZ200
		JCCV32B	

Figure 1-15 SMP/E SMPCSI query for the GLOBAL zone

CROSS-ZONE QUERY

The SMP/E CROSS-ZONE QUERY panel lets you interrogate the maintenance level of a specific module or load module. Figure 1-16 shows an example of a cross-zone query request against the DFHSMGF module. This shows us that in the target library this module has an RMID level of UQ68396, which means that a PTF (UQ68396) has been applied to this module.

```

Entry Type:  MOD
Entry Name:  DFHSMGF

To return to the previous panel, enter END .

To select an entry from a zone, enter S next to the zone.

    * - Entry not found in zone.
    ** - Zone could not be allocated or is not initialized.


```

----- Status -----					
ZONE	FMID	RMID	LASTUPD	DISTLIB	UMID(S)
-----	-----	-----	-----	-----	-----
CIC22DZ	HCI6200	HCI6200	HCI6200	ADFHMOD	
CIC22TZ	HCI6200	UQ68396	HCI6200	ADFHMOD	
GLOBAL	*				

Figure 1-16 SMP/E Cross-Zone Query for a MODULE

Note: What is reflected in the SMP/E environment does not necessarily reflect what is running in your problem system environment. It shows what maintenance has been received, applied, and accepted, but does not show what libraries or data sets have been migrated to higher level systems.

Getting release information from the dump

IPCS, the Interactive Problem Control System, which we discuss later, can also be used to verify the operating system or product release, as well as abend symptom data as follows: Using IPCS, we can format the Communication Vector Table (CVT) to determine the release

of z/OS that is running. The IPCS command that can be used is the CBFORMAT command, which means Control Block Format, and is usually abbreviated as CBF. Figure 1-17 shows the result of an IPCS CVT format.



Figure 1-17 IPCS Communication Vector Table format

This is the first line of the formatted CVT control block and tells us that we are running z/OS V1R2, as indicated by the PRODN value, SP7.0.2, and the FMID for this release of z/OS is HBB7705, as indicated in the PRODI field. The MDL field indicates that this version of z/OS is running on a 2064 processor.

In CICS, if we format the dump using IPCS VERBX ‘CSA=2’ we can review the data at offset x’9F’ which displays the CICS release level; for example, 53 or 62.

We can also interrogate the maintenance that has been applied to modules using IPCS as follows:

In CICS, for example, issue the IPCS command VERBX DFHPD530 ‘LD=1’ and locate the PROGRAM STORAGE MAP. Figure 1-18 on page 22 shows an example of an IPCS format of the CICS Loader domain.

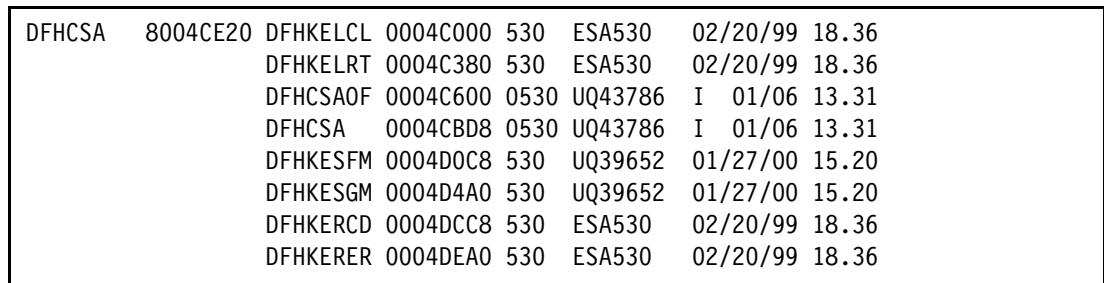


Figure 1-18 CICS IPCS format of the Loader domain

In DB2® you can run the DIAGNOSE DISPLAY MEPL utility to format the module information. Figure 1-19 shows an example of the DB2 Diagnose Display MEPL process.

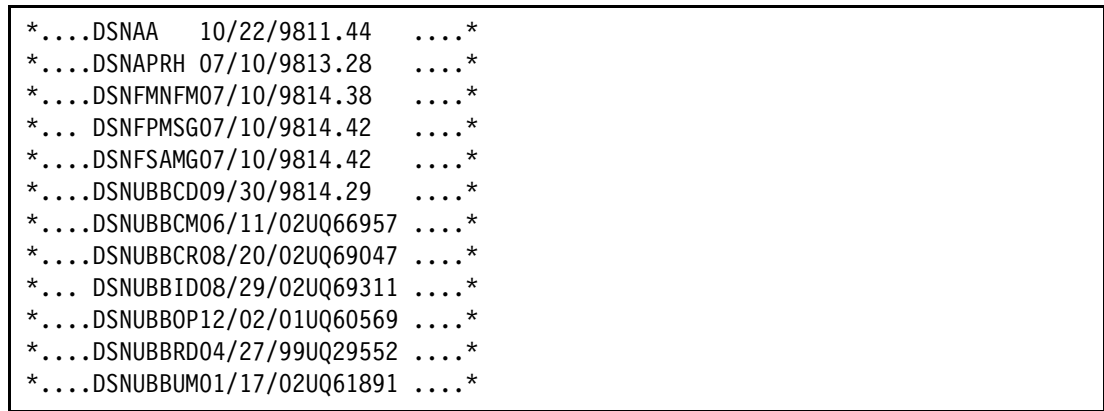


Figure 1-19 DB2 Diagnose Display MEPL output

1.11 SDSF and RMF

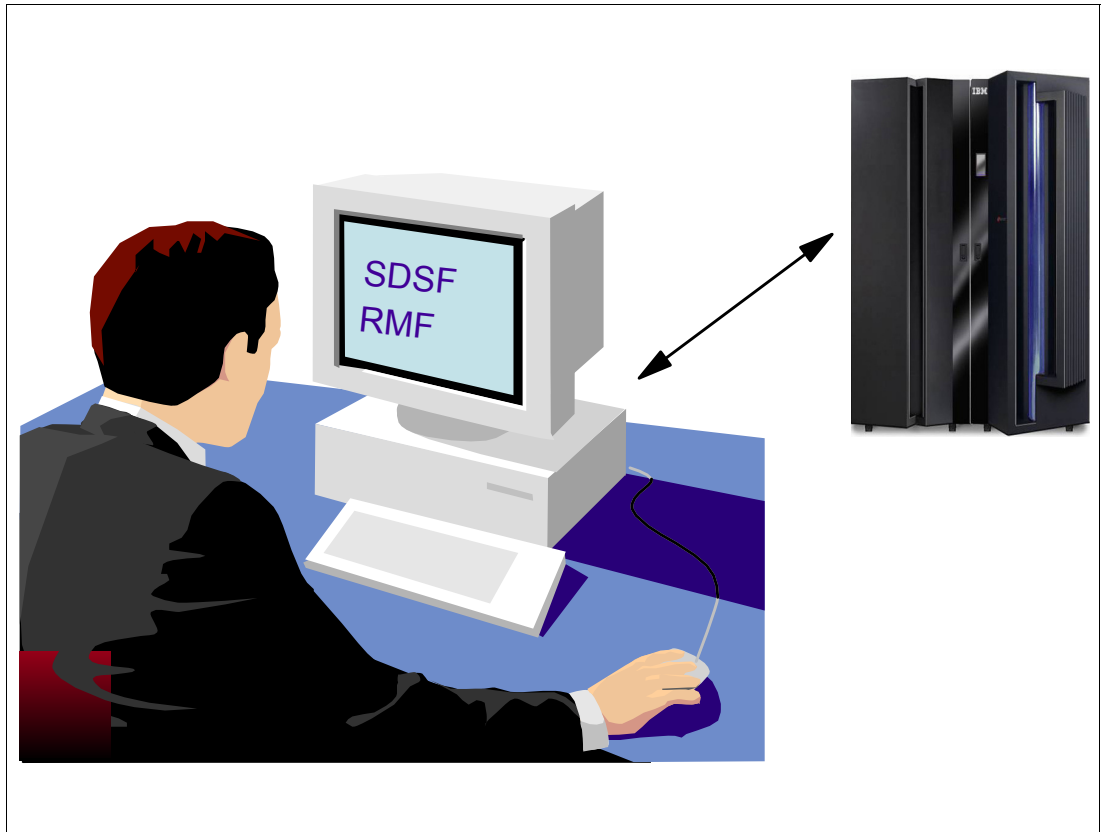


Figure 1-20 Problem determination with SDSF

System Display and Search Facility (SDSF)

SDSF is a program that runs on TSO/E and uses Interactive System Productivity Facility (ISPF) panels.

SDSF provides a powerful and secure way to monitor, manage and control your z/OS sysplex. Its easy-to-use interface lets you control the following:

- ▶ Jobs and output
- ▶ Devices, such as printers, readers, lines, and spool offloaders
- ▶ System resources, such as WLM scheduling environments, the members of your JES2 MAS, and JES2 job classes
- ▶ System log and action messages

Resource Measurement Facility (RMF)

RMF is designed to ease the management of single or multiple system workloads and to enable faster reaction to system delays. Detecting a possible bottleneck early means that corrective actions can be taken earlier. System delays are avoided or at least remedied at an early stage.

Using RMF for problem analysis

Use output from RMF, SMF, or another system monitoring program to look for problems. Find someone in your installation who is familiar with the program and can interpret the output. Some of the kinds of problems you should look for are:

- ▶ A program using a lot of storage, whether it is real, virtual, auxiliary or extended storage
- ▶ Data set contention
- ▶ ENQ contention
- ▶ Tuning problems
- ▶ System running over capacity



Problem resolution steps

As a system programmer the important part of your job is to keep your system running and avoid application slowdowns or outages. If an error or problem occurs you should be able to collect all necessary information and documentation to fix it or to ask for assistance providing the collected documentation. If you need IBM support you should provide also a severity indication depending on the system impact. You should be able to find a search argument according to the error information to check for known problems or calling IBM support center.

The following problem resolution steps provide a debug guideline:

- ▶ Identifying and document the problem
- ▶ Prioritize the problem
- ▶ Analyze the problem and ask for assistance if necessary
- ▶ Implement the resolution and close the problem

2.1 Identifying a problem

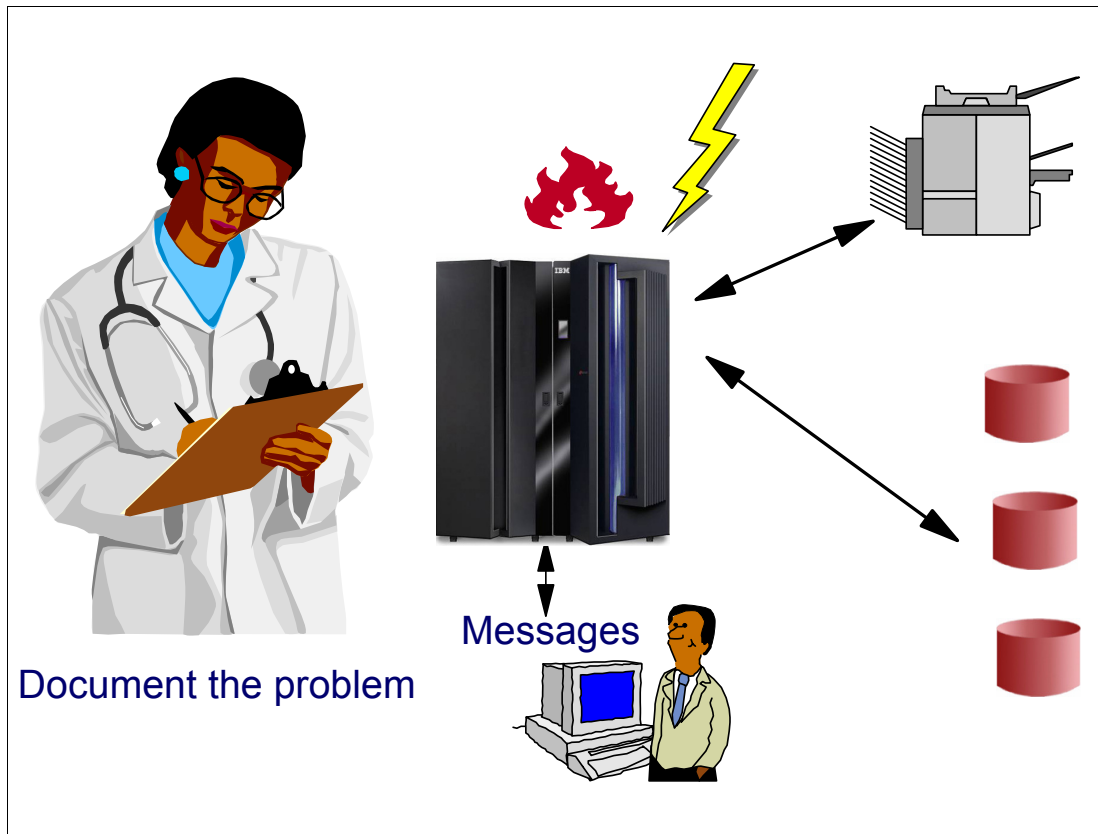


Figure 2-1 Identifying a problem

What caused the problem

Depending on the system or application impact in case of an error the most important questions you must ask include:

- ▶ Is the process that is causing the problem a new procedure, or has it worked successfully before?
- ▶ If it was an existing procedure that was previously successful, what has changed?
- ▶ What messages are being generated that could indicate what the problem is? These could be presented on the terminal if the process is conversational, or in the batch or subsystem job log, or in the system log (SYSLOG).

Note: Review the *z/OS MVS System Messages*, SA22-763x and *z/OS MVS Systems Codes*, SA22-7626 manuals.

- ▶ Can the failure be reproduced, and if so what steps are being performed?
- ▶ Has the failing process generated a dump?

All of these questions will enable you to develop an appropriate plan of action to aid with the resolution. You can never be criticized for providing too much diagnostic data, but too little information only delays the solving of the problem.

Document the problem

Documentation of the problem and the analysis steps taken can assist with not only initial resolution, but will also assist if the problem occurs again. For larger more complex problems regular documentation during the analysis process can highlight areas that will become more crucial as the investigation progresses. This will enable you to develop a flow chart and reference point listing that can be referred to throughout your analysis. Document the final resolution for future reference.

Identifying the problem

A system problem can be described as any problem on your system that causes work to be stopped or degraded. The steps involved in diagnosing these problems are different for each type of problem.

Before you can begin to diagnose a system problem, however, you have to know what kind of problem you have. To identify a system problem, look at the following:

- ▶ System processing witnessed by the operator.
- ▶ The dump, in which the system records information about the system problem. It is important to remember that the error triggering a dump might be a symptom itself, and the information needed to diagnose the root cause might not be captured in that dump. Depending on what type of dump the system or the operator takes, you can determine the type of system problem you need to diagnose and whether you will need to collect additional data.
- ▶ The logrec data set, which contains a history of the errors encountered by the system.
- ▶ The console log.

2.2 Prioritize problem resolution

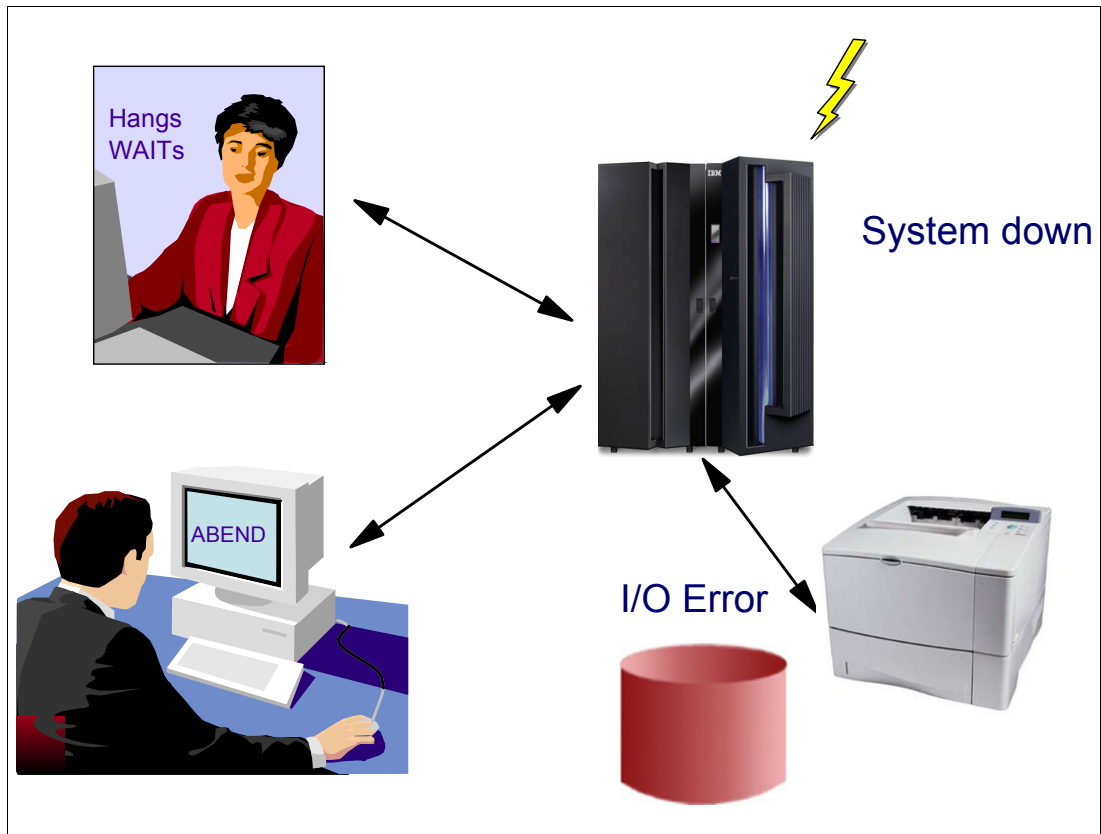


Figure 2-2 Prioritize problem resolution

Prioritize problem resolution

Your prime objective as a system programmer is to ensure system availability, and in the event of a major subsystem failure, for example, a Customer Information Control System (CICS) failure, or worse still the whole z/OS system, your focus will be on the speedy restoration of the system.

Subsystem failures will often generate their own diagnostic data, and the recovery process is often fairly straightforward. These systems will generally perform cleanup processes during recovery and thereby restore system availability. If the subsystem fails during recovery, then immediate problem analysis and resolution will be required.

System down

The worst-case scenario is that your complete z/OS system is down. Swift system recovery is required, but a decision must be made to determine whether the currently preserved main storage should be dumped via a stand-alone dump routine prior to the recovery Initial Program Load (IPL). The IPL process clears main storage; therefore, any failure information will be lost. The stand-alone dump process will take some time but could be extremely valuable should the problem reoccur.

System programmer actions

Depending on the nature of the problem, system programmers can take actions related to the type of problem that has occurred.

- | | |
|---------------------|---|
| Abend | Review the dump to determine if further diagnosis is required.

Review system messages to determine the abend's impact on the installation. |
| Hang or WAIT | Use the DUMP command to obtain an SVC dump. If the SVC dump does not provide the necessary information, ask the operator to take a stand-alone dump.

Have the operator check to see whether the system console is responsive. If it is not, take a stand-alone dump. If it is, take an SVC dump of the user's address space. |
| I/O error | Check for messages indicating I/O errors. |

2.3 Problem severity

- ❑ IBM Support Center
 - Severity of problem - report to IBM
- ❑ Four severity levels
 - Severity 1 (SEV 1)
 - Severity 2 (SEV 2)
 - Severity 3 (SEV 3)
 - Severity 4 (SEV 4)

Figure 2-3 Reporting the severity of a problem to the IBM Support Center

Report problems to IBM

When you need to report a problem to the IBM Support Center, you will be asked what the severity of the problem is. We set severity from SEV-1 (highest severity, meaning worst problems) to SEV-4 (lowest severity, meaning least important problems). It's important to be realistic when reporting the severity of an issue, so we can prioritize it properly.

Severity 1 (SEV 1)

Production system down, critical business impact, unable to use the product in a production environment, no workaround is available.

Severity 2 (SEV 2)

Serious problem that has a significant business impact; use of the product is severely limited, but no production system is continuously down. SEV-2 problems include situations where customers are forced to restart processes frequently, and performance problems that cause significant degradation of service but do not render the product totally unusable. In general, a very serious problem for which there is an unattractive but functional workaround would be SEV-2, not SEV-1.

Severity 3 (SEV 3)

Problems that cause some business impact but that can be reasonably circumvented; situations where there is a problem but the product is still usable. For example, short-lived problems or problems with components that have failed and then recovered and are back in

normal operation at the time the problem is being reported. The default severity of new problem reports should be SEV-3.

Severity 4 (SEV 4)

This severity is for minor problems that have minimal business impact. While we are all aware of the pressure that customers and management place on the speedy resolution of their problems, the correct problem severity enables all involved support teams to react to and manage the problems according to the “real” severity of the problem. While a “customer is unhappy SEV1” is in many cases valid for business reasons, it does not preclude the fact that a customer with a “production system down SEV1” is more important.

2.4 Analyze a problem - ask for assistance

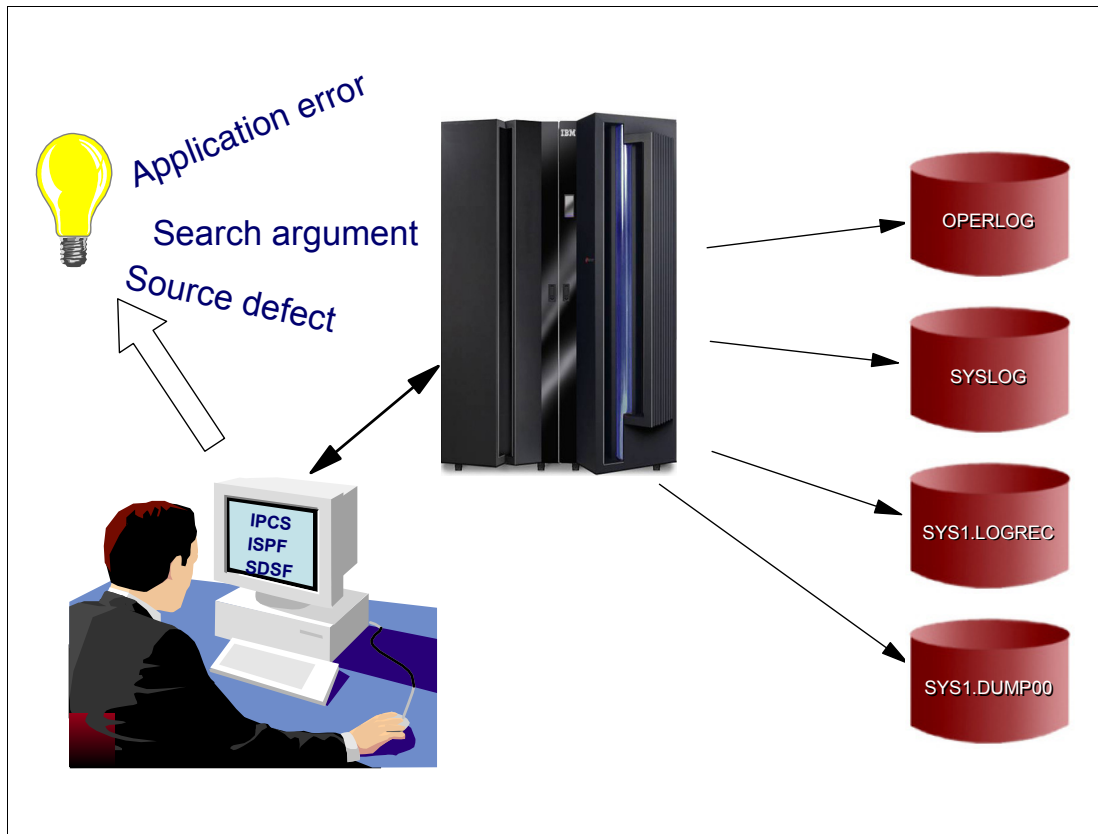


Figure 2-4 Analyze a problem

Analyze the problem

Before you start the process of what could be described as the more complex analysis procedures, you should review all of the data you currently have that may solve your problem. Have you:

1. Looked in the system log for any relevant messages or abend information?
2. Looked in the job log for any relevant messages or abend information?
3. Reviewed the meanings of any messages or codes in the relevant manuals?
4. Reviewed the system error log, SYS1.LOGREC, which contains information about hardware and software failures?

Problem analysis

Problem analysis is, like any process, a skill that develops the more you use it. Of course, problems vary in their complexity and frequency, and it is possible that tasks requiring this type of review may be infrequent in your environment. The ultimate aim is to have little need to perform complex problem diagnosis. This is why a sound methodology is necessary to assist with your analysis.

It is necessary to retain a focus during the analysis process and be aware that there are often alternative ways to approach a problem. To ask for assistance with a problem is not a sign of failure, but an indication you are aware that another person's views could speed up the resolution. A fresh idea can often stimulate and enhance your thought processes.

Solving a problem

Solving a problem is a combination of:

1. Your ability to understand the problem.
2. The quality of the diagnostic data you can review.
3. Your ability to use the diagnostic tools at your disposal.

Ask for assistance

You will hopefully be aware that some assistance may be required when you are making little progress with your diagnosis. What you and your manager are seeking is a speedy resolution, and it is far more professional to use all the facilities and technical expertise available. The IBM Support Center is there to assist you with your problems and the diagnostic data you have collected, and the analysis steps you have already performed will be of great help to the Support Center when they review the problem.

2.5 Gather Messages and Logrec

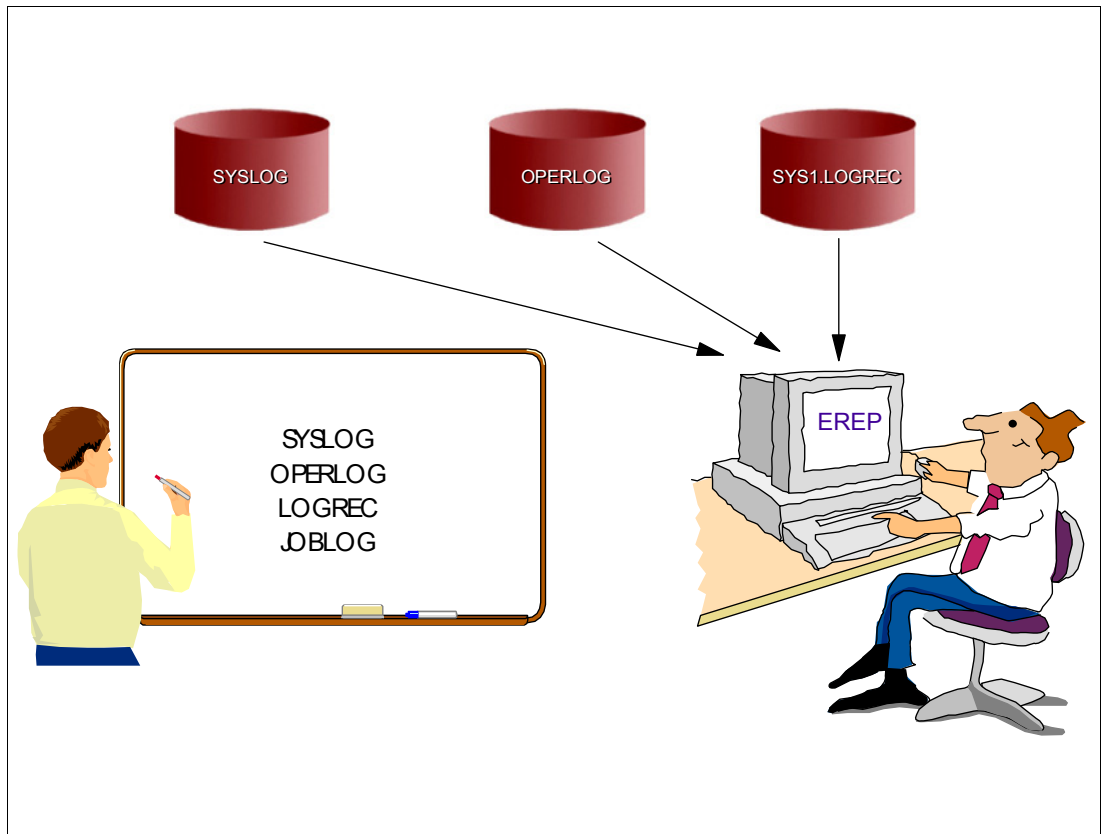


Figure 2-5 Gather messages and logrec

Gathering information

Often, the most readily available source of data identifies the key piece of information that will resolve the problem, and often, this source of data is overlooked. The first places to look when reviewing a problem are:

- ▶ The console log
- ▶ The system log
- ▶ An error log related to a specific product
- ▶ The whole system

While a system dump or a trace is often required, the logs may provide enough detail to solve the problem. The location of the relevant logs varies from product to product, and system to system.

Collect and analyze messages and logrec records about the problem. Look at any messages or software, symptom, and hardware records for logrec around the time of the problem.

Diagnostic data sources

The main sources of diagnostic data are contained in the messages provided by the system in the following logs:

► Console log

Messages sent to a console with master authority are intended for the operators. The system writes in the hard-copy log all messages sent to a console, regardless of whether the message is displayed.

► SYSLOG

The SYSLOG is a SYSOUT data set provided by the job entry subsystem (either JES2 or JES3). SYSOUT data sets are output spool data sets on direct access storage devices (DASD). An installation should print the SYSLOG periodically to check for problems. The SYSLOG consists of the following:

- All messages issued through WTL macros
- All messages entered by LOG operator commands
- Usually, the hard-copy log
- Any messages routed to the SYSLOG from any system component or program

► Job log

Messages sent to the job log are intended for the programmer who submitted a job. Specify the system output class for the job log in the MSGCLASS parameter of the JCL JOB statement.

► OPERLOG

Operations log (OPERLOG) is an MVS system logger application that records and merges messages about programs and system functions (the hardcopy message set) from each system in a sysplex that activates OPERLOG. Use OPERLOG rather than the system log (SYSLOG) as your hardcopy medium when you need a permanent log about operating conditions and maintenance for all systems in a sysplex.

► Hard-copy log

The hard-copy log is a record of all system message traffic:

- Messages to and from all consoles
- Commands and replies entered by the operator

In a dump, these messages appear in the master trace. With JES3, the hard-copy log is always written to the SYSLOG. With JES2, the hard-copy log is usually written to the SYSLOG but can be written to a console printer, if the installation chooses.

► Logrec

Logrec log stream is an MVS System Logger application that records hardware failures, selected software errors, and selected system conditions across the sysplex. Using a logrec log stream rather than a logrec data set for each system can streamline logrec error recording.

2.6 SYSLOG processing

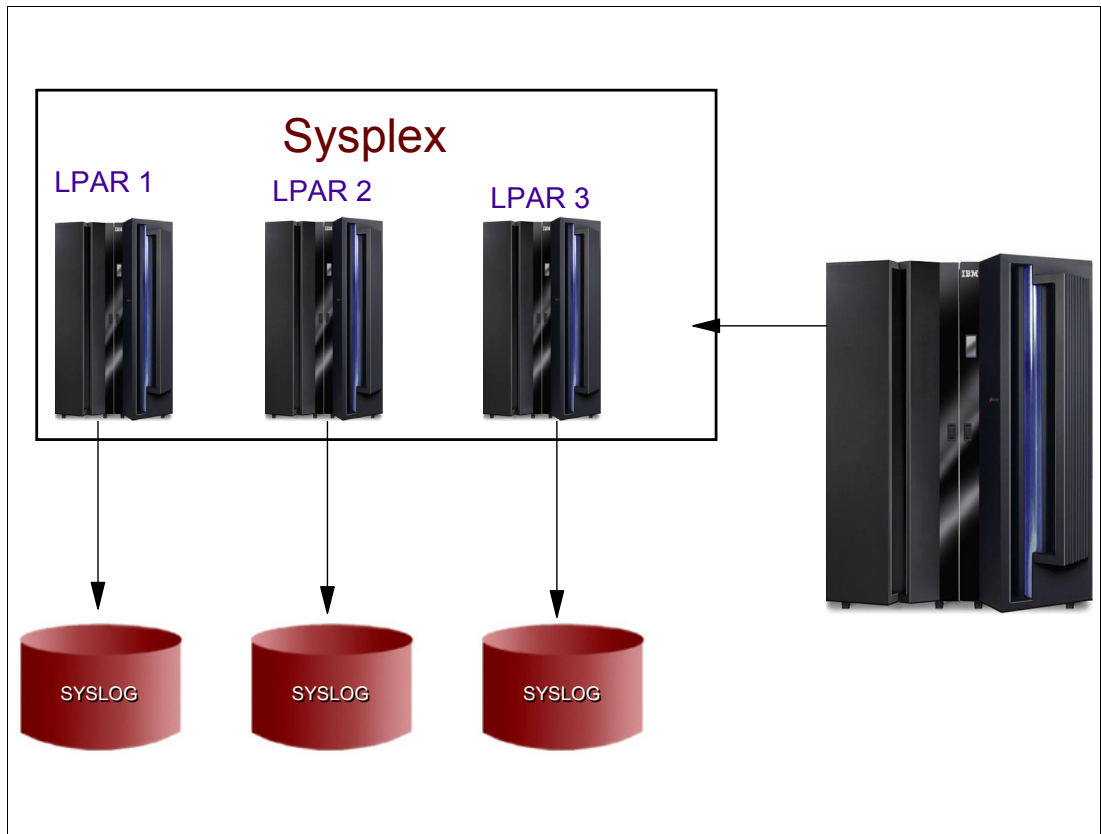


Figure 2-6 SYSLOG processing

SYSLOG processing

The system log (SYSLOG) is a direct access data set that stores messages and commands. It resides in the primary job entry subsystem's spool space. It can be used by application and system programmers (through the WTL macro) to record communications about programs and system functions. You can use the LOG command to add an entry to the system log.

Several kinds of information can appear in the system log:

- ▶ Job time, step time, and data from the JOB and EXEC statements of completed jobs entered by user-written routines
- ▶ Operating data entered by programs using a write to log (WTL) macro instruction
- ▶ Descriptions of unusual events that you enter using the LOG command
- ▶ The hardcopy message set

On z/OS, the SYSLOG can be viewed via the Spool Display and Search Facility (SDSF) using the LOG option. A small amount of the SYSLOG is also stored in memory and is included when an address space is dumped. This is referred to as master trace (MTRACE) data and can be accessed via the IPCS using the VERBX MTRACE command.

2.7 SYSLOG messages

```
M 0020000 SC70      2005185 19:40:01.33 RMFGAT  00000090
E                                     808 00000090
NR4000000 SC70      2005185 19:40:01.97 RMFGAT  00000090
M 0020000 SC70      2005185 19:41:00.43 STC27300 00000090
E                                     139 00000090
NR4000000 SC70      2005185 19:41:00.93 STC27300 00000090
N FFFF000 SC70      2005185 19:44:35.16 MQSCCHIN 00000090
N FFFF000 SC70      2005185 19:44:35.19 MQSCCHIN 00000090
S
N FFFF000 SC70      2005185 19:44:35.21 MQSCCHIN 00000090
S
-----
IEC070I 203-204,RMFGAT,RMFGAT,SYS00753,3E14,SBOX01,RMF3.SC70.B, 808
IEC070I RMF3.SC70.B.DATA,UCAT.VSBOX01
ERB813I III: ACTIVE MONITOR III DATA SET IS NOW 'RMF3.SC70.A'
IEC070I 203-204,RMFGAT,RMFGAT,SYS00882,3E14,SBOX01,RMF3.SC70.B, 139
IEC070I RMF3.SC70.B.DATA,UCAT.VSBOX01
ERB813I III: ACTIVE MONITOR III DATA SET IS NOW 'RMF3.SC70.A'
+CSQX449I =MQSC CSQXREPO Repository manager restarted
+CSQX037E =MQSC CSQXREPO Unable to get message from
SYSTEM.CLUSTER.COM
MAND.QUEUE, MQCC=2 MQRC=2016
+CSQX448E =MQSC CSQXREPO Repository manager stopping because of
errors. Restart in 600 seconds
```

Figure 2-7 Examples of SYSLOG messages

SYSLOG messages

Figure 2-7 shows an example of the MVS SYSLOG. The time stamps that would normally be seen to the left of the data shown in the bottom half of the figure are shown in the top part of the figure.

Message description

A description of the first message in Figure 2-7 follows:

```
M 0020000 SC70      2005185 19:40:01.33 RMFGAT  00000090
M
0020000
SC70      System name that issued the message
2005185   Julian date (Day 185 of year 2005)
19:40:01.33 Time that the message was issued
RMFGAT    Address space name that issued the message
00000090
```

Message IEC070I

The IEC070I message is displayed on the first 2 lines. A description of the first message in Figure 2-7 follows in the bottom part of the figure:

```
IEC070I 203-204, RMFGAT, RMFGAT, SYS00753, 3E14, SBOX01, RMF3.SC70.B, 808
IEC070I RMF3.SC70.B.DATA, UCAT.VSBOX01
-----Message description-----
IEC070I rc[{sfi}]- ccc, jjj, sss, ddname, dev, volser, xxx, dsname, cat
```

IEC070I message description

Explanation: An error occurred during EOVS (end-of-volume) processing for a VSAM data set.

In the message text:

- ▶ 203 is the return code (rc). This field indicates the specific cause of the error. For an explanation of this return code, see message IEC161I.
 - sfi is the subfunction information (error information returned by another subsystem or component). This field appears only for certain return codes, and its format is shown with those codes to which it applies. When a catalog LOCATE request fails, this field appears for return code 032 or 034.
- ▶ 204 is a problem-determination function (PDF) code. The PDF code is for use by the IBM Support Center if further problem determination is required. If the PDF code has meaning for the user, it is documented with the corresponding reason code (rc).
- ▶ RMFGAT (ccc) is the job name.
- ▶ RMFGAT (sss) is the step name. If the step is part of a procedure, this field contains an eight-character procedure step name, with trailing blanks, followed by the name of the job step that called the procedure, without trailing blanks. The two names are not separated by a comma.
- ▶ SYS00753 (ddname) is the data definition (DD) name.
- ▶ 3E14 (dev) is the device number, if the error is related to a specific device.
- ▶ SBOX01(volser) is the volume serial number, if the error is related to a specific volume.
- ▶ RMF3.SC70.B (xxx) is the name of the cluster that contained the data set being processed when the error was detected, or when not available, the data set name specified in the DD statement indicated in the access method control block (ACB).
- ▶ RMF3.SC70.B.DATA (dsname) is the name of the data set being processed when the error was detected.
- ▶ UCAT.VSBOX01 (cat) is the catalog name.

System programmer response

If the error recurs and the program is not in error, look at the messages in the job log for more information. Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all printed output and output data sets related to the problem.

2.8 OPERLOG (operations log)

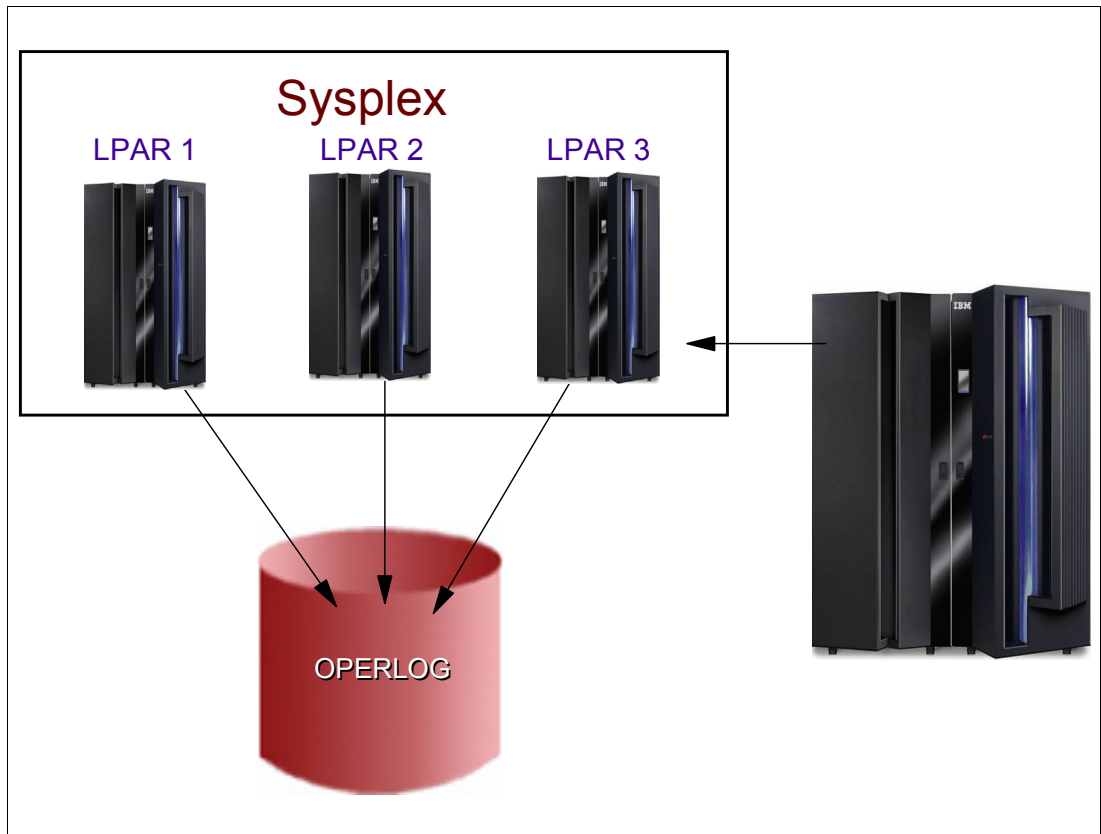


Figure 2-8 OPERLOG processing

OPERLOG

The operations log (OPERLOG) is a log stream that uses the system logger to record and merge communications about programs and system functions from each system in a sysplex. The operations log is operationally independent of the system log. An installation can choose to run with either or both of the logs. If you choose to use the operations log as a replacement for SYSLOG, you can prevent the future use of SYSLOG.

You can use the operations log (OPERLOG) to record messages and commands from all the systems in a sysplex. The operations log centralizes log data in a sysplex. The OPERLOG panel displays the data from a log stream, a collection of log data used by the MVS System Logger to provide the merged, sysplex-wide log.

OPERLOG message

Following is a message from the OPERLOG—the same message described in Figure 2-7 on page 37.

```
M 0020000 SC70      2005185 19:40:01.33 RMFGAT  00000090
E                                     808 00000090
-----
IEC070I 203-204,RMFGAT,RMFGAT,SYS00753,3E14,SB0X01,RMF3.SC70.B
IEC070I RMF3.SC70.B.DATA,UCAT.VSBOX01
```

2.9 Job error logs

NP	DDNAME	StepName	ProcStep	DSID	Owner
	JESJCLIN			1	CICSTS
	JESMSG LG	JES2		2	CICSTS
	JESJCL	JES2		3	CICSTS
	JESYSMSG	JES2		4	CICSTS
	\$INTTEXT	JES2		5	CICSTS
	CAFF	SCSCPAA1		101	CICSTS
	CINT	SCSCPAA1		103	CICSTS
	DFHCXRF	SCSCPAA1		104	CICSTS
	COUT	SCSCPAA1		105	CICSTS
	CEEMSG	SCSCPAA1		106	CICSTS
	CEEOUT	SCSCPAA1		107	CICSTS
	PLIMSG	SCSCPAA1		108	CICSTS
	CRPO	SCSCPAA1		109	CICSTS
	MSGUSR	SCSCPAA1		110	CICSTS

Figure 2-9 Display of CICS SYSOUT data sets obtained with the SDSF DA operand

Job error log data sets

Each individual product has its own log file on the z/OS platform that may contain data that may be valuable when diagnosing a problem. It is particularly important to look for events that precede that actual failure, because the problem, in many cases, will have been caused by a previous action. Figure 2-9 shows the SYSOUT data sets that might be associated with a CICS address space.

The key SYSOUT data sets to review that may provide problem determination data are:

JESMSG LG and MSGUSR

The following data sets will contain Language Environment (LE) problem data usually associated with application problems:

CEEMSG and CEEOUT

MSGUSR data set

Figure 2-10 on page 41 shows an example of some transaction abend data included in the MSGUSR SYSOUT data set.

```
DFHIR3783 04/11/2005 01:25:50 SCSCPTA2 Transaction SX2 termid E39 -  
Connected transaction abended with message DFHAC2206 01:25:50 SCSCPAA4  
Transaction SX2  
failed with abend AFCV. Updates to local recoverable resources backed out.  
DFHAC2236 04/11/2005 01:25:50 SCSCPTA2 Transaction SX2 abend AZI6 in program  
*UNKNOWN  
term PB09. Updates to local recoverable resources will be backed out.  
DFHAC2262 04/11/2005 01:25 (sense code 0824089E).  
DFHAC2206 01:25:50 SCSCPAA4 Transaction SX2 failed with abend AFCV.  
Updates to local recoverable resources backed out.
```

Figure 2-10 CICS MSGUSR SYSOUT data set sample data

JESMSGGLG data set

The CICS JESMSGGLG SYSOUT data set includes information related to CICS startup and errors related to system problems, not specifically transaction related. Figure 2-11 is a sample taken from the CICS JES Message Log (JESMSGGLG).

```

+DFHTR0103 TRACE TABLE SIZE IS 64K
+DFHSM0122I SCSCPTA2 Limit of DSA storage below 16MB is 5,120K.
+DFHSM0123I SCSCPTA2 Limit of DSA storage above 16MB is 60M.
+DFHSM0113I SCSCPTA2 Storage protection is not active.
+DFHSM0126I SCSCPTA2 Transaction isolation is not active.
+DFHSM0120I SCSCPTA2 Reentrant programs will not be loaded into read-only
storage
+DFHDM0101I SCSCPTA2 CICS is initializing.
+DFHXS1100I SCSCPTA2 Security initialization has started.
+DFHXB0109I SCSCPTA2 Web domain initialization has started.
+DFHS00100I SCSCPTA2 Sockets domain initialization has started.
+DFHRX0100I SCSCPTA2 RX domain initialization has started.
+DFHRX0101I SCSCPTA2 RX domain initialization has ended.
+DFHLG0101I SCSCPTA2 Log manager domain initialization has started.
+DFHEJ0101 SCSCPTA2 291
Enterprise Java domain initialization has started. Java is a
trademark of Sun Microsystems, Inc.
+DFHDB0100I SCSCPTA2 Document domain initialization has started.
.
+DFHLG0103I SCSCPTA2 System log (DFHLOG) initialization has started.
+DFHLG0104I SCSCPTA2 340
System log (DFHLOG) initialization has ended. Log stream
***** is connected to structure
*****.
+DFHSI1519I SCSCPTA2 The interregion communication session was successfully
started
+DFHXB1007 SCSCPTA2 Initializing CICS Web environment.
+DFHXB1008 SCSCPTA2 CICS Web environment initialization is complete.
+DFHSI8430I SCSCPTA2 About to link to PLT programs during the third stage of
initialization
+EYUNX0001I SCSCPTA2 LMAS PLTPI program starting
+EYUXL0003I SCSCPTA2 CPSM Version 220 LMAS startup in progress
+EYUXL0103E SCSCPTA2 CICSplex SM subsystem (EYUX) not active
+EYUXL0024I SCSCPTA2 Waiting for CICSplex SM subsystem activation

```

Figure 2-11 CICS JESMSGLG output

2.10 Logrec data set

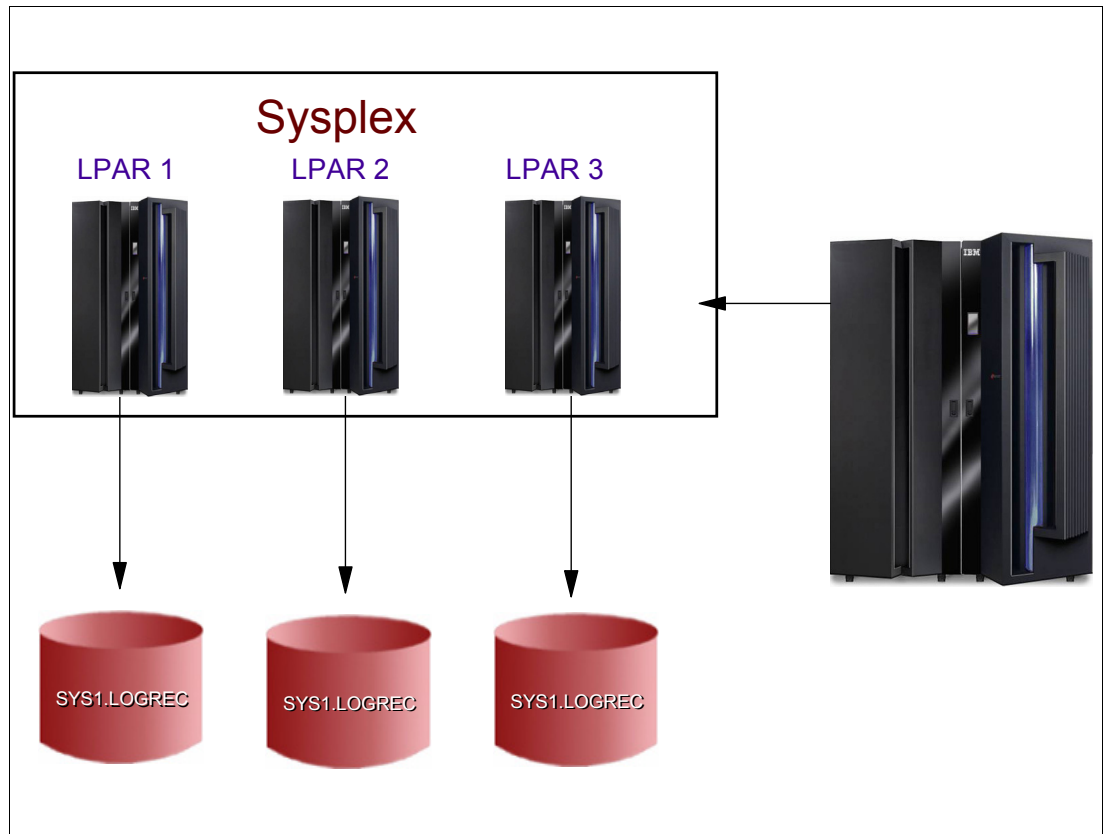


Figure 2-12 SYS1.LOGREC data sets

Logrec data set

The z/OS error log contains data related to hardware and software errors. This data is written to the SYS1.LOGREC data set and is also written to internal storage that is included in a dump. The SYS1.LOGREC data set can be interrogated using the ICFEREP1 program, or if the abend has triggered a dump, the EREP data can be reviewed using the IPCS VERBX LOGDATA command.

Figure 2-13 on page 44 shows the last error record contained in the error log generated when the VERBX LOGDATA command was issued for a dump being reviewed using IPCS. Generally, the error log entries at the end of the display, if they have an influence on the problem being reviewed, will have time stamps that relate to (or immediately precede) the actual abend.

```

JOBNAME: ITSOCIOI  SYSTEM NAME: SC48
ERRORID: SEQ=05462  CPU=0042  ASID=00CE  TIME=15:03:28.1

SEARCH ARGUMENT ABSTRACT

  PIDS/5740XYR00 RIDS/DSNXGRDS#L RIDS/DSNXRIVB AB/S00C7 PRCS/00000000
REGS/OCB2C
  REGS/B6B67 RIDS/DSNTFRCV#R

SYMPTOM          DESCRIPTION
-----          -
PIDS/5740XYR00   PROGRAM ID: 5740XYR00
RIDS/DSNXGRDS#L  LOAD MODULE NAME: DSNXGRDS
RIDS/DSNXRIVB    CSECT NAME: DSNXRIVB
AB/S00C7         SYSTEM ABEND CODE: 00C7
PRCS/00000000    ABEND REASON CODE: 00000000
REGS/OCB2C       REGISTER/PSW DIFFERENCE FOR R0C: B2C
REGS/B6B67       REGISTER/PSW DIFFERENCE FOR R0B: -6B67
RIDS/DSNTFRCV#R  RECOVERY ROUTINE CSECT NAME: DSNTFRCV

OTHER SERVICEABILITY INFORMATION

DATE ASSEMBLED:      01/29/04
MODULE LEVEL:        UQ84577
SUBFUNCTION:          RDS  SQL    DIAGNOSE

SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

  RECOVERY ROUTINE LABEL

TIME OF ERROR INFORMATION

  PSW: 077C1000 9E43EDFC  INSTRUCTION LENGTH: 04  INTERRUPT CODE: 0007
  FAILING INSTRUCTION TEXT: D5244420 B0219680 D5245820

```

Figure 2-13 Final record in logrec data from IPCS VERBX LOGDATA

Note: Do not ignore the valuable data that is written to the log files.

2.11 Analyzing EREP reports

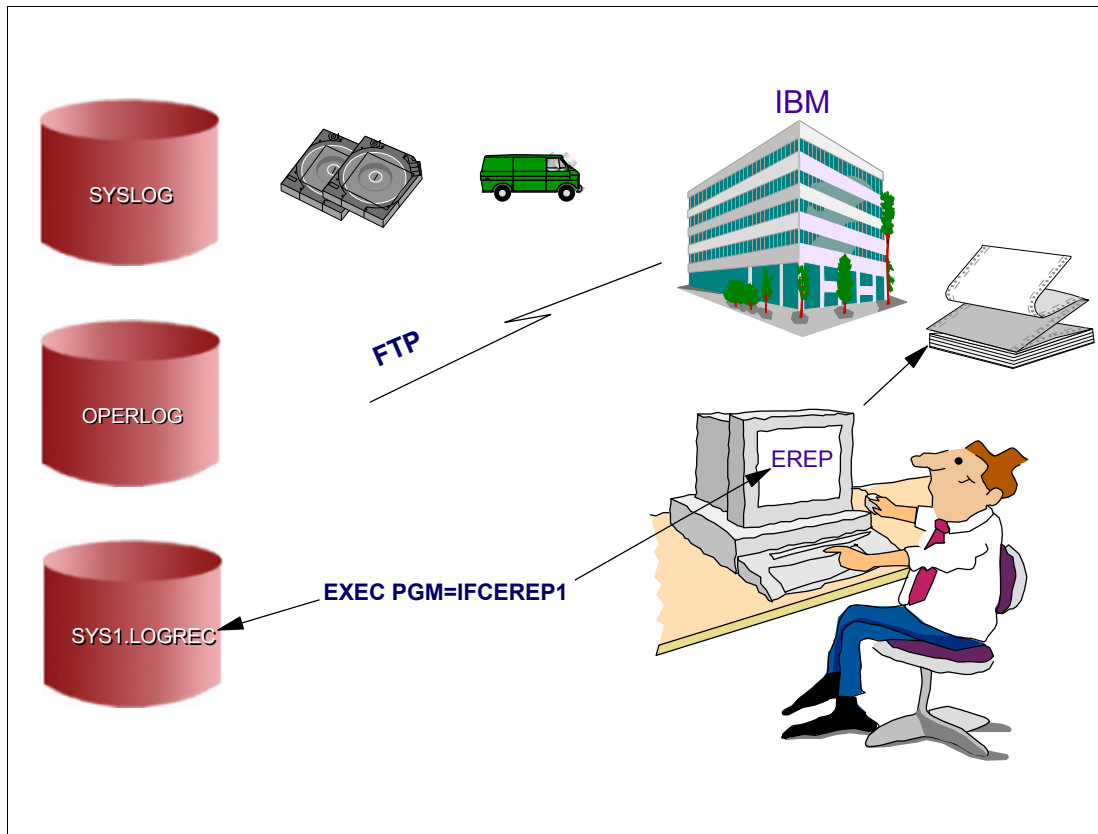


Figure 2-14 Gather messages and logrec

Environmental Record Editing and Printing Program (EREP)

The Environmental Record Editing and Printing Program (EREP) is a diagnostic application program that runs under the MVS, VM, and VSE operating systems. The purpose of EREP is to help IBM service representatives maintain your data processing installation. EREP edits and prints reports from the records placed in the error recording data set (ERDS) by the error recovery program (ERP) of your operating system. Some of these records are the result of device or system errors, while others are informational or statistical data. The service representative analyzes information in the EREP reports to determine whether a problem exists, what the problem is, and where the problem is located.

What EREP does

EREP processes the error records from your operating system to produce formatted reports. These EREP reports can show the status of the entire installation, an I/O subsystem, or an individual device, depending upon which report you request.

Important: EREP is a service tool that shows statistical data that helps your IBM service representative determine whether a problem is media related or hardware related.

1. EREP edits and prints records that already exist; it does not create the error records.
2. EREP is not designed to automate media maintenance or library management.

2.12 Using EREP

- ❑ EREP report types
 - System summary
 - Trends
 - Event history
 - System exception
 - Threshold summary
 - Detail edit and summary
- ❑ EREP records
 - Software and hardware
- ❑ Stages for building EREP records
- ❑ Setting EREP environment

Figure 2-15 Establishing an EREP environment

EREP report types

EREP reports vary in format depending on types shown in Table 2-1.

Table 2-1 EREP report types

Report Type	Format
System summary	Error data in summary form
Trends	Error data by daily totals
Event history	Error data in a time sequence by occurrence
System exception	The system exception series is a series of reports that list software and hardware error data in a variety of ways to help you identify problems within your subsystems.
Threshold summary	The threshold summary report shows all the permanent read/write errors, temporary read/write errors, and media statistics for each volume mounted.
Detail edit and summary	The detail edit and summary reports provide environmental information, hexadecimal dumps and summaries of errors to determine their nature and causes.

EREP records

Your operating system with its hardware and software captures statistical and error data, such as:

- ▶ A read error on a direct access device or tape volume
- ▶ A machine check on a processor
- ▶ An IPL of the operating system

Processing EREP data records

The system procedure executing EREP issues commands to write the buffered statistical data from the system-attached devices to the ERDS (error recording data set). The system ERP (error recovery program/processing) builds the records in the stages shown in Table 2-2.

Table 2-2 EREP processing stages

Stage	Action
1	The devices attached to the operating system generate sense data for the events encountered during the day. The sense data can be informational, error-related, or statistical.
2	The ERP of the operating system looks at the sense data. If the sense data indicates that a record should be built, the ERP takes the sense data and places it after the standard header information. The combination of the header information and the sense data becomes the error record.
3	The operating system ERP writes the records onto the system ERDS.

Setting up and running EREP

See *Environmental Record Editing and Printing Program (EREP) User's Guide*, GC35-0151 for the general guidelines for invoking and running EREP.

2.13 EREP reports

❑ Generating EREP reports

- Overview reports
- Analysis reports
- Detail reports

```
//EREPPRNT JOB ,ESTER,  
// MSGCLASS=T,NOTIFY=C961231,USER=C961231  
//\-----\/  
//\ STEP0: COPIES SYS1.LOGREC TO TEMPORARY DATA SET \/  
//\-----\/  
//S0 EXEC PGM=IFCEREPL,REGION=1024K,  
// PARM='ACC,ZERO=N'  
//SERLOG DD DISP=(OLD,KEEP),DSN=SYS1.LOGREC  
//ACCDEV DD DISP=(NEW,PASS),DSN=*&ERRDATA,  
// UNIT=SYSDA,SPACE=(CYL,(2,2)),  
// DCB=(RECFM=VB,BLKSIZE=6144)  
//DIRECTWK DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)  
//EREPPPT DD SYSOUT=A,DCB=BLKSIZE=133  
//TOURIST DD SYSOUT=A,DCB=BLKSIZE=133  
//SYSIN DD DUMMY
```

Figure 2-16 Generating EREP reports

EREP reports

EREP reports are designed to give you a variety of views of the data being processed. EREP produces:

- Overview reports** From which you can determine if there are problems
- Analysis reports** From which you can determine where there are problems
- Detail reports** From which you can determine what the problems are

Generating an EREP report

MVS systems require system controls that create the interface between EREP and the operating system. The following is an example of job control language (JCL) to execute a series of EREP reports as it would appear in a file without the annotation of the more detailed example provided in *Environmental Record Editing and Printing Program (EREP) User's Guide*, GC35-0151.

You run EREP by executing a procedure containing the operating system EREP command and its associated parameter and control statements. You can only request one type of report each time you execute the EREP command for your system. You may produce any number of different type reports by issuing additional EREP commands with the associated parameters and control statements.

Create MVS JCL

Define the input and output data sets using JCL DD statements. The JCL submits the job as a batch job or interactively via TSO. Put the IFCEREP1 program in the JCL EXEC statement. Include the EREP parameters on the EXEC statement or as part of SYSIN in-stream data with the EREP control statements, as shown in Figure 2-17.

```
//EREPPRNT JOB ,ESTER,
// MSGCLASS=T,NOTIFY=C961231,USER=C961231
//\-----\
//\ STEP0: COPIES SYS1.LOGREC TO TEMPORARY DATA SET \
//\-----\
//S0 EXEC PGM=IFCEREP1,REGION=1024K,
// PARM='ACC,ZERO=N'
//SERLOG DD DISP=(OLD,KEEP),DSN=SYS1.LOGREC
//ACCDEV DD DISP=(NEW,PASS),DSN=&&ERRDATA,
// UNIT=SYSDA,SPACE=(CYL,(2,2)),
// DCB=(RECFM=VB,BLKSIZE=6144)
//DIRECTWK DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//EREPPPT DD SYSOUT=A,DCB=BLKSIZE=133
//TOURIST DD SYSOUT=A,DCB=BLKSIZE=133
//SYSIN DD DUMMY
//\
//\-----\
//\ STEP1: PRINTS SYSTEM SUMMARY REPORT \
//\-----\
//S1 EXEC PGM=IFCEREP1,REGION=1024K,
// PARM='HIST,ACC=N,SYSUM'
//ACCIN DD DISP=(OLD,PASS),DSN=&&ERRDATA
//DIRECTWK DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//EREPPPT DD SYSOUT=A,DCB=BLKSIZE=133
//TOURIST DD SYSOUT=A,DCB=BLKSIZE=133
//SYSIN DD DUMMY
//\
//\-----\
//\ STEP2: PRINTS SYSTEM EXCEPTION REPORTS \
//\-----\
//S2 EXEC PGM=IFCEREP1,REGION=1024K,
// PARM='HIST,ACC=N,SYSEXN,TABSIZE=128K'
//ACCIN DD DISP=(OLD,PASS),DSN=&&ERRDATA
//DIRECTWK DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//EREPPPT DD SYSOUT=A,DCB=BLKSIZE=133
//TOURIST DD SYSOUT=A,DCB=BLKSIZE=133
//SYSIN DD DUMMY
//\
//\-----\
```

Figure 2-17 EREP save and report JCL example

2.14 EREP parameter and control statements

- ❑ When submitting JCL for EREP reports, use:
 - EREP report parameters
 - EREP selection parameters
 - EREP processing parameters
 - EREP control statements
 - EREP parameter combinations
- Example of parameter and control statements

```
//STEP1 EXEC PGM=IFCEREP1,PARM='CARD'

        HIST=Y
        ACC=N
        PRINT=PS
        TABSIZE=800K
        ZERO=NO
        ENDPARM
        / *
```

Figure 2-18 Parameter and control statements for EREP reports

Using parameter and control statements

The parameters and control statements can be grouped according to the kinds of information they convey to the EREP program as shown in Figure 2-19 on page 51 and Figure 2-20 on page 51.

These parameter and control statements determine the following:

- ▶ Which report to produce
- ▶ Which records to select for the requested report
- ▶ How to control the processing of error records and report output
- ▶ How to direct EREP processing and supply more information about the system's configuration

This provides organization to the requested reports.

Using PARM=CARD

In the JCL, specify PARM='CARD' and enter the parameters and control statements on the SYSIN statement, as follows:

```
//STEP1 EXEC PGM=IFCEREP1,PARM='CARD'
```

EREP summary report

The system summary report, using the SYSUM parameter, provides an overview of errors for each of your installation's principal parts, or subsystems. SYSUM produces a condensed two-part system summary report of all errors for the principal system elements, such as CPU, channels, storage, SCP, and the I/O subsystem.

```
//HILG1A JOB (7904),MSGLEVEL=(1,1),MSGCLASS=K,REGION=6000K,  
//      NOTIFY=HILG,CLASS=A  
//STEP1 EXEC PGM=IFCEREPI,PARM='CARD'  
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR  
//ACCIN DD DSN=VSA1.EREP.RECCRW,DISP=SHR  
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,5,,CONTIG)  
//EREPT DD SYSOUT=*  
//TOURIST DD SYSOUT=*  
//SYSIN DD *  
HIST=Y  
ACC=N  
SYSUM  
TABSIZE=800K  
ZERO=NO  
ENDPARM  
/*
```

Figure 2-19 EREP summary report

EREP SYSEXN report

The SYSEXN parameter produces a system exception report series covering processors, channels, DASD, optical, and tape subsystems.

```
//HILG1A JOB (7904),MSGLEVEL=(1,1),MSGCLASS=K,REGION=6000K,  
//      NOTIFY=HILG,CLASS=A  
//STEP1 EXEC PGM=IFCEREPI,PARM='CARD'  
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR  
//ACCIN DD DSN=VSA1.EREP.RECCRW,DISP=SHR  
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,5,,CONTIG)  
//EREPT DD SYSOUT=*  
//TOURIST DD SYSOUT=*  
//SYSIN DD *  
HIST=Y  
ACC=N  
SYSEXN  
TABSIZE=800K  
ZERO=NO  
ENDPARM  
/*
```

Figure 2-20 EREP exception report

2.15 Copy logs to tape

Send SYSLOG data set to IBM support center

```
//HILG1A JOB (7904),MSGLEVEL=(1,1),MSGCLASS=K,REGION=6000K,
//          NOTIFY=HILG,CLASS=A
//STEP1 EXEC PGM=IEHINITT
//LABEL1 DD UNIT=(3480,1,DEFER),VOL=(,RETAIN),STORCLAS=NONSMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
LABEL1 INITT SER=SHARK,DISP=REWIND
//GENER1 EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=SYS1.LOG.DATA,DISP=SHR
//SYSUT2 DD DSN=HILG.LOG.DATA,DISP=(,KEEP),
//          DCB=*.SYSUT1,
//          UNIT=3480,LABEL=(1,SL),VOL=(,RETAIN,SER=SHARK),
//          STORCLAS=NONSMS
```

Figure 2-21 JCL to create SYSLOG on tape

SYSLOG to support center

Sometimes it might be necessary to copy log data sets to a tape and send them to IBM or any other support center. The following JCL can be used to label the tape and copy data:

To send data to IBM you don't need the data on a tape. You can send the data using FTP to a server. Ask your support center for the address.

2.16 Implement a resolution

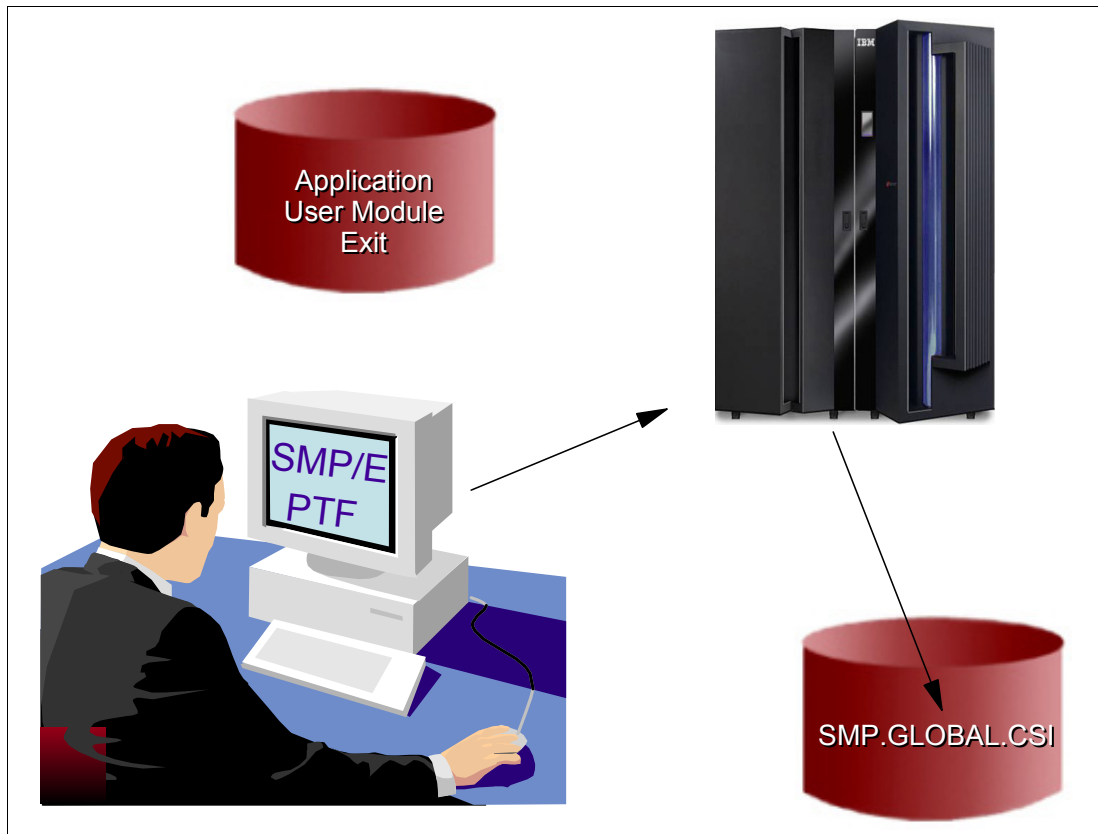


Figure 2-22 Implement a resolution

Implement the resolution

Successful diagnosis of the problem will result in a number of possible resolutions:

- ▶ User Error

This will require you to correct your procedure to ensure a satisfactory resolution is implemented. If your procedure is impacting other users, then prompt action is encouraged.

- ▶ Software implementation error

You must ensure that all installation procedures have been correctly executed and any required customization has been performed correctly. Until you can be sure of a successful implementation, it is advisable to remove this software, or regress to a previous level of the software until more extensive testing can be done in an environment that will not impact production workloads.

- ▶ Software product fault

If the fault is identified as a failure in software a fix might already have been developed to solve this problem. This fix is identified as a Program Temporary Fix (PTF) and will need to be installed into your system. If the problem is causing a major impact, it is suggested that you expedite your normal migration process and promote the fix to the problem system to hopefully stabilize that environment.

If the problem has not been previously reported, an authorized program analysis report (APAR) will be created and a PTF will be generated.

► Hardware fault

This is the resolution that will be controlled by the hardware service representative, but may require some reconfiguration tasks, depending on the nature of the problem. Consultation with the hardware vendor's service representative will clarify the requirements.

Close the problem

When you have tested and implemented the problem resolution, ensure that all parties involved with this problem are informed of the closure of this issue.

It should be noted that during your career you will experience some problems that occur only once, and even with the best diagnostic data, cannot be recreated or solved, by anyone. When this happens there is a point in time where you must accept the fact that this anomaly was in fact just that, an anomaly.



Common problem types

z/OS can process large amounts of work efficiently because it keeps track of storage in a way that makes its storage capacity seem greater than it is. It's a complex system made up of many components, similar to the human body. And, like the human body, z/OS can experience problems that need to be diagnosed and corrected.

The following are examples of problems you might encounter while running z/OS:

- ▶ An abnormal end occurs in processing, known as an abend.
 - Application program abends
 - System program abends
- ▶ A job remains hung in the system.
 - System, subsystem and application hang.
- ▶ The system or process repetitively loops through a series of instructions.
 - System, subsystem and application loop
- ▶ I/O errors.
- ▶ System wait states.
- ▶ Processing slows down.

For system problems, z/OS displays symptoms that will help you with your diagnosis. Problem source identification, called PSI, is the determination of what caused the error. Why was an abend issued? What program is using so much of system storage? What component caused the hang? Which program is looping?

3.1 Common problem types

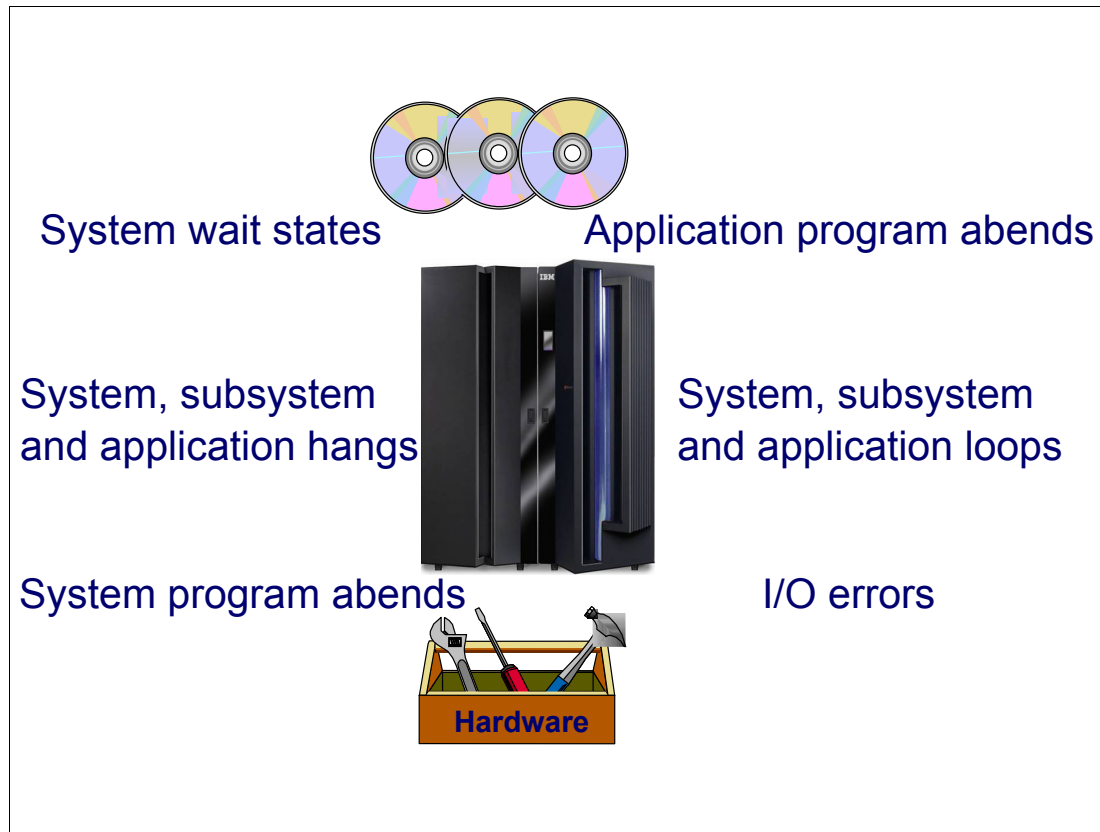


Figure 3-1 Common problem types

Application program abends

Application program abends are always accompanied by messages in the system log (SYSLOG) and the job log indicating the abend code and usually a reason code. Many abends also generate a symptom dump in the SYSLOG and job log. A symptom dump is a system message, either message IEA995I or a numberless message, which provides some basic diagnostic information for diagnosing an abend. Often the symptom dump information can provide enough information to diagnose a problem.

For a system-detected problem, the system abnormally ends a task or address space when it determines that the task or address space cannot continue processing and produce valid results.

System program abends

Like application program abends, system program abends are usually accompanied by messages in the system log (SYSLOG), and if there is a SYS1.DUMPxx data set available or dynamic dump data set allocation at the time of the abend, and this dump code was not suppressed by the dump analysis and elimination (DAE) facility, then an SVC dump will be taken. SVC dumps will be discussed later in this chapter.

I/O errors

I/O errors are most likely caused by a hardware failure or malfunction. The visible symptom will be an abend, accompanied by messages in the SYSLOG that include reason codes,

which can identify the type of error, and sense data, which will offer more detailed, hardware-specific information.

I/O errors can also be the result of software conditions that create a situation where subsequent operations will appear as I/O errors. This could be the result of a corruption in a data set, or data set directory, and the rectification process may be as simple as redefining the data set.

System wait states

The basic summation of a wait state is: the "machine is dead and will not IPL". You will usually experience this condition during the IPL process, and the disabled wait state code will indicate the problem. The cause is often as simple as the system not being able to find some data that is crucial to the IPL process on the IPL volume. Wait codes are documented in *z/OS MVS System Codes*, SA22-7626.

The types of waits are:

- ▶ Disabled wait with a wait state code - The system issues a wait state code and stops. The operator can see the wait state code on the system console. This wait is called a coded wait state or a disabled wait. There are two types of disabled wait state codes, restartable and non-restartable.

For a non-restartable wait state code, the operator must reIPL the system. For a restartable wait state code, the operator may restart the system.

- ▶ Enabled wait - The system stops processing without issuing a wait state code when the dispatcher did not find any work to be dispatched.

The operator sees a WAIT indicator on the system console, followed by a burst of activity caused by system resources manager (SRM) processing, followed by the WAIT indicator, followed by a burst of activity, and so on. An indication of an enabled wait is a PSW of `X'07xxxxxx xxxxxxxx'`.

A special type of enabled wait is called a *no work wait* or a *dummy wait*.

System hangs and loops

The operator usually takes a stand-alone dump for one of the following types of problems:

- ▶ Disabled wait
- ▶ Enabled wait
- ▶ Loop
- ▶ Partial system hang

3.2 Stand-alone dumps

- ❑ For certain problem types:
 - The stand-alone dump program produces a stand-alone dump of storage
 - Use for:
 - A system that fails
 - The system stops processing
 - The system enters a wait state with or without a wait state code
 - The system enters an instruction loop or hangs
 - The system is processing slowly

Figure 3-2 Conditions for taking stand-alone dumps

Stand-alone dumps

When an operator takes a stand-alone dump, it is important to determine the conditions of the system at the time the dump was taken. Because a stand-alone dump can be requested for various problem types, the collection of problem data is imperative for determining the cause of the error.

The objectives for analyzing the output of a stand-alone dump are:

- ▶ Gather symptom data
- ▶ Determine the state of the system
- ▶ Analyze preceding system activity
- ▶ Find the failing module and component

Determine symptoms

Operational conditions should be determined to understand the exact circumstances that caused the dump to be taken, as follows:

- ▶ Was the system put into a wait state?
- ▶ Were the consoles hung or locked up?
- ▶ Were commands being accepted on the master console without a reply?
- ▶ Was a critical job or address space hung?

3.3 Symptom dump output

- ❑ For system and application program abends
 - Normally a symptom dump is displayed

```
IEA995I SYMPTOM DUMP OUTPUT
  SYSTEM COMPLETION CODE=0C4  REASON CODE=00000004
  TIME=16.44.42  SEQ=00057  CPU=0000  ASID=000C
  PSW AT TIME OF ERROR 078D0000  00006FEA  ILC 4  INTC 04
  ACTIVE LOAD MODULE=ABENDER  ADDRESS=00006FD8
  OFFSET=00000012
  DATA AT PSW 00006FE4 - 00105020  30381FFF  58E0D00C
  GPR 0-3  FD000008  00005FF8  00000014  00FD6A40
  GPR 4-7  00AEC980  00AFF030  00AC4FF8  FD000000
  GPR 8-11 00AFF1B0  80AD2050  00000000  00AFF030
  GPR 12-15 40006FDE  00005FB0  80FD6A90  00006FD8
  END OF SYMPTOM DUMP
```

Figure 3-3 SYMPTOM dump data as shown in the MVS SYSLOG and related job log

Symptom dumps

A symptom dump is a system message, either message IEA995I or a numberless message, that provides some basic diagnostic information for diagnosing an abend. Often the symptom dump information can provide enough data to diagnose a problem.

Symptom dumps appear in the following places:

- ▶ For SYSUDUMP and SYSABEND ABEND dumps: in message IEA995I, which is routed to the job log.
- ▶ For a SYSMDUMP ABEND dump: in message IEA995I in the job log and in the dump header record.
- ▶ For an SVC dump: in the dump header record.
- ▶ For any dump in a Time Sharing Option/Extensions (TSO/E) environment: displayed on the terminal when requested by the TSO/E PROFILE command with the WTPMSG option.
- ▶ In response to a DISPLAY DUMP,ERRDATA operator command, which displays information from SYS1.DUMPxx data sets on direct access.

Symptom dump output

Figure 3-3 shows the symptom dump for an abend X'0C4' with reason code X'4'. This symptom dump shows that:

- ▶ Active load module ABENDER is located at address X'00006FD8'.

- ▶ The failing instruction was at offset X'12' in load module ABENDER.
- ▶ The address space identifier (ASID) for the failing task was X' 000C'.

If the information in a symptom dump is insufficient, you can capture additional dump data by including specific DD statements, as discussed in the following section.

Note: Abend codes starting with U are user abends, and are not issued by z/OS. Any program can issue a user abend. Its meaning is determined by the program. Language Environment (LE) shows these kinds of abends according to the LE option settings because z/OS will not handle them.

3.4 Waits, hangs, and loops

❑ System waits, hang conditions, and program loops

Waiting for resource



Resource owner



Figure 3-4 Wait scenarios

System, subsystem, and application hangs

"Hangs" are usually caused by a task, or tasks, waiting for an event that will either never happen, or an event that is taking an excessive amount of time to occur. If one of the waiting tasks is a fundamental system task, or is holding control of a resource, for example a data set, then other tasks will queue up and wait for the required resource to become available. As more tasks enter the system they will also join the queue until the system eventually stops, or the task causing the contention is cancelled. Unfortunately, by the time the system grinds to a halt, the operating system will no longer process any operator commands, so an IPL will be the only alternative. A system hang is more specifically known as an *enabled wait* state.

Hangs and loops

One of the difficult things to determine is whether a system or subsystem is in a hung or looping state. While the symptoms in many cases are similar, for example, the inability to process other units of work, or transactions; or the inability to get the system or subsystem to accept commands—the key difference is whether there is CPU and EXCP activity that indicates the system is still performing work.

If no other tasks can be dispatched within a subsystem, and the CPU activity is high, often 100%, this is generally a symptom that you have a loop condition. Loops can usually be categorized as either *enabled*, *disabled* or a *spin* loops.

Loops are caused by a program, application, or subsystem attempting to execute the same instructions over and over again. The most severe loop condition causes the task

experiencing the condition to use all available CPU resources, and subsequently no other task is allowed to gain control. The only way to alleviate the problem is to cancel the problem task, or if this is unsuccessful, an IPL is necessary. The three types of loop conditions are:

- Enabled** Enabled loops are usually caused by a programming error, but do not impact other jobs in the system, unless the looping task is a subsystem, which will generally impact the whole system.
- Disabled** Disabled loops will not allow an interrupt to be processed, and are generally identified by continuous 100 percent CPU utilization.
- Spin** Spin loops occur when one processor in a multiple-processor environment is unable to communicate with another processor, or is unable to access a resource being held by another processor.

A CPU entering a disabled loop will often be presented to the operators as a spin loop, where the system will cycle (or spin) through the available CPUs.

There are many tools that can be used to assist with hang or loop problem diagnosis, and many of the system monitoring tools will enable you to interrogate at the transaction or thread level and to cancel or purge the individual unit of work or task associated with the loop.

It is important to remember that the monitoring tools should have a high dispatching priority to enable them to get control when required.

It is good to remember that trace data can be used to assist with loop and hang diagnosis, and even 20 seconds of trace data can help identify a looping sequence and often the associated unit of work or transaction. For example, the CICS Auxtrace facility or CICS internal tracing with all CICS components traced at level 1 and a dump of the suspected problem regions can show via a quick IPCS review the type of problem you are experiencing.

An indication of a dummy wait or no work wait is a PSW of X'070E0000 00000000' and GPRs containing all zeroes. Diagnosis is required for this type of wait only when the system does not resume processing.

Processing slows down

In case of system processing slows—central processor at 100% utilization or a job using a high percentage of central processor storage—use an online monitor such as RMF to determine where the problem originates.

3.5 SLIP command

❑ Types of SLIP commands

- SLIP SET - "Setting a SLIP Trap"
- SLIP MOD - "Modifying an Existing SLIP Trap"
- SLIP DEL - "Deleting an Existing SLIP Trap"

❑ Using SLIP commands

- On a console with MVS master authority
- On a TSO terminal in OPERATOR mode
- In a TSO CLIST
- In an IEACMD00, COMMNDxx, or IEASLPxx parmlib member

Figure 3-5 Using SLIP commands

Types of SLIP commands

The SLIP command controls SLIP (serviceability level indication processing). It is a diagnostic aid that intercepts or traps certain system events and specifies what action to take. Using the SLIP command, you can set, modify, and delete SLIP traps. You must specify SET, MOD, or DEL immediately following SLIP, as shown in Figure 3-5.

SLIP command examples

```
SLIP SET[,options],END - Command for an error event trap (non-PER)
SLIP SET,IF[,options],END - Command for an instruction fetch PER trap
SLIP SET,SBT[,options],END - Command for a successful branch PER trap
SLIP SET,SA|SAS[,options],END - Commands for a storage alteration PER trap
SLIP MOD[,options] - Command to modify an existing trap
SLIP DEL[,options] - Command to delete an existing trap
```

Note: If you specify IF, SBT, SA, or SAS, they must immediately follow SET. Specify END at the end of all SLIP SET commands.

Using SLIP commands

Use a SLIP command only at the direction of the system programmer. You can enter a SLIP command as follows:

- On a console with MVS master authority

- ▶ On a TSO terminal in OPERATOR mode
- ▶ In a TSO CLIST

In the CLIST, use the line continuation character at the end of each line and the END parameter at the end of the last line.
- ▶ In an IEACMD00, COMMNDxx, or IEASLPxx parmlib member

While you can enter a SLIP command in any of these members, IBM recommends that you place your SLIP commands in IEASLPxx and enter a SET SLIP=xx command to activate the member. IEACMD00 and COMMNDxx require that a command be on a single line. Also, SLIP may process commands in IEACMD00 and COMMNDxx in any order, but processes commands in IEASLPxx in the order in which they appear.

For a sysplex containing similar systems, certain problems might require identical SLIP traps on those similar systems. To set up these traps, do the following:

- ▶ Assign similar names to identical jobs on different systems. The names should form a pattern, such as JOB1, JOB2, JOB3, and so on.
- ▶ Create one IEASLPxx member containing the trap you need for the problem.
- ▶ Place the member in the shared parmlib data set or in the parmlib data set for each of the similar systems.
- ▶ In systems using JES2 or JES3, activate the member or members with the following command entered on one of the systems:

```
ROUTE *ALL,SET SLIP=xx
```

3.6 Storage overlays

- ❑ System problems in MVS are often caused by storage overlays that:
 - Destroy data, control blocks, or executable code
- ❑ Overlays result in:
 - MVS detects an error and issues an abend code
 - Referencing the data or instructions can cause an immediate error
 - Bad data is used to reference a second location, which then causes another error

Figure 3-6 Problems with storage overlays

Storage overlays

Storage overlays can affect your system during IPL and during production lifetime. The system can crash if any of the system-related control blocks have been overlaid. Data overlay may be recoverable but you still need to determine why you get the overlay and who is storing data to an area not owned or where data has already been located.

If the data that causes the overlay is still stored at the same storage address, you can use a storage alteration SLIP (SA) to locate the culprit. If the data is stored randomly in a storage area, it's quite difficult to find the responsible module or program.

System problems

Always be aware of the possibility of a storage overlay when analyzing a dump. System problems in MVS are often caused by storage overlays that destroy data, control blocks, or executable code. The results of such an overlay vary. For example:

- ▶ The system detects an error and issues an abend code, yet the error can be isolated to an address space. Isolating the error is important in discovering whether the overlay is in global or local storage.
- ▶ Referencing the data or instructions can cause an immediate error such as a specification exception (abend X'0C4') or operation code exception (abend X'0C1').
- ▶ The bad data is used to reference a second location, which then causes another error.

3.7 Storage overlay during IPL

❑ Storage overlay of PSA or related control blocks

- Take a stand-alone dump
- Use IPCS to format the dump

❑ Analysis of dump

- Identify failing CP
- Identify failing module
- Create a trap to find the overlay
- Diagnose the cause

Figure 3-7 Analyzing storage overlays

Storage overlays during IPL

When you recognize that the contents of a storage location are not valid and subsequently recognize the bit pattern as a certain control block or piece of data, you can generally identify the erroneous process or component and start a detailed analysis.

WAIT 014

A WAIT 014 is usually the result of an overlay of a critical control block such as the PSA, ASCB, SGTE, or PGTE. Typically the last program running on the CP caused the overlay of the PSA or related control blocks. The system enters a non-restartable wait state.

Dump to analyze overlay

To determine the control block that has been overlaid and the module that did the overlay ask the system programmer to provide a standalone dump. Use IPCS to format the dump and start with the debug.

Identify the failing processor

Enter `IP ST WORKSHEET` and examine the common system data area (CSD) CPU online mask. There is one bit for each processor online. To determine which processor was taken offline look at:

```
CSD   Available CPU mask: FC00   Alive CPU mask: 7C00   No. of active CPUs: 0005
```

Where:

FC00 shows the available CPU mask. Bits 0 to 5 are set to one.

7C00 shows the alive CPU mask. Bits 1 to 5 are set to one

This means that CPUs 1 to 5 are active and CPU 0 is the failing processor.

In addition, the IPCS command ST WORKSHEET also shows the automatic CPU recovery (ACR) pair leading to failing and recovery processors.

Identify the failing module and overlaid control block

Examine the last program interrupt on the failing processor:

Program old PSW at PSA+x'28' identifies the failing module
ILIC (Instruction Length Interrupt Counter) is at PSA+x'8C'

Use the PSW address and the ILC to determine the offset in the failing module. Examine that code to obtain the control block field that was being referenced. This is typically a PSA field. If possible, use known/valid control block values to determine the extent of the overlay. For detailed control block information see the volumes on z/OS MVS Data Areas, as follows:

- ▶ z/OS MVS Data Areas, Volume 1 (ABEP - DALT), GA22-7581
- ▶ z/OS MVS Data Areas, Volume 2 (DCCB - ITZYRETC), GA22-7582
- ▶ z/OS MVS Data Areas, Volume 3 (IVT - RCWK), GA22-7583
- ▶ z/OS MVS Data Areas, Volume 4 (RD - SRRA), GA22-7584
- ▶ z/OS MVS Data Areas, Volume 5 (SSAG - XTLST), GA22-7585

Provide a trap to catch the overlayer

A storage alteration (SA) trap could be supplied to catch the overlayer.

Note: The trap should only be set on a field that is not ordinarily updated.

Create a SLIP trap to wait when the PSA+x'200' is overlaid, as follows:

```
SLIP SET,SA,ASA=SA,A=WAIT,RA=(200,203),END
```

ASA=SA prevents the trap from hitting on a data space update.

Diagnosing the cause

From the SA dump of the WAIT 014, determine the *Window of Error* by:

- ▶ Examining the system trace to identify the last program that successfully ran on the failing CP.
- ▶ Identify the failing instruction address via LCCAPPSW, LCCA+x'88'.

These two events define the Window of Error, and the code that executed in the Window probably caused the error.

3.8 Storage overlay in a production system

❑ Determine overlay area

- Find storage address
- Determine data overlayed

❑ Set a SLIP trap

- Determine if storage or a register is needed
- Sample SLIP trap

```
SLIP SET,IF,N=(IAXUA,237A),A=(SVCD,REFAFTER),SUMLIST=(009C.5000,6000),  
REFAFTER=(009C.5000.EQC(2),009C.5098,1REQ,00000000),END
```

Figure 3-8 Setting a SLIP trap for an overlay

Storage overlay SLIP trap

Depending on the overlaid area, it could be possible to repair the overlaid control block or storage information. To fix the overlay you need to know the storage address and the data that has been overlaid. The SLIP definition provides the possibility to check the control block using the indirect pointing.

Use a powerful option where SLIP will modify the storage or register as part of the action taken when the PER trap hits.

Use with caution and ensure accuracy. This will allow correction of an overlay or improperly specified register or storage, but if the target is not correct, or the refresh data is incorrect, further potential damage may occur.

The following SLIP shows an example of how to get a dump and repair the overlaid area. The SLIP indicates the module name is located in ASID X'9C' at offset x'5000', and refreshes the first two bytes to zeroes and sets R1=0.

```
SLIP SET,IF,N=(IAXUA,237A),A=(SVCD,REFAFTER),SUMLIST=(009C.5000,6000),  
REFAFTER=(009C.5000.EQC(2),009C.5098,1REQ,00000000),END
```

3.9 SLIP to catch the overlayer

- ❑ Determine how to set the SLIP
- ❑ Check if SLIP does not match
- ❑ PER traps
- ❑ DEBUG option
- ❑ Environments where SLIP PER not supported

```
SLIP SET,SA,ASA=SA,RA=(2D0,2D3),A=SVCD,ID=HILG,  
SDATA=(ALLNUC,PSA,SQA,CSA,RGN,LPA,TRT,SUM),END
```

Figure 3-9 Setting SLIP traps

Sample SLIP trap

The following SLIP is an example how to catch the program overlaying the storage area on offset x'2D0' length 4 bytes.

```
SLIP SET,SA,ASA=SA,RA=(2D0,2D3),A=SVCD,ID=HILG,  
SDATA=(ALLNUC,PSA,SQA,CSA,RGN,LPA,TRT,SUM),END
```

Determining if SLIP matches

Check the following to see whether the SLIP is not matching:

- ▶ Issue a D SLIP=XXXX (where XXXX is the trap id) to verify that the trap was set as intended.
- ▶ With the LPAMOD or PVTMOD keywords, verify that it specifies the load module name, not the CSECT name.
- ▶ Be sure that MODE=HOME, JOBNAME= or ASID is specified with PVTMOD for a module that is loaded into private storage.
- ▶ PER traps:
 - Check the PSA+X'98' for the residual PER address stored by the hardware when the PER interrupt is presented. The PER trap is not active if 0 or the PER bit is not on in the PSW.
 - Check control registers 9, 10, and 11 to determine whether they are set correctly. These registers are the STATUS REGS, as follows:

- CR 9 - PER EVENT TYPE
- CR 10 - BEGIN RANGE
- CR 11 - END RANGE

Note: Any SLIP trap affects system performance, but PER traps can have a measurable effect on performance. Therefore, use conditions to filter the events being checked for matches, especially for PER traps. Improper use of PER traps can cause severe performance problems.

- Use the DEBUG option with A=TRACE to see which keyword is not matching on the SLIP trap. With DEBUG a GTF record will be cut regardless of whether the trap matches, and will contain a key indicating which keyword did not match.

For a SLIP SET trap, the DEBUG option allows you to determine why a trap is not working as you expected by indicating which of the conditions you established is not being met. DEBUG provides trap information each time the trap is tested rather than just when it matches.

The generalized trace facility (GTF) and its trace option for SLIP records must be active. Each DEBUG trace record contains SLIP information plus two bytes: the first byte contains a value indicating the failing parameter and the second byte contains zero.

- With PVTMOD, A=IGNORE, traps will not match if the local lock is not available at the time the PER interrupt is presented and SLIP module IEAVTSL2 is checking for a match. See DOC APAR OY37341.

SLIP has a default match limit of 1 on all traps that specify, or default to, ACTION=SVCD. The match limit can be changed by the MATCHLIM parameter when setting the SLIP trap. You can further qualify the SLIP trap by using other parameters, such as DATA and PVTMOD.

SLIP PER environment

SLIP PER is not active or is not supported in the following environments:

- Program check, machine check, and restart FLIHs
- Some RSM™ modules
- Dispatcher
- Lock manager (cannot SLIP on lock words)
- DAT-OFF code (SLIP only supports virtual addresses)
- Any code that turns the PER bit off in the PSW

If any of the above cases apply, use the CP address compare hardware function or a software detection trap.



Dump processing

Dumps can provide useful diagnosis data. But you need to check the dump-related options to be sure all information needed will be dumped.

Generally, the system automatically captures a dump when it detects a serious error with an operating system component (for example, JES, VTAM, etc.), a subsystem (for example, CICS, DB2, MQ, etc.), or application program. For most system or subsystem failures an SVC (Supervisor Call) dump is generated and written out to a predefined, or dynamically defined, dump data set. You do, however, have the ability to manually capture a dump should you need to capture specific diagnostic data.

The DUMP command requests a system dump (SVC dump) of virtual storage. The data set may be either a pre-allocated dump data set named SYS1.DUMPxx, or an automatically allocated dump data set named according to an installation-specified pattern. You should request only one dump at a time on one system. A system writes only one SVC dump at a time, so it does not save time to make several requests at once.

4.1 Getting or requesting dumps

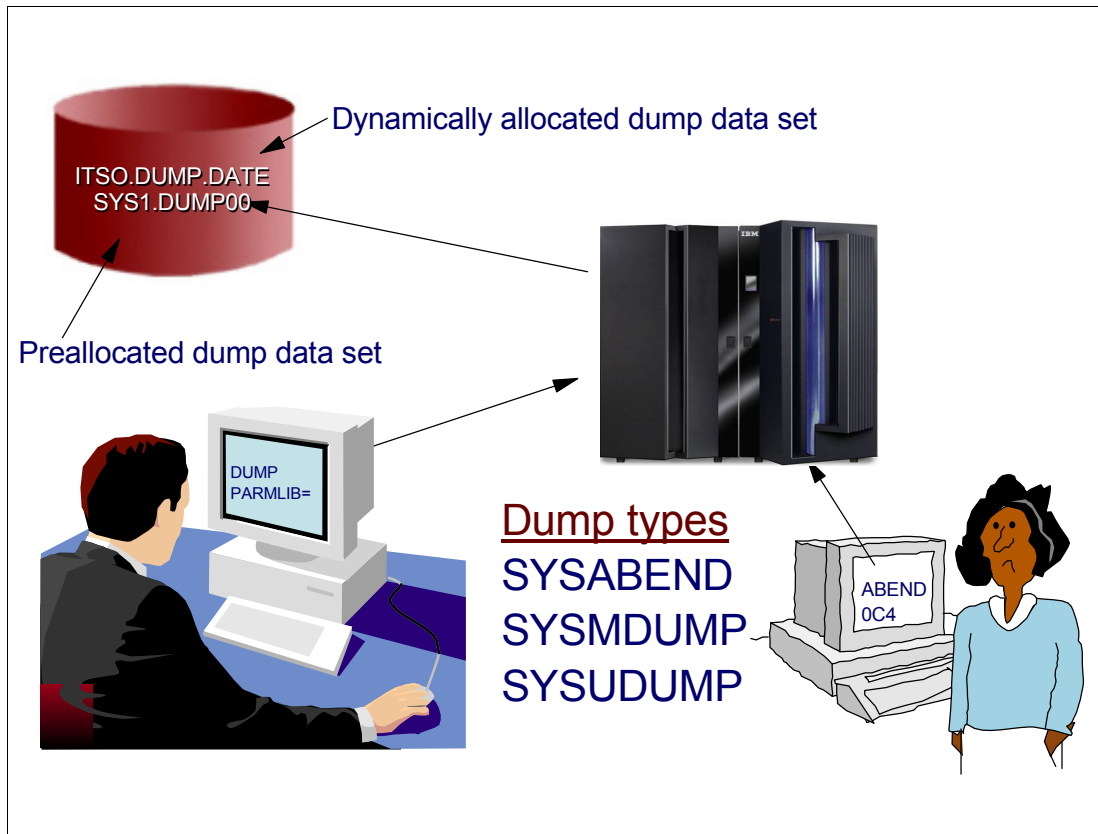


Figure 4-1 Getting or requesting dumps

Diagnostic data - dumps

Different types of dumps can be used to analyze problems. The dump types and the procedures that can be used to initiate these processes are discussed later in detail.

Dumps could best be described as a *SNAPSHOT* of the system at the time a failure is detected by the operating system or application, or at the time the system is dumped by the operator (console dump) via the DUMP command or the stand-alone dump procedure.

Following are the dump types that will be discussed:

- ▶ Abend dumps
- ▶ SLIP dumps
- ▶ SNAP dumps
- ▶ Stand-alone dumps
- ▶ SVC dumps

ABEND dump types

Use an ABEND dump when ending an authorized program or a problem program because of an uncorrectable error. These dumps show:

- ▶ The virtual storage for the program requesting the dump.
- ▶ System data associated with the program.

The system can produce three types of ABEND dumps:

- SYSABEND** The largest of the ABEND dumps, containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program. This dump is formatted.
- SYSMDUMP** Contains a summary dump for the failing program, plus some system data for the failing task. SYSMDUMP dumps are the only ABEND dumps that are unformatted and must be formatted with IPCS.
- SYSUDUMP** The smallest of the ABEND dumps, containing data and areas only about the failing program. This dump is formatted.

Specifying dumps via JCL

You can obtain SYSABEND, SYSUDUMP, and SYSMDUMP dumps by specifying the appropriate DD statement in your JCL, as follows:

- ▶ SYSABEND dumps are formatted as they are created and can be directed to either DASD, TAPE, or SYSOUT.

```
//SYSABEND DD SYSOUT=*
```
- ▶ SYSUDUMP dumps are formatted as they are created and can be directed to either DASD, TAPE, or SYSOUT.

```
//SYSUDUMP DD SYSOUT=*
```
- ▶ SYSMDUMP dumps are unformatted and must be analyzed using the Interactive Problem Control System (IPCS). These data sets must reside on either DASD or TAPE. Figure 4-2 shows an example of a SYSMDUMP DD statement.

```
//SYSMDUMP DD DSN=MY.SYSMDUMP,DISP=(,CATLG),UNIT=DISK,  
//          SPACE=(CYL,(50,20),RLSE),  
//          LRECL=4160,BLKSIZE=4160
```

Figure 4-2 SYSMDUMP DD statement

4.2 Slip commands

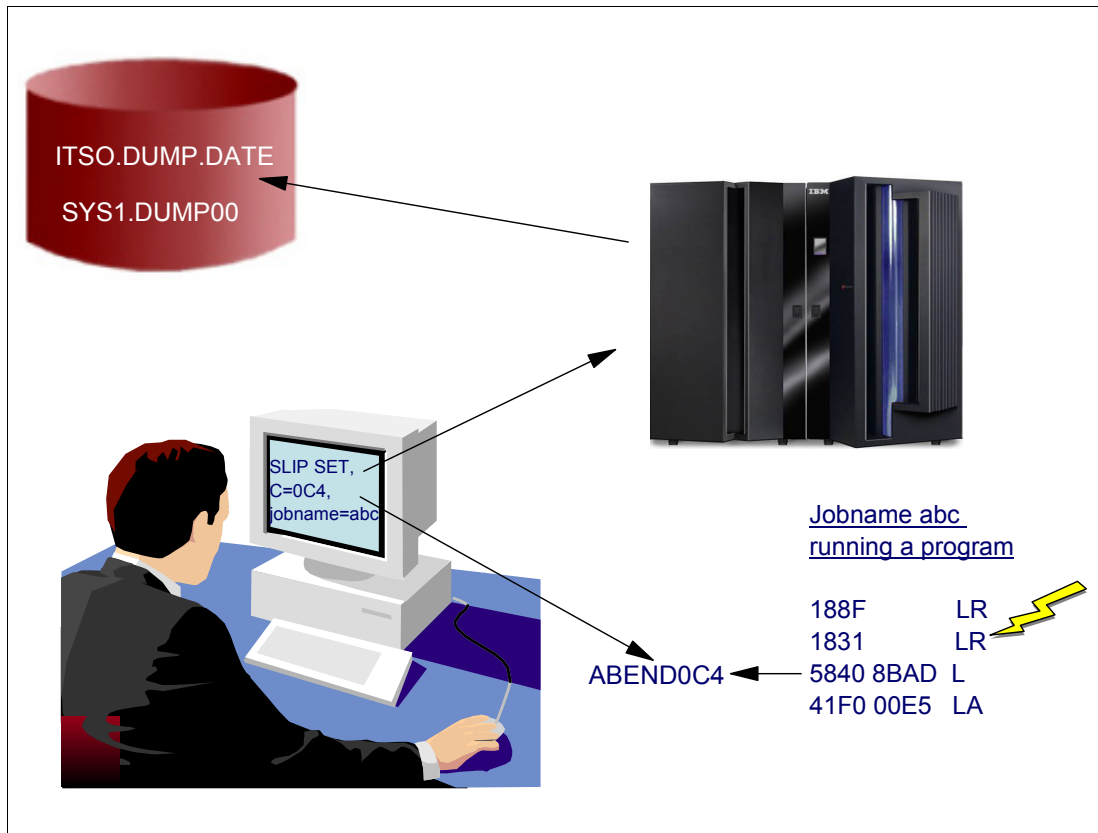


Figure 4-3 SLIP commands

SLIP commands

The SLIP command controls SLIP (serviceability level indication processing), a diagnostic aid that intercepts or traps certain system events and specifies what action to take. Using the SLIP command, you can set, modify, and delete SLIP traps. Following are the SLIP commands:

SLIP SET[,options],END	Command for an error event trap (non-PER)
SLIP SET,IF[,options],END	Command for an instruction fetch PER trap
SLIP SET,SBT[,options],END	Command for a successful branch PER trap
SLIP SET,SAISAS[,options],END	Commands for a storage alteration PER trap
SLIP MOD[,options]	Command to modify an existing trap
SLIP DEL[,options]	Command to delete an existing trap

Setting a SLIP dump

In Figure 4-3, the operator is setting a SLIP that forces a dump in jobname abc, which is executing a program that has an 0C4 abend at instruction 58408BAD every time it executes. Setting the SLIP forces the program to take a dump on the occurrence of the 0C4. The command is set as follows, as shown in the figure:

```
SLIP SET,C=0C4,JOBNAME=ABC
```

Using SLIP with ABEND dumps

ABEND dumps can be suppressed using the SLIP command in member IEASLPxx in SYS1.PARMLIB. These commands used to reside in member IEACMDxx in SYS1.PARMLIB but it is recommended that you move any SLIP commands from IEACMDxx to IEASLPxx to avoid restrictions found in other parmlib members. For example,

- ▶ IEASLPxx supports multiple-line commands; IEACMD00 does not.
- ▶ IEASLPxx does not require any special command syntax; IEACMD00 does.

Figure 4-4 shows the SLIP commands in SYS1.PARMLIB member IEASLP00.

```
SET,C=013,ID=X013,A=NOSVCD,J=JES2,END SLIP SET,C=028,ID=X028,A=NOSVCD,END SLIP
SET,C=47B,DATA=(15R,EQ,0,OR,15R,EQ,8),ID=X47B,A=NODUMP,END SLIP
SET,C=058,DATA=(15R,EQ,4,OR,15R,EQ,8,OR,15R,EQ,C,OR,15R,EQ,10,OR,
15R,EQ,2C,OR,15R,EQ,30,OR,15R,EQ,3C),ID=X058,A=NODUMP,END SLIP
SET,C=0E7,ID=X0E7,A=NOSVCD,END SLIP SET,C=0F3,ID=X0F3,A=NODUMP,END SLIP
SET,C=13E,ID=X13E,A=NODUMP,END SLIP SET,C=222,ID=X222,A=NODUMP,END SLIP
SET,C=322,ID=X322,A=NODUMP,END SLIP SET,C=33E,ID=X33E,A=NODUMP,END SLIP
SET,C=422,ID=X422,A=NODUMP,END SLIP SET,C=622,ID=X622,A=NODUMP,END SLIP
SET,C=804,ID=X804,A=(NOSVCD,NOSYSU),END SLIP
SET,C=806,ID=X806,A=(NOSVCD,NOSYSU),END SLIP
SET,C=80A,ID=X80A,A=(NOSVCD,NOSYSU),END SLIP
SET,C=9FB,ID=X9FB,A=NOSVCD,J=JES3,END SLIP
SET,C=B37,ID=XB37,A=(NOSVCD,NOSYSU),END SLIP
SET,C=D37,ID=XD37,A=(NOSVCD,NOSYSU),END SLIP
SET,C=E37,ID=XE37,A=(NOSVCD,NOSYSU),END SLIP
SET,C=EC6,RE=0000FFXX,ID=XEC6,A=NODUMP,END SLIP
SET,C=EC6,RE=0000FDXX,ID=XXC6,A=NOSVCD,END
```

Figure 4-4 SLIP commands in SYS1.PARMLIB member IEASLP00

4.3 SLIP dumps

❑ SLIP dumps

➤ SLIP using IGC0003E

— SLIP processing

➤ SLIP using MSGID

❑ SLIP dump using a z/OS UNIX reason code

➤ Obtaining a dump on a specific reason code

Figure 4-5 Taking SLIP dumps using the SLIP command

SLIP dumps

The SLIP command is set via the z/OS operator SLIP SET command. This is a most powerful tool and allows for great complexity to be used to trigger a dump for a specific situation. It can be used to check storage associated with an event and trigger a dump when that event is true. We are going to concentrate on the most common use of the SLIP command: where it is set to trigger a dump when a specific message is written to the console. There are two forms of this command, as follows:

- ▶ The first, being the “old” way, where we interrogate storage being used by the WTOR routine
- ▶ The second, the later and more understandable version of the message SLIP

SLIP using IGC0003E

It is not necessary to set SLIP traps individually and run a failing job multiple times, using one trap for each execution until a dump is taken. You can set SLIP PER traps at multiple points in a load module as follows: use a non-IGNORE PER trap to monitor the range that encompasses all of the points in which you are interested, followed by several IGNORE PER traps to prevent the SLIP action from being taken on the intervening instructions, in which you are not interested.

Figure 4-6 shows a SLIP command example.

```
SLIP SET,IF,LPAMOD=(IGC0003E,0),  
DATA=(1R?+4,EQ,C3E2D8E7,1R?+8,EQ,F1F1F1C5),  
JOBNAME=ssidCHIN,  
JOBLIST=(ssidMSTR,ssidCHIN),  
DSPNAME=('ssidCHIN'.CSQXTRDS),  
SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),  
MATCHLIM=1,END
```

Figure 4-6 SLIP SET example

SLIP processing

This SLIP command example would interrogate the Register 1 storage owned by the WTOR routine, IGC0003E, and check for the values, starting at offset 4, to see if they match, CSQX (x'C3E2D8E7), and the Register 1 values starting at offset 8, 111X (x'F1F1F1C5). If the matching message was written, in this case, by job ssid CHIN, then the MQ MSTR and CHIN address spaces, and associated CHIN data space, will be dumped for a maximum match limit of 1 time. No further dumps will be taken if this job generates this message again.

SLIP using MSGID

Figure 4-7 shows the new form of the SLIP message, and as you can see, is much more user friendly because the MSGID can be included in its ASCII form, not as a HEX representation.

```
SLIP SET,MSGID=CSQX111E,  
JOBNAME=ssidCHIN,  
JOBLIST=(ssidMSTR,ssidCHIN),  
DSPNAME=('ssidCHIN'.CSQXTRDS),  
SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),  
MATCHLIM=1,END
```

Figure 4-7 SLIP SET using the MSGID parameter

Another simple use of the SLIP is to capture a dump when a specific application abend occurs. For example, you might be getting an S0C4 abend in an application program and require an SVC dump to assist with this, instead of an application or transaction dump. Figure 4-8 shows an example of a completion code SLIP.

```
SLIP SET,ENABLE,COMP=0C4,ERRTYP=PROG,JOBNAME=JOBXYZ,LPAMOD=MOD01,END
```

Figure 4-8 Completion code SLIP example

This example will capture an SVC dump when there is an S0C4 program check interruption while module MOD01 and job JOBXYZ are in control.

SLIP dump using a z/OS UNIX reason code

If a z/OS UNIX reason code is obtained and additional information is required, the IBM Support Center personnel may ask that you set a SLIP to collect a dump or trace on a recreation of the problem. Included below are the general instructions on how to gather this data.

Obtain a dump on a specific reason code

Figure 4-9 shows an example of a SLIP that will produce a dump on the issuance of a specific reason code.

```
SLIP SET,IF,A=SYNCSVCD,  
RANGE=(10?+8C?+F0?+1F4?),DATA=(13R??+b0,EQ,xxxxxxx),DSPNAME=('OMVS' .*),  
SDATA=(ALLNUC,PSA,CSA,LPA,TRT,SQA,RGN,SUM),j=jobname,END
```

Figure 4-9 Register 13 reason code SLIP example

Where:

- ▶ xxxxxxxx = the 8-digit (4 byte) reason code that is to be trapped.
- ▶ j=jobname is the optional jobname that is expected to issue the error (for example j=IBMUSER).

Note: In rare instances the above SLIP will not capture the requested reason code if the module in question does not use R13 as a data register. Your IBM software support provider can check the specific reason code and determine if this is the reason the SLIP did not match.

4.4 SNAP dumps

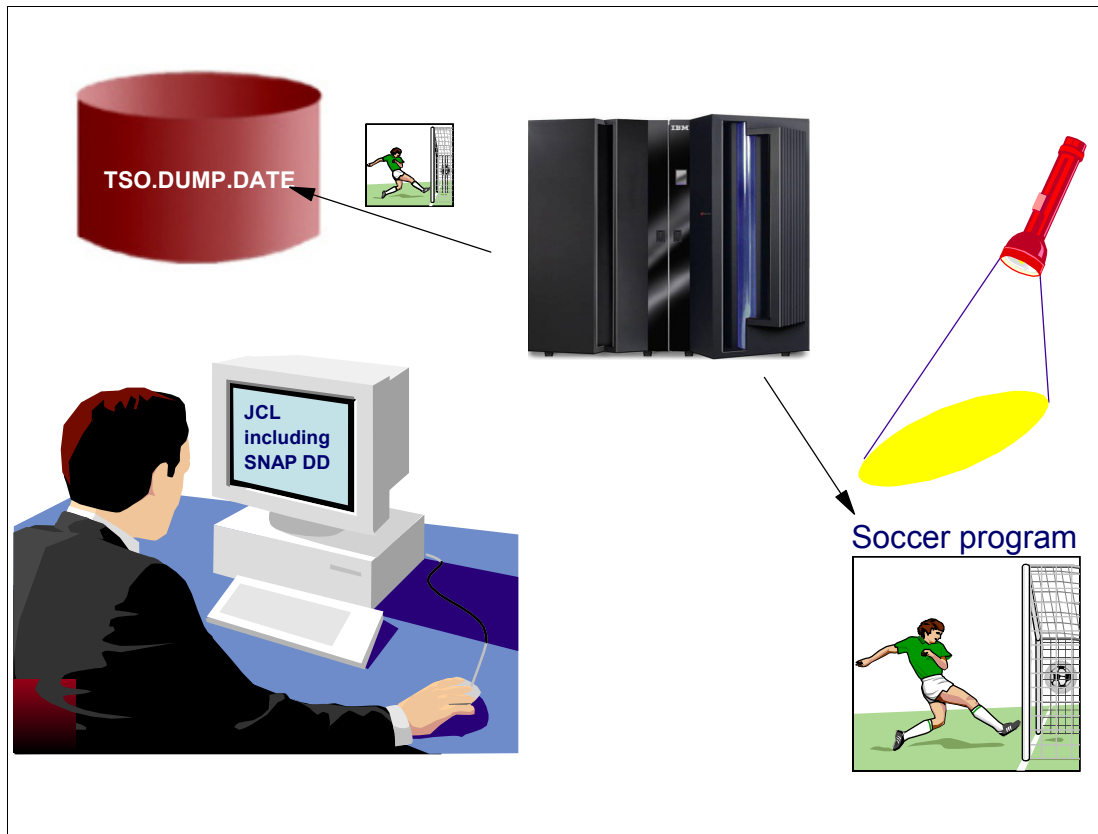


Figure 4-10 SNAP dump processing

SNAP dump

A SNAP dump is like getting a snapshot of yourself while kicking a ball. You can go back later and look at what you did wrong so that you can improve.

A SNAP dump shows virtual storage areas that a program, while running, requests the system to dump. A SNAP dump, therefore, is written while a program runs, rather than during abnormal end. The program can ask for a dump of as little as a 1-byte field to as much as all of the storage assigned to the current job step. The program can also ask for some system data in the dump. A SNAP dump is especially useful when testing a program. A program can dump one or more fields repeatedly to let the programmer check intermediate steps in calculations. For example, if a program being developed produces incorrect results, requests for SNAP dumps can be added to the program to dump individual variables. The first time incorrect storage is encountered should narrow down the section of code causing the error.

Note: A SNAP dump is written while a program runs, rather than during abnormal end.

Obtaining a SNAP dump

Obtain a SNAP dump by taking the following steps:

1. Code a DD statement in the JCL for the job step that runs the problem program to be dumped with a ddname other than SYSUDUMP, SYSABEND, SYSMDUMP, or another restricted ddname. The statement can specify that the output of the SNAP dump should be written to one of the following:

- Direct access storage device (DASD). For example,
//SNAP1 DD DSN=MY.SNAP.DUMP,DISP=(OLD)
 - Printer. Note that a printer is not recommended because the printer cannot be used for anything else while the job step is running, whether a dump is written or not.
 - SYSOUT. SNAP dumps usually use SYSOUT. For example,
//SNAP1 DD SYSOUT=X
 - Tape. For example,
//SNAP1 DD DSN=SNAP.TO.TAPE,UNIT=TAPE,DISP=(OLD)
2. In the problem program:

- a. Specify a data control block (DCB) for the data set to receive the dump. For a standard dump, which has 120 characters per line, the DCB must specify:

```
BLKSIZE=882 or 1632
DSORG=PS
LRECL=125
MACRF=(W)
RECFM=VBA
```

For a high-density dump, which has 204 characters per line and will be printed on an APA 3800 printer, the DCB must specify:

```
BLKSIZE=1470 or 2724
DSORG=PS
LRECL=209
MACRF=(W)
RECFM=VBA
```

- b. Code an OPEN macro to open the DCB.

Before you issue the SNAP or SNAPX macro, you must open the DCB that you designate on the DCB parameter, and ensure that the DCB is not closed until the macro returns control. To open the DCB, issue the DCB macro with the following parameters, and issue an OPEN macro for the data set:

```
DSORG=PS,RECFM=VBA,MACRF=(W),BLKSIZE=nnn,LRECL=xxx,
and DDNAME=any name but SYSABEND, SYSMDUMP or SYSUDUMP
```

If the system loader processes the program, the program must close the DCB after the last SNAP or SNAPX macro is issued.

- c. Code a SNAP or SNAPX assembler macro to request the dump. We recommend the use of the SNAPX macro as this allows for programs running in Access-Register (AR) mode to cause the macro to generate larger parameter lists. In the following example, the SNAPX macro requests a dump of a storage area, with the DCB address in register 3, a dump identifier of 245, the storage area's starting address in register 4, and the ending address in register 5:

```
SNAPX DCB=(3),ID=245,STORAGE=((4),(5))
```

Repeat this macro in the program as many times as wanted, changing the dump identifier for a unique dump. The system writes all the dumps that specify the same DCB to the same data set.

- d. Close the DCB with a CLOSE assembler macro.

Customizing SNAP dumps

An installation can customize the contents of SNAP dumps through the IEAVADFM or IEAVADUS installation exits. IEAVADFM is a list of installation routines to be run and IEAVADUS is one installation routine. The installation exit routine runs during control block formatting of a dump when the CB option is specified on the SNAP or SNAPX macro. The routine can format control blocks and send them to the data set for the dump. See *z/OS MVS Installation Exits*, SC28-1753, for more information.

4.5 Stand-alone dumps

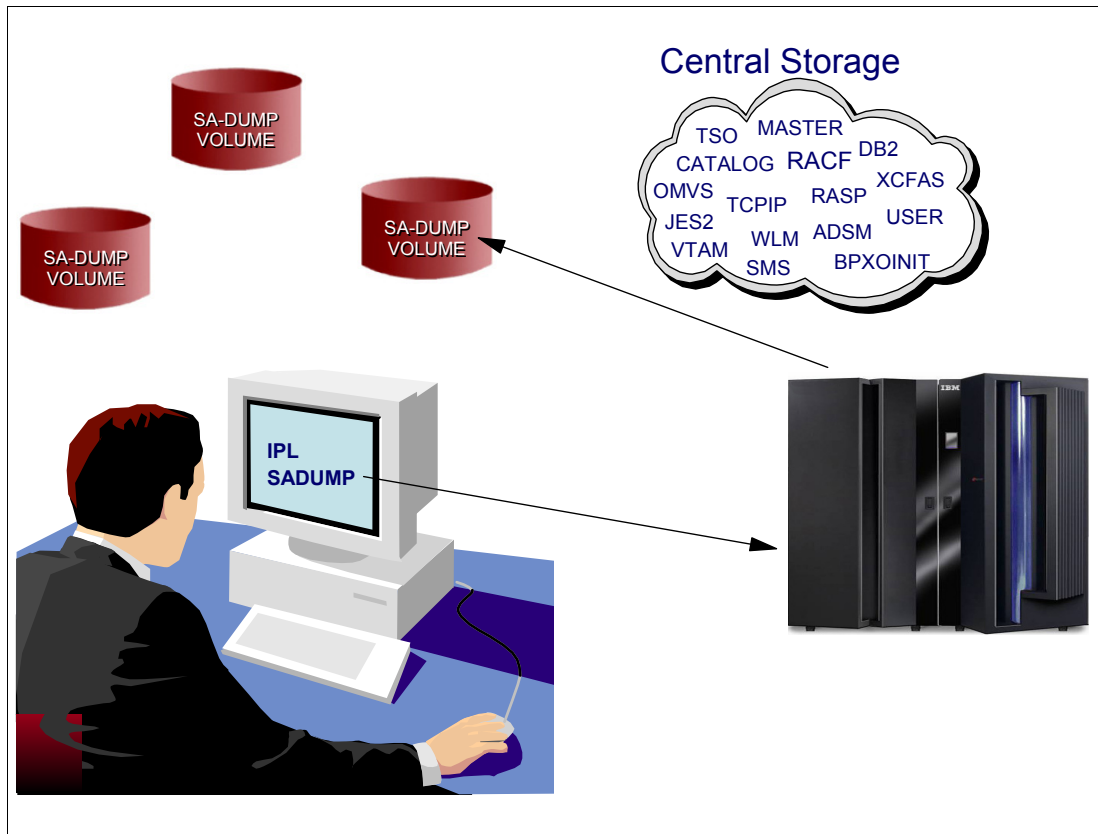


Figure 4-11 Stand-alone dump

Stand-alone dump

Stand-alone dumps are not produced by z/OS but by a program called SADMP, which is IPLed in place of z/OS. When to use a stand-alone dump is shown in Figure 4-11.

The stand-alone dump program and the stand-alone dump together form what is known as the stand-alone dump service aid. The term stand-alone means that the dump is performed separately from normal system operations and does not require the system to be in a condition for normal operation. The stand-alone dump program produces a high-speed, unformatted dump of main storage and parts of paged-out virtual storage on a tape device or a direct access storage device (DASD). The stand-alone dump program, which you create, must reside on a storage device that can be used to IPL.

Produce a stand-alone dump when the failure symptom is a wait state with a wait state code, a wait state with no processing, an instruction loop, or slow processing. Stand-alone dumps can be analyzed using IPCS.

Allocating the stand-alone dump data set

Prior to z/OS V1R7, in the SYS1.SAMPLIB data set, you can use the AMDSADDD REXX™ utility to allocate and initialize the SADMP dump data set. The dump data set must be both allocated and initialized using the AMDSADDD REXX or IPCS SADMP dump data set utilities panel created in z/OS V1R7, shown in Figure 4-12 on page 82.

```

----- SADMP DASD Dump Data Set Utility -----
Command ==>

Enter/verify parameters.
Use ENTER to perform function, END to terminate.

Function ==> R ( C - Clear, D - Define, R - Reallocate)
DSNAME    ==>

Volume serial numbers:  (1-32)
    1- 8  VOL001
    9-16
   17-24
   25-32

Unit ==> 9345  (3380, 3390, or 9345)
Cylinders ==> 500      (cylinders per volume)
DSNTYPE(LARGE) ==> N (Y or N)

Optional SMS classes: (May be required by installation ACS routines)
StorClas ==>          DataClas ==>          MgmtClas ==>

```

Figure 4-12 SADMP DASD Dump Data Set Utility panel

Note: Beginning with z/OS v1R7, you can use the SADMP DASD dump data utility and select option 3.6 to use a panel to create, clear, and reallocate SADMP data sets on DASD. This utility performs the same functions associated with the AMDSADDD REXX utility. You can also use AMDSADDD, but references to the REXX utility in SYS1.SAMPLIB no longer exist. You must now refer to this utility in SYS1.SBLSCLIO. The data set is placed in SBLSCLIO rather than SAMPLIB because it is no longer a sample.

You can EXEC this REXX utility from the ISPF data set utility option 3.4, and either VIEW (V), BROWSE (B) or EDIT (E) the data set. You can issue the following command from the ISPF option line and the utility prompts you as shown in Figure 4-13 on page 83.

```
EXEC 'SYS1.SBLSCLIO(AMDSADDD)'
```

```

What function do you want?
Please enter DEFINE if you want to allocate a new dump data set
Please enter CLEAR if you want to clear an existing dump data set
Please enter REALLOC if you want to reallocate and clear an existing
    dump data set
Please enter QUIT if you want to leave this procedure
define

    Please enter VOLSER or VOLSER(dump_dataset_name)
SDD01A(WTSCPLX1.SADMP.SDD01A)
    Please enter the device type for the dump data set
    Device type choices are 3380 or 3390 or 9345
3390
    Please enter the number of cylinders
10014
    Do you want the dump data set to be cataloged?
    Please respond Y or N
Y
    TIME-08:59:31 AM. CPU-00:00:03 SERVICE-549023 SESSION-01:18:42 APRIL 9,

    Initializing output dump data set with a null record:
    Dump data set has been successfully initialized

    Results of the DEFINE request:

        Dump data set Name   : WTSCPLX1.SADMP.SDD01A
        Volume               : SDD01A
        Device Type          : 3390
        Allocated Amount     : 10014
    ***

```

Figure 4-13 Prompts issued by the AMDSADDD REXX utility

Or, you can just enter this command and execute without prompts:

```

TSO EXEC 'SYS1.SBLSCLI0(AMDSADDD)' 'DEFINE SDD01A(WTSCPLX1.SADMP.SDD01A) 3390
10014 YES LARGE'

```

Note: The size used, 10014, and the data set type, LARGE, are new with z/OS V1R7. See 6.4, “IPCS support of large data sets” on page 141.

4.6 The SADMP program

- ❑ Produces a high-speed unformatted dump
 - Central storage and parts of virtual storage
 - Dump to tape or DASD
 - Created with AMDSADM macro
- ❑ Default DASD device

Figure 4-14 SADMP processing of dumps

The SADMP program

The SADMP program produces a high-speed, unformatted dump of main storage and parts of paged-out virtual storage on a tape device or a direct access storage device (DASD). The SADMP program that you create must reside on a storage device that can be used to IPL.

Create the SADMP program by using the AMDSADM macro to produce the following:

- ▶ A SADMP program that resides on DASD, with output directed to a tape volume or to a DASD dump data set
- ▶ A SADMP program that resides on tape, with output directed to a tape volume or to a DASD dump data set

Create the SADMP program with the following JCL as an example.

```

//SADMPGEN JOB MSGLEVEL=(1,1)
//OSG      EXEC PGM=AMDSAOSG
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//DPLTEXT  DD DSN=SYS1.NUCLEUS(AMDSADPL),DISP=SHR
//IPLTEXT  DD DSN=SYS1.NUCLEUS(AMDSAIPD),DISP=SHR
//PGETEXT  DD DSN=SYS1.NUCLEUS(AMDSAPGE),DISP=SHR
//GENPRINT DD DSN=SADMP.LIST,DISP=OLD
//GENPARMS DD *
            AMDSADMP IPL=DSYSDA,VOLSER=SP00L2,          X
            CONSOLE=(1A0,3277)
            END
/*

```

ADMSADMP macro

AMDSADMP processing does not allocate the data set or check to see that a valid MVS data set name has been provided. Therefore, you should ensure that:

- ▶ The AMDSADDD REXX utility is used to allocate and initialize the same data set name specified on the OUTPUT= keyword.
- ▶ The data set name specified should be fully qualified (without quotes).
- ▶ The necessary data set management steps are taken so that the SADMP dump data sets will not be placed into a migrated state or moved to a different volume.
- ▶ Alphabetic characters appearing in the dump data set name should be specified as capital letters.

You need to answer some questions when you plan for a stand-alone dump. Some typical questions follow:

- ▶ Should I take a stand-alone dump to DASD or to tape?
- ▶ Can I use my current version of the stand-alone dump program to dump a new version of z/OS?

Default DASD device

If the default DASD device is to be used and no dump data set name is provided, the SADMP program will assume that the default dump data set name is SYS1.SADMP if the DDSPROMPT=NO parameter was also specified. Otherwise, if DDSPROMPT=YES was specified, the SADMP program will prompt the operator at run-time for a dump data set name to use.

- ▶ At run-time, only a null response to message AMD001A will cause the SADMP program to use the default device and/or dump data set name.
- ▶ Do not place a data set that is intended to contain a stand-alone dump on a volume that also contains a page or swap data set that the stand-alone dump program may need to dump. When SADMP initializes a page or swap volume for virtual dump processing, it checks to see if the output dump data set also exists on this volume. If it does, the SADMP program issues message AMD100I and does not retrieve any data from page or swap data sets on this volume. Thus, the dump may not contain all of the data that you requested. This lack of data may impair subsequent diagnosis.
- ▶ You cannot direct output to the SADMP residence volume.

4.7 Using stand-alone dumps

- ❑ Stand-alone dump steps
 - Select processor that stopped
 - Is a STORE STATUS required ?
 - Find a ready device (Tape or DASD)
 - IPLing SADMP
 - Select a console defined to AMDSADMP

Figure 4-15 SADMP steps to create the dump

Stand-alone dump procedure

Use the following procedure to initialize the SADMP program and dump storage:

1. Select a processor that was online when the system was stopped.
2. If the processor provides a function to IPL a stand-alone dump without performing a manual STORE STATUS, use this function to IPL SADMP. If you do not use such a function, perform a STORE STATUS before IPLing a stand-alone dump. If the operator does not store status, virtual storage is not dumped.

The hardware store-status facility stores the current program status word (PSW), current registers, the processor timer, and the clock comparator into the unprefix save area (PSA). This PSA is the one used before the nucleus initialization program (NIP) initialized the prefix register.

If you IPL the stand-alone dump program from the hardware console, it is not necessary to perform the STORE STATUS operation. Status is automatically stored when stand-alone dump is invoked from the hardware console and automatic store status is on.

If the operator does not issue the STORE STATUS instruction before IPLing a stand-alone dump, the message ONLY GENERAL PURPOSE REGS VALID might appear on the formatted dump. The PSW, control registers, and so on, are not included in the dump.

Note: Do not use the LOAD CLEAR option because it erases main storage, which means that you will not be able to diagnose the failure properly.

3. Make the residence device ready. If it is a tape, mount the volume on a device attached to the selected processor and ensure that the file-protect ring is in place. If it is a DASD volume, ensure that it is write-enabled.

4. IPL SADMP

SADMP does not communicate with the operator console. Instead, SADMP loads an enabled wait PSW with wait reason code X' 3E0000'. The IPLing of the stand-alone dump program causes absolute storage (X'0' through X'18' and storage beginning at X'110') to be overlaid with CCWs. You should be aware of this and not consider it as a low storage overlay.

Note: SADMP uses the PSW to communicate with the operator or systems programmer.

SADMP waits for a console I/O interrupt or an external interrupt.

5. Select the system console or an operator console with a device address that is in the console list that you specified at SADMP generation time (in the CONSOLE keyword of AMDSADMP). At SADMP run time, the operator can choose either a console specified with the CONSOLE= keyword or the system console to control SADMP operation. If an operator console is chosen, press Attention or Enter on that console. (On some consoles, you might have to press Reset first.) This causes an interruption that informs SADMP of the console's address. Message AMD001A appears on the console.
 - a. Make an output device ready. When you dump to devices that have both real and virtual addresses (for example, dumping a VM system), specify only the real address to the SADMP program. If you are dumping to tape, ensure that the tape cartridge is write-enabled. If you are dumping to DASD, ensure that the DASD data set has been initialized using the AMDSADDD REXX utility.
 - b. Reply with the device number for the output device. If you are dumping to a DASD device and DDSPROMPT=YES was specified on the AMDSADMP macro, message AMD002A is issued to prompt the operator for a dump data set. If DDSPROMPT=NO was specified, message AMD002A is not issued and the SADMP program assumes that the dump data set name is SYS1.SADMP.

Note: Pressing Enter in response to message AMD001A will cause the SADMP program to use the default device specified on the OUTPUT= keyword of the AMDSADMP macro. If the default device is a DASD device, then pressing the Enter key in response to message AMD001A will cause the SADMP program to use both the default device and the dump data set specified on the OUTPUT= keyword of the AMDSADMP macro. If no dump data set name was provided on the OUTPUT= keyword and the DDSPROMPT=YES keyword was specified, message AMD002A is issued to prompt the operator for a dump data set. If DDSPROMPT=NO was specified, then the SADMP program assumes that the dump data set name is SYS1.SADMP.

If you reply with the device number of an attached device that is not of the required device type, or if the device causes certain types of I/O errors, SADMP might load a disabled wait PSW. When this occurs, use procedure B to restart SADMP.

4.8 SADMP processing

- ❑ Procedures for processing the dump
 - Specify a dump title
 - Respond to a prompt message - AMD011A
 - Ready the output device
 - Monitor AMD095I message issued every 30 seconds
 - Specifying PROMPT option on AMDSADMP macro
 - Is additional storage required to be dumped
- ❑ Considerations while taking the dump

Figure 4-16 SADMP processing considerations

Processing the SADMP

SADMP prompts you, with message AMD011A, for a dump title. When no console is available, run SADMP without a console.

- ▶ Ready the default output device that was specified on the OUTPUT parameter on the AMDSADMP macro. For tapes, ensure that the tape cartridge is write-enabled. For DASD, ensure that the dump data set has been initialized using the AMDSADDD REXX utility.

Note: You can create different versions of the stand-alone dump program to dump different types and amounts of storage. You can create different versions of the stand-alone dump program by coding several AMDSADMP macros and varying the values of keywords on the macros.

- ▶ Enter an external interruption on the processor that SADMP was IPLed from. SADMP proceeds using the default output device and/or the default dump data set. No messages appear on any consoles; SADMP uses PSW wait reason codes to communicate to the operator.

Message AMD005I

When SADMP begins and finishes dumping main storage, it issues message AMD005I to display the status of the dump. SADMP may end at this step.

When SADMP begins dumping real storage it issues message AMD005I. Message AMD095I is issued every 30 seconds to indicate the progress of the dump. Message AMD005I will be issued as specific portions of real storage have been dumped, as well as upon completion of the real dump. SADMP may end at this step.

PROMPTs specified

If you specified PROMPT on the AMDSADMP macro, SADMP prompts you for additional storage that you want dumped by issuing message AMD059D.

SADMP dumps instruction trace data, paged-out virtual storage, the SADMP message log, and issues message AMD095I every 30 seconds to indicate the progress of the dump.

When SADMP completes processing, SADMP unloads the tape, if there is one, and enters a wait reason code X'410000'.

Considerations for taking the dump

Consider the following actions to take based on system availability and severity of problem:

- ▶ If I do dump to DASD, how much space do I need?
- ▶ Can I dump to multiple dump data sets?
- ▶ What can I name my DASD dump data sets?
- ▶ How much of the system should I dump?
- ▶ When should I specify the dump tailoring options?
- ▶ What type of security does the stand-alone dump program require?
- ▶ Should I use IEBGENER or the COPYDUMP subcommand to copy a dump to a dump to a data set?
- ▶ What is dumped when I run the stand-alone dump program?

4.9 SVC dumps

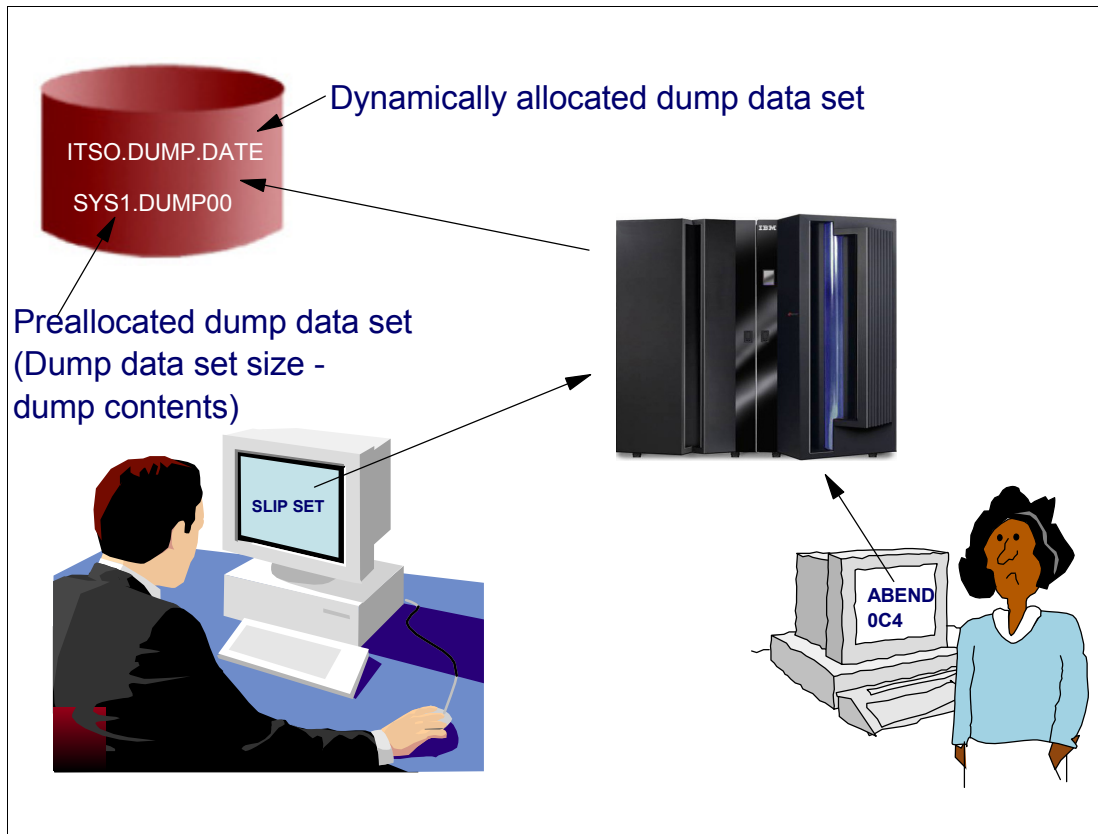


Figure 4-17 SVC dumps

SVC dump

An SVC dump provides a representation of the virtual storage for the system when an error occurs. Typically, a system component requests the dump from a recovery routine when an unexpected error occurs. However, an authorized program or the operator can also request an SVC dump when diagnostic dump data is needed to solve a problem.

SVC dumps can be used in different ways:

- ▶ Most commonly, a system component requests an SVC dump when an unexpected system error occurs, but the system can continue processing.
- ▶ An authorized program or the operator can also request an SVC dump (by using the SLIP or DUMP commands) when they need diagnostic data to solve a problem.

SVC dump contents

SVC dumps contain a summary dump, control blocks, and other system code, but the exact areas dumped depend on whether the dump was requested by a macro, command, or SLIP trap. SVC dumps can be analyzed using IPCS.

SVC dump processing stores data in dump data sets that you pre-allocate manually, or that the system allocates automatically, as needed. You can also use pre-allocated dump data sets as a back up in case the system is not able to allocate a data set automatically. To prepare your installation to receive SVC dumps, you need to provide SYS1.DUMPxx data

sets. These data sets will hold the SVC dump information for later review and analysis. This section describes how to set up the SVC dump data sets.

Note: An incomplete dump, or partial dump, is 99 percent of the time, useless.

Dump data set size

When the z/OS operating system initiates, or is instructed to dump an address space, or multiple address spaces, the data will be written to a dump dataset on a disk device. These data sets can be pre-allocated, as is the case with the traditional SYS1.DUMPxx data sets, or dynamically allocated, in which case a new dataset will be allocated whenever the system requests a dump.

In conjunction with the dump data set, the user-defined MAXSPACE parameter must be set to ensure that sufficient memory is allocated to retain the dump information in use by the address spaces and system areas. The recommended MAXSPACE in today's environment is 2500 MB, which is a lot different from the IBM default of 450 MB. This will need to be increased as products, such as DB2, start to make use of 64-bit virtual address ability.

Application-related dumps can be written to a data set pointed to by the SYSMDUMP DD statement in the JCL. The data written to the SYSMDUMP data set is always required to diagnose application-related problems running under Language Environment control.

The DCB requirements for dump data sets are as follows:

4.10 Allocating SYS1.DUMPxx data sets

- ❑ Name your SYS1.DUMPxx data sets - xx=00-99
- ❑ Blocksize must be 4160 bytes
 - RECFM=FBS (Fixed block spanned)
- ❑ Initialize with first record being an EOF
- ❑ Allocation requirements
 - UNIT - DISP - Volume - Space
- ❑ Managing SYS1.DUMPxx data sets
 - DUMPDS CLEAR,DSN=xx

Figure 4-18 Allocating the SYS1.DUMPxx data sets

Allocating SYS1.DUMPxx data sets

To prepare your installation to receive SVC dumps, you need to provide SYS1.DUMPxx data sets. These data sets will hold the SVC dump information for later review and analysis.

Allocate SYS1.DUMPxx data sets using the following requirements:

- ▶ Name the data set SYS1.DUMPxx, where xx is a decimal number of 00 through 99.
- ▶ Select a device with a track size of 4160 bytes. The system writes the dump in blocked records of 4160 bytes. If you want to increase the Block Size for the dump data set, you can do so as long as the blocking factor does not exceed 7, for example; 29120, and the Record Format (RECFM) must be Fixed Block Spanned (FBS).
- ▶ Initialize with an end of file (EOF) record as the first record.
- ▶ Allocate the data set before requesting a dump. Allocation requirements are:
 - UNIT** A permanently resident volume on a direct access device.
 - DISP** Catalog the data set (CATLG). Do not specify SHR.
 - VOLUME** Place the data set on only one volume. Allocating the dump data set on the same volume as the page data set could cause contention problems during dumping, as pages for the dumped address space are read from the page data set and written to the dump data set.
 - SPACE** An installation must consider the size of the page data set that will contain the dump data. The data set must be large enough to hold the amount of data as

defined by the MAXSPACE parameter on the CHNGDUMP command, VIO pages, and pageable private area pages. SVC dump processing improves service by allowing secondary extents to be specified when large dump data sets are too large for the amount of DASD previously allocated. An installation can protect itself against truncated dumps by specifying secondary extents and by leaving sufficient space on volumes to allow for the expansion of the dump data sets. For the SPACE keyword, you can specify CONTIG to make reading and writing the data set faster. Request enough space in the primary extent to hold the smallest SVC dump expected. Request enough space in the secondary extent so that the primary plus the secondary extents can hold the largest SVC dump. The actual size of the dump depends on the dump options in effect when the system writes the dump.

Note: Approximately 250 cylinders will be sufficient for most single address space SVC dump requirements.

Managing SVC dump data sets

The system writes only one dump in each SYS1.DUMPxx data set. Before the data set can be used for another dump it can be cleared by using the DUMPDS command with the CLEAR keyword. The format of the command is:

```
DUMPDS CLEAR,DSN=xx
```

Where xx is the SYS1.DUMPxx identifier. You can abbreviate the DUMPDS command to DD, for example:

```
DD CLEAR,DSN=01
```

4.11 Automatic allocation of SVC dump data sets

- ❑ SVC dump supports automatic allocation:
 - Automatic allocation when system writes the dump
 - Allocates from a set of DASD volumes of SMS classes
 - Can specify in COMMNDxx parmlib member
 - By operator after IPL with DUMPDS command
- ❑ Steps to initiate automatic allocation
 - Create a user ID for DUMPSRV address space
 - Authorize user ID
 - Create data set naming pattern
 - Add resources for dump using DUMPDS command

Figure 4-19 Automatic allocation of SVC dump data sets

Automatic allocation

SVC dump processing supports automatic allocation of dump data sets at the time the system writes the dump to DASD. The dump can be allocated from a set of DASD volumes or SMS classes. When the system captures a dump, it allocates a data set of the correct size from the resources you specify. If automatic allocation fails, pre-allocated dump data sets are used. If no pre-allocated SYS1.DUMPnn data sets are available, message IEA793A is issued, and the dump remains in virtual storage. SVC dump periodically retries both automatic allocation and writing to a pre-allocated dump data set until successful or until the captured dump is deleted either by operator intervention or by the expiration of the CHNGDUMP MSGTIME parameter governing message IEA793A.

DASD volumes and SMS classes

Once active, allocation to SMS classes and DASD volumes is done starting from the first resource you added with the DUMPDS ADD command until unsuccessful, then the next resource is used. If you have defined both DASD volumes and SMS classes, SMS classes are used first. Allocation to DASD volumes is not multivolume or striped, while allocation to SMS classes can be multivolume or striped, depending on how the storage class is set up by the installation.

COMMNDxx parmlib member

You can specify the command instructions to enable or disable automatic allocation either in the COMMNDxx parmlib member, to take effect at IPL, or from the operator console at any time after the IPL, to dynamically modify automatic allocation settings.

If you use COMMNDxx, you may want to specify DUMP=NO in the IEASYSxx parmlib member to prevent dumps taken during IPL from being written to SYS1.DUMPxx data sets.

DUMPDS command

The DUMPDS command provides the following flexibility:

- ▶ Activate automatic allocation of dump data sets
- ▶ Add or delete allocation resources
- ▶ Direct automatic allocation to SMS or non-SMS managed storage
- ▶ Deactivate automatic allocation of dump data sets
- ▶ Reactivate automatic allocation of dump data sets
- ▶ Change the dump data set naming convention

Steps to initiate automatic allocation

Automatic allocation can be set up using the following steps:

- ▶ Set up allocation authority
- ▶ Establish a name pattern for the data sets
- ▶ Define resources for storing the data sets
- ▶ Activate automatic allocation

Add resources for dump using DUMPDS command

The steps to initiate automatic dump data set allocation are:

- ▶ Associate the DUMPSRV address space with a user ID.
- ▶ Authorize DUMPSRV's user ID to create new dump data sets.
- ▶ Set up your installation data set name pattern using the DUMPDS command:

```
DUMPDS NAME=SC68;.&JOBNAME;.&Y&YR4;M&MON;.&D&DAY;T&HR;&MIN;.&S&SEQ;
```
- ▶ Add dump data set resources that can be used by the automatic allocation function:

```
DUMPDS ADD,VOL=(SCRTH1,HSM111)
DUMPDS ADD,SMS=(DUMPDA)
```
- ▶ Activate automatic dump data set allocation using the **DUMPDS** command:

```
DUMPDS ALLOC=ACTIVE
```

Note: These steps can be performed after IPL using the DUMPDS command from an operator console, or early in IPL by putting the commands in the COMMNDxx parmlib member and pointing to the member from the IEASYSxx parmlib member using CMD=xx.

4.12 Dumping multiple address spaces in a sysplex

❑ Create a SYS1.PARMLIB member - IEADMCxx

➤ Can create multiple members

❑ Requesting a dump

➤ Dumping dataspaces

❑ Considerations using SLIP entries

➤ SLIP examples

— IEASLPxx examples

Figure 4-20 Considerations for dumping multiple address spaces in a sysplex

Multiple address space dumps

To dump multiple address spaces in a sysplex environment, the following examples can be used as a guide. Create a SYS1.PARMLIB member using a IEADMCxx member containing the following DUMP parameters shown in Figure 4-21, as follows:

- ▶ job1 = IMS Control Region Jobname - job2 = IMS DLI region Jobname
- ▶ job3 = DBRC Region Jobname - job4 = IRLM Region Jobname (If IRLM DB Locking used)

```
JOBNAME=(job1,job2,job3,job4),  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ),  
REMOTE=(SYSLIST=('*job1','job2','job3','job4'),SDATA)
```

Figure 4-21 IEADMC11 example

Figure 4-22 shows the creation of a second SYS1.PARMLIB member, IEADMC12, containing the following DUMP parameters:

- ▶ job5 = CCTL Region 1 - job6 = CCTL Region 2 - job7 = CCTL Region 3

```
JOBNAME=(job5,job6,job7),  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ,XESDATA),  
REMOTE=(SYSLIST=('*job5','job6','job7'),SDATA)
```

Figure 4-22 IEADMC12 example

Requesting a dump

To request a dump to be captured as per the IEADMCI1 and IEADMCI2 parmlib members, issue the following MVS command:

```
DUMP TITLE=(IMS/CCTL sysplex DUMPS),PARMLIB=(I1,I2)
```

If the data space DSPNAME parameter is specified, for example:

```
DSPNAME=('job1'.*)
```

Two dump data sets are created on each MVS image in the sysplex matching the REMOTE specifications for the JOBNAMES. Then the same data space is dumped in the associated address spaces in the other systems if the DSPNAME parameter is included on the REMOTE statement. For example:

```
REMOTE=(SYSLIST=*( 'job1','job2','job3','job4'),SDATA,DSPNAME)
```

Considerations using SLIP entries

Figure 4-23 and Figure 4-24 shows an alternative where IEASLPxx has been used containing the following SLIP entries, using the IEASLPxx example, as follows:

- ▶ job1 = IMS Control Region Jobname
- ▶ job2 = IMS DLI region Jobname
- ▶ job3 = DBRC Region Jobname
- ▶ job4 = IRLM Region Jobname (If IRLM DB Locking is used)

```
SLIP SET,IF,N=(IEAVEDS0,00,FF),A=(SYNCSVCD,TARGETID),  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,GRSQ),  
JOBLIST=(job1,job2,job3,job4),ID=IMS1,TARGETID=(IMS2),  
REMOTE=(JOBLIST,SDATA),D,END
```

Figure 4-23 IEASLPxx example

- ▶ job5 = CCTL Region 1
- ▶ job6 = CCTL Region 2
- ▶ job7 = CCTL Region 3

```
SLIP SET,IF,N=(IEAVEDS0,00,FF),  
JOBLIST=(job5,job6,job7),ID=IMS2,  
SDATA=(CSA,PSA,RGN,SQA,SUM,TRT,XESDATA),  
REMOTE=(JOBLIST,SDATA),  
D,END
```

Figure 4-24 IEASLPxx example

Before activating the SLIP, ensure that any existing PER SLIP for each MVS image in the sysplex is disabled, as follows:

```
ROUTE *ALL,SLIP,MOD,DISABLE,ID=trapid
```

To activate the SLIP trap and trigger the associated SVC dumps, enter the following MVS commands:

```
SET SLIP=xx  
SLIP MOD,ENABLE,ID=IMS1
```

Two dumps are then be captured on each MVS image in the sysplex matching the REMOTE specifications.

4.13 Managing taking a dump

- ❑ Tasks can be canceled with a dump using:
 - **MVS CANCEL command**
 - CANCEL jobname,DUMP
- ❑ Dump analysis and elimination (DAE)
 - **Suppressing dumps**
 - Using ADYSETxx parmlib member with SET DAE=xx
- ❑ Handling partial dumps

Figure 4-25 Ways to manage taking a dump

Canceling jobs with a dump

Canceling a problem task can be initiated from either an MVS console or from an SDSF session running under TSO provided sufficient security privileges have been set up. The MVS console has the highest dispatching priority which allows commands to be issued at a sufficient level to handle most system loop or hang conditions. An IPL will be required if the problem task cannot be terminated using these procedures. Attempting to cancel a looping task via an SDSF session executing under TSO will often fail because the TSO session will have an insufficient dispatching priority to interrupt the loop process, but this is dependant on the severity of the looping process.

The CANCEL command can be performed as follows:

1. Issue the CANCEL jobname from the master console, where jobname is the looping task.
2. If the looping task is a TSO user, then issue, CANCEL U=tsouser.
3. Optionally, you might want to take a dump during the cancel. This is achieved by adding the DUMP option to the CANCEL command. For example,

```
CANCEL jobname,DUMP
```

It is recommended that a separate DUMP command be issued, and after this has been successfully processed, then CANCEL the task. This will dump according to the SYSABEND, SYSUDUMP, or SYSMDUMP DD statements specified in the JCL.

Dump analysis and elimination (DAE)

DAE suppresses dumps that match a dump you already have. Each time DAE suppresses a duplicate dump, the system does not collect data for the duplicate or write the duplicate to a data set. The ADYSETxx members in SYS1.PARMLIB control the DAE facility. If you find that dumps are being suppressed, as indicated by the following messages: IEA820I, IEA848I or IEA838I, please review DAE to ensure that you do not suppress this dump. A stop and start of DAE is required to reset the dump suppression count.

A stop of DAE is done by issuing a SET DAE=xx, where the xx in the ADYSETxx member contains a DAE=STOP,GLOBALSTOP command.

Restart DAE by SET DAE=xx, where xx is the active ADYSETxx parmlib member. This is often ADYSET00.

Partial dumps

How can you determine if the dump that has been captured is a complete dump? A partial, or incomplete dump will be missing some key areas of storage that in most cases will make the dump useless when it comes to efficient problem diagnosis.

The only other way to determine whether the dump is partial is to interrogate the dump using the Interactive Problem Control System (IPCS)—apart from the obvious message that will be generated in the z/OS system log that indicates a dump is partial, or that the dump MAXSPACE has been reached. Figure 4-26 shows an example of the IEA042I message.

```
IEA043I SVC DUMP REACHED MAXSPACE LIMIT - MAXSPACE=xxxxxxx MEG or
IEA611I PARTIAL DUMP ON dsname
```

Figure 4-26 IEA611I message indicating partial dump

Figure 4-27 shows the result of the IPCS LOCATE command that can be issued to interrogate the storage, which will indicate if the dump taken was partial. In this case we are looking at storage at address x'E0' for a length of 16 bytes.

```
Command ==> ip l e0. block(0) l(16)
***** TOP OF DATA *****
LIST E0. BLOCK(0) LENGTH(X'10') AREA
BLOCK(0) ADDRESS(E0.)
000000E0. 00000000 30000000 00000000 00000000 |.....|
*****END OF DATA *****
```

Figure 4-27 IPCS Storage Address Locate for IEA611I reason

The 4 words found at location X'E0' contain partial dump reason codes. These codes are mapped by DSECT SDRSN, and can be found in the z/OS data areas manual. The flags are also listed in *z/OS MVS System Messages, Vol 6 (GOS-IEA)*, SA22-7636 under message IEA611I. The description listed under IEA611I for x'30000000' in the second word is:

- 20000000 -The system detected an error in the SVC dump task and gave recovery control.
- 10000000 - The SVC dump task failed.

If the values displayed at location X'E0' are all zero, then the dump is not partial.

4.14 Customizing dumps using SDATA options

- ❑ Customizing dumps
 - SVC, SYSABEND, SYSUDUMP, SYSMDUMP
- ❑ Checking SDATA options
 - Use operator command - d d,o
- ❑ Creating SDATA options
- ❑ IPCS and SDATA options

Figure 4-28 Options to customize dumps

Customizing SVC dumps

You can customize the contents of an SVC dump, SYSABEND, SYSUDUMP, and SYSMDUMP dumps, to meet the needs of your installation. For example, you might want to add areas to be dumped, reduce the dump size, or dump Hiperspaces. In most cases, you will customize the contents of an SVC dump or summary dump via the SDATA parameter of the SDUMP or SDUMPX macro or with operator commands.

SDATA options

To check the SDUMP (SDATA) options in your system, enter the D D,O command on the operator console. Figure 4-29 shows an example where we can see that the SDUMP options are the default ones.

```
RESPONSE=MCEVS1
IEE857I 15.04.04 DUMP OPTION 796
  SYSABEND- ADD PARMLIB OPTIONS SDATA=(LSQA,TRT,CB,ENQ,DM,IO,ERR,SUM),
                                PDATA=(SA,REGS,LPA,JPA,PSW,SPLS)
  SYSUDUMP- ADD PARMLIB OPTIONS SDATA=(SUM), NO PDATA OPTIONS
  SYSMDUMP- ADD PARMLIB OPTIONS (NUC,SQA,LSQA,SWA,TRT,RGN,SUM)
  SDUMP- ADD OPTIONS (LSQA,TRT,XESDATA),BUFFERS=00000000K,
                    MAXSPACE=00000500M,MSGTIME=99999 MINUTES
```

Figure 4-29 Dump options

Creating SDATA options

If you need to add more options, you can use the following command:

```
CD SET,SDUMP=(PSA,LPA,RGN,SUM,SQA,CSA)
```

Enter D D,O again and you should see the update shown in Figure 4-30.

```
RESPONSE=MCEVS1
IEE857I 15.24.13 DUMP OPTION 514
  SYSABEND- ADD PARMLIB OPTIONS SDATA=(LSQA,TRT,CB,ENQ,DM,IO,ERR,SUM),
                PDATA=(SA,REGS,LPA,JPA,PSW,SPLS)
  SYSUDUMP- ADD PARMLIB OPTIONS SDATA=(SUM), NO PDATA OPTIONS
  SYSMDUMP- ADD PARMLIB OPTIONS (NUC,SQA,LSQA,SWA,TRT,RGN,SUM)
  SDUMP- ADD OPTIONS (PSA,SQA,LSQA,RGN,LPA,TRT,CSA,SUMDUMP,XESDATA),
                BUFFERS=00000000K,MAXSPACE=00000500M,
                MSGTIME=99999 MINUTES
  ABDUMP- TIMEENQ=0240 SECONDS
```

Figure 4-30 SDUMP options

IPCS and SDATA options

Figure 4-31 shows the result of the IPCS control block format of the CVT to interrogate the SDATA options that were in effect when the dump was taken. The command is:

```
cbf cvt+23c?+9c str(sdump) view(flags)
```

```
SDUMP_PL: 00FB357C

==> FLAGS SET IN SDUFLAG0:
  Set system non-dispatchable while dumping global storage.

==> FLAGS SET IN SDUFLAG1:
  SYSMDUMP request.
  SUMLIST specified.
  Ignore CHNGDUMP parameters.
  TSO user extension is present.
  48+ byte parameter list.

==> FLAGS SET IN SDUSDATA:
  Dump SQA.
  Dump LSQA.
  Dump rgn-private area.
  Dump LPA mod. for rgn.
  Dump trace data.
  Dump SWA.
  Do not dump all PSA.
```

Figure 4-31 Example of IPCS “cbf cvt+23c?+9c str(sdump) view(flags)” command

Even though the SDATA RGN parameter has been specified, the fact that some areas of RGN storage may have been paged out when the dump was taken can result in a “storage not available”.

4.15 Dump options and considerations

- ❑ IEADRM00 parmlib member
 - **SDATA=(parms,.....)**
 - **SDATA=(NUC,SQA,LSQA,SWA,TRT,LPA,CSA,RGN,GRSQ,ALLNUC,NOSYM,SUM)**
- ❑ SYSMDUMP considerations with z/OS UNIX
- ❑ CHNGDUMP command
 - **CHNGDUMP DEL** - Removing options from or resetting the system
 - **CHNGDUMP RESET** - Resetting dump mode to ADD and the dump options to initial values
 - **CHNGDUMP SET** - Setting the dump modes and options

Figure 4-32 Dump options in parmlib and commands

IEADMR00 parmlib member

IEADMR00 contains IBM defaults and/or installation parameters for ABDUMP, for use when an ABEND dump is written to a SYSMDUMP data set.

Note: During an IPL, an informational message will notify the operator if IEADMR00 is invalid or cannot be found. No prompting of the operator will occur. If the member contains both valid and invalid parameters, an informational message will indicate the valid options that were accepted before the error occurred.

SYSMDUMP data set

ABDUMP parameters for a SYSMDUMP data set may be specified as follows:

- ▶ The dump request parameter list pointed to by the DUMPOPT keyword of an ABEND macro. The list can be built by using the list form of the SNAP macro.
- ▶ The initial system dump options specified in IEADMR00. These options are added to the options on the dump request parameter list.
- ▶ The system dump options as altered by the CHNGDUMP command. With the CHNGDUMP command, options can be added to or deleted from the system dump options list. The CHNGDUMP command can also cause the dump request parameters to be ignored.

4.16 Catalog address space (CAS) dumps

❑ DUMP command for CAS

- F CATALOG,DUMPON
- F CATALOG,DUMPON(aaa,bbb,cc)
- F CATALOG,DUMPON(aaa,bbb,cc,nnn)
 - aaa The catalog rc in decimal (000-255), or ***
 - bbb The catalog rc in decimal (000-255), or ***
 - cc The catalog module identifier in CAS, or **
- DUMPON option
- DUMPOFF option

Figure 4-33 Catalog address space dumps

MODIFY CATALOG,DUMPON syntax

MODIFY or F CATALOG, DUMPON or DUMPOFF specifies whether CAS dynamic dumping is to occur. Dynamic dumping by CAS does not occur unless you specify DUMPON.

- ▶ MODIFY CATALOG,DUMPON
- ▶ MODIFY CATALOG,DUMPON(aaa,bbb,cc)
- ▶ MODIFY CATALOG,DUMPON(aaa,bbb,cc,nnn)

Where:

- aaa - The catalog return code in decimal (000-255), or ***
- bbb - The catalog reason code in decimal (000-255), or ***
- cc - The catalog module identifier in CAS, or **
- nnn - The limit number in decimal (000-999)

Options in parenthesis that follow the DUMPON parameter can be used to create a dump whenever a given return code, reason code, and module identifier occur. This dump can prove valuable to service personnel in solving problems. Normally, the return code, reason code, and module identifier are available on return from CAS and are printed by IDCAMS. The module identifier corresponds to the last two characters in the catalog module name. For example, the module identifier is A3 for IGG0CLA3. The return code, reason code, and module identifiers may be specified as a string of asterisks to indicate any value encountered will match the value of that field. This is referred to as a generic match. All three fields may not

be simultaneously specified as asterisks. Whenever a generic match is specified for a particular field, it will be assumed that field always matches the value being returned by catalog for a catalog request. As an example:

```
MODIFY CATALOG,DUMPON(008,042,**)
```

will create a dump for any return code 8, reason code 42, regardless of the module that detected the error. An option has been provided for a match count to obtain the nth occurrence of a return code, reason code, and module identifier. The match count decrements by one each time a return code, reason code, and module identifier is set in the catalog address space. If this option is not specified or is set to 000, then the first occurrence causes a dump.

Only one set of return codes, reason codes and module identifiers can be set at a time. Each entry overwrites the previous information. Once a match occurs, the information is cleared and the original DUMPON status is maintained. If DUMPON is entered without the additional options, certain conditions will produce dumps automatically. If then a DUMPON with options is entered, a match will cause a dump and the return code, reason code and module identifier will be cleared. The DUMPON status will remain on.

MODIFY CATALOG,REPORT,DUMP can be used to view the settings.

The header for the catalog dynamic dump will contain the return code and reason code in hex. For example:

```
CAS DYNAMIC DUMP-IGG0CLA9 RCX'00' RSNX'00'
```




z/OS trace processing

Another useful source of diagnostic data is the trace. Tracing collects information that identifies ongoing events that occur over a period of time. Some traces are running all the time so that trace data will be available in the event of a failure. Other traces must be explicitly started to trace a defined event.

In this chapter, the following trace activity is described:

- ▶ GTF trace
- ▶ Component trace
- ▶ Master trace
- ▶ GFS trace
- ▶ System trace
- ▶ SMS tracing

5.1 z/OS trace facilities

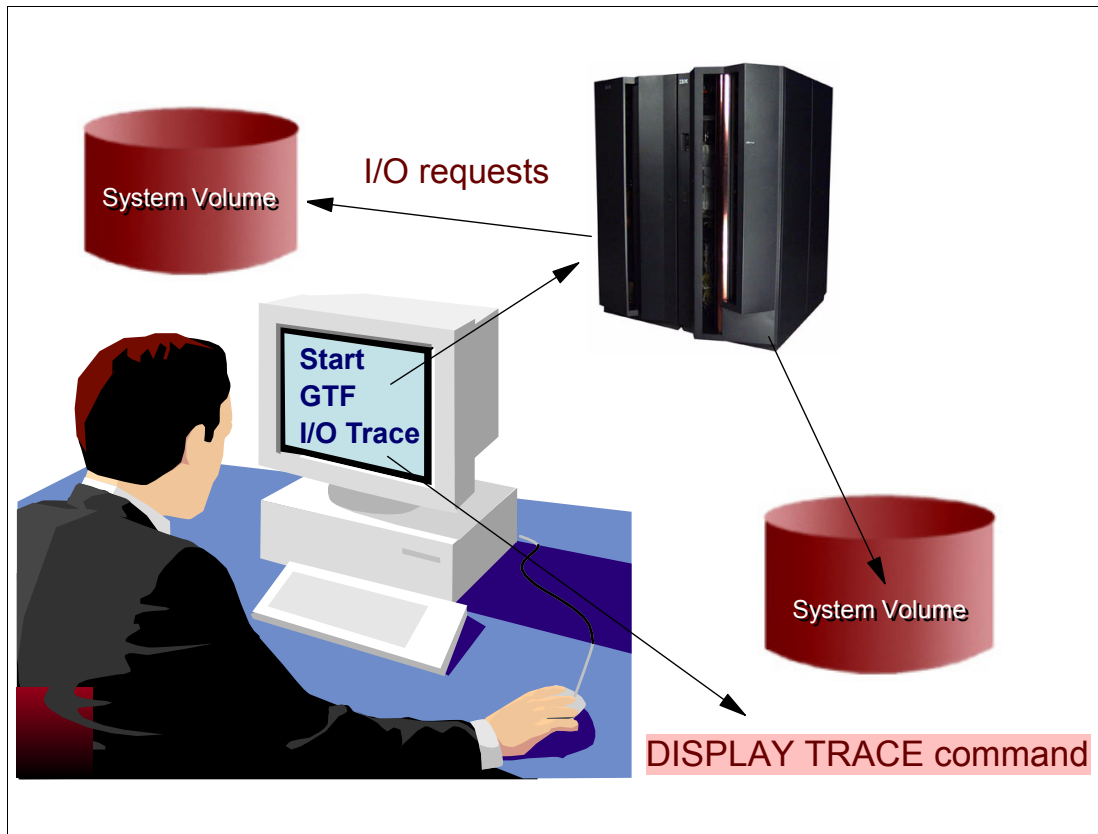


Figure 5-1 z/OS trace facilities

GTF trace facility

The generalized trace facility (GTF) is a service aid you can use to record and diagnose system and program problems. GTF is part of the MVS system product, and you must explicitly activate it by entering a START GTF command.

Use GTF to record a variety of system events and program events on all of the processors in your installation. If you use the IBM-supplied defaults, GTF lists many of the events that system trace lists, showing minimal data about them.

I/O trace

GTF builds an I/O record when an I/O interruption occurs and TRACE=SYSM, TRACE=SYS, TRACE=IO, or TRACE=IOP are the GTF options in effect. To trace PCI I/O interruptions, TRACE=PCI must also be in effect.

SYS1.TRACE

When you start GTF, a trace output data set is created and has the name SYS1.TRACE. The data set resides on a DASD that is large enough for the data set to contain 20 physical blocks. After completely filling the 20 physical blocks, GTF will overlay previously written records with new trace records, starting at the beginning of the output data set.

I/O requests

When you start GTF, one of the options is to trace I/O requests. GTF then requests recording of all nonprogram-controlled I/O interruptions. Unless you also specify the PCI trace option, GTF does not record program-controlled interruptions.

Using the DISPLAY TRACE command

To display the current trace option in effect issue the MVS DISPLAY TRACE command. Figure 5-2 shows an example of the output generated by the DISPLAY TRACE command.

It shows that we have system trace (ST) enabled, with 256K allocated for the system trace table on each processor and 3584K allocated to the system trace table buffers. Address space (AS) tracing is ON and branch tracing is OFF, as is explicit software tracing. Master tracing is ON with a master trace table size of 24K. This also displays the status of component and sub-component traces.

```
RESPONSE=MCEVS1
IEE843I 19.30.33 TRACE DISPLAY 177
      SYSTEM STATUS INFORMATION
ST=(ON,0256K,03584K) AS=ON BR=OFF EX=ON MT=(ON,024K)
COMPONENT MODE  COMPONENT MODE  COMPONENT MODE  COMPONENT MODE
-----
SYSGRS    MIN   SYSTCPRT  OFF   SYSJES2    OFF   SYSANT00  MIN
SYSANT01  MIN   SYSRRS    MIN   SYSSPI     OFF   SYSJES    OFF
SYSSMS    OFF   SYSOPS    ON    SYSXCF     ON    SYSLLA    MIN
SYSXES    ON    SYSTTRC   OFF   SYSTCPDA   OFF   SYSRSM    OFF
SYSAOM    OFF   SYSVLF    MIN   IRLM       OFF   SYSTCPIP  OFF
SYSLOGR   ON    SYSOMVS   ON    SYSWLM     MIN   SYSTCPIS  OFF
SYSTCPRE  OFF   SYSIOS    MIN   JRLM       OFF   SYSIEFAL  ON
```

Figure 5-2 Display trace command output

To get information for a single trace such as SYSOMVS, issue D TRACE,COMP=SYSOMVS. The output of this command (Figure 5-3) shows that the internal trace buffer size for OMVS is 4 MB.

```
RESPONSE=MCEVS1
IEE843I 19.36.21 TRACE DISPLAY 490
      SYSTEM STATUS INFORMATION
ST=(ON,0256K,03584K) AS=ON BR=OFF EX=ON MT=(ON,024K)
COMPONENT      MODE BUFFER HEAD SUBS
-----
SYSOMVS        ON   0004M
ASIDS          *NONE*
JOBNAMES       *NONE*
OPTIONS        ALL
WRITER         *NONE*
```

Figure 5-3 Display trace,comp=tracename output

5.2 GTF trace definitions

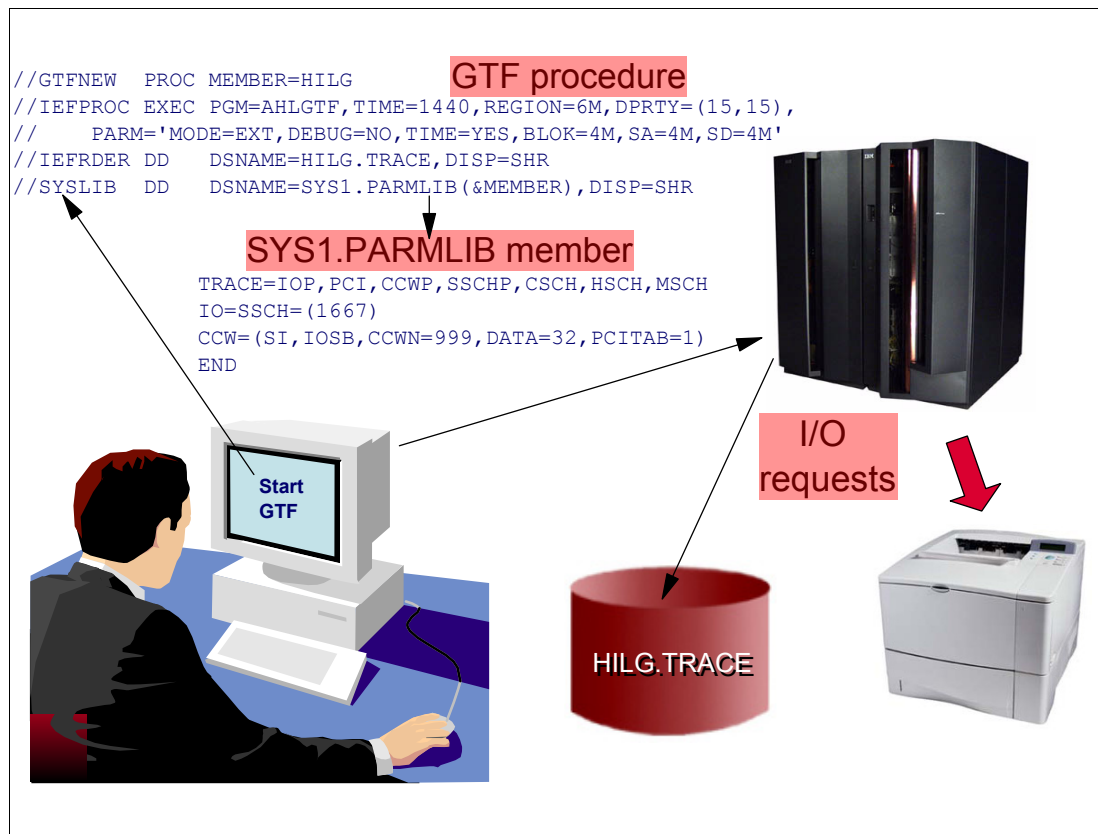


Figure 5-4 GTF processing

Start GTF trace

Use a GTF trace to show system processing through events occurring in the system over time. The installation controls which events are traced. GTF tracing uses more resources and processor time than a system trace. Use GTF when you are familiar enough with the problem to pinpoint the one or two events required to diagnose your system problem. GTF can be run to an external data set as well as a buffer.

GTF procedure

When you activate GTF, it operates as a system task, in its own address space. The only way to activate GTF is to enter a START GTF command from a console with master authority. Using this command, the operator selects either the IBM or your cataloged procedure for GTF. The cataloged procedure defines GTF operation; you can accept the defaults that the procedure establishes, or change the defaults by having the operator specify certain parameters on the START GTF command.

Because GTF sends messages to a console with master authority, enter the command only on a console that is eligible to be a console with master authority. Otherwise, you cannot view the messages from GTF that verify trace options and other operating information.

IBM supplies the GTF cataloged procedure, which resides in SYS1.PROCLIB. This procedure defines GTF operation, including storage needed, where output is to go, recovery for GTF, and the trace output data sets. Figure 5-5 on page 111 shows the format of the IBM-supplied GTF procedure.

```
//GTF PROC MEMBER=GTFARM
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
// TIME=1440,REGION=2880K
//IEFRDER DD DSN=SYS1.TRACE,UNIT=SYSDA,SPACE=(TRK,20),
// DISP=(NEW,KEEP)
//SYSLIB DD DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

Figure 5-5 GTF procedure

SYS1.PARMLIB member for GTF

GTFARM provides default or installation-defined trace options to control the generalized trace facility (GTF). The member is read only when the operator (or an automatic command) issues START GTF. GTFARM is not used during system initialization.

The member name on the START GTF command can be the same as the IBM-supplied cataloged procedure, GTF. The PROC statement of that procedure identifies GTFARM as the member from which GTF will get its trace parameters. If the installation wants to place the GTFARM member in a data set other than SYS1.PARMLIB, specify the alternate data set in the SYSLIB DD statement and then specify a member from that PDS using the MEMBER keyword, as shown in Figure 5-4 on page 110. If the installation wants to substitute another member in place of GTFARM, as shown in the figure, the operator may enter the replacement member name on the START command with the MEMBER keyword.

Trace data to external devices

The two primary locations that are used to store GTF trace data are as follows:

- ▶ A data set on DASD
- ▶ Internal storage

The benefit of writing to internal storage is that if the trace is being taken to be reviewed in conjunction with a dump, the GTF in-storage buffers will be dumped along with the address space. You will have trace and dump data taken at the same time, and this can be reviewed using IPCS.

Note: If you need to trace for an extended period of time, then writing to an external device is advisable.

5.3 Implementing GTF trace

- ❑ Defining GTF trace options
 - GTF procedure options
 - GTFPARM member options
 - GTF trace options
- ❑ Starting GTF
 - {START | S} {GTF | membername}.identifier
- ❑ Stopping GTF
 - {STOP | P} identifier
 - Display identifier - D A,LIST
- ❑ GTF tracing for reason code interrogation

Figure 5-6 Implementing GTF tracing

Defining the GTF trace options

The GTF options can be specified through either system prompting in response to the START GTF command or in a predefined parmlib member or data set member. However, GTF will not use certain combinations of options. Figure 5-7 shows the GTF trace option meanings.

SYSM	Selected system events
USR	User data that the GTRACE macro passes to GTF
TRC	Trace events associated with GTF itself
DSP	Dispatchable units of work
PCI	Program-controlled I/O interruptions
SRM	Trace data associated with the system resource manager (RSM)

Figure 5-7 GTF trace options

Note: For these combinations, and regarding other GTF options, see *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085.

GTF procedure options

We recommend that GTF be started with the following parameters, which are specified in the GTF procedure in SYS1.PROCLIB:

```
PARM='MODE(INT)' and REGION=2880K
```

Options specified on the PARM parameter specify where GTF writes trace data and the amount of storage needed for GTF to collect and save trace data in various dump types (Figure 5-8).

```
MODE={INT|EXT|DEFER}  
SADMP={nnnnnnK|nnnnnnM|40K}  
SDUMP={nnnnnnK|nnnnnnM|40K}  
NOPROMPT  
ABDUMP={nnnnnnK|nnnnnnM|0K}  
BLOK={nnnnn|nnnnnK|nnnnnM|40K}  
SIZE = {nnnnnnK|nnnnnnM|1024K}  
TIME=YES  
DEBUG={YES|NO}
```

Figure 5-8 GTF parameters on the PARM= in the GTF procedure

The GTF parameters SADMP, SDUMP, ABDUMP and BLOK parameters should all be set to at least 10 MB.

GTFPARM member

Figure 5-7 on page 112 shows the IBM-supplied GTFPARM parmlib member, which contains the GTF trace options, as follows:

```
TRACE=SYSM,USR,TRC,DSP,PCI,SRM
```

Note: The member containing predefined trace options does not have to reside in the parmlib member. GTF will accept any data set specified in the SYSLIB DD statement of the cataloged procedure, or in the START command, as long as that data set's attributes are compatible with those of SYS1.PARMLIB.

Starting GTF

To invoke GTF, the operator issues the following START command:

```
{START|S}{GTF|membername}.identifier
```

After the operator enters the START command, GTF issues message AHL100A or AHL125A to allow the operator either to specify or to change trace options. If the cataloged procedure or START command did not contain a member of predefined options, GTF issues message AHL100A so the operator may enter the trace options you want GTF to use. If the procedure or command did include a member of predefined options, GTF identifies those options by issuing the console messages AHL121I and AHL103I. Then you can either accept these options, or reject them and have the operator respecify new options. Figure 5-9 shows the sequence of messages that appear on the console when starting GTF.

```

START GTF.EXAMPLE1
AHL121I TRACE OPTION INPUT INDICATED FROM MEMBER GTFPARM OF PDS SYS1.PARMLIB
TRACE=SYSM,USR,TRC,DSP,PCI,SRM
AHL103I TRACE OPTIONS SELECTED--SYSM,USR,TRC,DSP,PCI,SRM
*451 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
REPLY 451,U
AHL031I GTF INITIALIZATION COMPLETE

```

Figure 5-9 GTF start-up messages

Stopping GTF

The operator can enter the STOP command at any time during GTF processing. The amount of time you let GTF run depends on your installation and the problem you are trying to capture, but a common time is between 15 and 30 minutes.

To stop GTF processing, have the operator enter the STOP command. This command must include either the GTF identifier specified on the START command, or the device number of the GTF trace data set if you specified MODE=EXT or MODE=DEFER to direct output to a data set.

If you are not sure of the identifier, or the device number of the trace data set, ask the operator to enter the DISPLAY A,LIST command. Figure 5-10 shows the result of this command and the GTF identifier displayed is EVENT1.

```

DISPLAY A,LIST
IEE114I 14.51.49 2005.181 ACTIVITY FRAME LAST F E SYS=SY1
JOBS M/S TS USERS SYSAS INITS ACTIVE/MAX VTAM OAS
00000 00003 00000 00016 00000 00000/00000 00000
LLA LLA LLA NSW S VLF VLF VLF NSW S
JES2 JES2 IEFPROC NSW S
GTF EVENT1 IEFPROC NSW S
.....

```

Figure 5-10 D A,LIST command

The operator must enter the STOP command at a console with master authority. The general format of the command is:

```
{STOP|P} identifier
```

When the STOP command takes effect, the system issues message AHL006I. If the system does not issue this message, then GTF tracing continues, remaining active until a STOP command takes effect, or until the next initial program load (IPL). When this happens, you will not be able to restart GTF tracing. In this case, you can use the FORCE ARM command to stop GTF. If there were several functions started with the same identifier on the START command, using the same identifier on the STOP command will stop all those functions.

GTF tracing for reason code interrogation

In some instances your software support provider may ask you to capture a GTF trace that will contain all the reason codes issued by a particular job. This is more likely if the reason code is not reported externally. If you choose to look at such a GTF trace, be aware that many reason codes are issued validly and do not represent real errors (that is, reason codes that indicate file not found are usually quite valid).

Prior to setting the slip below you would need to start GTF with options TRACE=SLIP. The slip that would be set is:

```
SLIP SET,IF,A=TRACE,RANGE=(10?+8C?+F0?+1f4?),TRDATA=(13R??+B0,+B3),END
```

After recreating the problem, stop GTF and format the output using the IPCS command GTFTRACE.

5.4 Component trace (CTTRACE)

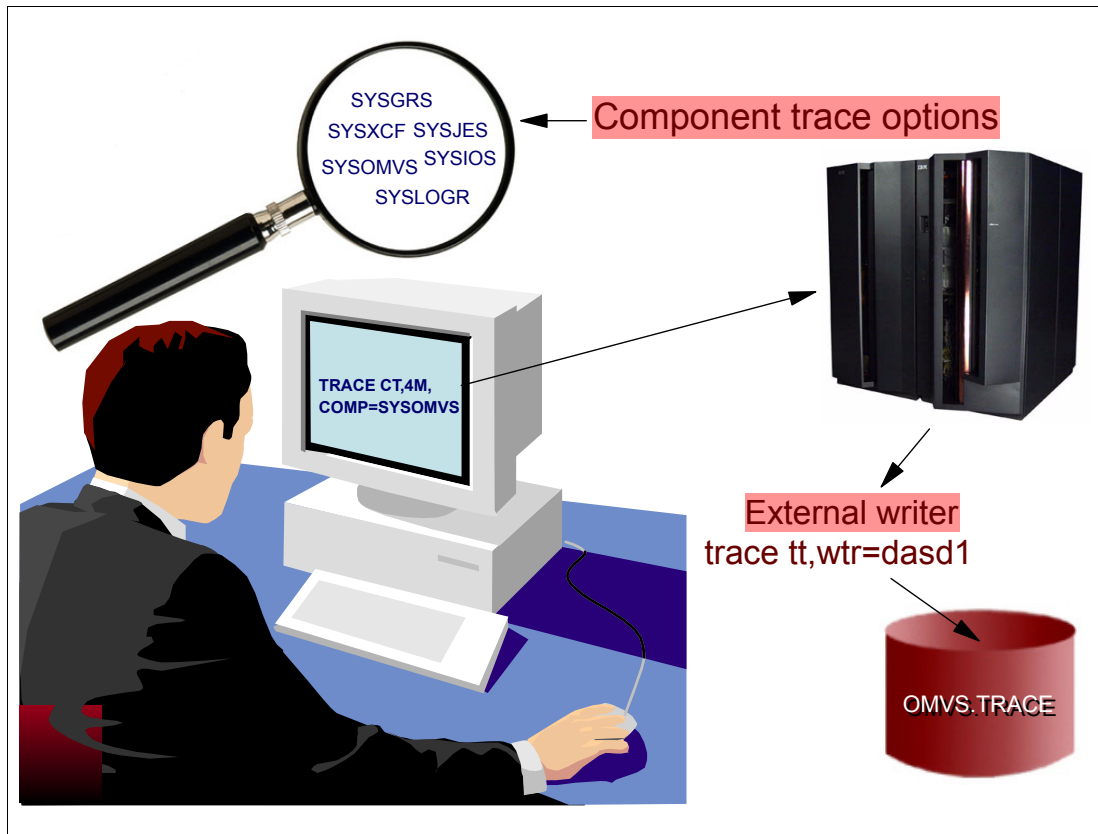


Figure 5-11 Implementing component trace (CTTRACE)

Component trace

The component trace service provides a way for MVS components to collect problem data about events. Each component that uses the component trace service has set up its trace in a way that provides the unique data needed for the component.

A component trace provides data about events that occur in the component. The trace data is intended for the IBM Support Center, which can use the trace to:

- ▶ Diagnose problems in the component
- ▶ See how the component is running

If the IBM Support Center requests a trace, the Center might specify the options, if the component trace uses an `OPTIONS` parameter in its parmlib member or `REPLY` for the `TRACE CT` command. The options are:

```
SYSAPPC SYSDLF SYSDSOM SYSGRS SYSIEFA SYSIOS SYSJES SYSjes2 SYSLLA SYSLOGR  
SYSOMVS SYSOPS SYSRRS SYSRSM SYSTRC SYSSPI SYSVLF SYSWLM SYSXCF SYSXES
```

You will typically use component trace while recreating a problem. The installation, with advice from the IBM Support Center, controls which events are traced for a system component. GTF does not have to be active to run a component trace.

External writer for tracing

Transaction trace supports the use of an external writer for processing transaction trace records. An external writer can be specified on the initial command that activates transaction trace, or specified standalone while transaction trace is active. Transaction trace uses the MVS TRACE command with the TT keyword to start an external writer. For example:

```
trace tt,wtr=prt1
```

Component trace messages are issued in response to this command. Transaction trace writes trace data in a transaction trace data space in the trace address space. If an external writer has been defined, the record is also written to the external writer. IPCS is used to view the transaction trace records.

Transaction trace external writer processing can be stopped with the use of the WTR=OFF keyword. For example:

```
trace tt,wtr=off
```

Component trace messages are issued in response to this command.

The transaction trace TRACE TT command allows the transaction trace data space size to be changed. The data space can be from 16 K to 999 K or 1 MB to 32 MB. For example:

```
trace tt,bufsiz=2m
```

The following message is issued:

```
ITZ002I 'BUFSIZ' IS SET TO 0002M
```

Note: In the example in Figure 5-11 on page 116, the operator is requesting a component trace for SYSOMVS and the external writer writes the data to a DASD data set named OMVS.TRACE

5.5 Implementing component trace

- ❑ **SYS1.PARMLIB definitions**
 - IBM-supplied CTIITT00 member
 - PARM=CTIITT00 can be specified on a TRACE CT command
- ❑ **Specifying trace options**
- ❑ **Collecting trace records**
- ❑ **Starting component trace**

Figure 5-12 Options for implementing component trace

Parmlib member definitions

The CTncccx parmlib member specifies component trace options. There is a table in *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085, that shows whether a component has a parmlib member. It indicates whether the member is a default member needed at system or component initialization, and whether the component has default tracing. Some components run default tracing at all times when the component is running; default tracing is usually minimal and covers only unexpected events. Other components run traces only when requested. When preparing your production SYS1.PARMLIB system library, do the following:

- ▶ Make sure the parmlib contains all default members identified in the table. If the parmlib does not contain the default members at initialization, the system issues messages. The table contains the following members:

SYSAPPC SYSDLF SYSDSOM SYSGRS SYSIEFA SYSIOS SYSJES SYSjes2 SYSLLA SYSLOGR
SYSOMVS SYSOPS SYSRRS SYSRSM SYSTRC SYSSPI SYSVLF SYSWLM SYSXCF SYSXES
- ▶ Make sure that the IBM-supplied CTIITT00 member is in the parmlib. PARM=CTIITT00 can be specified on a TRACE CT command for a component trace that does not have a parmlib member; CTIITT00 prevents the system from prompting for a REPLY after the TRACE CT command. In a sysplex, CTIITT00 is useful to prevent each system from requesting a reply.

Trace options for support center

If the IBM Support Center requests a trace, the Center might specify the options, if the component trace uses an OPTIONS parameter in its parmlib member, or REPLY for the

TRACE CT command. You must specify all options you would like to have in effect when you start a trace. Options specified for a previous trace of the same component do not continue to be in effect when the trace is started again. If the component has default tracing started at initialization by a parmlib member without an OPTIONS parameter, you can return to the default by doing one of the following:

- ▶ Stop the tracing with a TRACE CT,OFF command.
- ▶ Specify OPTIONS() in the REPLY for the TRACE CT command or in the CTnccccxx member.

Collecting trace records

Depending on the component, the potential locations of the trace data are:

- ▶ In address-space buffers, which are obtained in a dump
- ▶ In data-space buffers, which are obtained in a dump
- ▶ In a trace data set or sets, if supported by the component trace

If the trace records of the trace you want to run can be placed in more than one location, you need to select the location. For a component that supports trace data sets, you should choose trace data sets for the following reasons:

- ▶ Because you expect a large number of trace records
- ▶ To avoid interrupting processing with a dump of the trace data
- ▶ To keep the buffer size from limiting the amount of trace data
- ▶ To avoid increasing the buffer size

Starting component trace

Select how the operator is to request the trace. The component trace is started by either of the following:

- ▶ A TRACE CT operator command without a PARM parameter, followed by a reply containing the options
- ▶ A TRACE CT operator command with a PARM parameter that specifies a CTnccccxx parmlib member containing the options

To start a component trace, the operator enters a TRACE operator command on the console with MVS master authority. The operator replies with the options that you specified. Instead of using ON on the START command you can provide a trace buffer size, depending on the component you would like to start the trace, as follows:

```
trace ct,on,comp=sysxcf
* 21 ITT006A ....
r 21,options=(serial,status),end
```

This example requests the same trace using parmlib member CTWXCF03. When TRACE CT specifies a parmlib member, the system does not issue message ITT006A.

```
trace ct,on,comp=sysxcf,parm=ctwxcf03
```

It is possible to provide the CTRACE buffer size request on the start command. The following shows the START TRACE command for USS requesting 4 MB:

```
trace ct,4M,comp=sysomvs
```

5.6 Component trace for System Logger

- ❑ PARMLIB member example
 - CTnccccxx - Parmlib member skeleton
 - CTILOG00 - System Logger parmllib member
- ❑ System Logger trace as an example
- ❑ Set up CTRACE options
- ❑ Operator command to display trace status
- ❑ SYS1.PARMLIB defintions
- ❑ Starting the trace

Figure 5-13 Setting up component trace for System Logger

Parmlib member example

An example for a parmllib definition for z/OS UNIX is:

```
CTnccccxx      -  
CTILOG00 -      z/OS UNIX parmllib member  
(which must be specified in the BPXPRM00 member)
```

Where LOG is the ccc, and 00 is the xx and L is the n. For some components, you need to identify the component's CTnccccxx member in another parmllib member. See the parmllib member listed in the default member column in the table in *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085.

Tracing System Logger

More subsystems are now using the z/OS System Logger for logging activity that can be used during unit-of-recovery processing. This data was previously managed by the subsystems, such as CICS, DB2, and MQ, but now the System Logger address space (IXGLOGR) manages the system and subsystem log data. This can reside in a Coupling Facility, or on DASD.

CTTRACE options

Problems with Logger process will often require some additional trace data, which can be collected by setting up the CTRACE for System Logger data as follows:

- Issue the following command to display the current SYSLOGR trace status:
D TRACE,COMP=SYSLOGR
- To update the CTRACE component for the z/OS System Logger, edit the SYS1.PARMLIB member CTILOGxx. CTILOG00 is the supplied Logger CTRACE member.

Parmlib definitions

Figure 5-14 shows the CTILOGxx parmlib member and the specified options.

```
TRACEOPTS ON
BUFSIZE(8M)
OPTIONS('CONNECT','DATASET','SERIAL','STORAGE',
'LOGSTRM','MISC','RECOVERY','LOCBUFF')
```

Figure 5-14 CTILOGxx parmlib member

For CTRACE, we recommend a 10 MB buffer size. The default is 2 MB.

Operator command to display status

Figure 5-15 shows the results of the DISPLAY TRACE command for component SYSLOGR.

```
IEE843I 01.11.36 TRACE DISPLAY 967
      SYSTEM STATUS INFORMATION
ST=(ON,0064K,00128K) AS=ON BR=OFF EX=ON MT=(ON,024K)
COMPONENT      MODE BUFFER HEAD SUBS
-----
SYSLOGR        MIN 0002M
ASIDS          *NONE*
JOBNAMES       *NOT SUPPORTED*
OPTIONS        MINIMAL TRACING ONLY
WRITER         *NONE*
```

Figure 5-15 DISPLAY TRACE,COMP=SYSLOGR output

Starting the trace

This parmlib member will be used when you issue the following command:

```
TRACE CT,COMP=SYSLOGR,PARM=CTILOGxx
```

There is minimal overhead with the MVS Logger CTRACE.

To start the CTRACE for the z/OS Logger and change the trace parameters, you can issue:

```
TRACE CT,8M,COMP=SYSLOGR
R xx,OPTIONS=(ALL),END
```

5.7 Master trace

- ❑ **Parmlib definitions**
 - **SCHEDxx member in SYS1.PARMLIB**
- ❑ **Starting the master trace**
 - **Change the trace table size and then start**
 - **TRACE MT,500K**
 - **TRACE MT**
 - **TRACE MT,OFF**
- ❑ **Master trace table output with IPCS**

Figure 5-16 Master trace

Master trace

Master trace maintains a table of the system messages that are routed to the hardcopy log. This creates a log of external system activity, while the other traces log internal system activity. Master trace is activated automatically at system initialization, but you can turn it on or off using the TRACE command.

Master trace can help you diagnose a problem by providing a log of the most recently issued system messages. For example, master trace output in a dump contains system messages that may be more pertinent to your problem than the usual component messages issued with a dump.

Use the master trace to show the messages to and from the master console. Master trace is useful because it provides a log of the most recently-issued messages. These can be more pertinent to your problem than the messages accompanying the dump itself. Master tracing is usually activated at IPL time and the data can be reviewed with IPCS and is saved when an SVC dump or stand-alone dump is taken.

Parmlib definitions

At initialization, the master scheduler sets up a master trace table of 24 kilobytes. A 24-kilobyte table holds about 336 messages, assuming an average length of 40 characters. You can change the size of the master trace table or specify that no trace table be used by changing the parameters in the SCHEDxx member in SYS1.PARMLIB.

Starting the master trace

You can change the size of the master trace table using the TRACE command. For example, to change the trace table size to 500 kilobytes, enter:

```
TRACE MT,500K
```

Start, change, or stop master tracing by entering a TRACE operator command from a console with master authority. For example, to start the master tracing:

```
TRACE MT
```

To stop master tracing:

```
TRACE MT,OFF
```

You can also use the TRACE command to obtain the current status of the master trace. The system displays the status in message IEE839I. For example, to ask for the status of the trace, enter:

```
TRACE STATUS
```

Master trace table output

The following shows a sample of the master trace table. This is an in-storage copy of the system log (SYSLOG) and the amount of data contained in the table is dependant on the size of the table. Figure 5-17 shows a sample of the data contained in the Master Trace (MTRACE).

```
2003062 03:48:04.21 STC08076 00000090 ITS010 SYS 1: READY FOR COMMUNICATION
2003062 03:48:33.24 STC04022 00000094 >+CSQX500I =MQM1 CSQXRCTL Channel MQM1.ITS0810 started
2003062 03:49:03.39 STC04022 00000094 >+CSQX202E =MQM1 CSQXRCTL Connection or remote listener
152 00000094 > channel MQM1.ITS0810,
152 00000094 > connection 9.9.9.90,
152 00000094 > TRPTYPE=TCP RC=00000468
2003062 03:49:03.42 STC04022 00000094 >+CSQX599E =MQM1 CSQXRCTL Channel MQM1.ITS0810 ended
2003062 03:50:01.85 ZZ4NM002 00000294 $RALL,R=*,D=W91A.*,Q=789
2003062 03:50:01.89 ZZ4NM002 00000084 $HASP683 NO JOBS OR DATA SETS REROUTED
```

Figure 5-17 IPCS MTRACE output

5.8 GFS trace

- ❑ **DIAGxx parmlib member**
 - **Defines GFS trace control parameters**
 - **VSM TRACE GETFREE (ON) ASID (3, 5-9) LENGTH (24) DATA (ALL)**
- ❑ **Starting a GTF trace for GFS data**
 - **Use a GTF cataloged procedure**
- ❑ **Stopping GTS trace**
 - **Create a DIAGxx parmlib member**
 - **VSM TRACE GETFREE(OFF)**
- ❑ **Obtaining GFS trace data**
 - **Use IPCS GTFTRACE command**

Figure 5-18 GFS trace

GFS trace

GFS trace is a diagnostic tool that collects information about the use of the GETMAIN, FREEMAIN, or STORAGE macro. You can use GFS trace to analyze the allocation of virtual storage and identify users of large amounts of virtual storage. Use the generalized trace facility (GTF) to get the GFS trace data output.

DIAGxx parmlib member

The DIAGxx parmlib member syntax is shown in Appendix C.1.1, “DIAGxx parmlib member syntax” on page 310.

IBM provides the following parmlib members:

- | | |
|-------------------------|---|
| DIAG00 (default) | Sets storage tracking on and GFS trace off. |
| DIAG01 | Sets storage tracking on but does not change GFS trace settings. |
| DIAG02 | Sets storage tracking off but does not change GFS trace settings. |

The following procedure explains how to request a GFS trace:

1. In the DIAGxx parmlib member, set the VSM TRACE GETFREE parameter to ON and define the GFS trace control data.
 - a. The following DIAGxx parmlib member starts GFS trace and limits the trace output to requests to obtain or release virtual storage that is 24 bytes long and resides in address spaces 3, 5, 6, 7, 8, and 9, as follows:

```
VSM TRACE GETFREE (ON) ASID (3, 5-9) LENGTH (24) DATA (ALL)
```

Note: If you want the IPCS GTFTRACE output to be formatted, you must include the TYPE and FLAGS data items on the DATA keyword specification of the DIAGxx parmlib member.

You will need another DIAGxx parmlib member defined to stop GFS tracing specifying:

```
VSM TRACE GETFREE (OFF)
```

2. Ask the operator to enter the SET DIAG=xx command to activate GFS trace using the definitions in the DIAGxx parmlib member.
3. Start a GTF trace (ask the operator to enter a START membername command on the master console). The membername is the name of the member that contains the source JCL (either a cataloged procedure or a job). Tell the operator to specify a user event identifier X'F65' to trace GTF user trace records.

Starting a GTF trace for GFS data

The operator starts GTF tracing with cataloged procedure GTFPROC to get GFS data in the GTF trace output. The contents of cataloged procedure GTFPROC are shown in Figure 5-19.

The operator then replies to messages AHL100A with the USRP option. When message AHL101A prompts the operator for the keywords for option USRP, the operator replies with USR=(F65) to get the GFS user trace records in the GTF trace output.

```
//GTF      PROC MEMBER=GTFPROC
// * Starts GTF
//IEFPROC EXEC PGM=AHLGTF,REGION=32M,TIME=YES,
//  PARM='MODE=EXT,DEBUG=NO,TIME=YES,BLOK=40K,SD=OK,SA=40K'
//IEFRDER DD DSN=MY.GTF.TRACE,
//          DISP=SHR,UNIT=3390,VOL=SER=VOL001
```

Figure 5-19 GTF procedure for GFS trace

Stopping GTF trace

To stop the GTF trace, ask the operator to enter a STOP procname command on the master console. To stop GFS trace, create a DIAGxx parmlib member with:

```
VSM TRACE GETFREE(OFF)
```

The operator then enters the SET DIAG=xx command, where xx points to the created DIAGxx parmlib member.

Obtaining GFS trace data

GTF places the GFS trace data in a user trace record with event identifier X' F65'. To obtain GFS trace data, do one of the following:

1. When GTF writes the trace data to a data set, format and print the trace data with the IPCS GTFTRACE subcommand.
2. When GTF writes trace data only in the GTF address space, use IPCS to see the data in an SVC dump. Request the GTF trace data in the dump through the SDATA=TRT dump option.
3. Issue the IPCS GTFTRACE subcommand to format and see the trace in an unformatted dump. See the output in Appendix C.1.2, "GFS trace data" on page 310.

5.9 System trace

- ❑ Using system trace
- ❑ Controlling trace table size
 - **TRACE ST,256K**
- ❑ Tracing branch instructions
 - **TRACE ST,BR=ON**
 - Such as BALR, BASR, BASSM, and BAKR
- ❑ Problem determination
 - System tracing will be captured in all dump situations by default, except during a SNAP dump
 - **SDATA=TRT** must be specified

Figure 5-20 System trace

System trace

System trace provides an ongoing record of hardware and software events occurring during system initialization and operation. The system activates system tracing at initialization and the tracing runs continuously, unless your installation has changed the IBM-supplied system tracing. After system initialization, you can use the TRACE operator command on a console with master authority to customize system tracing.

Because system trace usually runs all the time, it is very useful for problem determination. While system trace and the general trace facility (GTF) list many of the same system events, system trace also lists events occurring during system initialization, before GTF tracing can be started. System trace also traces branches and cross-memory instructions, which GTF cannot do.

System trace writes trace data in system trace tables in the trace address space. It maintains a trace table for each processor. You can obtain the trace data in a dump that includes option **SDATA=TRT**.

Using system trace

Use system trace to see system processing through events occurring in the system over time. System tracing is activated at initialization and, typically, runs continuously. It records many system events, with minimal detail about each. The events traced are predetermined, except for branch tracing. This trace uses fewer resources and is faster than a GTF trace.

System trace tables reside in fixed storage on each processor. The default trace table size is 64 kilobytes per processor, but you can change it using the TRACE ST command. We do not recommend running with trace tables smaller than the default 64 kilobytes.

Controlling trace table size

You might, however, want to increase the size of the system trace table from the default 64 kilobytes. Issue the following command to increase the system trace table size to 256K:

```
TRACE ST,256K
```

Tracing branch instructions

System tracing allows you the option of tracing branch instructions, such as BALR, BASR, BASSM, and BAKR, along with other system events. If you want to trace branch instructions, use the BR=ON option on the TRACE ST command when you start tracing, as follows:

```
TRACE ST,BR=ON
```

Note: With branch tracing on, this can affect your system performance and use very large amounts of storage. Do not use branch tracing as the default for system tracing on your system. You should only use it for short periods of time to solve a specific problem. The default system tracing does not include branch instructions.

Problem determination

Because system trace usually runs all the time, it is very useful for problem determination. While system trace and the general trace facility (GTF) lists many of the same system events, system trace also lists events occurring during system initialization, before GTF tracing can be started. System trace also traces branches and cross-memory instructions, which GTF cannot do.

System tracing will be captured in all dump situations by default, except during a SNAP dump where SDATA=TRT must be specified. Figure 5-21 shows some sample SYSTRACE data.

01	000A	00AEF430	SVCR	7B	070C0000	868985D2	00000000	00000000	04379238
01	000A	00AEF430	PGM	011	070C2000	868985F2	00040011	12004000	
01	000A	00AEF430	*RCVY	PROG			940C4000	00000011	00000000
01	000A	00AEF430	SSRV	12D		813DE814	00AEF430	000C8000	FF3A0000
							00000000		
01	000A	00AEF430	SSRV	12D		813DE830	00AEF430	000B8000	00000000
							00000000		
01	000A	00AEF430	DSP		070C2000	812FADEA	00000000	00FD0E20	12004780
01	000A	00AEF430	*SVC	D	070C2000	812FADEC	00000000	00FD0E20	12004780
01	000A	00AEF430	SSRV	78		86A0A4AE	0000FF50	000000C8	00AFB5D8

Figure 5-21 IPCS SYSTRACE output

5.10 SMS tracing

- ❑ Start and stop SMS tracing
 - SETSMS TRACE (ON or OFF)
- ❑ Control size of SMS trace table
 - SETSMS SIZE(255M)
- ❑ SMS tracing by jobname

SETSMS TRACE(ON),TYPE(ALL),SIZE(1M),DESELECT(ALL),SELECT(ALL),JOBNAME(SMS)
- ❑ Diagnosing SMS problems
 - Take a dump of the SMS address space
- ❑ Using IPCS for SMS
 - VERBX SMSDATA 'TRACE'

Figure 5-22 SMS tracing

SMS tracing

If you need to trace the interaction between a data set allocation and SMS, collecting SMS trace data may be of assistance. The procedures to collect and review SMS trace data are as follows. To start and stop SMS tracing, use:

```
SETSMS TRACE (ON or OFF)
```

Control size of SMS trace table

The SIZE parameter specifies the size of the trace table in kilobytes. If you omit K or M, the default unit is K. The default value is 128K. The maximum is 255000K or 255M. This value is rounded up to the nearest 4K. Issue the following MVS command:

```
SETSMS SIZE(255M)
```

Select SMS tracing by jobname

You can select tracing by jobname and this limits SMS to tracing to the specified address space. If you enter jobname(*), all address spaces are traced. If you specify ASID, omit jobname. Issue the following MVS command:

```
SETSMS TRACE(ON),TYPE(ALL),SIZE(1M),DESELECT(ALL),SELECT(ALL),JOBNAME(SMS)
```

Diagnosing SMS problems

Take a dump of the SMS address space. For example:

```
DUMP COMM=(any dump title you desire)
R #,JOBNAME=SMS,CONT
R #,SDATA=(LPA,CSA,ALLNUC,GRSQ,LSQA,SWA,PSA,SQA,TRT, RGN,SUM)
```

Using IPCS for SMS

The SMS IPCS verb exit (SMSDATA) is intended for the use of diagnostic programmers who are working with the IBM Support Center to resolve an SMS-related problem. Invoke IPCS and review the SMS trace by issuing the following IPCS command:

```
VERBX SMSDATA 'TRACE'
```

The SMSDATA verb exit performs the following functions:

- ▶ Validates control block chains in the SMS address space.
- ▶ Formats the control blocks in the SMS address space.
- ▶ Formats the trace table in the SMS address space.
- ▶ Formats the control blocks associated with the SMS automatic data areas.

5.11 Trace data using an external writer

❑ Obtaining trace data with the external writer

- Trace data set on DASD or tape rather than requesting a dump

❑ Creating the external writer

```
//CTWTR PROC
//IEFPROC EXEC PGM=ITTTTRCWR
//TRCOUT01 DD DSN=ibmuser.ctrace1,VOL=SER=xxxxxx,UNIT=xxxx,
//          SPACE=(CYL,(xxx),,CONTIG),DISP=NEW,CATLG)
//SYSPRINT DD SYSOUT=*
```

❑ External writer example

- TRACE CT,WTRSTART=ctwrtr
- TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpproc)

Figure 5-23 Collecting trace data with an external writer

Obtaining trace data with the external writer

By using the external writer, you can write application trace buffers directly to a trace data set on DASD or tape rather than requesting a dump. While you might still view your trace buffers by requesting a dump, the advantages of using the external writer are:

- ▶ You do not need to code a component trace buffer find exit routine for IPCS processing.
- ▶ Depending on the size of the trace data set, you can capture more trace data without using valuable system resources such as central or auxiliary storage.

Note: While component trace runs under the master scheduler address space, you need to verify that the priority of the external writer is at least equal to, and preferably greater than the priority of the component being traced. For example, if you are tracing COMP(SYSXES) for JOBNAME(IRLMA), the dispatching priority of the external writer should be equal to or greater than that assigned to IRLMA.

Creating the external writer

Create source JCL to invoke an external writer, which will send the component trace output to one or more trace data sets. Add a procedure to the SYS1.PROCLIB system library or a job as a member of the data set in the IEFJOBS or IEFPSI concatenation.

An external writer is not specific for a component but can be used by any application. So you can use the same source JCL, shown in Figure 5-24 on page 131, again for other tracing later, if needed.

```
//CTWTR PROC
//IEFPROC EXEC PGM=ITTTTCWR
//TRCOUT01 DD DSN=ibmuser.ctrace1,VOL=SER=xxxxxx,UNIT=xxxx,
//          SPACE=(CYL,(xxx),,CONTIG),DISP=NEW,CATLG)
//SYSPRINT DD SYSOUT=*
```

Figure 5-24 External writer procedure

External writer example

The following shows an example for TCPIP CTRACE to an external writer:

- ▶ Start the writer for TCPIP CTRACE where ctwrtt is a writer for CTRACE

```
TRACE CT,WTRSTART=ctwrtt
```

- ▶ Start CTRACE

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpproc)
```

```
R xx,JOBNAME=(tcpproc,otherappljobname),options=(validoptions),WTR=ctwrtt,END
```

Note: Where validoptions=(PFS,TCP,SOCKET,ENGINE,SOCKAPI) for z/OS systems.



IPCS dump debugging

IPCS provides an interactive, online facility for diagnosing software failures. Using data sets and active system storage, IPCS analyzes information and produces reports that can be viewed at a Time Sharing Option Extensions (TSO/E) terminal, or can be printed.

SVC dumps, stand-alone dumps, and some traces are unformatted and need to be formatted before any analysis can begin. IPCS provides the tools to format dumps and traces in both an online and batch environment. IPCS provides you with commands that will let you interrogate specific components of the operating system and allows you to review storage locations associated with an individual task or control block. IPCS allows you to quickly review and isolate key information that will assist with your problem determination process.

Some dumps, such as CEEDUMP, are in a readable format. To debug these dumps you have to browse them.

Dumps produced by an MVS system fall into two categories:

- ▶ Formatted dumps: SYSABEND and SYSUDUMP ABEND dumps and SNAP dumps. IPCS cannot be used with formatted dumps.
- ▶ Unformatted dumps: SVC dumps, SYSMDUMP ABEND dumps, and stand-alone dumps. IPCS formats and analyzes unformatted dumps.

When you submit unformatted dump data sets to IPCS, it simulates dynamic address translation (DAT) and other storage management functions to recreate the system environment at the time of the dump. IPCS reads the unformatted dump data and translates it into words. For example, IPCS can identify the following:

- ▶ Jobs with error return codes
- ▶ Resource contention in the system
- ▶ Control block overlays

IPCS also helps your own dump analysis. For example, you can:

- ▶ Format control blocks. IPCS inserts field names into the output and displays the data in columns by field.
- ▶ Browse unformatted dump storage. IPCS allows you to easily follow pointers to other locations in the dump. It also retains addresses of certain locations in the dump.

- ▶ Reduce the size of a stand-alone dump. You can reduce the size of a stand-alone dump as you transfer it from tape to a direct access storage device (DASD) for IPCS processing.

This chapter gives a brief overview of how to work with IPCS and get at least a useful search argument looking for known problems or asking for IBM support, as follows:

- ▶ Setting IPCS defaults
- ▶ ASIDs to be dumped
- ▶ The VERBX MTRACE command
- ▶ The IPCS SUMMARY command
- ▶ IPCS virtual storage commands
- ▶ Using IPCS to browse dumps
- ▶ Searching IBM problem databases

6.1 IPCS dump debugging

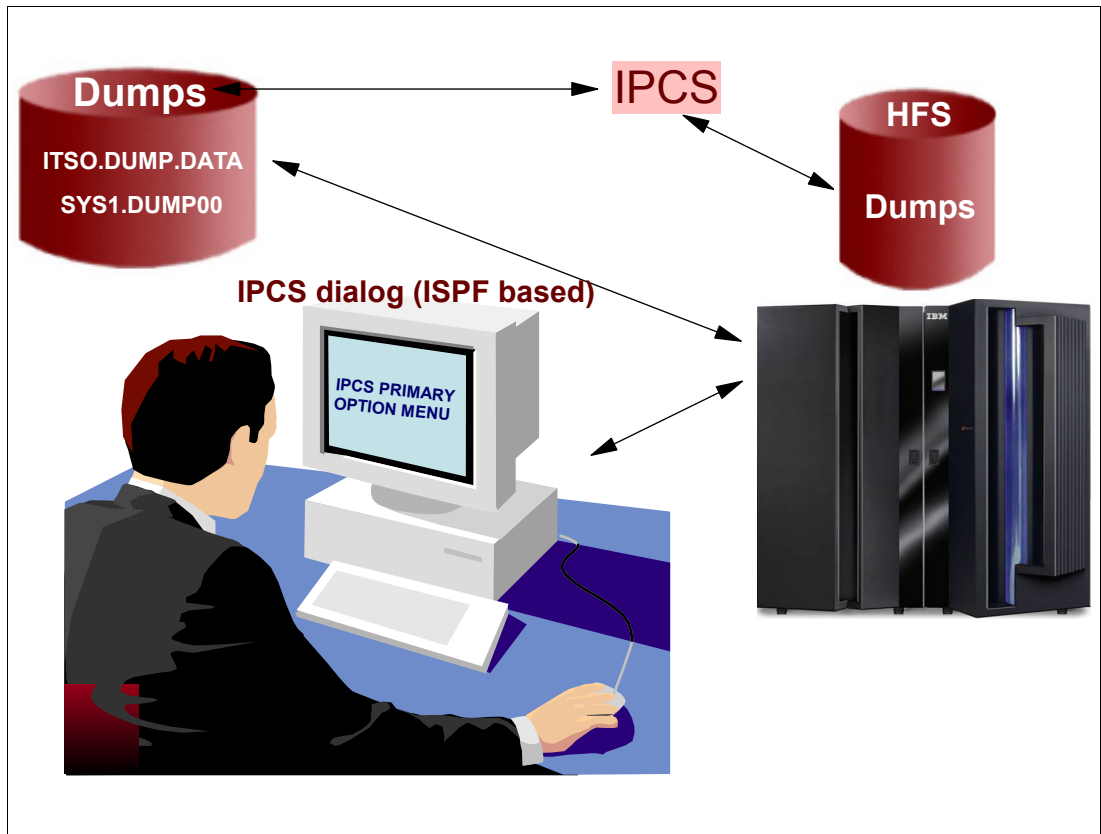


Figure 6-1 IPCS dump debugging

IPCS dump debugging

The interactive problem control system (IPCS) is a tool provided in the MVS system to aid in diagnosing software failures. IPCS provides formatting and analysis support for dumps and traces produced by MVS, other program products, and applications that run on MVS.

IPCS decides whether the source data set should be treated as a system dump by comparing the data set to the following criteria.

The dump data must be stored on a data set with sequential (PS), direct (DA), or unidentified (*) organization. With z/OS V1R2 and higher, IPCS also allows data stored on hierarchical file systems (HFS) to be accessed.

IPCS dialog

Analysis of dumps is through the IPCS full-screen dialog (IPCS dialog) that is supplied with IPCS. The IPCS dialog is an interactive dialog that you use at a terminal. It organizes the problem analysis process into seven options:

- ▶ Set IPCS defaults.
- ▶ View formatted dump data.
- ▶ Generate and edit dump analysis reports.
- ▶ Submit dump analysis jobs for batch processing.
- ▶ Run IPCS subcommands, CLISTs, and REXX execs.

- ▶ Copy dump and trace data from one data set to another.
- ▶ Manage dump and trace data set sources.

IPCS Primary Option Menu

As part of customizing access to IPCS, IBM recommends that you or your installation provide an option for starting the IPCS dialog from an ISPF selection panel, usually the ISPF Primary Option Menu. To start the IPCS dialog from such an ISPF panel, select the option for IPCS.

After you select the IPCS option and press Enter, the system displays the IPCS Primary Option Menu. Figure 6-6 on page 143 shows the IPCS Primary Option Menu panel.

DUMPs on DASD and HFS

Dumps produced by an MVS system fall into two categories:

Formatted dumps SYSABEND and SYSUDUMP ABEND dumps and SNAP dumps.
IPCS cannot be used with formatted dumps.

Unformatted dumps SVC dumps, SYSMDUMP ABEND dumps, and stand-alone dumps.
IPCS formats and analyzes unformatted dumps.

When you submit unformatted dump data sets to IPCS, it simulates dynamic address translation (DAT) and other storage management functions to recreate the system environment at the time of the dump. IPCS reads the unformatted dump data and translates it into words. For example, IPCS can identify the following:

- ▶ Jobs with error return codes
- ▶ Resource contention in the system
- ▶ Control block overlays

6.2 IPCS command processing

- ❑ IPCS processes commands, subcommands, CLISTs, and REXX execs
 - TSO/E commands for IPCS
 - IPCS subcommands
 - IPCS primary and line commands
 - REXX EXECs and CLISTs
 - ISPF primary commands

Figure 6-2 IPCS command processing

IPCS and commands

IPCS is a problem-state, key 8 program that runs in a TSO/E user's address space. IPCS operates in interactive and batch environments supported by TSO/E. The base of IPCS is a TSO/E command processor. The TSO/E command "IPCS" activates the IPCS command processor. All commands used to perform IPCS functions are "subcommands" of the IPCS command. You can use IPCS functions from any TSO/E line mode session.

TSO/E commands for IPCS

IPCS provides three commands to be invoked from the TSO/E READY prompt. Other TSO/E commands may have unique processing features when issued from an IPCS dialog session. The commands are:

- ▶ IPCS
- ▶ IPCSDDIR
- ▶ SYSDSCAN

IPCS subcommands

Once you enter the IPCS command to begin an IPCS session, the IPCS subcommands are your main tools for performing dump and trace analysis. These commands allow you to analyze, format, view, retrieve, and copy dump and trace data, and to maintain an IPCS session. You may use subcommands in any mode.

IPCS primary and line commands

An additional set of IPCS commands are available for use in the full-screen dialog. These commands control various panel functions. The primary commands are entered on the COMMAND or OPTION line of the IPCS dialog. The line commands are used in the prefix area of an IPCS dialog.

Use the IPCS primary command to invoke an IPCS subcommand, CLIST, or REXX exec from any of the panels of the IPCS dialog. The subcommand, CLIST, or REXX EXEC is entered exactly as though it was being invoked under IPCS in line mode. If the subcommand, CLIST, or REXX EXEC sends a report to the terminal, you view the report using the dump display reporter panel. The syntax is as follows:

```
IPCS      { subcommand }
IP        { clist       }
          { rexx-exec   }
```

REXX EXECs and CLISTs

You can invoke REXX EXECs and CLISTs from an IPCS session. These procedures can enter subcommands or use other REXX and CLIST functions to analyze dumps and traces. IPCS provides functions to store data in REXX or CLIST variables and to print data to the IPCS dialog or print data set.

ISPF primary commands

For interactive use, the IPCS dialog uses ISPF dialog support to run as an interactive, full-screen application. This application uses the IPCS command processor. z/OS IPCS exploits data spaces, if permitted, to free virtual storage to allow large, complex analysis routines to function.

6.3 IPCS dump debug example

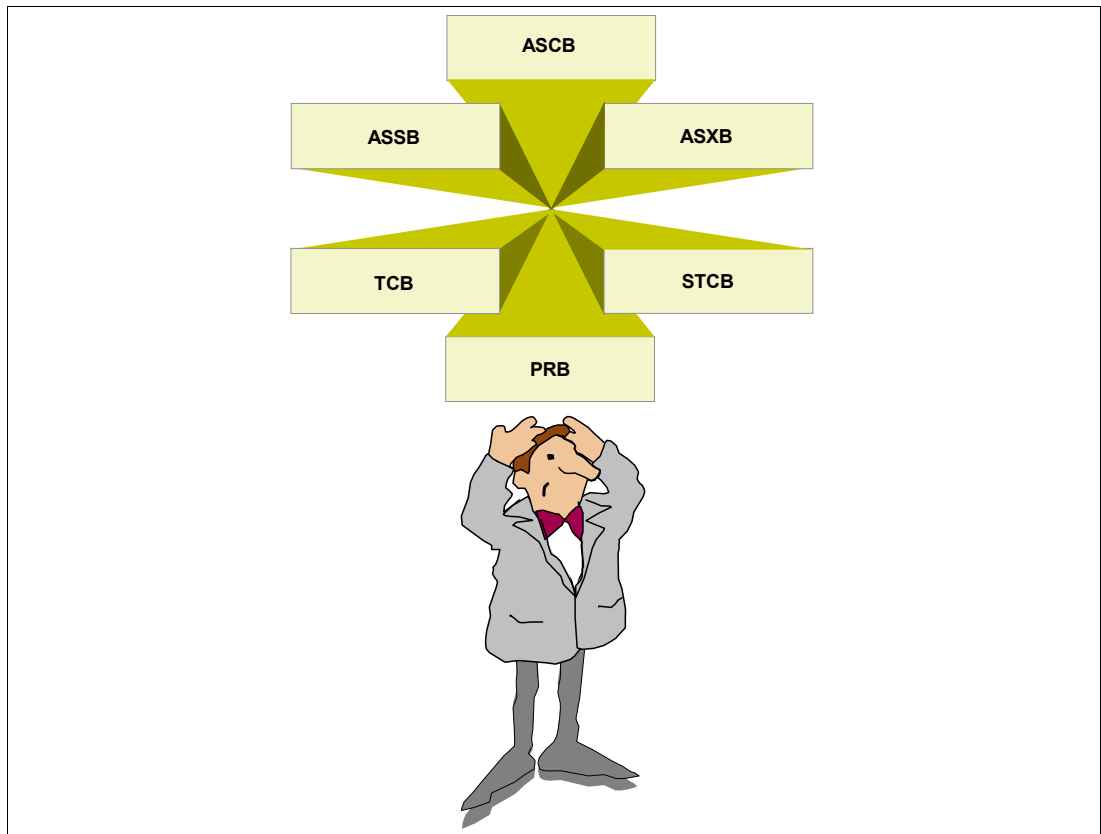


Figure 6-3 Dump debug

Dump debug example

Digging in a dump like one from UNIX System Services is like walking through New York without a map. And if you have a map you might get lost anyway. This chapter provides steps for how to start looking at a dump. It describes IPCS commands used to debug a UNIX System Services hang scenario. A system programmer reports a TSO user process is hanging in UNIX System Services. He dumped the TSO user and OMVS address space. In addition, he dumped the OMVS data spaces.

IPCS dump commands

After the dump has been initialized by IPCS, check whether the dump is a complete one, as follows:

```
Command ==> ip l e0. block(0) l(16)

LIST E0. BLOCK(0) LENGTH(X'10') AREA
E0. LENGTH(X'10')==>All bytes contain X'00'
```

Next, list the dumped address spaces:

```
Command ==> ip cbf rtct;f astb

ASTB

SDAS SDF4 SDF5
```

	----	----	----
001	0001	F8	00
002	000E	80	00
003	007F	80	00
004	0000	00	00

Get the jobnames for address space E and 7F:

Command ==> IP SELECT ASID(X'E',X'7F')

ASID	JOBNAME	ASCBADDR	SELECTION CRITERIA
----	-----	-----	-----
000E	OMVS	00F42E80	ASID
007F	TKBNR	00F9FA00	ASID

The IPCS SUMMARY FORMAT command

Let us now have a look at address space 7F, which is the hanging TSO user. To format the control block fields, use the SUMMARY FORMAT subcommand. Format specifies a report containing the major control blocks associated with the specified address space.

Command ==> IP SUMM FO ASID(X'7F')

ASCB Address Space Control Block

Contains information and pointers needed for address space control

ASSB Address Space Secondary Block

Allows address space related information to be maintained above 16 megabytes

ASXB Address Space Extension Block

Contains information and pointers needed for address space control.

TCB Task Control Block

The TCB serves as a repository for information and pointers associated with a task.

STCB Secondary Task Control Block

The STCB allows task-related information to be kept above 16 megabytes

RB Request Block

Part of the RB is mapped by IHARB and part is mapped by IKJRB.

Maps out the following Request Blocks:

IRB (Interrupt Request Block), which is not the same as an Interruption Response Block. See the IRB data area description.

PRB (Program Request Block)

SIRB (System Interrupt Request Block)

SVRB (SuperVisor Request Block for SVC routines)

TIRB (Task Interrupt Request Block)

The RB control block contains information needed by the supervisor concerning programs and routines, including save areas for all general registers, extended registers, a save area for SVC routines, and additional data needed for control.

Figure 6-4 The SUMMARY FORMAT subcommand

6.4 IPCS support of large data sets

- ❑ **DSNTYPE=LARGE supported**
 - Dumps
 - Traces
 - Other data sets viewed via RBA or BLOCK(n)
 - Print file
 - Table of contents file
- ❑ **Growth and complexity makes performance a concern**
 - Dumps and traces blocked, compressed, and striped
 - Dump directory with large CISIZE, large BUFSPACE, and striped
- ❑ **Operational considerations**

Figure 6-5 IPCS and large data sets

IPCS large data sets

Files directly supported by IPCS may have the DSNTYPE=LARGE attribute in z/OS V1R7. If you are planning to run larger LPARs, it makes sense to set aside some time to plan for larger dumps and traces.

DSNTYPE=LARGE

The DSNTYPE=LARGE is supported in:

- ▶ Dumps
- ▶ Traces
- ▶ Other data sets viewed via RBA or BLOCK(n)
- ▶ Print file
- ▶ Table of contents file

Large dumps and traces

Large dumps and traces make performance more of a concern. So consider the following:

- ▶ Large BLKSIZEs, compression, and striping are all supported. Each can make a significant difference.
- ▶ Good allocation for dump directories can make a significant difference in IPCS efficiency. Compression is not recommended because directories are updated very rapidly during

IPCS analysis, but focusing on primary space, secondary space, CISIZE, BUFSPACE, and striping can really help. If you anticipate the need to work with really large media, the VSAM extended addressing option should be used.

- ▶ Ensure large CISIZE for the DATA portion. BLSCDDIR CLIST is updated to help. A DSNTYPE=LARGE data set can only be used if the dump is both written and processed on a V1R7 system or a later release. A VSAM linear data set with either an extended format or conventional format with a control interval size (CISIZE) of 32K can be substituted. Neither extended sequential nor VSAM data sets, other than linear data sets with the required CISIZE, should be used.
- ▶ In addition, consider the following options:
 - Ensure large (but not excessive) BUFSPACE for the directory.
 - Consider striping.
 - Avoid compression because of intensive updating during IPCS analysis.

Operational considerations

SADMP runs very much the same way as prior releases. From the perspective of the operator who runs SADMP, DSNTYPE=LARGE data sets are treated just the same as the ones used previously. The operational changes are as follows:

- ▶ SADMP tries harder to ensure that data needed to process every SADMP is written to it early. Several page data set pages may be brought in concurrently to achieve this acceleration if independent paths are available.
- ▶ An alteration of some messages tells the operator about progress through the three phases, and, if the operator is sensitive to such things, a modest acceleration of capturing data from page data sets may be sensed. Some messages are changed to reflect the following logic and inform the operator about the phases, as follows:
 - Primary phase dumps vital MVS data (PSAs, CVT, and so forth).
 - Second phase dumps ASIDs 1-4.
 - Third phase dumps the rest.

If installation priorities mandate cutting the dumping process short, this makes it more likely that the truncated dump will be useful. We do not recommend truncation, but we recognize that your specific business priorities may require it.

6.5 Setting the IPCS defaults

```
BLSPPRIM ----- z/OS 01.07.00 IPCS PRIMARY OPTION MENU -----
OPTION ==> 0_

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

*****
* USERID - ROGERS
* DATE   - 05/07/08
* JULIAN - 05.189
* TIME   - 14:22
* PREFIX - ROGERS
* TERMINAL- 3278T
* PF KEYS - 24
*****

Enter END command to terminate IPCS dialog
```

- ☐ When you Enter 0, the IPCS Default Option panel is displayed and you modify the following fields:

```
Source ==> DSNAME('SYS1.DUMP01')
Address Space ==> ASID(X'0001')
Message Routing ==> NOPRINT TERMINAL
Message Control ==> FLAG(WARNING) NOCONFIRM VERIFY
Display Content ==> MACHINE REMARK REQUEST STORAGE SYMBOL
```

Figure 6-6 Selecting the IPCS default options

Setting the IPCS defaults

Option 0 from the Primary Option Menu enables you to identify the data set that contains the dump you will be analyzing. Figure 6-7 shows the part of the IPCS default option menu that you change to gain access to the dump you want to process.

You may change any of the defaults listed in Figure 6-7. The defaults shown before any changes are LOCAL. Change scope to GLOBAL to display global defaults.

```
Scope ==> LOCAL (LOCAL, GLOBAL, or BOTH)
```

If you change the Source default, IPCS will display the current default Address Space for the new source and will ignore any data entered in the Address Space field.

Creating the defaults

The initial display will show Source ==> NODSNAME and no value in Address Space. When you enter your dump DSNAME (in single quotes), you must manually change the NODSNAME for DSNAME. Pressing Enter will then update the Address Space field with the primary ASID for the dump.

```

Source ==> DSNAME('SYS1.DUMP01')
Address Space ==> ASID(X'0001')
Message Routing ==> NOPRINT TERMINAL
Message Control ==> FLAG(WARNING) NOCONFIRM VERIFY
Display Content ==> MACHINE REMARK REQUEST STORAGE SYMBOL

```

Figure 6-7 IPCS Default Option Panel

If the dump was captured via the DUMP COMM command, the ASID will always equal x'0001', the Master Address space, but the dump data set will also include any address spaces that you requested to be dumped.

You will be able to change the Address Space ASID when you know what ASID dump data you need to review.

After setting the IPCS defaults, return to the IPCS Primary Option menu (Figure 6-6 on page 143) and select Option 6, Command. The first IPCS command you enter will start the initialization process for the dump you have specified.

Figure 6-8 shows the messages that are issued during the initialization process.

```

TIME-05:14:53 AM. CPU-00:00:46 SERVICE-673781 SESSION-00:48:42 APRIL 13
BLS18122I Initialization in progress for DSNAME(¢ SYS1.DUMP03¢ )
BLS18124I TITLE=COMPID=DF115,CSECT=IGWLGMO+1264,DATE=02/18/94,MAINTID= NONE
RC=00000024,RSN=12088C01
BLS18222I ESA mode system
BLS18160D May summary dump data be used by dump access? Enter Y to use, N to
bypass
Y Note. Enter Yes
BLS18123I 4,616 blocks, 19,202,560 bytes, in DSNAME(¢ SYS1.DUMP03¢ )
IKJ56650I TIME-05:15:05 AM. CPU-00:00:46 SERVICE-702725 SESSION-00:48:53 APRIL
13
***

```

Figure 6-8 IPCS Dump Initialization messages

After the initialization process, the address space field in the IPCS Default Values panel will now contain the address space identifier (ASID) information stored in the dump data set SYS1.DUMP00. For example:

```
Address Space ==> ASID(X'009E')
```

6.6 IPCS utility menu

- ❑ New Option 6
 - SADMP dump data set utility
- ❑ The panel allows you to clear, define, or reallocate a SADMP dump data set

```
----- IPCS UTILITY MENU -----
OPTION ==>

1 COPYDDIR - Copy dump directory data
2 COPYDUMP - Copy a dump data set
3 COPYTRC  - Copy trace data
4 DSLIST   - Process list of data set names
5 DAE      - Process DAE data
6 SADMP    - SADMP dump data set utility

Enter END command to terminate
```

```
*****
* USERID  - ROGERS
* DATE    - 05/08/23
* JULIAN   - 05.235
* TIME     - 12:21
* PREFIX   - ROGERS
* TERMINAL - 3278T
* PF KEYS  - 24
*****
```

Figure 6-9 New option on IPCS UTILITY MENU - SADMP

IPCS utility panel

When you select Option 3 from the IPCS panel shown in Figure 6-6 on page 143, you receive the panel shown in Figure 6-9. The IPCS Utility Menu panel provides three options for copying data, an option for listing the names of your source data sets, and an option for the dump analysis and elimination (DAE) data set. To invoke it, select Option 3 (Utility) from the IPCS Primary Option Menu panel.

SADMP option

When you choose Option 6, the new SADMP DASD Dump Data Set Utility panel shown in Figure 6-10 on page 146 is displayed. Use the SADMP option to perform the tasks associated with creating, clearing, and reallocating of SADMP data sets on DASD.

Note: This new option is available with z/OS V1R7.

6.7 SADMP dump data set utility

```
----- SADMP DASD Dump Data Set Utility -----
Command ==>

Enter/verify parameters.
Use ENTER to perform function, END to terminate.

Function ==> R ( C - Clear, D - Define, R - Reallocate)
DSNAME    ==>

Volume serial numbers:  (1-32)
    1- 8  VOL001
    9-16
   17-24
   25-32

Unit ==> 9345  (3380, 3390, or 9345)
Cylinders ==> 500      (cylinders per volume)
DSNTYPE(LARGE) ==> N (Y or N)

Optional SMS classes: (May be required by installation ACS routines)
StorClas ==>          DataClas ==>          MgmtClas ==>
```

Figure 6-10 Panel to define SADMP processing

SADMP panel

This utility performs the same functions associated with the AMDSADDD REXX utility. You can also use AMDSADDD, but references to SAMPLIB must now refer to ABLSCLI0. The data set is placed in SBLSCLI0 rather than SAMPLIB because it is no longer a sample.

Note: Systems and the applications that they support tend to get larger and more complex over time. This impacts the dumps and traces that they produce and, in turn, may create problems for you when you attempt to analyze problems using IPCS.

The REXX utility AMDSADDD resides in SYS1.SBLSCLI0. You can use the AMDSADDD REXX utility to:

- ▶ Allocate and initialize the SADMP data set.
- ▶ Clear (reinitialize) the data set.
- ▶ Reallocate and initialize the data set.

The IPCS SADMP dump data set utility panel, shown in Figure 6-10, performs the same functions as the AMDSADDD REXX utility.

6.8 Using IPCS subcommands

- ❑ IPCS Primary Option menu - Option 6
- ❑ STATUS subcommand
 - STATUS FAILDATA subcommand
 - Locates instruction that failed causing the dump

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

===> _

----- IPCS Subcommands and Abbreviations -----
ADDUMP      DROPDUMP, DROPD      LISTDUMP, LDMP      RENUM,      REN
ANALYZE      DROPMAP, DROPM     LISTMAP, LMAP      RUNCHAIN, RUNC
ARCHECK      DROPSYM, DROPS     LISTSYM, LSYM      SCAN
ASCBEXIT,   ASCBX      EPTRACE      LISTUCB, LISTU     SELECT
ASMCHK,     ASMK       EQUATE,      EQU, EQ    LITERAL     SETDEF,     SETD
CBFORMAT,   CBF        FIND,        F          LPAMAP      STACK
CBSTAT      FINDMOD, FMOD      MERGE       STATUS,      ST
CLOSE       FINDUCB, FINDU     NAME        SUMMARY,    SUMM
COPYDDIR    GTFTRACE, GTF      NAME TokN   SYSTRACE
COPYDUMP    INTEGER       NOTE,       N          TCBEXIT,   TCBX
COPYTRC     IPCS HELP, H      OPEN        VERBEXIT,   VERBX
CTRACE      LIST,        L      PROFILE,    PROF     WHERE,      W
```

Figure 6-11 IPCS subcommands

Select the IPCS subcommand entry panel

Once you enter the IPCS command to begin an IPCS session, the IPCS subcommands are your main tools for performing dump and trace analysis. These commands allow you to analyze, format, view, retrieve, and copy dump and trace data, and to maintain an IPCS session. You may use subcommands in any mode.

Return to the IPCS Primary Option menu and select Option 6. When you press Enter, the IPCS Subcommand Entry panel is displayed.

STATUS subcommand

Use the STATUS subcommand to display data usually examined during the initial part of the problem determination process.

STATUS produces different diagnostic information depending on the report type parameter or parameters entered: SYSTEM, CPU, WORKSHEET, and FAILDATA.

Locate failing instruction

Use the IPCS subcommand STATUS FAILDATA to locate the specific instruction that failed and to format all the data in an SVC dump related to the software failure. This report gives information about the CSECT involved in the failure, the component identifier, and the PSW address at the time of the error.

Diagnostic report output

The IPCS STATUS FAILDATA command shows a diagnostic report that summarizes the failure. The following show the FAILDATA information. Figure 6-12 shows an example of the IPCS STATUS FAILDATA report.

```
* * * DIAGNOSTIC DATA REPORT * * *
SEARCH ARGUMENT ABSTRACT
PIDS/5695DF115 RIDS/IGWLHHLS#L RIDS/IGWLGMO AB/S00F4 PRCS/00000024
REGS/0E00C REGS/0B225 RIDS/IGWLHERR#R
Symptom Description
-----
PIDS/5695DF115 Program id: 5695DF115
RIDS/IGWLHHLS#L Load module name: IGWLHHLS
RIDS/IGWLGMO Csect name: IGWLGMO
AB/S00F4 System abend code: 00F4
PRCS/00000024 Abend reason code: 00000024
REGS/0E00C Register/PSW difference for R0E: 00C
REGS/0B225 Register/PSW difference for R0B: 225
RIDS/IGWLHERR#R Recovery routine csect name: IGWLHERR
OTHER SERVICEABILITY INFORMATION

Recovery Routine Label: IGWFRCS
Date Assembled: 02/18/94
Module Level: NONE
SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE
Subfunction
Time of Error Information
PSW: 075C2000 82CC5BCC Instruction length: 02 Interrupt code: 000D
Failing instruction text: 41F00024 0A0D5880 D19C5840
```

Figure 6-12 IPCS STATUS FAILDATA output

Note: The STATUS FAILDATA data in this case shows that the load module that was pointed to by the program status word (PSW) was IGWLHHLS, the CSECT within that load module was IGWLGMO, the abend code (0F4), and the abend reason code (0024). This information is also displayed during the initialization of the dump data set but is not formatted as it is here.

With the information we currently have we could perform a search of the IBM problem databases for a possible solution, but in this instance we will pursue the problem using IPCS to help you develop a better understanding of problem analysis techniques.

6.9 SADMP analysis and COPYDUMP

- ❑ IPCS analysis of dump in place not recommended for multi-volume dumps to DASD
- ❑ IEBGENER and similar programs not recommended for transcription of multi-volume dumps to DASD
- ❑ IPCS COPYDUMP recognizes SADMP “striping”
 - Now has ability to merge the records from a multi-volume SADMP and recapture the prioritized order used by SADMP to get the most important data into the dump data sets first
- ❑ Use compressed extended sequential data set as a target - IBM testing has seen roughly 40% saving of DASD for these large data sets

Figure 6-13 SADMP analysis

SADMP analysis considerations

When doing SADMP analysis, consider the following when processing dumps:

- ▶ IPCS analysis of dumps in place is not recommended for multi-volume dumps to DASD.
- ▶ Use IPCS COPYDUMP since it produces a dump that IPCS can process more efficiently than one copied by IEBGENER or similar programs. This subcommand, COPYDUMP, can be issued from the panel shown in Figure 6-11 on page 147.

Dump testing with CITIZE

Use a compressed extended sequential data set as a target. This could save about 40% of DASD for large data sets. Figure 6-14 on page 150 shows an 87 GB dump, with SADMP, unloaded using IEBGENER. This performance test was to see whether dump directory performance could be improved by simply striping it. Appropriate SMS classes, with a dump directory striped 5 ways, was used to try to improve performance. The result was a dump initialization that completed in 36 minutes.

The version of IPCS with which all preceding runs had taken place was z/OS V1R6 IPCS. A dump directory striped 5 ways and using z/OS V1R7 IPCS resulted in a one third reduction in initialization time and brought it down to 24 minutes.

Dump initialization elapsed Time (minutes)	IPCSDDIR Characteristics
3600	4K CISIZE, V1R6 IPCS
54	24K CISIZE, V1R6 IPCS
36	24K CISIZE, 5 stripes, V1R6 IPCS
24	24K CISIZE, 5 stripes, V1R7 IPCS

Figure 6-14 Improvement in dump directory size for performance

Striping support

Striping spreads sections, or stripes, of a data set across multiple volumes and uses independent paths, if available, to those volumes. The multiple volumes and independent paths accelerate sequential reading and writing of the data set, reducing the time during which dump I/O competes with production I/O.

It is recommended that the number of stripes match the number of volumes you use. This combination will yield the best performance because MVS data management allows random access to any record as though it appeared on a single volume. This is particularly useful during an IPCS analysis of a dump. The savings when loading the data set are real but smaller, the result of reducing the number of times end of volume processing comes into play.

In a striped data set, when the last volume receives a stripe, the next stripes are placed on the first volume, the second volume, the third, and so on to the last volume, then back to the first volume.

6.10 IPCS COPYDUMP

- ❑ Use COPYDUMP to copy the SADMP dump data sets from the data sets which they were initially written into
- ❑ Creates a second type of extended format dump data set
 - This makes the special repositories that an installation tends to set aside for SADMP use maximally available for reuse
 - This produces a dump data set that IPCS can process more efficiently

Figure 6-15 Using the IPCS COPYDUMP subcommand

IPCS COPYDUMP

IPCS COPYDUMP is the recommended method to copy an SADMP dump data set. IPCS COPYDUMP can run without a dump directory being employed. IPCS COPYDUMP is enhanced with z/OS V1R7 as follows:

- ▶ Input may be a list of ddnames or dsnames to accommodate SADMP overflow. SADMP can fill one dump data set, ask the operator for another, and write overflow records to the second. It can also go from a 2nd to a 3rd and so on. IPCS COPYDUMP has been updated to accept a list of input data sets to bring such dumps back together for analysis.
- ▶ Original multi-volume SADMP detected:
 - All volumes accessed in parallel.
 - Records merged to recover SADMP placement of important data first.
 - DSNTYPE=LARGE supported for input and output.

Use COPYDUMP to copy the SADMP dump data sets from the data sets which they were initially written into to a second type of extended format dump data set. This makes the special repositories that the installation tends to set aside for SADMP use maximally available for reuse, and produces a dump data set that IPCS can process more efficiently. SADMP sees a multi-volume dump data set as though it were volume-count separate sequential repositories. DFSMS sees all records on volume 1 followed by all records on volume 2, and so on. Transcription multi-volume SADMPs using COPYDUMP reconciles the two views and produces a data set where the most important records appear early in the dump data set, not scattered across N volumes.

6.11 Using subcommands

- ❑ **SELECT ALL command**
 - What ASIDs have been dumped
- ❑ **SELECT CURRENT command**
 - Display address space executing when dump is taken
- ❑ **SUMMARY FORMAT and VERBEXIT LOGDATA**
 - Use for SLIP dumps and DUMPs from a console

Figure 6-16 Using subcommands

What ASIDs have been dumped

The SELECT ALL command shows what address spaces were *active* when the dump was taken. It *does not show* what address spaces are included in the dump. Figure 6-19 on page 153 shows an example of the IPCS SELECT ALL command.

SELECT CURRENT command

The SELECT CURRENT command displays the address space that was executing at the point in time the dump was initiated. If the dump was issued via a console dump command, the SELECT CURRENT command will display the Master scheduler address space. Figure 6-17 shows the IPCS SELECT CURRENT output.

ASID	JOBNAME	ASCBADDR	SELECTION CRITERIA
000A	CONSOLE	00FB4400	CURRENT

Figure 6-17 IPCS SELECT CURRENT output

This shows that the CONSOLE ASID was dispatched at the time of theabend.

If the dump was taken while in cross-memory mode, both address spaces involved in the cross-memory operation will be included in the dump. Figure 6-18 on page 153 shows the IPCS SELECT CURRENT output, showing the ASIDs involved in the cross-memory function.

ASID	JOBNAME	ASCBADDR	SELECTION CRITERIA
----	-----	-----	-----
0033	CICSFILE	00F4E680	CURRENT
008E	CICSJG03	00ED8100	CURRENT

Figure 6-18 IPCS SELECT CURRENT cross-memory output

SLIP and console dumps

For SLIP dumps or dumps initiated from consoles, use SUMMARY FORMAT or VERBEXIT LOGDATA instead of STATUS FAILDATA. Any valid IPCS command would have started the initialization process and the related display that results after initialization. It should be noted that the dump is only initialized the first time it is referenced via IPCS, and will only be initialized again if the dump is deleted from the IPCS inventory.

ASID	JOBNAME	ASCBADDR	SELECTION CRITERIA
----	-----	-----	-----
0001	*MASTER*	00FD1480	ALL
0002	PCAUTH	00FDB880	ALL
0003	RASP	00FBDA00	ALL
0004	TRACE	00FBD880	ALL
0005	DUMPSRV	00FBD700	ALL
0006	XCFAS	00FB4700	ALL
0007	GRS	00FB4580	ALL
0008	SMSPDSE	00FA1480	ALL
0009	SMSVSAM	00FA1300	ALL
000A	CONSOLE	00FB4400	ALL
000B	WLM	00FB4280	ALL
000C	ANTMAIN	00FB4100	ALL
000D	ANTAS000	00FA3780	ALL
000E	OMVS	00FAF080	ALL
0010	IEFSCHAS	00FBFE80	ALL
0011	JESXCF	00FBFD00	ALL
0012	ALLOCAS	00FBF300	ALL
0013	IOSAS	00FBF180	ALL
0014	IXGLOGR	00FA3600	ALL
0015	SMF	00FA3480	ALL
007C	CMAS	00F43400	ALL
007D	CAS	00F43580	ALL
007E	EYUX140	00F43280	ALL
0080	MQT1CHIN	00F45700	ALL
0081	MQC1CHIN	00F38500	ALL
0082	NETMOPS	00F57B80	ALL
0084	NETMSNA	00F57880	ALL
0086	IOAOMON1	00F57580	ALL
0087	XCOM	00F57280	ALL
0088	CICSCCTR	00F57100	ALL
0089	DWTSPAS	00F63B80	ALL
008B	CICSUA1B	00F4E280	ALL
008C	CICSCA3B	00F47A00	ALL
008D	CICSTA3B	00F47880	ALL

Figure 6-19 IPCS SELECT ALL output

6.12 Analyzing dumps

- ❑ Identifying address spaces in a dump
 - CBFORMAT command to format control blocks
 - CBF CVT
 - FIND command to locate words
 - CBF RTCT
 - RTCT to locate ASIDs
 - FIND ASTB
- ❑ SELECT ASIDLIST command
 - `select asidlist(x'3eb',x'45f',x'445',x'8a',x'10f')`

Figure 6-20 Analyzing dumps

Identify address spaces in a dump

To identify which address spaces are contained in the dump, you can also use IPCS as follows:

1. Format the CVT (IPCS command CBF CVT)
Use the CBFORMAT (CBF) primary command to format a control block. CBF CVT formats the CVT control block which contains the ASIDs that are in the dump.
2. Issue a FIND command for RTCT to locate the address of the Recovery Termination Control Table (RTCT).
Use the FIND primary command to search through all dump output for a single occurrence of a specified value.
3. At offset +x'10C' in the RTCT begins a list of 1-word entries for the address spaces in the dump. The first half of the word contains the ASID.

268	(10C)	CHARACTER	64	RTCTASTB	SVC DUMP ASID TABLE
-----	-------	-----------	----	----------	---------------------

Figure 6-21 on page 155 shows the commands required to determine what address spaces are contained in the dump. The field SDAS contains the ASIDs that are present in the dump.

cbf rtct;f astb			
ASTB			
	SDAS	SDF4	SDF5
	----	----	----
001	03EB	F8	00
002	045F	F8	00
003	0445	F8	00
004	008A	F8	00
005	010F	F8	00
006	0000	00	00
007	0000	00	00

Figure 6-21 IPCS control block format output of RTCT for the ASTB (SVC DUMP ASID TABLE)

SELECT ASIDLIST command

The select address space identifier (ASID) service scans the ASCBs in a dump by following the pointers in the ASVT and then generates a list of entries for selected address spaces within that dump. The select ASID service returns a list of ASCBs meeting selection criteria. The ASID service also creates storage maps entries for ASCBs, which indirectly improve performance.

The select address space identifier (ASID) service scans the ASCBs in a dump by following the pointers in the ASVT and then generates a list of entries for selected address spaces within that dump. The select ASID service returns a list of ASCBs meeting selection criteria. The ASID service also creates storage maps entries for ASCBs, which indirectly improves performance.

Figure 6-22 shows the result of the following IPCS SELECT ASIDLIST command where you use the ASID values returned in the previous format of the RTCT ASTB shown in Figure 6-21. In Figure 6-22, the ASIDs and associated JOBNAMEs that are contained in the dump are displayed.

```
select asidlist(x'3eb',x'45f',x'445',x'8a',x'10f')
```

ASID	JOBNAME	ASCBADDR	SELECTION CRITERIA
----	-----	-----	-----
008A	MQT1CHIN	00ED8A00	ASID
010F	IMSTFAFM	00F0E280	ASID
03EB	IMSTCTL	00F60D00	ASID
0445	MQT1MSTR	00F17700	ASID
045F	IMSTDLI	00FA2B80	ASID :

Figure 6-22 IPCS SELECT ASIDLIST command output

6.13 IPCS trace commands - MTRACE

- ❑ Tracing master trace table
 - VERBX MTRACE subcommand
 - IPCS Trace Processing panel
 - Option 3 - MTRACE
 - IPCS MVS Dump Component Data Analysis panel
 - MTRACE option

Figure 6-23 Tracing the master trace table with IPCS

VERBX MTRACE subcommand

The VERBEXIT MTRACE subcommand has no parameters. Specify the MTRACE verb name on the VERBEXIT subcommand to display:

Figure 6-24 on page 157 shows an example of the VERBX MTRACE output display the master trace table which is similar to the SYSLOG output.

The VERBX MTRACE command displays the following:

- ▶ The master trace table entries for the dumped system. This table is a wraparound data area that holds the most recently issued console messages in a first-in, first-out order.

The MTRACE output in Figure 6-25 on page 157 shows a small sample of what is contained in the MTRACE. In this sample we see details of the symptom dump for our problem.

All data that is displayed on the MVS master console will be captured in the master trace table. The amount of data kept is related to the master trace table buffer size.

- ▶ The NIP hard-copy message buffer
- ▶ The branch entry and NIP time messages on the delayed issue queue

```

00000090 IEA995I SYMPTOM DUMP OUTPUT
137 00000090 SYSTEM COMPLETION CODE=0F4 REASON CODE=00000024
137 00000090 TIME=17.21.42 SEQ=00084 CPU=0000 ASID=0008
137 00000090 PSW AT TIME OF ERROR 075C2000 82CC5BCC ILC 2 INTC 0D
137 00000090 NO ACTIVE MODULE FOUND
137 00000090 NAME=UNKNOWN
137 00000090 DATA AT PSW 02CC5BC6 - 41F00024 0A0D5880 D19C5840
137 00000090 GPR 0-3 12088C0C 440F4000 00000008 00000583
137 00000090 GPR 4-7 00FD1060 12088C0C 06BA3998 7F7697C8
137 00000090 GPR 8-11 00FD102C 02CC79A5 02CC69A6 02CC59A7
137 00000090 GPR 12-15 82CC49A8 7F769B48 82CC5BC0 00000024
137 00000090 END OF SYMPTOM DUMP

```

Figure 6-24 IPCS VERBX MTRACE output

IPCS Trace Processing panel

The MTRACE can also be obtained by using the IPCS Trace Processing panel. The IPCS Trace Processing panel, shown in Figure 6-25, displays a menu of trace formatting options. Invoke it by selecting option 7 (TRACE) from the Analysis of Dump Contents panel or by entering option 2.7 from the IPCS Primary Option Menu panel, shown in Figure 6-6 on page 143.

After choosing a trace processing option (and specifying parameters for certain options), IPCS processes the request for the current default source and displays the formatted trace data on a dump display reporter panel.

```

BLSPTRC1 ----- IPCS Trace Processing
-----
OPTION ==>

To display trace information, enter the corresponding option number.

    1 CTRACE      - Component trace
    2 GTFTRACE    - Generalized trace facility
    3 MTRACE      - Master trace
    4 SYSTRACE    - System traces
    5 MERGE       - Merge multiple traces
    T TUTORIAL    - Details on these traces

Enter END command to terminate IPCS trace processing.

```

Figure 6-25 Using IPCS Trace Processing panel for trace information in the dump

IPCS MVS Dump Component Data Analysis panel

Entering 2.6 on the IPCS primary option menu panel displays the Dump Component Data Analysis panel, bypassing the Analysis of Dump Contents Menu panel. From this IPCS panel you can specify the MTRACE option. For a display of this panel, see Figure C-2 on page 312.

6.14 SYSTRACE command

- ❑ Examining the system trace
- ❑ SYSTRACE command
 - SYSTRACE ALL TIME(LOCAL)
- ❑ Reviewing system trace data
 - Use FIND command to locate *SVC
 - Locate the trace entry that indicates the abend

Figure 6-26 Using the system trace with the SYSTRACE command

Examining the system trace

The system trace table describes the events in the system leading up to the error. The trace table is helpful when the PSW does not point to the failing instruction, and to indicate what sequence of events preceded the abend.

Because system trace usually runs all the time, it is very useful for problem determination. While system trace and the general trace facility (GTF) lists many of the same system events, system trace also list events occurring during system initialization, before GTF tracing can be started. System trace also traces branches and cross-memory instructions, which GTF cannot do.

SYSTRACE command

The system trace can be examined by issuing the SYSTRACE command from the IPCS subcommand entry panel shown in the visual. Issuing the SYSTRACE command on its own will display trace entries associated with the dumped ASID only. Issuing the SYSTRACE ALL command will display all system trace entries. To display the time field in local time, add the TIME(LOCAL) parameter. A complete system trace command is as follows:

```
SYSTRACE ALL TIME(LOCAL)
```

Reviewing system trace data

Figure 6-27 shows a small sample of the system trace. The time stamps would appear on the right-hand side of the display but have been removed for presentation reasons.

The system trace report marks important or significant entries with an asterisk (*). The system trace data can be best reviewed by going to the end of the trace output, and issuing a FIND “*SVC” PREV command. This should help you locate the trace entry that indicates the abend. Another useful trace point to search for is *RCVY, which indicates a recovery action. Entries prior to this can assist with problem diagnosis. An SVC D is the abend SVC. Note that the PSW, which is the same as identified in previous steps will point to the next instruction to be processed.

The SVC trace entries are as follows:

- ▶ An SVC trace entry is for processing of a Supervisor Call (SVC) instruction.
- ▶ An SVCE trace entry is for an error during processing of an SVC instruction.
- ▶ An SVCR trace entry is for return from SVC instruction processing.

SYSTRACE Example 1 (*SVC)									
CP	ASID	TCB	TRACE	ID	PSW	R15	R0	R1	
00	0008	007FD720	*SVC	D	075C2000 82CC5BCC	00000024	12088C0C		
00	0008	007FD720	SSRV	78	828BC3F0	0000FF50	000000C8		
						00080000			
00	0008	007FD720	SSRV	78	828BC41A	0000FF70	00000FB0		
						00080000			
00	0008	007FD720	EXT	1005	070C0000 813B54AC	00001005			
SYSTRACE Example 2 (*RCVY)									
00	0153	008DA530	SSRV	78	40E5269C	4050E612	000002B8		
						01530000			
00	0153	008DA530	SSRV	78	80E52704	4050E612	00000080		
						01530000			
02	0013	008C5E88	*RCVY	PROG		940C4000	00000011		
02	0013	008C5E88	SSRV	78	8109CADC	4000EF50	00000818		
						00010000			
02	0013	008C5E88	*RCVY	FRR	070C0000 9056FBE8	940C4000	00000011		

Figure 6-27 IPCS SYSTRACE ALL output

The actual SVC identified in the SYSTRACE is the hexadecimal identification. This must be converted to decimal to enable the correct research, for example:

The SYSTRACE entry for SVC 78 would convert to a decimal SVC number of 120, which, when referencing *z/OS MVS Diagnosis Reference*, SY28-1084, would identify the GETMAIN/FREEMAIN SVC.

This is an example of just one of the many trace entries that are created during the life of a z/OS task. For a further explanation of other trace entries, you can reference *z/OS Diagnosis: Tools and Service Aids*, SY28-1085.

6.15 IPCS SUMMARY subcommand

❑ SUMMARY subcommand

- SUMMARY subcommand parameters
- SUMMARY FORMAT
 - Displays task control block (TCB) and other control block information
- TCB summary
 - RTM2WA area

Figure 6-28 Using SUMMARY subcommand to locate failing TCB

SUMMARY subcommand

Use the SUMMARY subcommand to display or print dump data associated with one or more specified address spaces.

Using SUMMARY produces different diagnostic reports depending on the report type parameter, FORMAT, KEYFIELD, JOBSUMMARY, and TCBSUMMARY, and the address space selection parameters, ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, and JOBLIST. Specify different parameters to selectively display the information you want to see. See Figure C-3 on page 313 for a display of all the parameters with the SUMMARY subcommand.

Note: Installation exit routines can be invoked at the system, address space, and task level for each of the parameters in the SUMMARY subcommand.

SUMMARY FORMAT command

The SUMMARY FORMAT command displays task control block (TCB) and other control block information. By issuing the MAX DOWN, or M PF8 command the TCB summary will be located.

TCB summary

The TCB summary can be located at the end of an IPCS summary format report as shown in the following example. By reviewing the data in the CMP field, we see that TCB 007FD588

has a non-zero CMP field that reflects the 440F44000 abend. Figure 6-29 shows the TCB Summary.

```
* * * * T C B S U M M A R Y * *
JOB SMXC ASID 0008 ASCB 00FBC580 FWDP 00FBC400 BWD 00F4E600 PAGE
TCB AT CMP NTC OTC LTC TCB BACK PAGE
007FE240 00000000 00000000 00000000 007FDE88 007FF1D8 00000000 000014
007FF1D8 00000000 00000000 007FE240 00000000 007FDE88 007FE240 000018
007FDE88 00000000 007FF1D8 007FE240 007FD588 007FDB70 007FF1D8 000021
007FDB70 00000000 00000000 007FDE88 00000000 007FD588 007FDE88 000024
007FD588 440F4000 02000000 00000000 00000000 00000000 007FBFB8 000026
```

Figure 6-29 TCB Summary at the bottom of the SUMMARY FORMAT display

RTM2WA area

By issuing a FIND “TCB: 007FD588” prev command, the failing TCB data is displayed in the Summary Format display. From this point, you can locate the RTM2WA area. This can contain information that in many cases identifies the failing program.

In the TCB summary, find the task control block (TCB) for the failing task. This TCB has the abend code as its completion code in the CMP field. In the TCB summary, obtain the address of the recovery termination manager 2 (RTM2) work area (RTM2WA) for the TCB.

In the RTM2WA summary, obtain the registers at the time of the error and the name and address of the abending program.

If the RTM2WA summary does not give the abending program name and address, probably an SVC instruction abnormally ended.

If the RTM2WA summary gives a previous RTM2WA for recursion, the abend for this dump occurred while an ESTAE or other recovery routine was processing another, original abend. In recursive abends, more than one RTM2WA may be created. Use the previous RTM2WA to diagnose the original problem.

6.16 What is VERBX

- ❑ **IPCS VERBEXIT subcommand**
 - Supports a product specific exit routine
- ❑ **VERBX example for CICS**
 - Format the CICS dispatcher data in the dump
- ❑ **Verb exit routine**
 - Generates a unique diagnostic report
 - Can process:
 - Installation application storage
 - IBM component data areas and storage
- ❑ **Define verb exit routine**

```
EXIT EP(name) VERB(verb_name) AMASK(X'aaFFFFFF')  
ABSTRACT('text') HELP(helppanel)
```

Figure 6-30 VERBEXIT subcommand for exit routines

IPCS VERBEXIT subcommand

You use the VERBEXIT subcommand to run an installation-supplied or IBM-supplied verb exit routine. One of the more common IPCS commands is VERBEXIT (VERBX). VERBX supports a product-specific exit routine that can be used to format the dump. See Figure C-4 on page 314.

VERBX example

For example, to format dump data for CICS/TS Release 1.3 we would use the exit routine DFHPD530. This program is supplied with CICS/TS Release 1.3 to enable you to format the CICS/TS-specific data.

For example, the commands could be used as follows:

- ▶ Format the CICS Dispatcher data contained in the dump.
`VERBX DFHPD640 'DS=1'`
- ▶ Format the IMS save area.
`VERBX IMSDUMP 'imsjobname FMTIMS savearea'`
- ▶ Format the DB2 thread data.
`VERBX DSNWDMP 'verbx dsnwdmp 'sumdump=no,subsys=itso,ds=1'`

Verb exit routine

A verb exit routine can generate a unique diagnostic report that is not currently available in IPCS. A verb exit routine can process either:

- ▶ Installation application storage
- ▶ IBM component data areas and storage

Verb exit routines can be defined in BLSCUSER, in the IPCSPARM concatenation data set, or invoked by name. Define the verb exit routine in the BLSCUSER parmlib member with the following statement:

```
EXIT EP(name) VERB(verb_name) AMASK(X'aaFFFFFF')  
ABSTRACT('text') HELP(helppanel)
```

The variables are as follows:

name	The exit routine name.
verb_name	The exit routine verb name.
aa	Can be either: <ul style="list-style-type: none">▶00 - Indicates 24-bit storage accessing.▶7F - Indicates 31-bit storage accessing.
text	The abstract shown on the component data analysis panel entry associated with this verb exit.

6.17 IPCS VERBX LOGDATA command

- ☐ LOGDATA verb name in VERBEXIT subcommand
- ☐ LOGDATA report
- ☐ Examining the LOGREC buffer
- ☐ Viewing the LOGDATA report
- ☐ LOGREC reports

Figure 6-31 VERBEXIT LOGDATA subcommand and LOGREC reports

LOGDATA verb

Specify the LOGDATA verb name on the VERBEXIT subcommand to format the LOGREC buffer records that were in storage when the dump was generated. LOGDATA locates the LOGREC records in the LOGREC recording buffer and invokes the EREP program to format and print the LOGREC records. The records are formatted as an EREP detail edit report.

LOGDATA report

Use the LOGDATA report to examine the system errors that occurred just before the error that caused the dump to be requested.

Examining the LOGREC buffer

Use the IPCS subcommand VERBEXIT LOGDATA to view the LOGREC buffer in a dump. This report might repeat much of the information contained in the STATUS FAILDATA report, but it helps to identify occasions when multiple error events caused the software failure.

Viewing the LOGDATA report

When viewing the VERBEXIT LOGDATA report, skip the hardware records to view the software records. Search for the first software record. Figure 6-32 on page 165 shows the start of the last error log entry displayed.

```

TYPE: SOFTWARE RECORD REPORT: SOFTWARE EDIT REPORT DAY.
(SVC 13) REPORT DATE: 103.99
FORMATTED BY: IEAVTFDE HBB6601 ERROR DATE: 103.99
MODEL: 9021 HH:MM:SS
SERIAL: 060143 TIME: 17:21.42
JOBNAME: MSTJCL00 SYSTEM NAME:
ERRORID: SEQ=00080 CPU=0000 ASID=0008 TIME=17:21:42.3
SEARCH ARGUMENT ABSTRACT
PIDS/5695DF115 RIDS/IGWLHHLS#L RIDS/IGWLG MOT AB/S00F4 PRCS/00000024
REGS/OC7E8 RIDS/IGWLHERR#R
SYMPTOM DESCRIPTION
-----
PIDS/5695DF115 PROGRAM ID: 5695DF115
RIDS/IGWLHHLS#L LOAD MODULE NAME: IGWLHHLS
RIDS/IGWLG MOT CSECT NAME: IGWLG MOT
AB/S00F4 SYSTEM ABEND CODE: 00F4
PRCS/00000024 ABEND REASON CODE: 00000024
REGS/0E00C REGISTER/PSW DIFFERENCE FOR ROE: 00C
REGS/OC7E8 REGISTER/PSW DIFFERENCE FOR ROC: 7E8
RIDS/IGWLHERR#R RECOVERY ROUTINE CSECT NAME: IGWLHERR
OTHER SERVICEABILITY INFORMATION
RECOVERY ROUTINE LABEL: IGWFRCS D
DATE ASSEMBLED: 02/18/94
MODULE LEVEL: NONE
SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE
SUBFUNCTION
TIME OF ERROR INFORMATION
PSW: 075C2000 82CC5190 INSTRUCTION LENGTH: 02 INTERRUPT CODE: 000D
FAILING INSTRUCTION TEXT: 41F00024 0A0DBF0F D1D44780

```

Figure 6-32 VERBX LOGDATA output

System error log

Another valuable source of diagnostic information in the dump are the system error log entries, which are created for all hardware and software error conditions. To review these records the VERBX LOGDATA command can be used and the last records should relate to the abend. This is not always the case, but reviewing this data from the last entry and moving backwards in time can often present information that relates to the problem or may indicate what the cause was. This may indicate a hardware or software error. In our case, the logdata does include records for our problem and is representative of data already found.

6.18 Using the SYS1.LOGREC

❑ Viewing SYS1.LOGREC

- Batch job - EXEC PGM=IFCEREP1

❑ LOGREC data in a CF log stream

- Batch job accessing the log stream

❑ LOGREC data in a log stream

- Contains records for all systems in a sysplex
- Component information

Figure 6-33 Using SYS1.LOGREC data

Viewing SYS1.LOGREC

The system error log can also be interrogated via a batch utility. The program used to extract this data from either the online error log data set, SYS1.LOGREC, or a historical error log data set is, IFCEREP1. This program can be used to produce hardware and software failure reports in both a summary and detailed format. Figure 6-34 shows the JCL required to process a software summary report.

```
//LOGREC JOB,.....  
//STEP1 EXEC PGM=IFCEREP1,PARM=CARD  
//SYSPRINT DD SYSOUT=*  
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR  
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,5,,CONTIG)  
//EREPT DD SYSOUT=(*,DCB=BLKSIZE=133)  
//TOURIST DD SYSOUT=(*,DCB=BLKSIZE=133)  
//SYSIN DD *  
PRINT=PS  
TYPE=SIE  
ACC=N  
TABSIZ=512K  
ENDPARM  
//
```

Figure 6-34 IFCEREP1 sample JCL

LOGREC data in a CF

If your LOGREC data is stored in a Coupling Facility (CF) log stream data set you can use the IFCEREP1 program to access this. Figure 6-35 shows the JCL that will enable you to produce error log reports from the log stream data set.

```
//LOGREC1 JOB,.....  
//EREPL0G EXEC PGM=IFCEREP1,REGION=4M,  
// PARM=(¢ HIST,ACC=N,TABSIZE=512K,PRINT=PS,TYPE=SIE¢ )  
//ACCIN DD DSN=sysplex.LOGREC.ALLRECS,  
// DISP=SHR,  
// SUBSYS=(LOGR,IFBSEXIT,¢ FROM=(1999/125),TO=YOUNGEST¢ ,  
// ¢ SYSTEM=SC42¢ ) ,  
// DCB=(RECFM=VB,BLKSIZE=4000)  
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,5,,CONTIG)  
//TOURIST DD SYSOUT=*,DCB=BLKSIZE=133  
//EREPT DD SYSOUT=*,DCB=BLKSIZE=133  
//SYSABEND DD SYSOUT=*  
//SYSIN DD DUMMY
```

Figure 6-35 IFCEREP1 JCL to format Coupling Facility LOGREC data

LOGREC reports

When generating error log reports from log stream data it should be remembered that the log stream data set contains error information for all systems in the sysplex connected to the Coupling Facility. You should use the SYSTEM option of the SUBSYS parameter to filter the log stream records. Date and time parameters will also assist with the filtering.

Component information

Other information is included in the error log information in the component data. This can assist with isolating the specific product that is being affected and the maintenance level of the module that detected the failure. The maintenance level or service release level is also known as the PTF level, or you might be requested for the replacement modification identifier (RMID). It should be noted that the maintenance level of the failing load module is not necessarily the maintenance level of the failing CSECT, or module, within the load module.

Figure 6-36 shows some of the component data that can be located in the system error log.

```
COMPONENT INFORMATION:  
COMPONENT ID: 5695DF115  
COMPONENT RELEASE LEVEL: 1B0  
PID NUMBER: 5695DF1  
PID RELEASE LEVEL: V1R2  
SERVICE RELEASE LEVEL: UW04733  
DESCRIPTION OF FUNCTION: PDSE LATCH SUPPORT  
PROBLEM ID: IGW00000  
SUBSYSTEM ID: SMS
```

Figure 6-36 LOGREC error component data

6.19 IPCS virtual storage commands

❑ Virtual storage information

➤ Obtain by using VERBX VSMDATA subcommand

VERBX VSMDATA 'LOG SUMMARY'

Summary of Key Information from LDA (Local Data Area) :

```
STRTA = 34000 (ADDRESS of start of private storage area)
SIZE = BCC000 (SIZE of private storage area)
CRGTP = B6000 (ADDRESS of current top of user region)
LIMIT = BCC000 (Maximum SIZE of user region)
LOAL = 8E000 (TOTAL bytes allocated to user region)
HIAL = 4E000 (TOTAL bytes allocated to LSQA/SWA/229/230 region)
SMFL = FFFFFFFF (IEFUSI specification of LIMIT)
SMFR = FFFFFFFF (IEFUSI specification of VVRG)

ESTRA = CE00000 (ADDRESS of start of extended private storage area)
ESIZA = 73200000 (SIZE of extended private storage area)
ERGTP = CE63000 (ADDRESS of current top of extended user region)
ELIM = 73200000 (Maximum SIZE of extended user region)
ELOAL = 61000 (TOTAL bytes allocated to extended user region)
EHIAL = 388000 (TOTAL bytes allocated to extended LSQA/SWA/229/230)
SMFEL = FFFFFFFF (IEFUSI specification of ELIM)
SMFER = FFFFFFFF (IEFUSI specification of EVVRG)
```

Figure 6-37 Virtual storage data

Virtual storage information

Interrogating Virtual Storage usage in a dump is achieved by using the IPCS VERBX VSMDATA command. Some examples of this command are:

```
VERBX VSMDATA 'LOG SUMMARY'
VERBX VSMDATA 'OWNCOMM' (Check Common Storage Tracking)
VERBX VSMDATA 'OWNCOMM DETAIL ALL SORTBY(ASIDADDR)'
VERBX VSMDATA 'OWNCOMM DETAIL ASID(ddd) SORTBY(TIME)'
VERBX VSMDATA 'NOGLOBAL,JOBNAME(xxxxDBM1)'
```

The VERBX VSMDATA parameters are shown in Figure C-5 on page 314.

The VERBX VSMDATA command also supports a SUMMARY parameter, which provides a more concise report designed specifically for diagnosis of out of storage conditions. This report, generated by the VERBEXIT VSMDATA 'SUMMARY' subcommand, formats key data from the following VSM control blocks:

- ▶ Address queue anchor table (AQAT)
- ▶ Allocated element (AE)
- ▶ Double free element (DFE)
- ▶ Descriptor queue element (DQE)
- ▶ Free block queue element (FBQE)
- ▶ Free queue element (FQE)

- ▶ Global data area (GDA)
- ▶ Local data area (LDA)

The end of the VSMDATA LOG SUMMARY display has this interesting summary that can be very helpful for assisting with S878/80A abends. Figure 6-38 on page 170, and Figure 6-39 on page 171 show a sample of the data displayed for the Virtual Storage Manager™.

LOCAL STORAGE DATA SUMMARY
LOCAL STORAGE MAP

Extended	80000000	<- TOP OF EXT. PRIVATE
LSQA/SWA/229/230	80000000	<- MAX EXT. USER REGION ADDRESS
	7EB8E000	<- ELSQA BOTTOM
(Free Extended Storage)		
	30C77000	<- EXT. USER REGION TOP
Extended User Region		
	2AF00000	<- EXT. USER REGION START
:	:	
: Extended Global Storage	:	
=====		<- 16M LINE
: Global Storage	:	
:	A00000	<- TOP OF PRIVATE
LSQA/SWA/229/230		
	9B3000	<- LSQA BOTTOM
(Free Storage)		
	986000	<- MAX USER REGION ADDRESS
	564000	<- USER REGION TOP
User Region		
	6000	<- USER REGION START
: System Storage	:	
:	0	

Input Specifications:

Region Requested => 0
 IEFUSI/SMF Specification => SMFL : 980000 SMFEL: 79E00000
 SMFR : 880000 SMFER: 79800000
 Actual Limit => LIMIT: 980000 ELIM : 55100000

Summary of Key Information from LDA (Local Data Area) :

STRTA = 6000 (ADDRESS of start of private storage area)
 SIZA = 9FA000 (SIZE of private storage area)
 CRGTP = 564000 (ADDRESS of current top of user region)
 LIMIT = 980000 (Maximum SIZE of user region)
 LOAL = 54F000 (TOTAL bytes allocated to user region)
 HIAL = 4D000 (TOTAL bytes allocated to LSQA/SWA/229/230 region)
 SMFL = 980000 (IEFUSI specification of LIMIT)
 SMFR = 880000 (IEFUSI specification of VVRG)
 ESTR = 2AF00000 (ADDRESS of start of extended private storage area)
 ESIZA = 55100000 (SIZE of extended private storage area)
 ERGTP = 30C77000 (ADDRESS of current top of extended user region)
 ELIM = 55100000 (Maximum SIZE of extended user region)
 ELOAL = 5CE7000 (TOTAL bytes allocated to extended user region)
 EHIAL = C07000 (TOTAL bytes allocated to extended LSQA/SWA/229/230)
 SMFEL = 79E00000 (IEFUSI specification of ELIM)
 SMFER = 79800000 (IEFUSI specification of EVVRG)

Figure 6-38 VERBX VSMDATA storage map output

Subpool usage summary

This SUMMARY report also generates the following:

- ▶ Global storage map
- ▶ Global subpool usage summary
- ▶ Local storage map
- ▶ Local subpool usage summary

Note: The Global and Local subpool usage summaries reflect pages that have all or some of the page allocated. You can find information about the allocation of a particular page in the VSM control blocks representing the page.

Following is a SUBPOOL storage usage summary for each TCB.

LOCAL SUBPOOL USAGE SUMMARY						
TCB/OWNER	SP#	KEY	BELOW	ABOVE	TOTAL	
-----	---	---	-----	-----	-----	
9FF410	129	0	340000	3100000	3440000	
9FF410	130	8	100000	1200000	1300000	
9FF410	130	9	80000	200000	280000	
9FF410	131	8	4000	62B000	62F000	
9FF410	132	4	0	1E000	1E000	
9FF410	132	8	C000	86000	92000	
LSQA	205	0	0	A3000	A3000	
LSQA	215	0	0	19000	19000	
LSQA	225	0	0	15000	15000	
9FFE88	229	0	0	D000	D000	
9FFBF8	229	0	0	1C000	1C000	
9FF5A8	229	0	1000	2000	3000	
9FF410	229	0	0	1000	1000	
9FF180	229	0	0	1000	1000	
9FB280	229	0	0	8000	8000	
9ECE88	229	0	0	1000	1000	
9ECAD8	229	0	3000	9000	C000	

Figure 6-39 VERBX VSMDATA subpool usage summary

6.20 Using IPCS to browse storage

☐ Browse storage in a dump using IPCS


➤ Select address to display

```
----- IPCS - ENTRY PANEL -----
Command ==>

CURRENT DEFAULTS:
Source ==> DSNAME('DUMP.D0506.H15.SC65.GABERT1.S00011')
Address space ==> ASID(X'008A')

OVERRIDE DEFAULTS:                                     (defaults used for blank fields)
Source ==> DSNAME('DUMP.D0506.H15.SC65.GABERT1.S00011')
Address space ==>
Password      ==>

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



☐ Place an address in the field, hit Enter and the storage is displayed

Figure 6-40 Browsing storage with IPCS

Browse storage in a dump using IPCS

Another function of IPCS is the ability to browse storage locations with the dump. There will be many times when you will need to look at storage locations in a dump using IPCS. Normally you browse storage locations once you have been viewing other options in the dump. Select the BROWSE (Option 1) from the IPCS primary option menu, shown in Figure 6-6 on page 143. The next panel will identify the current dump data set, as shown in Figure 6-41 on page 173.

Select address to display

Once you place an address in the Address (Pointer: field) in Figure 6-41 on page 173, the address appears in Figure 6-42 on page 173.

```
----- IPCS - ENTRY PANEL -----
Command ==>

CURRENT DEFAULTS:
Source ==> DSNAME('DUMP.D0506.H15.SC65.GABERT1.S00011')
Address space ==> ASID(X'008A')

OVERRIDE DEFAULTS:                                     (defaults used for blank fields)
Source ==> DSNAME('DUMP.D0506.H15.SC65.GABERT1.S00011')
Address space ==>
Password      ==>

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

Figure 6-41 IPCS BROWSE storage panel

From this panel you can do a select (S) on the address, as shown in Figure 6-42.

```
DSNAME('DUMP.D0506.H15.SC65.GABERT1.S00011') POINTERS -----
Command ==>                                     SCROLL ==> CSR
ASID(X'008A') is the default address space
PTR   Address      Address space      Data type
S0003 01329D48.    ASID(X'008A')      AREA
Remarks:
```

Figure 6-42 IPCS BROWSE storage **S**elect option

Storage address displayed

Figure 6-43 shows the storage at location 01329D48—storage starting at that address.

```
ASID(X'008A') ADDRESS(01329D48.) STORAGE -----
Command ==>                                     SCROLL ==> CSR
01329D48      50E0D004  91802021  | &\}.j... |
01329D50  47806124  45E0631A  47F0629E  BF8F208C  | ../..\...0..... |
01329D60  47806184  58B08008  9501B018  47706162  | ../d....n...../. |
01329D70  1F11BF17  B0014100  00484130  00FA45E0  | ..... \ |
```

Figure 6-43 IPCS BROWSE selected storage

6.21 Using IPCS to find the failing instruction

☐ Find failing instruction in a dump

- Use STATUS FAILDATA subcommand - gets report
- Report shows the following:

```
PSW: 070C1000 81329D48   Instruction length: 02   Interrupt code: 000D
Failing instruction text: 00181610 0A0D50E0 D0049180
```

☐ Failing instruction text

☐ Detail edit report for a software record

- Report is produced by EREP and, through the VERBEXIT LOGDATA subcommand, under IPCS
- Use detail edit report for a software record to determine the cause of an abend, and the recovery action that the system or application needs to take

Figure 6-44 Finding the failing instruction in a dump

Find failing instruction in a dump

Normally when analyzing certain dumps, one of the first things to determine is to find the failing instruction. The STATUS FAILDATA report also helps you find the exact instruction that failed. This report—an example is shown in Figure C-6 on page 315 and Figure C-7 on page 316—provides the PSW address at the time of the error and the failing instruction text. Note that the text on this screen is not always the failing instruction text. Sometimes the PSW points to the place where the dump was taken and not the place where the error occurred. See Figure 6-11 on page 147, for the STATUS subcommand. On that display you issue the STATUS FAILDATA subcommand.

From the report, the PSW and failing instruction text are as follows:

```
PSW: 070C1000 81329D48   Instruction length: 02   Interrupt code: 000D
Failing instruction text: 00181610 0A0D50E0 D0049180
```

Failing instruction text

This contains 12 bytes of the instruction stream at the time of the error, including the actual instruction that caused the abend. Starting at the end of the sixth byte, subtract the instruction length to indicate the failing instruction. In the preceding example, the failing instruction is X'0A0D'.

Detail edit report for a software record

The detail edit report for a software record shows the complete contents of an error record for an abnormal end, including the system diagnostic work area (SDWA). The report is produced by EREP and, through the VERBEXIT LOGDATA subcommand, under IPCS.

Use the detail edit report for a software record to determine the cause of an abend, and the recovery action that the system or application has either taken or not taken. This report enables you to locate where an error occurred, similar to the analysis of an SVC dump. Once you locate the error, you can develop a search argument to obtain a fix for the problem.

See *Environmental Record Editing and Printing Program (EREP) User's Guide*, GC35-01511, for information about producing a detail edit report for an SDWA-type record. See *z/OS MVS Interactive Problem Control System (IPCS) Commands*, SA22-7594, for information about the VERBEXIT LOGDATA subcommand.

6.22 Analyzing for resource contention

- ❑ Resource contention analysis in dumps
 - Use the IPCS subcommand ANALYZE
 - Command is used to detect resource contention
- ❑ ANALYZE RESOURCE subcommand
 - Report lists the jobs that hold the device group and the jobs requiring, or waiting for, the device group
- ❑ ANALYZE RESOURCE XREF subcommand
 - For each job that holds a device group, the report lists all other device groups that job holds
 - For each job waiting for a device group, the report lists all other device groups that job holds

Figure 6-45 Subcommands to analyze resource contention

Resource contention analysis in dumps

You can obtain information related to resource contention by using the IPCS subcommand ANALYZE. This subcommand displays contention information for I/O, ENQs, suspend locks, allocatable devices and real storage.

This command is used to detect resource contention. Specifying GRSQ in the SDATA options makes the information more reliable. Generally the most useful information is found at the bottom of this example report, shown in Figure 6-46 on page 177. The top is generally I/O device contention and isn't usually relevant.

ANALYZE RESOURCE subcommand

The ANALYZE RESOURCE subcommand produces a report that identifies each resource, or device group, that is experiencing contention. Under each resource, it lists the jobs that hold the device group and the jobs requiring, or waiting for, the device group. For example, the resource name in contention in Figure 6-46 on page 177 is:

```
MAJOR=IGDCDSXS MINOR=SYSD.DFSMS.COMMDS SCOPE=SYSTEMS
```

Note: Scope=SYSTEMS means multi-system, and scope=SYSTEM means single system.

```

RESOURCE #0011:
NAME=MAJOR=IGDCDSXS MINOR=SYSD.DFSMS.COMMDS SCOPE=SYSTEMS
RESOURCE #0011 IS HELD BY:
JOBNAME=SMS ASID=0025 TCB=009EB0F0 SYSNAME=CM01
RESOURCE #0011 IS REQUIRED BY:
JOBNAME=SMS ASID=0026 TCB=009EB0F0 SYSNAME=PR02
JOBNAME=SMS ASID=0026 TCB=009EB0F0 SYSNAME=PR03
JOBNAME=SMS ASID=0028 TCB=009EC660 SYSNAME=SP02
JOBNAME=SMS ASID=0027 TCB=009EB0F0 SYSNAME=TS01

```

Figure 6-46 Resource contention data from the IPCS ANALYZE command

Note: Holders of and waiters on resources are identified in the output. ASIDs and TCBs (where appropriate) are provided and if the scope is SYSTEMS, the resource is the holding system name.

ANALYZE RESOURCE XREF subcommand

If you add the XREF keyword to the ANALYZE RESOURCE subcommand, IPCS would add the following information to the previous report:

- For each job that holds a device group, the report lists all other device groups that job holds.
- For each job waiting for a device group, the report lists all other device groups that job holds.

Report using XREF keyword

An example of the output from a report generated using the XREF keyword is shown in Figure 6-47.

```

                                CONTENTION REPORT BY RESOURCE NAME

RESOURCE #0001:
    NAME=GROUP record, group = SYSMCS , member = all members

RESOURCE #0001 IS HELD BY:

    JOBNAME=XCFAS      ASID=0006  UNKNOWN#00000001=00000000
    DATA=Local lock owner
        Request id = 000131B8
        Request code = 00000002

```

Figure 6-47 ANALYZE RESOURCE XREF report

6.23 Searching IBM problem databases

- ❑ IBM database searches
 - Evaluate available diagnostic data - use abend code
 - Look in z/OS MVS Systems Codes, SA22-7626
- ❑ Use search arguments
 - Abend codes - Messages IDs - Return and reason codes (RCxx) - Reason codes (RSNxxxx)
- ❑ Partial dump checks
 - Possibly key areas of storage is missing
 - Commands to determine this

Figure 6-48 Providing database search information

IBM database searches

At this point in time we have evaluated some of the available diagnostic data from the dumps. Look in *z/OS MVS Systems Codes*, SA22-7626 to find the meaning of an 0F4 abend. Figure 6-49 shows the explanation from this manual for a 0F4 abend.

Explanation: An error occurred in DFSMSdfp support.
Source: DFSMSdfp
System Action: Prior to the ABEND error occurring, a return code was placed in the general register 15 and a reason code in general register 0. An SVC dump has been taken unless the failure is in IGWSPZAP where register 15 contains 10. The DFSMSdfp recovery routines retry to allow the failing module to return to its caller. See DFSMS/MVS DFSMSdfp Diagnosis Guide for return code information.
Programmer Response: System error. Rerun the job.
System Programmer Response: If the error recurs and the program is not in error, search problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide the JCL, the SYSOUT output for the job, and the logrec data set error record.

Figure 6-49 Documented abend S0F4 Information

Note: The 9 volumes of the *z/OS MVS System Messages and Systems Codes*, SA22-763x manuals should always be your first reference point for possible causes.

Build search arguments for IBM databases

These are recommended formats to be used when querying the problem database or reporting problems. These are not the only formats that are used, and some creativity and imagination can assist with expanding your search. These search arguments are also called a symptom string. If the problem being diagnosed was already reported and the symptoms entered in the database, the search will produce a match. Figure 6-50 displays what we know of the abend details.

```
LOAD MODULE NAME: IGWLHLS - Maintenance level UW04733
CSECT NAME: IGWLGMO - Maintenance level UW03389
SYSTEM ABEND CODE: 00F4
ABEND REASON CODE: 00000024
RSN=12088C01
```

Figure 6-50 Abend details

This information can be used to build the IBM problem database search arguments. The search arguments should use the following formats:

Abend: The format should be ABENDxxx or ABENDSxxx, where xxx is the abend code.

Messages The format should be MSGxxxxxxx, where xxxxxx is the message code.

Return and Reason Codes

The format should be RCxx, where xx is the reason or return code. A reason code alternative is:

Reason Codes The format can be RSNxxxxx, where xxxxx is the reason code.

Partial dump check

A partial or incomplete dump will be missing some key areas of storage that in most cases will make the dump useless when it comes to efficient problem diagnosis. We should first check whether the dump is ok. The command shown in Figure 6-51 can provide this information, but there could be another problem which will not be shown by this command. If the dump has been transferred via FTP, it could be that not all data has been sent correctly.

```
Command ==> l e0. block(0) 1(16)
```

Figure 6-51 Partial dump check command

The dump will not be a partial dump if you receive the following information:

```
LIST E0. BLOCK(0) LENGTH(X'10') AREA
E0. LENGTH(X'10')==>All bytes contain X'00'
```

If you get a bad return like the one shown below, you need to refer to the z/OS data areas manual. These codes are mapped by the SDRSN control block.

```
LIST E0. BLOCK(0) LENGTH(X'10') AREA
BLOCK(0) ADDRESS(E0)
000000E0. 00000000 30000000 00000000 00000000
```

For this example, you will find:

- 20000000 - The system detected an error in the SVC dump task and gave recovery control
- 10000000 - The SVC dump task failed



z/OS Language Environment

Language Environment provides a common run-time environment across multiple high level languages (HLLs). These languages include:

- ▶ COBOL
- ▶ C/C++
- ▶ PL/I
- ▶ FORTRAN
- ▶ Assembler (not HLL)

Language Environment establishes a common run-time environment for all participating HLLs. It combines essential run-time services, such as routines for run-time message handling, condition handling, and storage management. All of these services are available through a set of interfaces that are consistent across programming languages. You may either call these interfaces yourself, or use language-specific services that call the interfaces. With Language Environment, you can use one run-time environment for your applications, regardless of the application's programming language or system resource needs.

Language Environment consists of:

- ▶ Basic routines that support starting and stopping programs, allocating storage, communicating with programs written in different languages, and indicating and handling conditions.
- ▶ Common library services, such as math services and date and time services, that are commonly needed by programs running on the system. These functions are supported through a library of callable services.
- ▶ Language-specific portions of the run-time library. Because many language-specific routines call Language Environment services, behavior is consistent across languages.

7.1 Language Environment ABEND and CEEDUMP handling

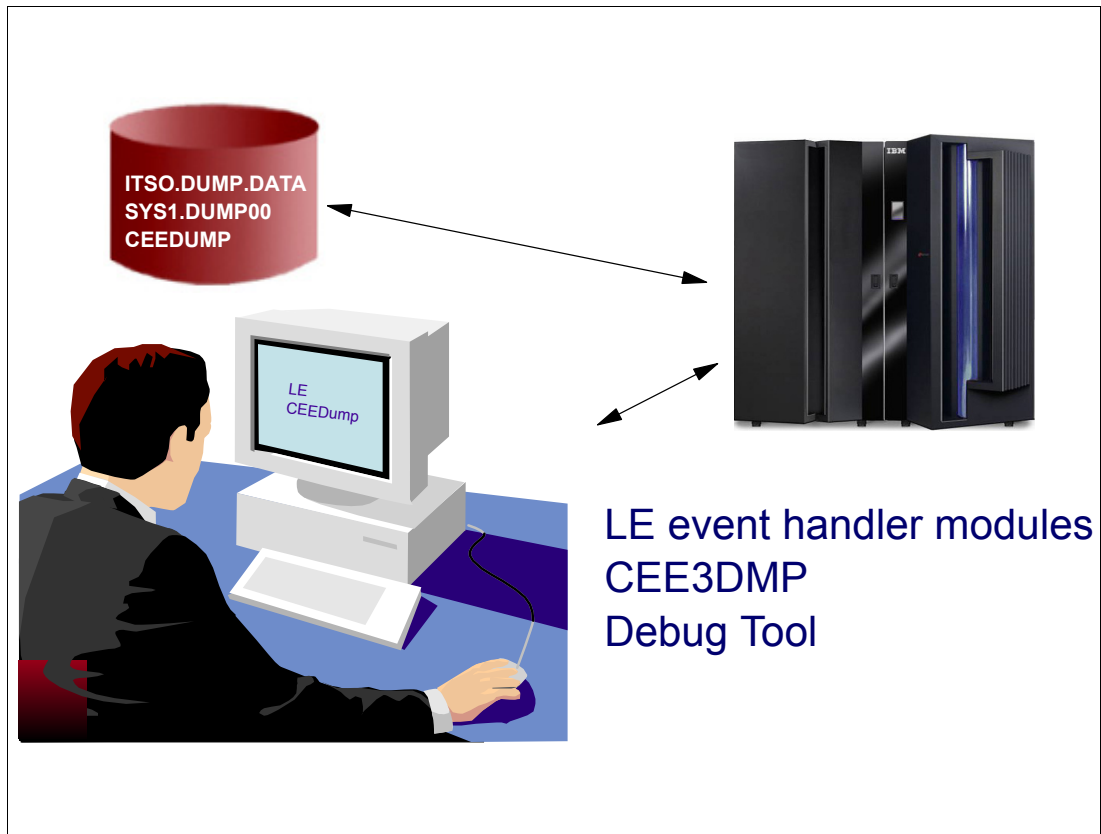


Figure 7-1 Language Environment ABEND and CEEDUMP handling

Run-time environment

A run-time environment provides facilities, such as storage control, system time and date functions, error processing, message processing and other system functions to the high-level languages. The run-time library is “called” by the user program to perform these functions. Before Language Environment, each high-level language had its own run-time library, but Language Environment has combined the functionality required by each language into a single run-time environment. Currently, most problems in Language Environment and member language routines can be determined with the use of a debugging tool or through information provided in the Language Environment dump.

Language Environment event handler modules

There are two common execution library (CEL) modules that will indicate a failure, but the cause will be elsewhere. The first is CEEHDSP, which schedules the Language Environment CEEDUMP to be taken. The second module is CEEPLPKA, which will always indicate an ABENDU4039 or ABENDU4038 no matter what the original error. Your diagnostic methodology should exclude failures in these two modules.

- The Language Environment event handler modules are identified as CEEExxx where xxx represents the language, as follows:
 - 003** C/C++ Run-time (that is, CEEEV003)
 - 005** COBOL
 - 007** FORTRAN
 - 008** DCE

010	PL/I
012	Debug Tool

CEE3DMP

For non-64-bit, the CEE3DMP callable service generates a dump of the run-time environment for Language Environment and the member language libraries at the point of the CEE3DMP call. You can call CEE3DMP directly from an application routine.

Depending on the CEE3DMP options you specify, the dump can contain information about conditions, tracebacks, variables, control blocks, stack and heap storage, file status and attributes, and language-specific information.

All output from CEE3DMP is written to the default ddname CEEDUMP. CEEDUMP, by default, sends the output to the SDSF output queue. You can direct the output from the CEEDUMP to a specific SYSOUT class by using the environment variable, `_CEE_DMPTARG=SYSOUT(x)`, where x is the output class.

Debug Tool

Debug tools are designed to help you detect errors early in your routine. IBM offers Debug Tool, a comprehensive compile, edit, and debug product that is provided with the C/C++, Enterprise COBOL for z/OS, COBOL for OS/390 and VM, COBOL for MVS and VM, PL/I for MVS and VM, VisualAge® PL/I, and VisualAge for Java™ compiler products.

You can use the IBM Debug Tool to examine, monitor, and control how your routines run, and debug your routines interactively or in batch mode. Debug Tool also provides facilities for setting breakpoints and altering the contents and values of variables. Language Environment run-time options can be used with Debug Tool to debug or analyze your routine. Refer to the Debug Tool publications for a detailed explanation of how to invoke and run Debug Tool.

7.2 Common Language Environment messages

- ❑ Run-time messages provide users with:
 - Information about a condition
 - Possible solutions for any errors that occurred
- ❑ Issued by Language Environment:
 - Common routines
 - Language-specific run-time routines
- ❑ Message content:
 - Message prefix
 - Message number
 - Severity code
 - Descriptive text

Figure 7-2 Debugging with run-time messages

Debugging with run-time messages

Run-time messages provide users with additional information about a condition, and possible solutions for any errors that occurred. They can be issued by Language Environment common routines or language-specific run-time routines and contain a message prefix, message number, severity code, and descriptive text.

The first step in debugging your routine is to look up any run-time messages. To find run-time messages, check the message file:

- ▶ On z/OS, run-time messages are written by default to ddname SYSOUT. If SYSOUT is not specified, then the messages are written to SYSOUT=*.
- ▶ On CICS, the run-time messages are written to the CESE transient data QUEUE.

Message content

In the following sample Language Environment message, the content is as follows:

CEE3206S The system detected a specification exception.

- ▶ The message prefix is CEE.
- ▶ The message number is 3206.
- ▶ The severity code is S.
- ▶ The message text is “The system detected a specification exception.”

7.3 Language Environment message abend prefixes

❑ Common LE messages and abends

- LE prefixes for messages
- Common CEL messages for 0Cx conditions
- Common CEL abends
- Condition codes

Figure 7-3 Understanding message prefixes and condition codes

Language Environment message abend prefixes

The message prefix indicates the Language Environment component that generated the message. The message prefix is the first three characters of the message number and is also the facility ID in the condition token. The following messages and abend prefixes can assist with problem diagnosis:

CEE	Is output by common execution library (CEL) modules, but may be reporting a problem elsewhere
IGZ	Is output by COBOL
IBM	Is output by PL/I
AFH	Is output by FORTRAN
EDC	Is output by C/C++

Some common CEL messages that indicate exception (0Cx) conditions are:

- ▶ CEE3201 = ABEND0C1
- ▶ CEE3204 = ABEND0C4
- ▶ CEE32xx = ABEND0Cy, where y is the hex equivalent of decimal xx

Message CEE3250 indicates a non-exception (0Cx) abend has occurred.

Common CEL abends

- U4038** Some “severe” error occurred but no dump was requested.
- U4039** Some “severe” error occurred and a dump was requested.
- U4083*** Backchain in error - only occurs after some other error.
- U4087*** Error during error processing.
- U4093*** Error during initialization.
- U4094*** Error during termination.

The * indicates that a reason code is required for this message to be meaningful.

Condition code token example

The following condition code example should show how to get the meaning of this information:

00030C84 59C3C5C5 xxxxxxxx

The condition code breaks down in the following way:

0003 | 0C84 | 59 | C3C5C5 | xxxxxxxx

0003 indicates severity and the other possibilities are:

- 0000** Informational (I)
- 0001** Warning (W)
- 0002** Error (E)
- 0003** Severe (S)
- 0004** Critical (C)

The other fields are as follows:

- 0C84** Hex message number (3204)
- 59** Flags (ignore)
- C3C5C5** Facility ID (message prefix)
- xxxxxxx** Instance specific information (internal use)

This token represents message CEE3204S.

7.4 Collecting debug documentation

☐ Specify run-time options

➤ `ABTERMENC(ABEND) TERMTHDACT(UADUMP) TRAP(ON)`

☐ Include a SYSMDUMP DD statement in the JCL

➤ `SPACE=(CYL,(100,100),RLSE),DISP=(NEW,DELETE,CATLG),
DSN=dump_dataset_name,LRECL=4160,RECFM=FBS`

☐ SDATA statement in the IEADMR00 parmlib member

Figure 7-4 Specifying information for debugging

Specifications to obtain debug documentation

There are several run-time options that affect debugging in Language Environment. The TEST run-time option, for example, can be used with a debugging tool to specify the level of control the debugging tool has when the routine being initialized is started, as follows:

- ▶ The ABPERC, CHECK, DEPTHCONDLMT, ERRCOUNT, HEAPCHK, INTERRUPT, TERMTHDACT, TRACE, TRAP, and USRHDLR options affect condition handling.
- ▶ The ABTERMENC option affects how an application ends (that is, with an abend or with a return code and reason code) when an unhandled condition of severity 2 or greater occurs.

Language Environment and batch methods for collecting dumps

Using the following methods, are ways to specify how to collect dumps in the event of an error or ABEND condition:

- ▶ Specify the following run-time options:

`ABTERMENC(ABEND) TERMTHDACT(UADUMP) TRAP(ON)`

For information about how to specify run-time options, refer to the section “Specifying Runtime Options under z/OS Batch” in *z/OS XL C/C++ User's Guide*, SC09-4767.

- ▶ Include a SYSMDUMP DD card in the JCL by specifying the following parameters:

`SPACE=(CYL,(100,100),RLSE),DISP=(NEW,DELETE,CATLG),
DSN=dump_dataset_name,LRECL=4160,RECFM=FBS`

► IEADMR00 parmlib member

IEADMR00 contains IBM defaults or installation parameters for ABDUMP, for use when an ABEND dump is written to a SYSMDUMP data set.

The system writes a SYSMDUMP as the core dump of a forked address space that runs a z/OS UNIX process. A core dump is written to an HFS file on behalf of the user experiencing the error. To obtain sufficient diagnostic data without consuming excessive storage in the file system, request the following options in IEADMR00:

SDATA=(RGN,TRT,SUM)

Note: Ensure that your IEADMR00 parmlib member reflects the following SDATA defaults:

SDATA=(NUC,SQA,LSQA,RGN,TRT,LPA,CSA,GRSQ,SUM)

See *z/OS MVS Initialization and Tuning Reference*, SA22-7592 for information about the IEADMR00 parmlib member and the SDATA parameter.

7.5 Language Environment and CICS debugging

- ❑ **CESE transient data queue**
 - Messages are written to this data queue
- ❑ **CICS system dump**
 - Dump is useful for diagnosing problems
 - Generate a U4039 ABEND by either:
 - CEMT SET TRD(4039) SYS ADD - (command)
 - TERMTHDACT(UADUMP) ABTERMENC(ABEND) TRAP(ON) - (Run-time options)
- ❑ **Procedure for 40xx dumps with CICS**

Figure 7-5 Debugging with Language Environment and CICS

Debugging under CICS

Under CICS, the Language Environment run-time messages, Language Environment traceback, and Language Environment dump output are written to the CESE transient data queue. The transaction identifier, terminal identifier, date, and time precede the data in the queue. The CESE transient data queue is defined in the CICS destination control table (DCT). The CICS macro DFHDCT is used to define entries in the DCT.

Under CICS, Language Environment run-time messages are written to the CESE transient data queue. A sample Language Environment message that appears when an application abends due to an unhandled condition from an EXEC CICS command is:

```
P039UTV9 19910916145313 CEE3250C The System or User ABEND AEIO was issued.  
P039UTV9 19910916145313          From program unit UT9CVERI at entry point  
                                +0000011E at P039UTV9 19910916145313  
                                at offset address 0006051E.
```

CICS system dump

Under CICS, a system dump provides the most useful information for diagnosing problems. However, if you have a Language Environment U4038 abend, CICS will not generate a system dump. In order to generate diagnostic information for a CICS run-time environment with a language Environment U4038 abend, you must create a Language Environment U4039 abend.

Perform the following steps to generate a system dump in a CICS run-time environment:

- Specify run-time options TERMTHDACT(UAONLY, UADUMP, or UATRACE), ABTERM(ABEND), and TRAP(ON). The TERMTHDACT suboption determines the level of detail of the Language Environment formatted dump.

TERMTHDACT(UADUMP) ABTERMENC(ABEND) TRAP(ON) produces a CICS transaction dump. It will never produce a SYSUDUMP/SYSABEND/SYSMDUMP since Language Environment's ESTAE routine does not get driven. Information APAR II13228 explains how to find the PSW and GPRs at the time of failure.

- In a CICS environment, you automatically receive the default transaction dump unless you disable it by using the CEMT transaction, and by specifying the dump code '4039'. Update the transaction dump table with the CICS supplied CEMT command:

```
CEMT SET TRD(4039) SYS ADD
```

A transaction dump should be produced for all Language Environment ABENDU40xx series abends, except ABENDU4038. If a transaction dump is not enough, request a CICS system dump.

Note: A CICS system dump of an ABENDU4038 is not helpful because it is taken at the time of the last termination, not at the point of detection. Instead, specify the following:

```
TERMTHDACT(UADUMP) ABTERMENC(ABEND) TRAP(ON)
```

This produces a CICS transaction dump with an ABENDU4039.

Note: SLIP commands on C=U40xx will not work in CICS. SLIP commands on C=0Cx will work in a CICS environment but not in batch.

Procedure for an SVC dump for 40xx abends under CICS

Here are the steps to get an SDUMP for a specific 40xx transaction abend under CICS:

1. Make sure the CICS region is started with the DUMP=YES SYSIN input (SIP) parameter.
2. Make sure the SYS1.DUMP- data sets are available. Most customers should have all this already set up.

As an alternative, the dynamic allocation facility may be used, as follows:

```
DUMPDS ALLOC ADD,VOL=xxxxxx  
DUMPDS ALLOC=ACTIVE
```

After these commands, MVS dynamically allocates data sets on the xxxxxx volume containing the dump with the following type of name:

```
SYS1.DUMP.D970910.T191701.SY1.S00001
```

3. Once the CICS region is up, log on and issue the following:

```
CEMT SET TRD(40xx) ADD SYS
```

Substituting the real dump code, for example: 4088 for 40xx. Following is a sample of what CICS sends back for this:

```
SET TRD(4088) ADD SYS  
STATUS: RESULTS - OVERTYPE TO MODIFY  
Trd(4088) Tra Sys      Loc Max( 999 ) Cur(0000)
```

4. Now run the transaction that creates the U40xx abend. A system, or SVC dump, should be produced at the point of the abend. This procedure will work for any transaction dump under CICS, not just U40xx abends.

7.6 Language Environment and UNIX System Services dumps

- ❑ **SYSDUMP for z/OS UNIX processes**
 - Set environment variable for where dump is written
 - `_BPXK_MDUMP=filename`
- ❑ **LE run-time options for z/OS UNIX**
 - `export _CEE_RUNOPTS="termthdact(suboption)"`
- ❑ **Steps to take system dumps**
 - Specify where to write the dump
 - Specify the run-time options
 - Rerun the failing program to take the dump

Figure 7-6 Taking dumps for z/OS UNIX processes

Dumps with UNIX System Services

You can dynamically request a SYSDUMP by using the SIGDUMP signal. Use the `_BPXK_MDUMP` environment variable to specify where the SYSDUMP is to be written to. You can also use `F BPXOINIT,DUMP=pid` to request a SYSDUMP. A SIGDUMP signal is then sent to the specified process. For both the SIGDUMP signal and the `F BPXOINIT,DUMP` command, the `_BPXK_MDUMP` environment variable must be set to an MVS data set name. If it is set to a UNIX file name or defaulted to OFF, then both the SIGDUMP signal and the `F BPXOINIT,DUMP` command may be ignored.

If you have a loop, hang, or wait condition in a z/OS UNIX process and need a dump or diagnosis, you need to dump several types of data:

- ▶ The kernel address space
- ▶ Any kernel data spaces that may be associated with the problem
- ▶ Any process address spaces that may be associated with the problem
- ▶ Appropriate storage areas containing system control blocks (for example, SQA, CSA, RGN, TRT)

Language Environment run-time options

Using UNIX System Services and the Language Environment run-time options. Consider the following steps to take system dumps:

1. To write the system dump to a data set, issue the command:

```
export _BPXK_MDUMP=filename
```

where filename is a fully qualified data set name with LRECL=4160 and RECFM=FBS, or
where filename is a fully qualified HFS filename.

2. Specify Language Environment run-time options:

```
export _CEE_RUNOPTS="termthdact(suboption) "
```

where suboption = UAONLY, UADUMP, UATRACE, or UAIMM. If UAIMM is set,
TRAP(ON,NOSPIE) must also be set. The TERMTHDACT suboption determines the level
of detail of the Language Environment formatted dump. For

3. Rerun the program and the dump will be written to the specified data set.

7.7 Understanding CEEDUMP

- ❑ CEE3DMP callable service
 - Writes dumps to default DDname CEEDUMP
 - Dump output goes to JES spool
- ❑ Batch job JCL example
 - `//CEEDUMP DD SYSOUT=*`
- ❑ z/OS UNIX and CEEDUMP

Figure 7-7 Using CEEDUMP

Additional hints to collect error information

All output from CEE3DMP, the callable service that generates a dump of the run-time environment for Language Environment, is written to the default ddname CEEDUMP. CEEDUMP, by default, sends the output to the JES output queue.

The IBM-supplied default settings for CEE3DMP are:

```
ENCLAVE(ALL) TRACEBACK  
THREAD(CURRENT) FI S VARIABLES NOBLOCKS NOSTORAGE  
STACKFRAME(ALL) PAGESIZE(60) FNAME(CEEDUMP)  
CONDITION ENTRY NOGENOPTS REGSTOR(96)
```

Batch JCL example

For batch, use a CEEDUMP DD card to route the dump to a specific SYSOUT or data set. If not specified, it will be dynamically allocated to SYSOUT=* by default.

In the following JCL example, The ddname of the dump output file can be CEEDUMP. If you do not define this ddname, Language Environment creates a default CEEDUMP file to contain the dump output. The LRECL of the dump output file must be at least 133 bytes to prevent dump records from wrapping. If you write the dump output to the SYSOUT file, make sure you change the default LRECL size of 121 to 133 to prevent from wrapping.

```
//SYSLOGD  PROC
//SYSLOGD  EXEC PGM=SYSLOGD,REGION=30M,TIME=NOLIMIT
//          PARM='POSIX(ON) ALL31(ON)/ -f /etc/syslogd.conf'
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSOUT    DD SYSOUT=*
//SYSERR    DD SYSOUT=*
//CEEDUMP   DD SYSOUT=*
```

Note: You can change the default SYSOUT class by specifying a CEEDUMP DD, or by setting the environment variable, `_CEE_DMPTARG=SYSOUT(x)`, where x is the preferred SYSOUT class.

z/OS UNIX and CEEDUMP

If your application is running under z/OS UNIX and is either running in an address space you issued a `fork()` to, or is invoked by one of the `exec` family of functions, the dump is written to the hierarchical file system (HFS). Language Environment writes the CEEDUMP to one of the following directories in the specified order:

- ▶ The directory found in environment variable `_CEE_DMPTARG`, if found
- ▶ The current working directory, if the directory is not the root directory (`/`), and the directory is writable
- ▶ The directory found in environment variable `TMPDIR` (an environment variable that indicates the location of a temporary directory if it is not `/tmp`)
- ▶ The `/tmp` directory

Examining dumps

In most cases Language Environment condition handling will trap original program checks (like `ABEND0C4`) and turn them into corresponding Language Environment conditions (like `CEE3204S`). After storing information about the original program check, Language Environment will terminate with an `ABENDU40xx`. When examining a dump of a `U40xx` the PSW and registers can be found in a control block called the `ZMCH`. `APAR II11016` is specifically written for those running Language Environment in a non-CICS environment, as the control block structure and condition handling changes when running under CICS. Depending on how the dump was produced it might be a formatted or unformatted one. The formatted one can be browsed. The unformatted one needs to be accessed by IPCS.

Figure 7-8 shows an `ABEND0C9` problem. The dump is a formatted one. These dumps are mostly useful for the program owner. Let us start with the joblog information:

```

IEA995I SYMPTOM DUMP OUTPUT
  USER COMPLETION CODE=4039 REASON CODE=00000000   TIME=21.45.36   SEQ=03447
CPU=0000   ASID=0153
PSW AT TIME OF ERROR 078D1000   A3E207B0   ILC 2   INTC 0D
  ACTIVE LOAD MODULE      ADDRESS=23E19D30 OFFSET=00006A80
  NAME=CEEPLPKA
  DATA AT PSW 23E207AA - 00181610 0A0D58D0 D00498EC
  GPR 0-3 84000000 84000FC7 00024478 23E207B0
  GPR 4-7 23E178A0 00000000 00024478 00025017
  GPR 8-11 23E238A5 23E228A6 000243D0 A3E206E0
  GPR 12-15 00015910 00026180 A3E22F1E 00000000
END OF SYMPTOM DUMP
IEA993I SYSMDUMP TAKEN TO JMONTI.LECOBED1.SYSMDUMP
IEF450I JMONTI0B GO - ABEND=SOC9 U0000 REASON=00000009

```

Figure 7-8 Joblog information

You should also get a program-related output (Figure 7-9).

```

CEE3209S The system detected a fixed-point divide exception.
From compile unit COB0LED2 at entry point COB0LED2 at statement 13
at compile unit offset +00000308 at address 23E029E0.

```

Figure 7-9 Program output

7.8 ZMCH control block

```
Machine State
+000348  MCH_EYE:ZMCH
+000350  MCH_GPR00:00026180      MCH_GPR01:00000000
+000358  MCH_GPR02:00000000      MCH_GPR03:0000000A
+000360  MCH_GPR04:00000000      MCH_GPR05:00046038
+000368  MCH_GPR06:00000000      MCH_GPR07:00FCCBF0
+000370  MCH_GPR08:23F1B100      MCH_GPR09:23F17700
+000378  MCH_GPR10:23E027E0      MCH_GPR11:23E028B0
+000380  MCH_GPR12:23E027D4      MCH_GPR13:000260C8
+000388  MCH_GPR14:A3E029D4      MCH_GPR15:A3E02916
+000390  MCH_PSW:078D2000 A3E029E2      MCH_ILC:0002      MCH_IC1:00
+00039B  MCH_IC2:09  MCH_PFT:00000000  MCH_FLT_0:00000000 00000000
+0003A8  MCH_FLT_2:00000000 00000000  MCH_FLT_4:00000000 00000000
+0003B8  MCH_FLT_6:00000000 00000000  MCH_EXT:00000000
+000418  MCH_FLT_1:00000000 00000000  MCH_FLT_3:00000000 00000000
+000428  MCH_FLT_5:00000000 00000000  MCH_FLT_7:00000000 00000000
+000438  MCH_FLT_8:00000000 00000000  MCH_FLT_9:00000000 00000000
```

Figure 7-10 Display of the ZMCH control block

IPCS VERBEXIT LEDATA

The Language Environment IPCS VERBEXIT LEDATA generates formatted output of the Language Environment run-time environment control blocks from a system dump. The LEDATA VERBEXIT is invoked with the ALL parameter. The system dump being formatted was obtained by specifying the TERMTHDACT(UADUMP) run-time option.

ZMCH (machine state information at time of exception)

The ZMCH control block, shown in Figure 7-10, shows the information at the time of the error. It includes PSW and the registers. Right at the beginning is the ZMCH eye catcher

Traceback information

Traceback information as shown in Figure 7-11 provides a module flow from involved modules. The traceback output is read from bottom to top.

```
CEE3DMP V2 R10.0: Condition processing resulted in the unhandled condition.  
02/26/01 9:48:42 PM
```

Information for enclave COBOLED1

Information for thread 8000000000000000

Traceback:

DSA Addr	PU Addr	PU Offset	Entry	Stmt	Load Mod	Service	Status
00024018	23E208A8	+000026A6	CEEHDSP		CEEPLPKA	UQ24548	Call
000260C8	23E026D8	+00000308	COBOLED2	13	COBOL1		Exception
00026018	23E00978	+0000033E	COBOLED1	14	COBOL1		Call

Figure 7-11 Traceback information

Condition information

The condition information is provided by the condition information block (CIB) address. Reg 13 points to DSA (save area) and register 12 points to CAA (common anchor area). The CIB also provides the storage area where we can see the instruction flow getting the error.

```
Condition Information for Active Routines  
Condition Information for COBOLED2 (DSA address 000260C8)  
CIB Address: 00024478  
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.  
Location:  
Program Unit: COBOLED2 Entry: COBOLED2 Statement: 13 Offset: +00000308  
Machine State:  
ILC..... 0002      Interruption Code..... 0009  
PSW..... 078D2000 A3E029E2  
GPR0..... 00026180  GPR1..... 00000000  GPR2..... 00000000  GPR3..... 0000000A  
GPR4..... 00000000  GPR5..... 00046038  GPR6..... 00000000  GPR7..... 00FCCBF0  
GPR8..... 23F1B100  GPR9..... 23F17700  GPR10.... 23E027E0  GPR11.... 23E028B0  
GPR12.... 23E027D4  GPR13.... 000260C8  GPR14.... A3E029D4  GPR15.... A3E02916  
Storage dump near condition, beginning at location: 23E029D0  
+000000 23E029D0 45E0913A 48208000 8E200020 48408002 1D244030 800445E0  
913A9140
```

Figure 7-12 Condition information

The PSW at the time of the error points to instruction 1D24, which is a DR (divide register). Looking at the register value you see that GPR4 is 00000000. This leads to our ABEND0C9.

The dump will show the CAA (common anchor area) control block pointed to by register 12 (GPR12) followed by the process control block (PCB), the region control block (RCB), and the enclaved data block (EDB). It shows that you can find the run-time options (runopts) in the formatted dump.

7.9 IPCS and Language Environment

- ❑ **IPCS LE problem diagnosis**
 - **Display LE run-time options**
 - VERBX CEERRIP - format the dump data
 - VERBX LEDATA - search for error TCB
 - **If no error TCB found**
 - Load SYSMDUMP into IPCS
 - Issue IP SUMM FORMAT command
 - Issue BOTTOM or MAX (PF8) command
 - Find the TCB
 - Issue IP VERBX LEDATA 'CM TCB(xxxxxxxxx)'
 - **Should now see ZMCH**

Figure 7-13 IPCS commands to diagnose an Language Environment dump

IPCS Language Environment problem diagnosis

All the information you can find in a formatted Language Environment dump can also be found in the SVC dump. IPCS provides some facilities to assist with Language Environment problem diagnosis. The IPCS commands VERBX LEDATA and VERBX CEEERRIP show the Language Environment run-time options and general information about your Language Environment environment at the time of the failure.

CEEERRIP

CEEERRIP is the Language Environment diagnostic module that is used to format the dump data. Figure 7-11 on page 197 shows the result of the VERBX CEEERRIP 'CEEDUMP' command and related traceback information.

LEDATA

LEDATA searches for an error TCB and formats the control blocks for that TCB. If there is no error TCB (shown in a console dump) the TCB or CAA keywords will need to be specified as follows:

1. Load SYSMDUMP into IPCS (instructions on how to get a SYSMDUMP with Language Environment can be found in info APAR II10573).
2. Issue the command:

```
IP SUMM FORMAT
```

3. Issue the command:

BOTTOM or MAX (PF8)

4. Find the TCB with a non-zero completion code. Now issue the command:

IP VERBX LEDATA 'CM TCB(xxxxxxx)'

Continue dump analysis if no ZMCH

If this does not format the ZMCH, locate the CAA with the following steps:

1. Issue the following command, where xxxxxxxx represents the address of the failing TCB:

F 'TCB: xxxxxxxx' PREV

2. Issue the command:

F RTM2WA

3. Press PF5 to search again. If there is a second RTM2WA for the failing TCB, then use the data contained in the RTM2WA.

4. Find the address in Register 12.

5. Issue the command “=1” to go into browse mode. Or select Option 1 from the IPCS Primary Option menu.

6. Issue the command L yyyyyyyy, where yyyyyyyy represents the address obtained from Register 12.

7. Now verify whether this is a valid CAA with the following:

- a. At the address in R12 verify that the value is “xxxx0800”.

- b. Issue L X-18 and the eyecatcher should be CEECAA.

This indicates we have found a valid CAA and can now issue the command:

IP VERBX LEDATA 'CM CAA(yyyyyyy)'

You have now formatted the ZMCH, so you can begin locating the values you were looking for.

Note: The above steps do *not* pertain to an ABENDU4036 dump.

Commands for additional dump information

The following IPCS VERBX commands can provide useful information from the dump:

VERBX CEEERRIP 'SUMMARY'	Like formatted dump
VERBX CEEERRIP 'CEEDUMP'	Traceback
VERBX CEEERRIP 'CM'	Condition Information
VERBX CEEERRIP 'NTHREADS(*)'	Traceback for each Language Environment-enabled TCB



Debug and maintenance tools

Debugging a dump does not always provide all necessary information. Sometimes you can locate a module name but cannot determine its maintenance level.

This chapter describes System Modification Program Extended (SMP/E) as the basic tool for installing and maintaining software in OS/390 or z/OS systems and subsystems. It controls the changes at the element level by:

- ▶ Selecting the proper levels of elements to be installed from a large number of potential changes
- ▶ Calling system utility programs to install the changes
- ▶ Keeping records of the installed changes

In other cases you may find a load module name (LMOD) but cannot find the CSECT or member name. AMBLIST is a utility that provides the internal CSECTs of a load module. In addition you can list the object code.

This chapter describes some of the diagnostic aids that can be used via members in SYS1.PARMLIB. These facilities enable you to simplify the diagnostic data collection process by enabling you to prepare data collection parameters in advance to ensure that complex dump procedures do not have to be typed in when a problem arises and prompt, error free action is required. The SYS1.PARMLIB members that can simplify the diagnostic data collection process include:

- ▶ IEAABD00
- ▶ IEADMP00
- ▶ IEADMR00
- ▶ IEADMCxx
- ▶ IEASLPxx

8.1 Using SMP/E

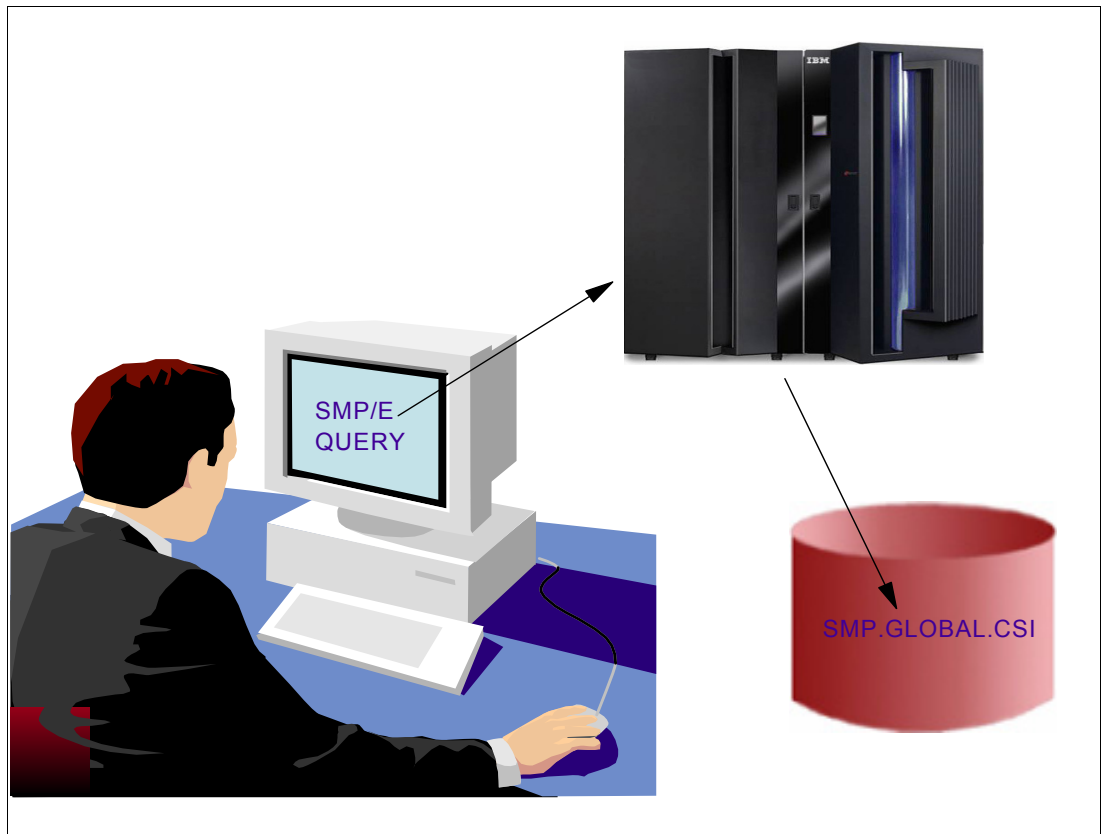


Figure 8-1 Using SMP/E

Using SMP/E

SMP/E is a tool designed to manage the installation of software products on your z/OS system and to track the modifications you make to those products. Usually, it is the system programmer's responsibility to ensure that all software products and their modifications are properly installed on the system. The system programmer also has to ensure that all products are installed at the proper level so all elements of the system can work together. At first, that might not sound too difficult, but as the complexity of the software configuration increases, so does the task of monitoring all the elements of the system.

To get a module level using SMP/E you should select the SMP/E PRIMARY OPTION MENU. It is shown in Figure 8-2.

----- SMP/E PRIMARY OPTION MENU ----- SMP/E 34.10

====> 3

- | | | |
|---|---------------------|--|
| 0 | SETTINGS | - Configure settings for the SMP/E dialogs |
| 1 | ADMINISTRATION | - Administer the SMPCSI contents |
| 2 | SYSMOD MANAGEMENT | - Receive SYSMODs and HOLDDATA
and install SYSMODs |
| 3 | QUERY | - Display SMPCSI information |
| 4 | COMMAND GENERATION | - Generate SMP/E commands |
| 5 | RECEIVE | - Receive SYSMODs, HOLDDATA and
support information |
| 6 | MIGRATION ASSISTANT | - Generate Planning and Migration Reports |
| 7 | ORDER MANAGEMENT | - Manage ORDER entries in the global zone |
| | | |
| D | DESCRIBE | - An overview of the dialogs |
| T | TUTORIAL | - Details on using the dialogs |
| W | WHAT IS NEW | - What is New in SMP/E |

Specify the name of the CSI that contains the global zone:

SMPCSI DATA SET ====> 'ZOSR17.GLOBAL.CSI'

(Leave blank for a list of SMPCSI data set names.)

Specify YES to have DD statements for SYSOUT and temporary
data sets generated. Specify NO, to use DDDEFS.

Generate DD statements ====> NO

Licensed Materials - Property of IBM

5694-A01 5655-G44

(C) Copyright IBM Corp. 1982, 2005

Figure 8-2 Get the module level and load module information

Enter the SMPCSI data set name and select Option 3 Query, as shown in Figure 8-2.

QUERY SELECTION MENU

====> 2

- | | | |
|---|------------------|--|
| 1 | CSI QUERY | - Display SMPCSI entries |
| 2 | CROSS-ZONE QUERY | - Display status of an entry in
all zones |
| 3 | SOURCEID QUERY | - Display SOURCEIDs for specified zone |
| | | |
| D | DESCRIBE | - Overview of using QUERY |
| | | |
| T | TUTORIAL | - Information on using QUERY |

To return to the SMP/E primary option menu, enter END .

Figure 8-3 Get the module level and load module information

Now select 2 Cross-Zone Query as shown in Figure 8-3.

8.2 Find a load module

- ❑ Select the CROSS-ZONE QUERY
 - Enter the module name
 - Entry type is MOD
- ❑ From the ENTRY Selection panel
 - Place an S next to the target zone module
- ❑ From the CSI QUERY - MOD ENTRY panel
 - Load module information

Figure 8-4 Steps to find a load module

Find a load module

From the QUERY SELECTION MENU, Option 2 displays the panel shown in Figure 8-5.

CROSS-ZONE QUERY

====>

Specify the entry type and name to be queried:

ENTRY TYPE	====>	MOD	Entry type to be queried. To display a selection list of all valid entry types, leave ENTRY TYPE and ENTRY NAME blank
ENTRY NAME	====>	ATRFMQUR	Entry name to be queried.

To return to the Query selection menu enter the END command

Figure 8-5 Get module level and load module information

Enter the entry type you would like to get information from. In our case, MOD (module), and then add the module name, ATRFMQUR, as shown in Figure 8-5. Then press Enter and you should get the panel shown in Figure 8-6.

```

CSI CROSS-ZONE QUERY - ENTRY SELECTION          Row 2 to 17 of 19
===>                                           SCROLL ==> PAGE

Entry Type:  MOD
Entry Name:  ATRFMQUR

To return to the previous panel, enter END .

To select an entry from a zone, enter S next to the zone.

    * - Entry not found in zone.
    ** - Zone could not be allocated or is not initialized.

      ----- Status -----
      ZONE      FMID      RMID      LASTUPD  DISTLIB  UMID(S)
      -----
S  MVSD700  HBB7720  UA24095  HBB7720  AOSC5
   MVSD710  *
   MVSD711  *
   MVSTA00  HBB7720  UA27072  HBB7720  AOSC5
   MVSTA10  *
   MVSTA11  *

```

Figure 8-6 Get the module level and load module information

To get load module-related information select the target zone, in our case MVSD700, place an S as shown in Figure 8-6, and press Enter.

```

CSI QUERY - MOD ENTRY          Row 1 to 2 of 2
===>                           SCROLL ==> PAGE

To return to the previous panel, enter END .

Primary Command: FIND

Entry Type:  MOD                      Zone Name: MVSD700
Entry Name:  ATRFMQUR                  Zone Type: DLIB

      FMID:    HBB7720                      LASTUPD: HBB7720  TYPE=ADD
      RMID:    UA24095                      DISTLIB: AOSC5

Link-edit Parameters:
RENT,REFR,OL,NCAL

      -----
      LMOD      ATRAMPVX
      CSECTS    ATRFMQUR
      ***** Bottom of data *****
      *****

```

Figure 8-7 Get module level and load module information

If you would like to check whether a PTF is installed, the Entry Type should be SYSMOD. Use Entry Type LMOD to look for load module information.

8.3 AMBLIST job to get LMOD and source information

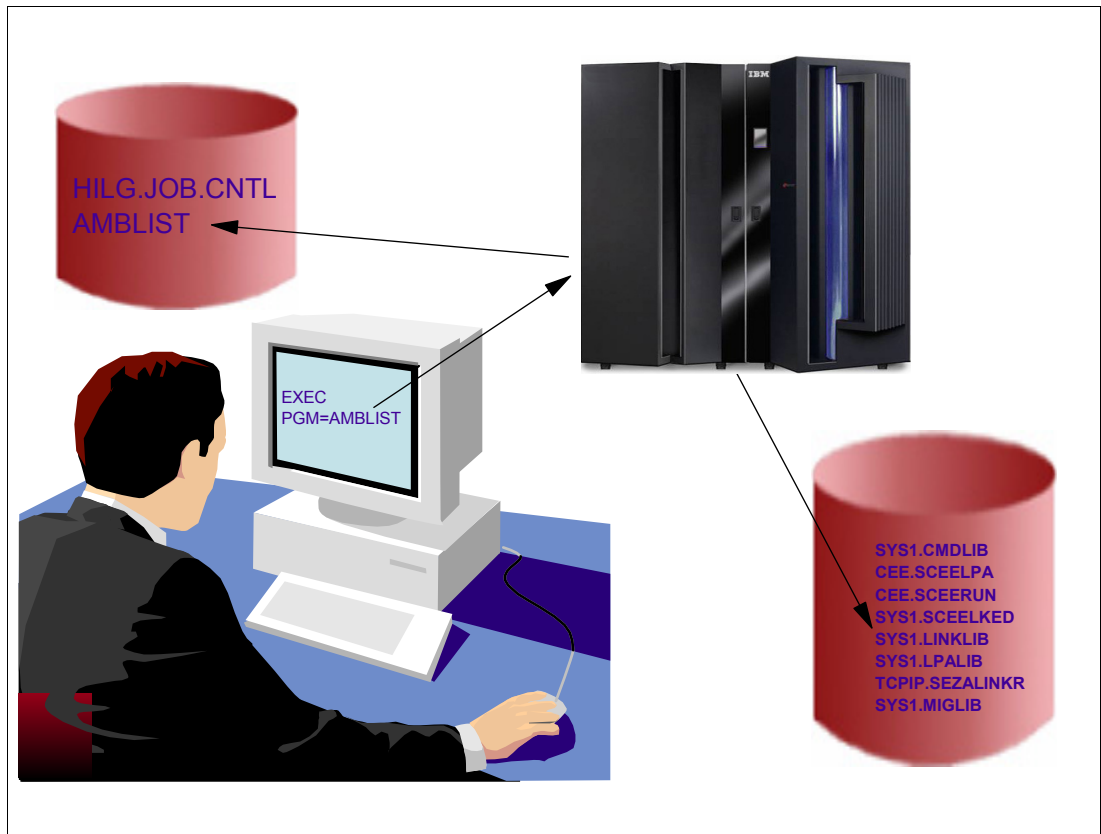


Figure 8-8 AMBLIST to get load module CSECTs and source listed

Using AMBLIST

AMBLIST is a very powerful utility that is easy to handle. It provides useful debug and diagnosis information. Use AMBLIST when you need information about the content of load modules and program objects or when you have a problem related to the modules on your system. AMBLIST is a program that provides lots of data about modules in the system, such as a listing of the load modules, map of the CSECTs in a load module or program object, list of modifications in a CSECT, map of modules in the LPA, and a map of the contents of the DAT-on (dynamic address translation) nucleus.

If you are analyzing a dump, for example, and can only get the load module name and not any CSECT name, you can use the AMBLIST JCL to get CSECT and, if necessary, source information. You need to know in which SYSLIB data set the load module resides. The following JCL shows an AMBLIST request for load Unix System Service module BPXINPVT, which should be in the SYS1.LINKLIB data set.

Obtaining AMBLIST output

To obtain AMBLIST output, you must code JCL, providing control statements as input to the job. These control statements dictate what type of information AMBLIST produces, as shown in Figure 8-9 on page 208. A snapshot of the output is shown in Figure 8-10 on page 208.

```
//HILGAA JOB 7904,HILGER,MSGLEVEL=(1,1),MSGCLASS=K,CLASS=A,
// NOTIFY=HILG3
//AMBLIST EXEC PGM=AMBLIST,REGION=OM
//*YSLIB DD DSN=SYS1.CMDLIB,DISP=SHR
//*YSLIB DD DSN=CEE.SCEELPA,DISP=SHR
//*YSLIB DD DSN=CEE.SCEERUN,DISP=SHR
//*YSLIB DD DSN=IOE.SIOELMOD,DISP=SHR
//*YSLIB DD DSN=SYS1.SCEELKD,DISP=SHR
//SYSLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//*YSLIB DD DSN=SYS1.LOTUS.LPALIB,DISP=SHR
//*YSLIB DD DSN=SYS1.LPALIB,DISP=SHR
//*YSLIB DD DSN=TCPIP.SEZALINK,DISP=SHR
//*YSLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//*YSLIB DD DSN=ISP.V4R4M0.SISPLPA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
LISTLOAD MEMBER=(BPXINPVT),OUTPUT=XREF
```

Figure 8-9 AMBLIST JCL job

```
MEMBER NAME: BPXINPVT
LIBRARY: SYSLIB
** ALIASES ** ENTRY POINT AMODE
BPXBDMI 001E80F8 31
BPXBDSI 001F1EF0 31
BPXFCSIN 002110E8 31
BPXFDNIN 002128F0 31
BPXFPINT 00214BE8 31
BPXFSLIT 00172228 31
BPXFSLM 00153E90 31
BPXFTCLN 00016858 31

CONTROL SECTION
LMOD LOC NAME LENGTH TYPE
00 BPXINPVT 3728 SD
3728 BPXTAVNO 4BD8 SD
8300 BPXPTCRE 25C8 SD
A8C8 P8CRECPY 6E SD
A938 P8CRERET 52 SD
A990 BPXNXFST 4460 SD
```

Figure 8-10 AMBLIST output

LISTLOAD control statement

Use the LISTLOAD control statement to obtain a listing of load module or program objects; see Figure 8-11 on page 209. The listed data can help you verify why certain link-edit errors might have occurred.

```
LISTLOAD
[OUTPUT={MODLIST|XREF|BOTH}]
[,TITLE=('title',position)]
[,DDN=ddname]
[,MEMBER={member| (member1,membern...)}]
[,RELOC=hhhhhhh]
[,ADATA={YES|NO}]
```

Figure 8-11 LISTLOAD control statement to obtain a listing of load module or program objects

8.4 IEAABD00, IEADMP00 and IEADMR00 members

- ❑ **IEAABD00 parmlib member**
 - Contains ABDUMP parameters for a SYSABEND dump data set
- ❑ **IEADMP00 parmlib member**
 - Contains ABDUMP parameters for a SYSUDUMP dump data set
- ❑ **IEASMR00 parmlib member**
 - Contains ABDUMP parameters for a SYSMDUMP dump data set
- ❑ **SDATA options for members**

Figure 8-12 SYS1.PARMLIB members for abend dumps

Parmlib members for abend dumps

IEAABD00 contains IBM defaults and/or installation-assigned parameters for ABDUMP, for use when an abend dump is written to a SYSABEND data set.

IEADMP00 contains IBM defaults and installation parameters for ABDUMP for use when an abend dump is written to a SYSUDUMP data set.

IEADMR00 contains IBM defaults and installation parameters for ABDUMP for use when an abend dump is written to a SYSMDUMP data set.

These members contain the SDATA and PDATA options that will be used when an abend dump is triggered.

SDATA options

Following are the SDATA options:

- | | |
|-----------------|---|
| ALLSDATA | All the following options are automatically specified (except ALLVNUC and NOSYM).

The following parameters request dump of specific SDATA areas, as indicated: |
| ALLVNUC | Entire virtual nucleus. SQA, LSQA, and the PSA are included. |
| NOSYM | No symptom dump is to be produced. |

SUM	Requests that the dump contain summary data, which includes the following: <ul style="list-style-type: none"> –Dump title. –Abend code and PSW at the time of the error. –If the PSW at the time of the error points to an active load module: (1) the name and address of the load module, (2) the offset into the load module indicating where the error occurred, and (3) the contents of the load module. –Control blocks related to the failing task. –Recovery termination control blocks. –Save areas. –Registers at the time of the error. –Storage summary consisting of 1K (1024) bytes of storage before and 1K bytes of storage after the addresses pointed to by the registers and the PSW. The storage will be printed only if the user is authorized to obtain it, and, when printed, duplicate addresses will be removed. –System trace table entries for the dumped address space.
NUC	Read/write portion of the control program nucleus. SQA, LSQA, and the PSA are included.
PCDATA	Program call information for the task being dumped.
SQA	The system queue area.
LSQA	Local system queue area for the address space. If storage is allocated for subpools 229, 230 and 249, they will be dumped for the current task.
SWA	Scheduler work area used for the failing task.
CB	Control blocks related to the failing task.
ENQ	Global resource serialization control blocks for the task.
TRT	System trace table and GTF trace, as available.
DM	Data management control blocks (DEB, DCB, IOB) for the task.
IO	IOS control blocks (UCB, EXCPD) for the task.
ERR	Recovery termination control blocks (RTM2WA, registers from the SDWA, SCB, EED) for the task.

8.5 PDATA options (only valid for IEADMP00)

ALLPDATA - Specifies all PDATA options

- PSW
- REGS
- SA
- SAH
- JPA
- LPA
- ALLPA
- SPLS
- SUBTASKS

Figure 8-13 The PDATA options for ABEND dumps

PDATA options for abend dumps

Following are the PDATA options:

- ALLPDATA** All the following options are automatically specified.
The following parameters request dump of specific PDATA areas, as indicated:
- PSW** Program status word at entry to abend.
- REGS** Contents of general registers at entry to abend.
- SA or SAH** SA requests save area linkage information and a backward trace of save areas.
This option is automatically selected if ALLPDATA is specified.
- SAH** Requests only save area linkage information.
- JPA** Contents of the job pack area that relate to the failing task. These include module names and contents.
- LPA** Contents of the LPA related to the failing task. These include module names and contents. Also includes active SVCs related to the failing task.
- ALLPA** Contents of both the job pack area and the LPA, as they relate to the failing task, plus SVCs related to the failing task.
- SPLS** User storage subpools (0-127, 129-132, 244, 251, and 252) related to the failing task.
- SUBTASKS** Problem data (PDATA) options requested for the designated task will also be in effect for its subtasks.

8.6 SDATA and PDATA recommendations

- ❑ SDATA and PDATA parameters that solve most problems
 - SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),
 - PDATA=(PSW,REGS,SPLS,ALLPA,SA)
- ❑ IEADMCxx parmlib member for dump commands
 - DUMP TITLE=(CICS Looping),PARMLIB=CI
 - DUMP COMM=(.....),PARMLIB=(xx)
- ❑ Setting SLIP traps
 - SYS1.PARMLIB(IEASLPxx)
 - SET SLIP=xx

Figure 8-14 SDATA and PDATA options for dumps and SLIP traps

SDATA and PDATA options

The following SDATA and PDATA parameters will provide you and IBM with sufficient data to solve most problems.

```
SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),  
PDATA=(PSW,REGS,SPLS,ALLPA,SA)
```

IEADMCxx (dump command parameter library)

IEADMCxx enables you to supply DUMP command parameters through a parmlib member. It enables the operator to specify the collection of dump data without having to remember and identify all the systems, address spaces, and data spaces involved.

This parmlib enables you to specify lengthy DUMP commands without having to reply to multiple writes to operator with reply (WTORs). Any errors in an original specification may be corrected and the DUMP command respecified.

IEADMCxx is an installation-supplied member of SYS1.PARMLIB that can contain any valid DUMP command. A dump command may span multiple lines and contain system static and (DUMP command SYMDEF defined) symbols and comments.

Figure 8-15 on page 214 shows a sample of what might be included in a SYS1.PARMLIB(IEADMCxx) member. As you can see, to key in this data when we need to capture a dump would be time-consuming and prone to errors. This simplifies the process

and when you need to capture a dump you can refer to the IEADMCxx member in the dump command. For example:

```
DUMP TITLE=(CICS Looping), PARMLIB=CI
```

where CI is the IEADMCxx parmlib member using the suffix, SYS1.PARMLIB(IEADMCCI).

The title is the name (1 to 100 characters) you want the dump to have. This title becomes the first record in the dump data set. COMM= and TITLE= are synonyms.

You can also use the parmlib parameter as follows:

```
DUMP COMM=(.....),PARMLIB=(xx)
```

```
TITLE=(DYNDUMP FOR IMS810I,IVP8IRC1,IVP8IDL1,IVP8IM11,
IVP8IM12,IVP8IM13,RRS,APPC)
JOBNAME=(IMS810I,IVP8IRC1,IVP8IDL1,IVP8IM11,IVP8IM12,IVP8IM13,
RRS,APPC)
DSPNAME=('APPC'.*, 'RRS'.*)
SDATA=(PSA,SQA,LSQA,RGN,LPA,TRT,CSA,SWA,SUM,ALLNUC,GRSQ)
```

Figure 8-15 IEADMCxx example

IEASLPxx (SLIP commands)

Use IEASLPxx to contain SLIP commands. The commands can span multiple lines, and the system processes the commands in order.

We recommend that you move any SLIP commands in the COMMNDxx and IEACMDxx parmlib members into a IEASLPxx parmlib member. By using IEASLPxx to contain your SLIP commands, you avoid restrictions found in other parmlib members.

Figure 8-16 shows a sample of what may be contained in SYS1.PARMLIB(IEASLPxx). In this example we are actually suppressing dumps.

```
SLIP SET,C=013,ID=X013,A=NOSVCD,J=JES2,END
SLIP SET,C=213,ID=X213,A=NOSVCD,END
SLIP SET,C=028,ID=X028,A=NOSVCD,END
SLIP SET,C=058,ID=X058,A=NODUMP,DATA=(15R,EQ,4,OR,15R,EQ,8,OR,
15R,EQ,C,OR,15R,EQ,10,OR,15R,EQ,2C,OR,15R,EQ,30,OR,
15R,EQ,3C),END
SLIP SET,C=0E7,ID=X0E7,A=NOSVCD,END
SLIP SET,C=0F3,ID=X0F3,A=NODUMP,END
SLIP SET,C=13E,ID=X13E,A=NODUMP,END
SLIP SET,C=1C5,RE=00090004,ID=X1C5,A=NODUMP,END
SLIP SET,C=222,ID=X222,A=NODUMP,END
SLIP SET,C=322,ID=X322,A=NODUMP,END
SLIP SET,C=33E,ID=X33E,A=NODUMP,END
SLIP SET,C=422,ID=X422,A=NODUMP,END
SLIP SET,C=47B,DATA=(15R,EQ,0,OR,15R,EQ,8),ID=X47B,A=NODUMP,END
SLIP SET,C=622,ID=X622,A=NODUMP,END
```

Figure 8-16 SYS1.PARMLIB(IEASLPxx)

Figure 8-17 on page 215 shows a much more complex SLIP that will capture dumps in multiple MVS images, when a certain message, IXC521I, is generated and Register 5 contains some specific data. It will dump the Console address space, the MSOPS address space, and also the XCFAS address space.

```
SLIP SET,MSGID=IXC521I,  
DATA=(5R?+0,EQ,C8C1E240,+4,EQ,D9C5C1C3),  
ACTION=SVCD,JOBLIST=(CONSOLE,XCFAS),  
DSPNAME=('XCFAS'.IXCDSL01),  
REMOTE=(SYSLIST=(SC55,SC66),  
JOBLIST=(CONSOLE,MSOPS,XCFAS),DSPNAME=('XCFAS'.IXCDSL01)),  
SDATA=(NUC,CSA,GRSQ,LPA,LSQA,PSA,RGN,SQA,SWA,TRT),  
MATCHLIM=3,ID=RON1,END
```

Figure 8-17 SLIP example with increased complexity

The SLIP is activated by issuing the SET SLIP=xx MVS command, where xx is the IEASLPxx parmlib member you want to activate.



SDSF and RMF

SDSF System Display and Search Facility

SDSF provides you with information to monitor, manage, and control your z/OS MVS/JES2 system. It can help you run your business and save you time and money.

SDSF panels provide current information about jobs, output, devices (including printers, punches, initiators, lines, spool offloaders, and spool volumes) and system resources, including nodes and WLM enclaves, anywhere in the JES2 Multi-Access Spool (MAS).

With SDSF panels, there is no need to learn or remember complex command syntax. SDSF's action characters, fields that can be overtyped, action bar, pull-downs, and pop-up windows allow you to select available functions.

RMF Resource Measurement Facility

RMF is designed to ease the management of single or multiple system workload and to enable faster reaction to system delays. Detecting a possible bottleneck early means that corrective actions can be taken earlier. System delays are avoided or at least remedied at an early stage.

System programmers are supported by several reports that ease their work, helping them to tune their system optimally. Consequently, this leads to fewer workload problems and, most important, increases system and operator productivity, a fact that makes the company as a whole more effective at less cost.

9.1 System Display and Search Facility (SDSF)

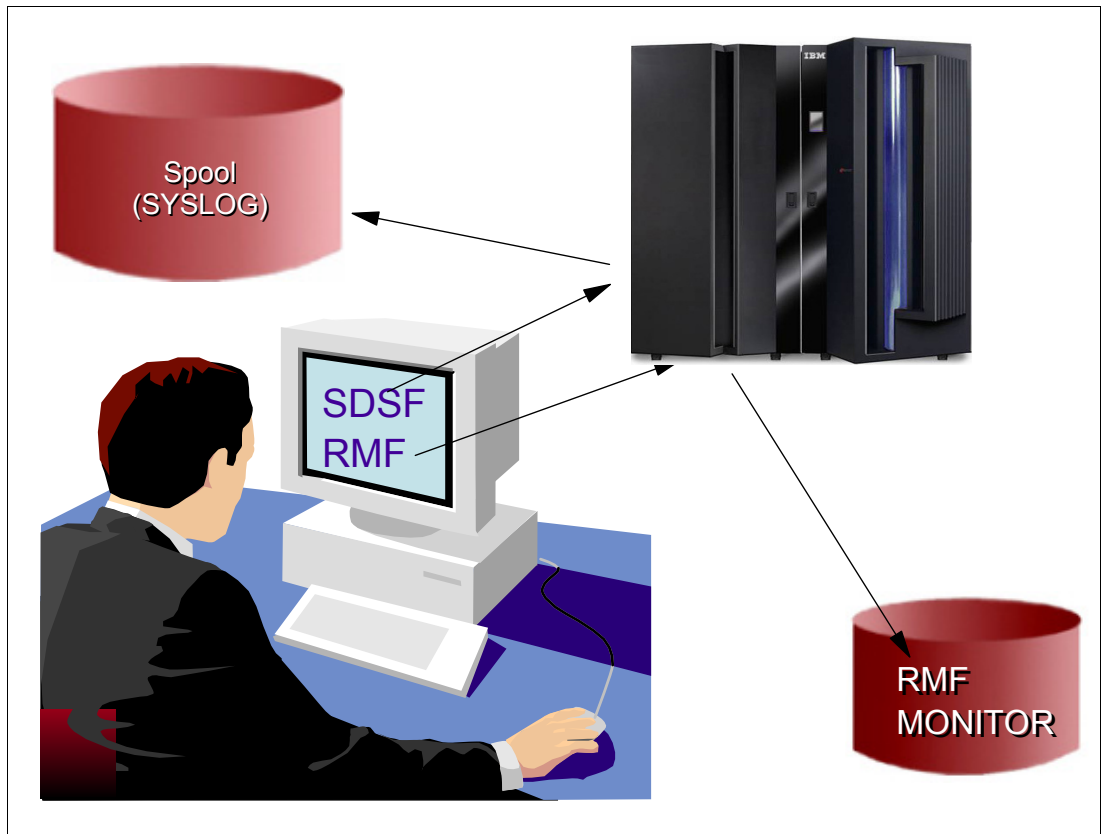


Figure 9-1 SDSF and RMF

System Display and Search Facility (SDSF)

SDSF gives you an easy and efficient way to monitor, manage, and control the key aspects of your MVS/JES2 system. Using SDSF, you can:

- ▶ Control job processing (hold, release, cancel, and purge jobs)
- ▶ Control output, and browse jobs without printing
- ▶ Control devices such as printers, lines, and initiators across the MAS
- ▶ Browse the syslog
- ▶ Manage system resources, such as members of the MAS, job classes, and WLM enclaves

With SDSF panels, there is no need to learn or remember complex command syntax. The SDSF action characters, overtypable fields, action bar, pull-downs, and pop-up windows allow you to select available functions. The SDSF primary option menu is shown in Figure 9-2 on page 219.

SDSF provides an easy way to manage work productively, as follows:

- ▶ Control jobs
- ▶ Control output
- ▶ Control devices
- ▶ Manage system resources

To become familiar with the panel handling and the output, select a function. If the RACF administration has been done correctly you should not be able to delete or destroy

processes. The following shows the active users on a system. To get this output select **DA**, as shown in Figure 9-3 on page 220

```
Display Filter View Print Options Help
-----
HGX7720 ----- SDSF PRIMARY OPTION MENU -----
COMMAND INPUT ==>                                SCROLL ==> PAGE

DA   Active users                                INIT  Initiators
I    Input queue                                PR    Printers
O    Output queue                                PUN    Punches
H    Held output queue                          RDR    Readers
ST   Status of jobs                            LINE   Lines
                                           NODE   Nodes
LOG   System log                                SO     Spool offload
SR    System requests                          SP     Spool volumes
MAS   Members in the MAS
JC    Job classes                                RM     Resource monitor
SE    Scheduling environments                  CK     Health checker
RES   WLM resources
ENC   Enclaves                                ULOG   User session log
PS    Processes

END    Exit SDSF

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1981, 2005. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

Figure 9-2 SDSF Primary Option Menu

Display Filter View Print Options Help											

SDSF DA SC69 SC69				PAG	0	CPU/L	4/ 4	LINE 1-26 (70)			
COMMAND INPUT ==>								SCROLL ==> PAGE			
PREFIX=* DEST=(ALL) OWNER=* SYSNAME=											
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO
	MASTER			STC14926	+MASTER+	NS	FF	6339	0.00	0.00	
	PCAUTH	PCAUTH				NS	FF	152	0.00	0.00	
	RASP	RASP				NS	FF	531	0.00	0.00	
	TRACE	TRACE				NS	FF	594	0.00	0.00	
	DUMPSRV	DUMPSRV	DUMPSRV			NS	FF	409	0.00	0.00	
	XCFAS	XCFAS	IEFPROC			NS	FF	21T	0.00	0.00	
	GRS	GRS				NS	FF	13T	0.00	0.00	
	SMSPDSE	SMSPDSE				NS	FF	58T	0.00	0.00	
	SMSVSAM	SMSVSAM	IEFPROC			NS	FF	11T	0.00	0.00	
	CONSOLE	CONSOLE				NS	FF	7931	0.00	0.00	
	WLM	WLM	IEFPROC			NS	FF	4641	0.00	0.00	
	ANTMAIN	ANTMAIN	IEFPROC			NS	FF	1298	0.00	0.00	
	ANTAS000	ANTAS000	IEFPROC			NS	FE	1137	0.00	0.00	
	DEVMAN	DEVMAN	IEFPROC			NS	FF	150	0.00	0.00	
	OMVS	OMVS	OMVS			NS	FF	352T	0.00	0.00	

Figure 9-3 Display active address spaces

9.2 Using the SYSLOG for debugging

- ❑ **SYSLOG data set on JES spool**
 - View SYSLOG using SDSF with JES2
 - View with (E)JES with JES3
 - Can be queued for printing
- ❑ **Viewing messages in a DUMP**
 - Messages appear in the master trace
- ❑ **Using SYSLOG for debugging**
 - Check messages in the SYSLOG for ABENDs

Figure 9-4 Using the SYSLOG for problem analysis

SYSLOG

The SYSLOG is a SYSOUT data set provided by the job entry subsystem (JES2 or JES3). SYSOUT data sets are output spool data sets on direct access storage devices (DASD). Print the SYSLOG periodically to check for problems. The SYSLOG consists of the following:

- ▶ All messages issued through WTL macros
- ▶ All messages entered by LOG operator commands
- ▶ Usually, the hard-copy logs
- ▶ Any messages routed to the SYSLOG from any system component or program

It can be used by application and system programmers to record communications about problem programs and system functions. The operator can use the LOG command to add an entry to the system log.

SYSLOG is queued for printing when the number of messages recorded reaches a threshold specified at system initialization. The operator can force the system log data set to be queued for printing before the threshold is reached by issuing the WRITELOG command. The SYSLOG can be viewed using SDSF with JES2 and with (E)JES with JES3 systems.

The SYSLOG, often referred to as the hard-copy log, is a record of all system message traffic, as follows:

- ▶ Messages to and from all consoles

- Commands and replies that are entered by the operator

In a dump, these messages appear in the master trace. With JES3, the hard-copy log is always written to the SYSLOG. With JES2, the hard-copy log is usually written to the SYSLOG but can be written to a console printer, if the installation chooses.

SYSLOG for debugging

To check for messages and abend information, have a look at the SYSLOG. To view the system log, enter `log` on the command line. Figure 9-5 shows an example starting with the time stamp and followed by the messages. In the complete output you will get more information. Data that would normally be seen to the left of the time stamp has been removed for presentation.

```
D U,,ALLOC,8052,1
IEE106I 17.04.19 UNITS ALLOCATED 881
UNIT      JOBNAME  ASID      JOBNAME  ASID      JOBNAME  ASID      JOBNAME  ASID
8052      *MASTER* 0000      *MASTER* 0001      DUMPSRV  0005      XCFAS    0006
8052      LLA      0018      JES2      001A      VTAM44   001B      NFSCLNT7 001D
8052      RMF      0024      ZFS       0028      APPC     002A      DFRMM    002E
8052      OPTSO    0030      RACF      0031      OSASF    004D      HAIMO    0050
8052      RMFGAT   0055      NFSMVS7   0058
D GRS,C
ISG343I 17.04.38 GRS STATUS 883
NO ENQ RESOURCE CONTENTION EXISTS
NO REQUESTS PENDING FOR ISGLOCK STRUCTURE
NO LATCH CONTENTION EXISTS
IEF126I HAIMO - LOGGED OFF - TIME=17.13.48 - ASID=0050 - SC69
$HASP395 HAIMO      ENDED
$HASP100 HAIMO      ON TSOINRDR
$HASP373 HAIMO      STARTED
IEF125I HAIMO - LOGGED ON - TIME=17.13.51 - ASID=0050 - SC69
IEF126I HAIMO - LOGGED OFF - TIME=17.14.05 - ASID=0050 - SC69
$HASP395 HAIMO      ENDED
IEA631I OPERATOR BOBH      NOW INACTIVE, SYSTEM=SC69      , LU=SC38TCC6
```

Figure 9-5 Sample SYSLOG output

9.3 RMF Resource Measurement Facility

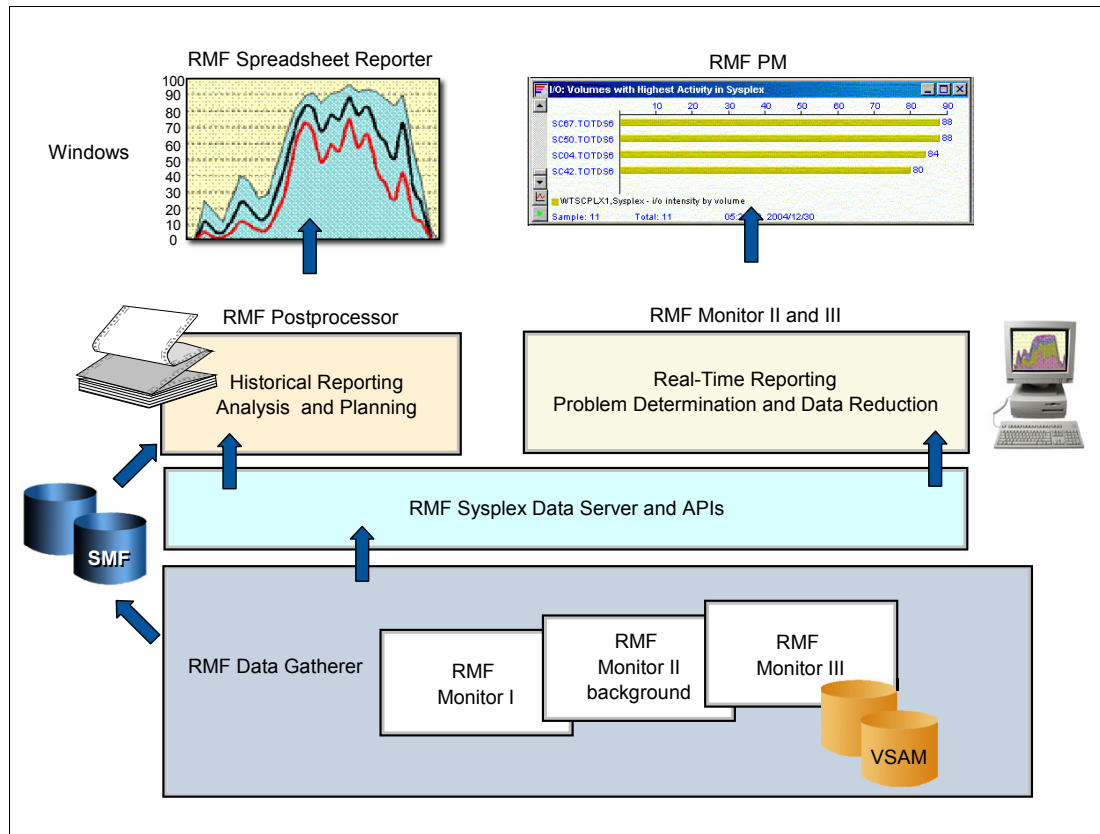


Figure 9-6 RMF data gatherer

RMF

Resource Measurement Facility (RMF) is shipped with every release of z/OS at the current level of support. It is integration tested with z/OS and includes the enhancements available with every new release. It's easier than ever to install RMF. RMF is an optional feature of z/OS.

RMF is designed to ease the management of single or multiple system workloads and to enable faster reaction to system delays. Detecting a possible bottleneck early means that corrective actions can be taken earlier. System delays are avoided or at least remedied at an early stage.

System programmers are supported by several reports that ease their work, helping them to tune their system optimally. Consequently, this leads to fewer workload problems and, most important, increases system and operator productivity, a fact that makes the company as a whole more effective at less cost.

RMF issues reports about performance problems as they occur, so that your installation can take action before the problems become critical. Your installation can use RMF to:

- ▶ Determine that your system is running smoothly
- ▶ Detect system bottlenecks caused by contention for resources
- ▶ Evaluate the service your installation provides to different groups of users
- ▶ Identify the workload delayed and the reason for the delay

- Monitor system failures, system stalls, and failures of selected applications

RMF monitors

RMF comes with three monitors, Monitor I, II and III. Monitor III with its ability to determine the “cause of delay” is where we start:

Monitor III provides short-term data collection and online reports for continuous monitoring of system status and solving performance problems.

Monitor III is a good place to begin system tuning. It allows the system tuner to distinguish between delays for important jobs and delays for jobs that are not as important to overall system performance.

Monitor I provides long-term data collection for system workload and resource utilization. The Monitor I session is continuous, and measures various areas of system activity over a long period of time. You can get Monitor I reports directly as real-time reports for each completed interval (single-system reports only), or you run the postprocessor to create the reports, either as single-system or as sysplex reports. Many installations produce daily reports of RMF data for ongoing performance management.

Monitor II provides online measurements on demand for use in solving immediate problems. A Monitor II session can be regarded as a snapshot session. Unlike the continuous Monitor I session, a Monitor II session generates a requested report from a single data sample. Since Monitor II is an ISPF application, you might use Monitor II and Monitor III simultaneously in split-screen mode to get different views of the performance of your system. In addition, you can use the Spreadsheet Reporter for further processing the measurement data on a workstation by help of spreadsheet applications. The following sections provide sample reports including the name of the corresponding macro.

Find a detailed description on how to create the reports and records and on how to use the macros in *RMF User's Guide*, SC28-1949.

9.4 RMF Monitor I data gathering

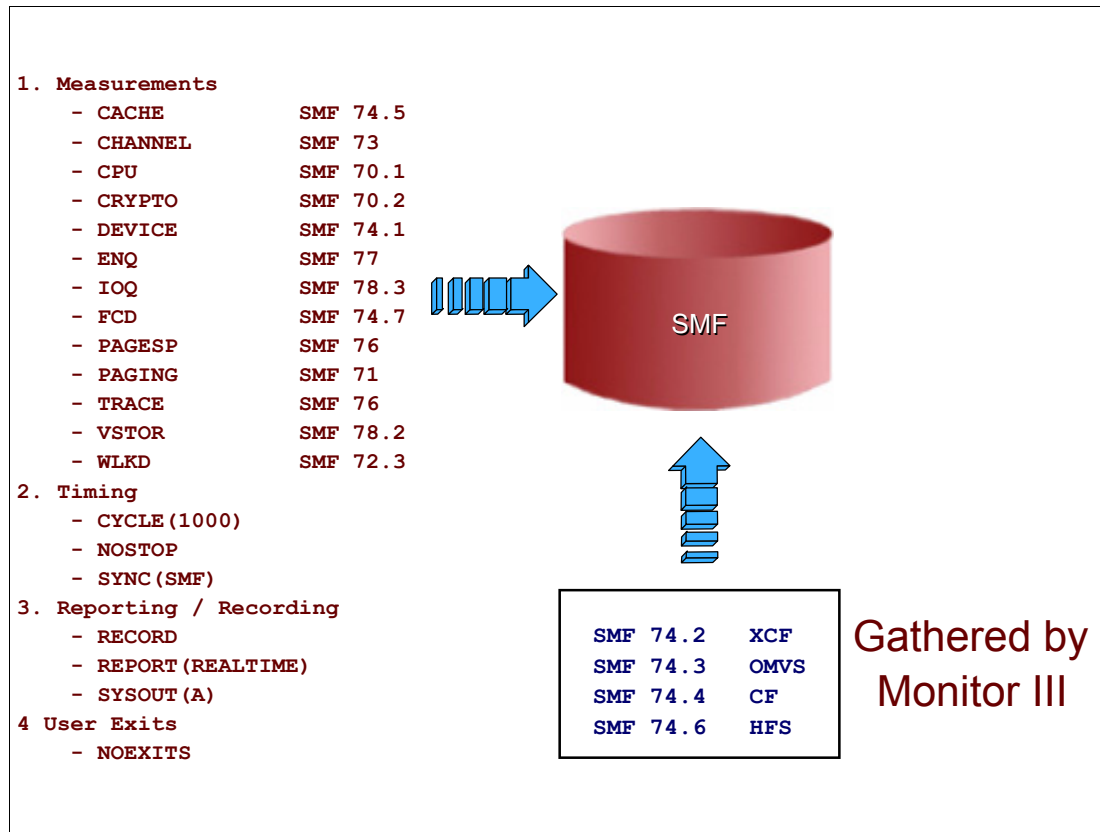


Figure 9-7 RMF Monitor output

Monitor I

Monitor I measures and reports the use of system resources (that is, the processor, I/O devices, storage, and data sets on which a job can enqueue during its execution). It runs in the background and measures data over a time period. Reports can be printed immediately after the end of the measurement interval, or the data can be stored in SMF records and printed later with the RMF postprocessor, which can be used to generate reports for exceptions, that is, conditions where user-specified values are exceeded. SMF data is kept in VSAM data sets as the postprocessor requires a sequential format. Use the SMF dump utility, IFASMFDP, to unload the data. Usually Generation Data Groups (GDGs) are the preferred target:

```
RMF.SMFDATA.SYSNAME(0)
```

IFASMFDP to unload JCL

Figure 9-8 on page 226 shows the JCL to unload the SMF data.

```
//SMFDUMP EXEC PGM=IFASMFDP
//IDD1 DD DISP=SHR,DSN=<input_smfdata_system1>
//IDD2 DD DISP=SHR,DSN=<input_smfdata_system2>
//SMFDATA DD DISP=(NEW,PASS),SPACE=(CYL,(10,10),RLSE),
// UNIT=SYSDA,DCB=(RECFM=VBS,LRECL=32760,BLKSIZE=0)
//SYSIN DD *
INDD(IDD1,OPTIONS(DUMP))
INDD(IDD2,OPTIONS(DUMP))
OUTDD(SMFDATA,TYPE(70:78))
```

Figure 9-8 IFASMFDP to unload JCL

Sort JCL job

To get an output sorted by date and time, the following sort job is required for sysplex-wide reporting.

```
//RMFSORT EXEC PGM=SORT
//SORTIN DD DISP=SHR,DSN=<input_smfdata_system1>
// DD DISP=SHR,DSN=<input_smfdata_system2>
//SYSIN DD *
SORT FIELDS=(11,4,CH,A,7,4,CH,A),EQUALS
MODS E15=(ERBPPE15,36000,,N),E35=(ERBPPE35,3000,,N)
```

Figure 9-9 Sort JCL

9.5 Monitor II data gathering

- ☐ **Monitor II is a snapshot reporter**
 - Collects the status of system resources (CPU, devices,..)
 - Collects the status of address spaces (resource usage,..)
- ☐ **Use Monitor II to:**
 - Continuously monitor resource usage
 - Determine the state of any address space in the system
 - Track CPU usage of problem address spaces
 - Collect supplemental information when analyzing performance problems with Monitor III
- ☐ **Choose background session to:**
 - Collect SMFA records for archiving and later processing
 - Automate snapshot reporting
- ☐ **Choose display session for:**
 - Immediate feedback
 - Online analysis

Figure 9-10 RMF Monitor II data gathering

Monitor II

Monitor II tells you what is happening right now on your system, how system resources are used, and how your address spaces are doing. Several standard reports are provided, and you can add your own reports. You cannot see older or historical data. You can only see what is happening right now on your system, or current data.

You can collect data to SMF data sets continuously for Monitor II reports. In this case, you decide beforehand which reports you will produce by specifying them to the Monitor II data gatherer. Later, you can write the reports using the postprocessor for the period you want to see. This is a useful method, for example, if you want to get information every third second about certain address spaces for one day or perhaps every day.

Starting Monitor II

To become familiar with RMF, start RMF Monitor II by issuing the TSO RMFMON command. Figure 9-11 shows the RMF display menu. Monitor II is a snapshot reporting tool for very fast information about how specific address spaces or system resources (processor, DASD volumes, storage) are performing. Monitor II has two modes for reporting on the performance of your system.

RMF DISPLAY MENU		
NAME	PFK#	DESCRIPTION
ARD	1	ADDRESS SPACE RESOURCE DATA
ASD	2	ADDRESS SPACE STATE DATA
ASRM	3	ADDRESS SPACE SRM DATA
CHANNEL	4	CHANNEL PATH ACTIVITY
DDMN	5	-----NOT APPLICABLE IN GOAL MODE-----
DEV	6	DEVICE ACTIVITY
PGSP	7	PAGE/SWAP DATA SET ACTIVITY
SENQ	8	SYSTEM ENQUEUE CONTENTION
SENQR	9	SYSTEM ENQUEUE RESERVE
SPAG	10	PAGING ACTIVITY
SRCS	11	CENTRAL STORAGE / PROCESSOR / SRM
TRX	12	-----NOT APPLICABLE IN GOAL MODE-----
ARDJ		ADDRESS SPACE RESOURCE DATA
ASDJ		ADDRESS SPACE STATE DATA
ASRMJ		ADDRESS SPACE SRM DATA
DEVV		DEVICE ACTIVITY
IOQUEUE		I/O QUEUING ACTIVITY
SDS		RMF SYSPLEX DATA SERVER
LLI		PROGRAM LIBRARY INFORMATION
ILOCK		IRLM LONG LOCK DETECTION

Figure 9-11 RMF display menu

ARD report

In the ARD report, the number of data lines in the report depends on the number of address space identifiers in the system that meet your selection criteria. The shown report is a sample for a system running in z/Architecture. Figure 9-12 shows the result of issuing the ARD command, showing data for each ASID. The key information we are looking for is who is consuming the CPU and/or EXCP cycles.

CPU= 2/ 2 UIC=2540 PR= 0 ESA2 ASD													
17:42:30	S	C	R	DP	CS	ESF	CS	TAR	X	PIN	ES	TX	SWAP
JOBNAME	SRVCLASS	P	L	LS	PR	F	TAR	WSS	M	RT	RT	SC	RV
MASTER	SYSTEM	1	NS		FF	4104		0		----		0	0
PCAUTH	SYSSTC	1	NS		FE	103		0	X	----		0	0
RASP	SYSTEM	1	NS		FF	348		0	X	----		0	0
TRACE	SYSSTC	1	NS		FE	985		0	X	----		0	0
DUMPSRV	SYSTEM	1	NS		FF	182		0		----		0	0
XCFAS	SYSTEM	1	NS		FF	1703		0	X	----		0	0
GRS	SYSTEM	1	NS		FF	942		691	X	----		0	0
SMSPDSE	SYSTEM	1	NS		FF	21.4K		9860	X	----		0	0
CONSOLE	SYSTEM	1	NS		FF	1277		0	X	----		0	0
WLM	SYSTEM	1	NS		FF	1092		0	X	----		0	0
ANTMAIN	SYSTEM	1	NS		FF	2203		0	X	----		1	998
ANTAS000	STC	1	NS		FB	203		0	X	----		1	998
OMVS	SYSTEM	1	NS		FF	25.3K		19.8K	X	----		0	0
JESXCF	SYSTEM	1	NS		FF	92		0	X	----		0	998
ALLOCAS	SYSTEM	1	NS		FF	2195		0	X	----		1	0
IOSAS	SYSTEM	1	NS		FF	1173		0	X	----		0	0
IXGLOGR	SYSTEM	1	NS		FF	752		0	X	----		0	0
SMS	SYSSTC	1	NS		FE	1150		0	X	----		0	998
SMF	SYSTEM	1	NS		FF	285		0	X	----		0	0
LLA	SYSSTC	1	NS		FE	1688		0	X	----		0	998
JES2AUX	STC	1	NS		FB	118		0		----		1	0

Figure 9-12 Output of the ARD command

To leave the RMF panel, enter end.

9.6 RMF Monitor III data gathering

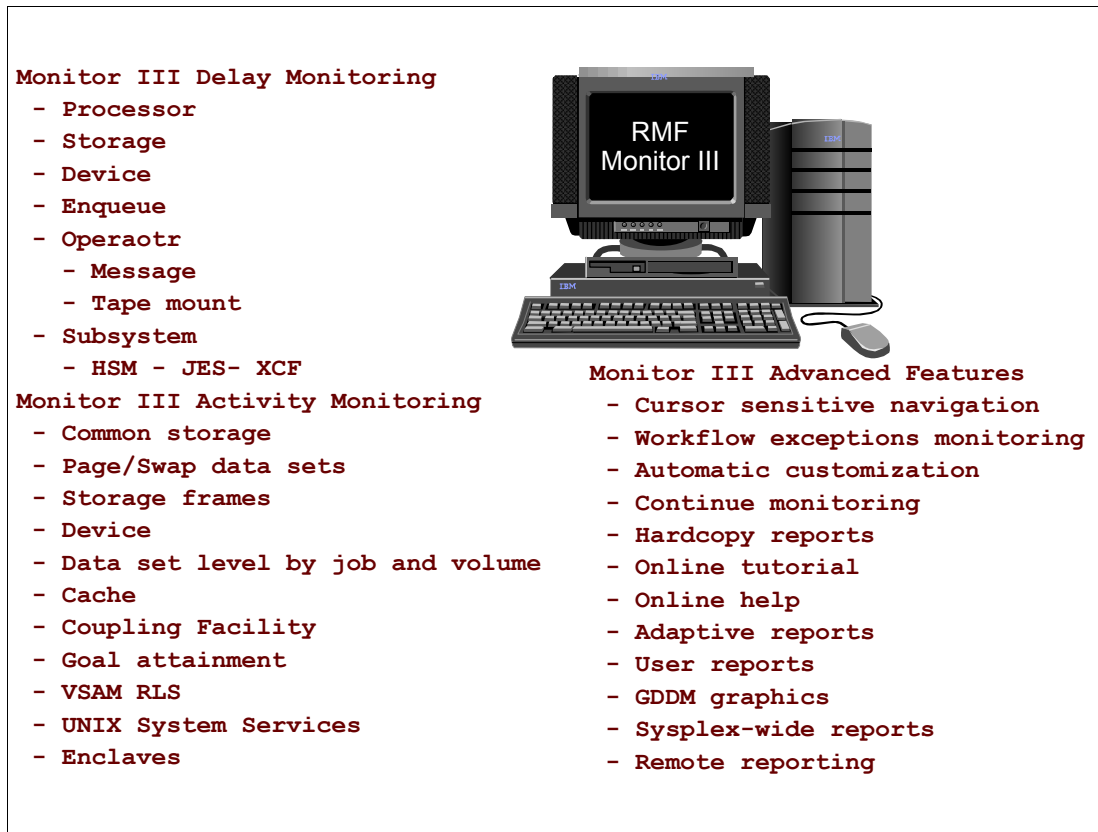


Figure 9-13 RMF Monitor III data gathering

Monitor III

Monitor III tells you how well your single system or sysplex is performing, and what is going on. This is presented at different levels:

- ▶ Sysplex-wide reports about the workloads, Coupling Facilities, and caching
- ▶ System-wide reports about the resources and address spaces

You can see what is happening right now, typically during the last 60 seconds. You can also see what happened recently or you might be able to see what happened the day before yesterday depending on your installation setup. Additionally, you can dynamically change the time frame you want to observe. For example, your actions might be:

- ▶ Using 10-minute time frames on one day, travelling backward and forward, to find the most interesting 10-minute period.
- ▶ Using one-minute time frames, travelling backward and forward, to find the most interesting one-minute period.

At that point, it should be easy to locate the system, partition, address space, device, or whatever it is that you want to examine.



z/Architecture and addressing

z/Architecture is the next step in the evolution from System/360™ to System/370™, System/370 Extended Architecture (370-XA), Enterprise Systems Architecture/370* (ESA/370), and Enterprise Systems Architecture/390® (ESA/390). In order to understand z/Architecture you have to be familiar also with the basics of ESA/390 and its predecessors.

An address space maps all of the available addresses, and includes system code and data as well as user code and data. Thus, not all of the mapped addresses are available for user code and data. This limit on user applications was a major reason for System/370 Extended Architecture (370-XA) and MVS/XA™. Because the effective length of an address field expanded from 24 bits to 31 bits, the size of an address space expanded from 16 megabytes to 2 gigabytes. An MVS/XA address space is 128 times as big as an MVS/370 address space.

This chapter describes:

- ▶ Program status word (PSW)
- ▶ Address space addressability
- ▶ Dumps in 31-bit and 64-bit modes

10.1 Program status word (PSW)

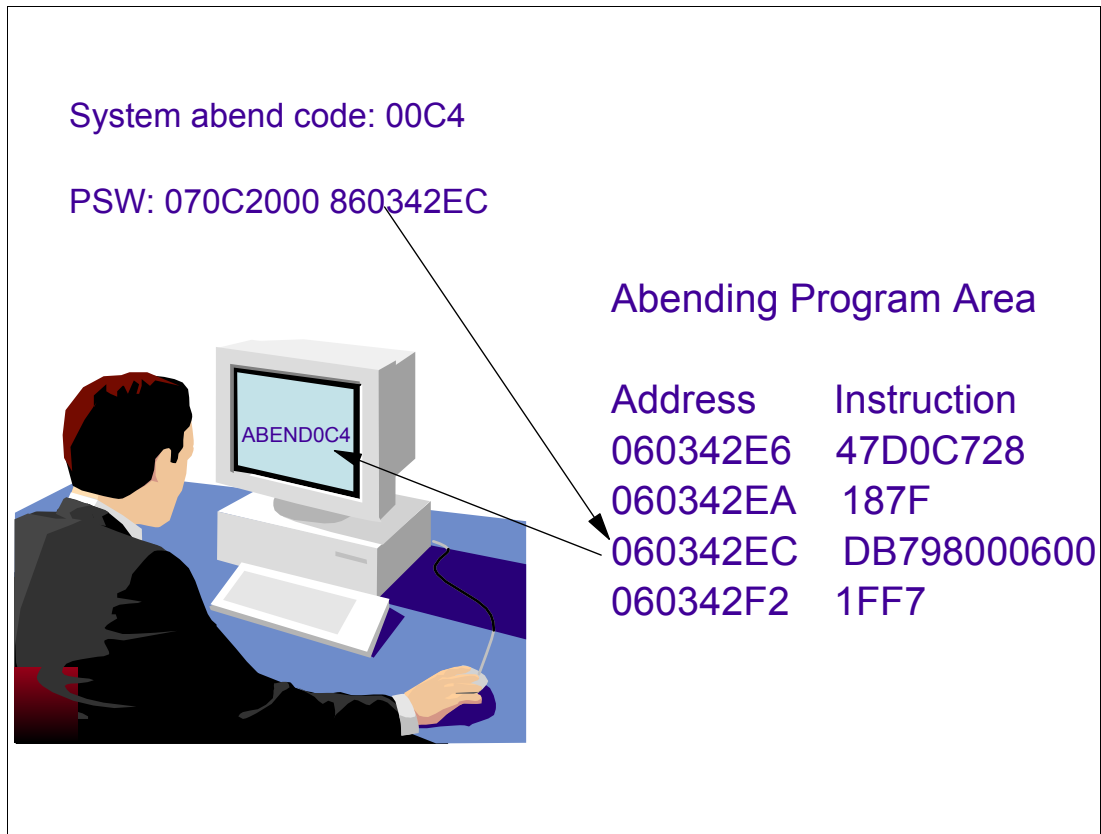


Figure 10-1 Program status word (PSW)

Program status word

One very important piece of information that will be crucial to your ability to diagnose a problem on z/OS is the program status word, more commonly referred to as the PSW. The PSW includes the instruction address, condition code, and other information to control instruction sequencing and to determine the state of the CPU. The active or controlling PSW is called the current PSW.

The PSW is so important because it keeps track of the progress of the system and the executing program. The current PSW usually points to the address of the next instruction to be executed. In some specific cases the PSW will point to the address of the failing instruction and this occurs when the interrupt code is 0010, which is a segment translation exception, or interrupt code 0011, which is a page translation exception.

What this means is that when a task abends and a dump is taken, the PSW is pointing to the next instruction that will be executed in the failing program. By subtracting the instruction-length code (ILC) from the PSW address, we will be looking at the failing instruction for which the abend was triggered.

Note: For page translation and segment translation errors, the PSW points to the failing instruction.

10.2 Program-status word (PSW)

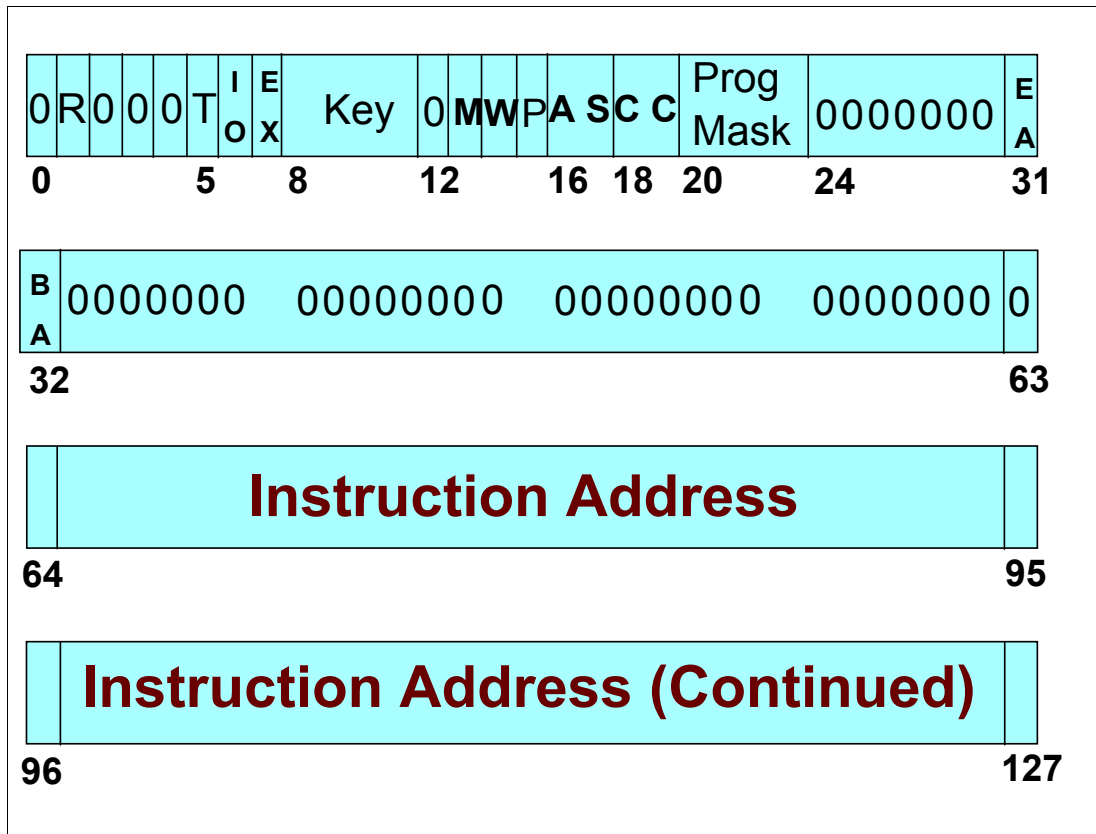


Figure 10-2 Program Status Word (PSW)

Current PSW

The current PSW is a storage circuit located within the CP. It contains information required for the execution of the currently active program, or, in other words, it contains the current state of a CP. It has 16 bytes (128 bits). The PSW includes the instruction address, condition code, and other information used to control instruction sequencing and to determine the state of the CP. The active or controlling PSW is called the current PSW. It governs the program currently being executed. Figure 10-2 describes the PSW from bits 0 to 127.

PER mask - R (bit 1)

Bit 1 controls whether the CP is enabled for interrupts associated with program-event recording (PER). When the bit is zero, no PER event can cause an interruption. When the bit is one, interruptions are permitted, subject to the PER-event-mask bits in control register 9.

DAT mode - T (bit 5)

Bit 5 controls whether implicit dynamic address translation of logical and instruction addresses used to access storage takes place. When the bit is zero, DAT is off, and logical and instruction addresses are treated as real addresses. When the bit is one, DAT is on, and the dynamic-address-translation mechanism is invoked.

I/O mask - IO (bit 6)

Bit 6 controls whether the CP is enabled for I/O interruptions. When the bit is zero, an I/O interruption cannot occur. When the bit is one, I/O interruptions are subject to the

I/O-interruption subclass-mask bits in control register 6. When an I/O-interruption subclass-mask bit is zero, an I/O interruption for that I/O-interruption subclass cannot occur; when the I/O-interruption subclass-mask bit is one, an I/O interruption for that I/O-interruption subclass can occur.

External mask - EX (bit 7)

Bit 7 controls whether the CP is enabled for interruption by conditions included in the external class. When the bit is zero, an external interruption cannot occur. When the bit is one, an external interruption is subject to the corresponding external subclass-mask bits in control register 0; when the subclass-mask bit is zero, conditions associated with the subclass cannot cause an interruption; when the subclass-mask bit is one, an interruption in that subclass can occur.

PSW key (bits 8-11)

Bits 8-11 form the access key for storage references by the CP. If the reference is subject to key-controlled protection, the PSW key is matched with a storage key when information is stored or when information is fetched from a location that is protected against fetching. However, for one of the operands of each of MOVE TO PRIMARY, MOVE TO SECONDARY, MOVE WITH KEY, MOVE WITH SOURCE KEY, and MOVE WITH DESTINATION KEY, an access key specified as an operand is used instead of the PSW key.

Machine-check mask - M (bit 13)

Bit 13 controls whether the CP is enabled for interruption by machine-check conditions. When the bit is zero, a machine-check interruption cannot occur. When the bit is one, machine-check interruptions due to system damage and instruction-processing damage are permitted, but interruptions due to other machine-check-subclass conditions are subject to the subclass-mask bits in control register 14.

Wait state - W (bit 14)

When bit 14 is one, the CP is waiting; that is, no instructions are processed by the CP, but interruptions may take place. When bit 14 is zero, instruction fetching and execution occur in the normal manner. The wait indicator is on when the bit is one. When in a wait state, the only way of getting out of that state is through an interruption, or IPL (a z/OS boot). Certain bits in the current PSW, when off, place the CP in a disabled state, that is, it does not accept interrupts. So, when z/OS because of any error reason (software or hardware) decides to stop a CP, it sets the PSW in a disabled and wait state, forcing an IPL as the way to get the CP back in a running state.

Problem state - P (bit 15)

When bit 15 is one, the CP is in the problem state. When bit 15 is zero, the CP is in the supervisor state. In the supervisor state, all instructions are valid. In the problem state, only those instructions are valid that provide meaningful information to the problem program and that cannot affect system integrity; such instructions are called unprivileged instructions. The instructions that are never valid in the problem state are called privileged instructions. When a CP in the problem state attempts to execute a privileged instruction, a privileged-operation exception is recognized. Another group of instructions, called semiprivileged instructions, are executed by a CP in the problem state only if specific authority tests are met; otherwise, a privileged-operation exception or a special-operation exception is recognized.

Address-space control -AS (bits 16-17)

Bits 16 and 17, in conjunction with PSW bit 5, control the translation mode.

Condition code - CC (bits 18-19)

Bits 18 and 19 are the two bits of the condition code. The condition code is set to 0, 1, 2, or 3, depending on the result obtained in executing certain instructions. Most arithmetic and logical operations, as well as some other operations, set the condition code. The instruction **BRANCH ON CONDITION** can specify any selection of the condition code values as a criterion for branching.

The part of the CP that executes instructions is called the arithmetic logic unit (ALU). The ALU has four internal bits that are set by certain instructions. At the end of such instructions this 4-bit configuration is mapped into bits 18 and 19 of the current PSW.

As an example, the instruction **COMPARE** establishes a comparison between two operands. The result of the comparison is placed in the CC of the current PSW, as follows:

- ▶ CC=00, the operands are equal
- ▶ CC=01, the first operand is lower
- ▶ CC=10, the first operand is greater

To test the contents of a CC (set by a previous instruction), use the **BRANCH ON CONDITION (BC)** instruction. It has an address of another instruction (branch address) to be executed depending on the comparison of the CC and a mask M. The instruction address in the current PSW is replaced by the branch address, if the condition code has one of the values specified by M; otherwise a normal instruction sequencing proceeds with the normal updated instruction address. Here are the types of codes:

- ▶ Condition code - bits 18 and 19 of the PSW
- ▶ Return code - a code associated with how a program ended
- ▶ Completion code - a code associated with how a task ended
- ▶ Reason code - a code passed in GPR 15, giving more details about how a task ended

Program Mask (bits 20-23)

During the execution of an arithmetic instruction, the CP may find some unusual (or error) condition, such as: overflows, loss of significance, underflow. In these cases, the CP generates a program interrupt. When this interrupt is treated by z/OS, usually the current task is abnormally ended (abend). However, in certain situations the programmer does not want an abend, so through the instruction **SET PROGRAM MASK (SPM)**, he or she can mask such interrupts by setting to off some of the program mask bits. Each bit is associated with one type of condition:

- ▶ Fixed point overflow (bit 20)
- ▶ Decimal overflow (bit 21)
- ▶ Exponent underflow (bit 22)
- ▶ Significance (bit 23)

Observe that the active program is informed about the above events through the condition code posted by the instruction where the events described happened.

The contents of the CP can be totally changed by two events:

- ▶ Loading a new PSW from storage along an interruption
- ▶ Executing the instruction **LPSW**, which copies 128 bits from memory to the current PSW.

Extended addressing mode - EA, BA (bits 31-32)

The combination of bits 31 and 32 specify the addressing mode (24, 31, or 64) of the running program. Bit 31 controls the size of effective addresses and effective address generation in conjunction with bit 32, the basic addressing mode bit. When bit 31 is zero, the addressing mode is controlled by bit 32. When bits 31 and 32 are both one, 64-bit addressing is specified.

10.3 64-bit addressing

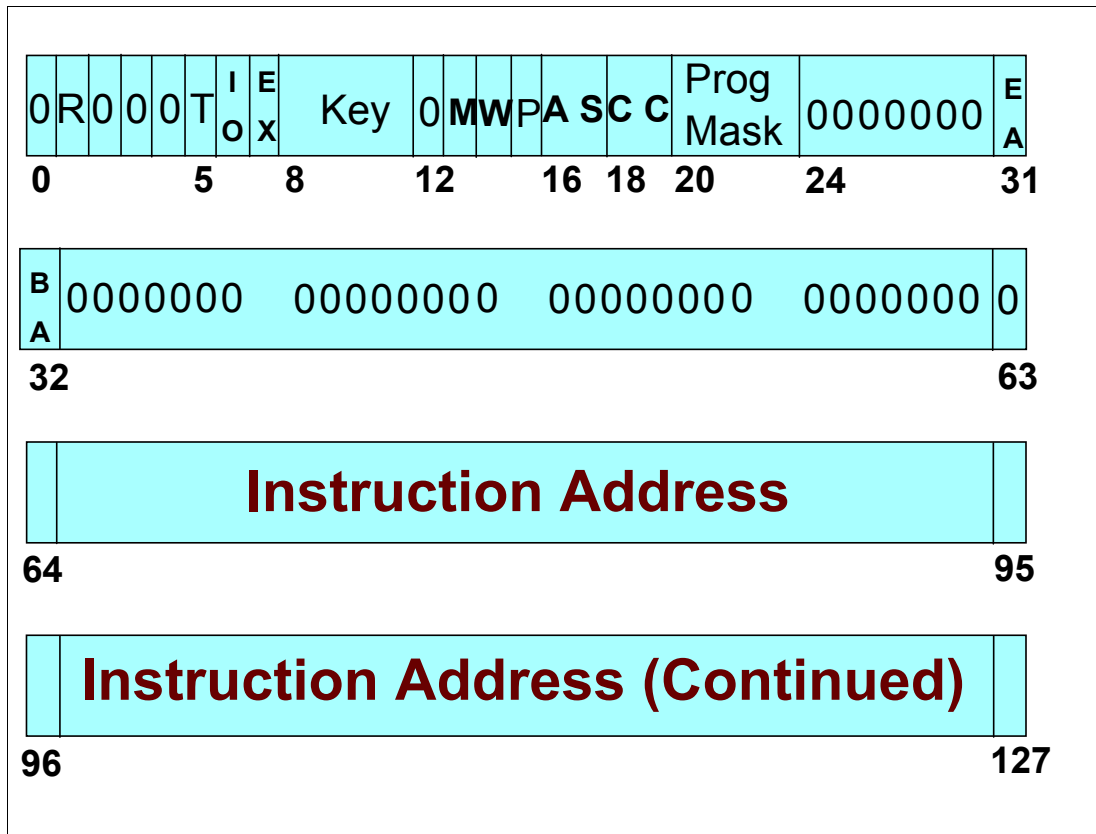


Figure 10-3 64-bit addressing

What is addressability

One of the major developments of the MVS operating system was the implementation of 31-bit addressing. Prior to MVS/XA the highest virtual storage location that could be addressed was 16 megabytes, or hexadecimal FFFFFFFF. Actually, it was one byte less than 16 megabytes, because we start at zero. As applications grew larger the 24-bit architecture limitations were recognized, and 31-bit addressability was introduced. The 31-bit standard increased the amount of addressable virtual storage to 2 gigabytes. The addressing mode of a program is determined by the high order bit (bit 32 of the PSW) of the instruction address. If this bit is set to 1 the processor is running in 31-bit mode. If it is 0 then the processor is running in 24-bit mode.

We have now taken the next step in storage addressability with z/OS, implementing 64-bit addressing. This means that the maximum storage that can be addressed is 2^{64} , or 16 exabytes. The highest address when running in 64-bit mode is X'FFFFFFFF_FFFFFFFF' as opposed to the previous 31-bit high address of X'7FFFFFFF'.

Format of the PSW

Prior to z/OS and 64-bit mode operations, the PSW was 64 bits in length and comprised of two 32-bit words. The first 32 bits (identified as bits 0 through 31) related to system state and mode status, but the second 32 bits (identified as bits 32 through 63 as shown in Figure 10-4 on page 238) indicated the addressing mode in the first bit and the address of the next instruction in bits 33 through 63. The second word is what will interest us in most cases, as shown in Figure 10-3.

For example,

PSW: 075C2000 82CC5BCC Instruction length: 02

Instruction address (bits 64 to 127)

Bits 64 to 127, shown in Figure 10-3 on page 236, point to the storage address of the next instruction to be executed by this CP. When an instruction is fetched from main storage, its length is automatically added to this field. It then points to the next instruction address. However, there are instructions such as a BRANCH that may replace the contents of this field, pointing to the branched instruction. The address contained in this PSW field may have 24, 31, or 64 bits, depending on the addressing mode attribute of the executing program. For compatibility reasons, old programs that use small addresses are still allowed to execute. When in 24- or 31-bit addressing mode, the leftmost bits of this field are filled with zeroes.

CP interrupts

The CP has an interrupt capability, which permits it to switch rapidly to another program in response to exceptional conditions and external stimuli. When an interrupt occurs, the CP places the current PSW in an assigned storage location, called the old-PSW location, for the particular class of interrupt. The CP fetches a new PSW from a second assigned storage location. This new PSW determines the next program to be executed. When it has finished processing the interrupt, the program handling the interrupt may reload the old PSW, making it again the current PSW, so that the interrupted program can continue.

There are six classes of interrupt: external, I/O, machine check, program, restart, and supervisor call. Each class has a distinct pair of old-PSW and new-PSW locations permanently assigned in real storage.

10.4 Next sequential instruction

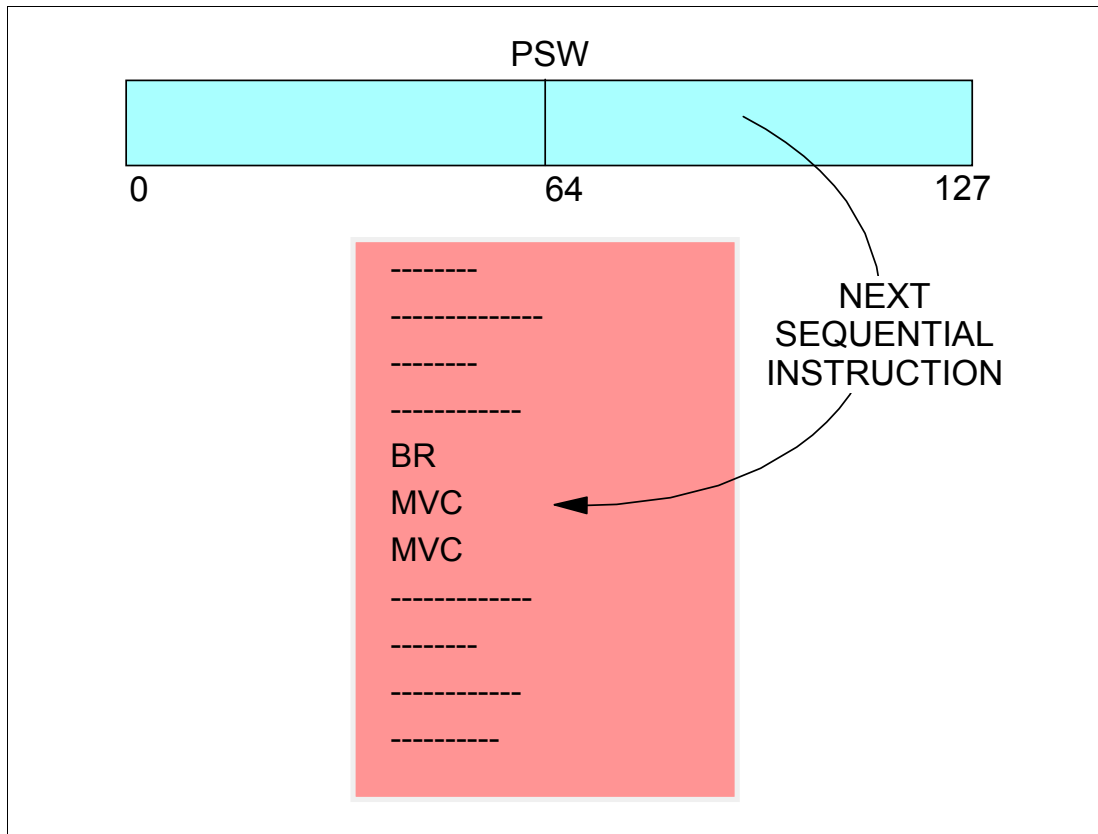


Figure 10-4 Next sequential instruction address

PSW second word

Using the PSW, example:

PSW: 075C2000 82CC5BCC Instruction length: 02

The second word of the PSW is 82CC5BCC. The first number, 8, indicates that this program is executing in 31-bit mode. In other words, this program runs above the 16-megabyte line. The number 8 in binary is 1000, which indicates the addressing mode bit 32 is ON. A value of zero decimal would be binary zero, 0000, indicating that the addressing mode bit 32 is OFF, which identifies that this location was below the 16-bit line, or in 24-bit mode.

The remaining data points to the next instruction to be executed. In this case, 2CC5BCC. For the sake of correctness the full address would be 02CC5BCC.

Subtracting the instruction length value, in this case, 2, from the PSW address, would result in 02CC5BCA, which would point to the failing instruction.

The PSW has now changed and the z/OS 128-bit PSW is converted by MVS to a 64-bit double word and the z/OS-formatted PSW is stored in control blocks. The PSW is represented as follows:

```

AMODE 24
07850000 00000000 00000000 00065788 078D0000 00065788
AMODE 31
04041000 80000000 00000000 00FE5768 040C1000 80FE5768
  
```

```
AMODE 64  
04045001 80000000 00000000 01685B28 040C5001_81685B28
```

The bold form of the PSW indicates the “converted” z/OS PSW. The underscore between the two words of the converted PSW indicates that this is a 64-bit (above the bar) address.

As you can see, it looks similar to the 31-bit PSW except for the non-zero value of bit 31 in the 1st word of the PSW, 040C500**1**, as well as the non-zero value in bit 32 of the PSW, which is the 1st bit of the second word, **8**1685B28. It is the use of bits 31 and 32 that indicates this is a 64-bit address. The address to interrogate in this case would be 1_81685B28.

In many cases, for most current applications, you will still be interrogating 31-bit storage addresses, but in the future, as more applications make use of the extended addressability, you will reference storage pointed to by the Addressing Mode (AMODE) 64-bit PSW.

10.5 64-bit address space

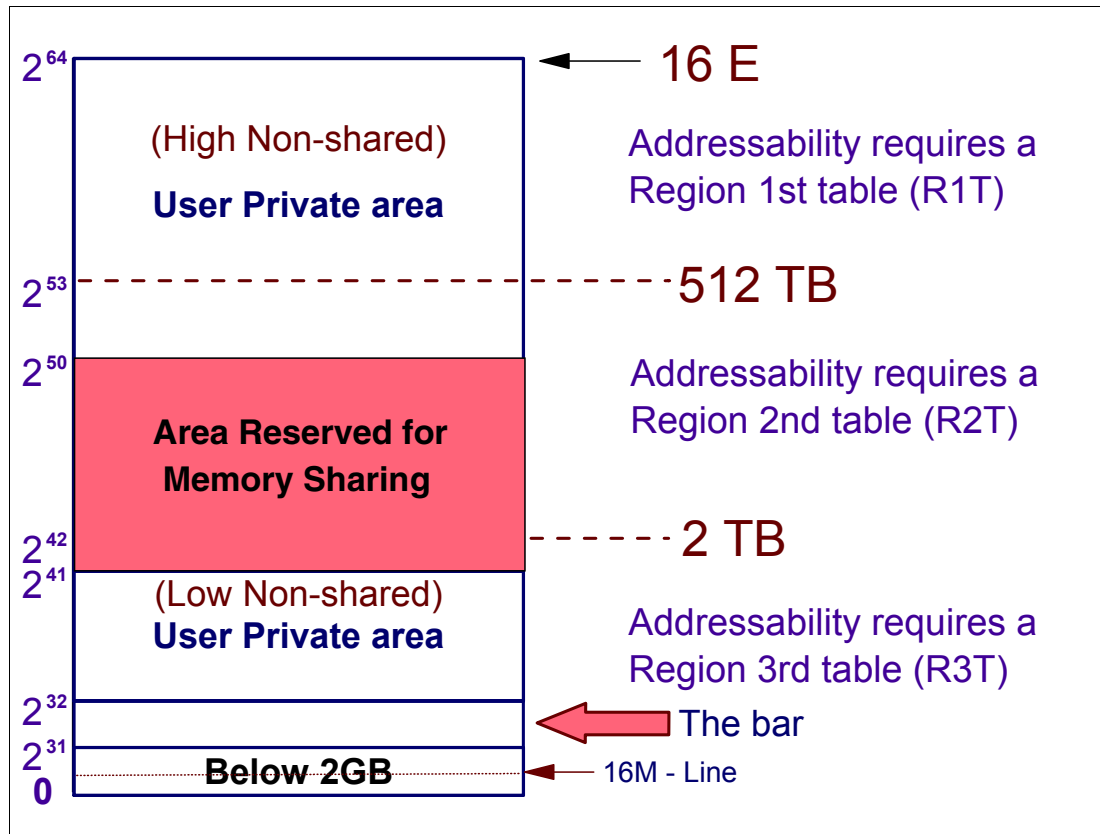


Figure 10-5 64-bit address space map

64-bit address space

With z/OS, the MVS address space expands to a size so vast that we need new terms to describe it. Each address space, called a 64-bit address space, is 16 exabytes in size; an exabyte is slightly more than one billion gigabytes. The new address space has logically 2^{64} addresses. It is 8 billion times the size of the former 2-gigabyte address space that logically has 2^{31} addresses. The number is 16 with 18 zeros after it:

16,000,000,000,000,000,000 bytes, or 16 exabytes

If you are coding a new program that needs to store large amounts of data, a 64-bit address space might work for you.

Introduction of 64-bit address space

As of z/OS V1R2, the address space begins at address 0 and ends at 16 exabytes, an incomprehensibly high address. The architecture that creates this address space provides 64-bit addresses. The address space structure below the 2-gigabyte address has not changed; all programs in AMODE 24 and AMODE 31 continue to run without change. In some fundamental ways, the address space is much the same as the XA address space.

In the previous 31-bit address space, a virtual line marks the 16-megabyte address. The 64-bit address space also includes the virtual line at the 16-megabyte address; additionally, it includes a second virtual line called the bar that marks the 2-gigabyte address.

The bar

The bar separates storage below the 2-gigabyte address, called *below the bar*, from storage above the 2-gigabyte address, called *above the bar*. The area above the bar is intended for data; no programs run above the bar. There is no area above the bar that is common to all address spaces, and no system control blocks exist above the bar. IBM reserves an area of storage above the bar for special uses to be developed in the future.

Memory sharing

Before z/OS V1R3, all programs in AMODE 31 or AMODE 24 were unable to work with data above the bar. To use virtual storage above the bar, a program must request storage above the bar, be in AMODE 64, and use the new z/Architecture assembler instructions.

As of z/OS V1R5, the following enhancements for 64-bit virtual storage have been added:

- ▶ 64-bit shared memory support
- ▶ Default shared memory addressing area between 2 terabytes and 512 terabytes

This shared memory is used by z/OS UNIX applications.

Using memory above the bar

The reason why someone designing an application would want to use the area above the bar is simple: the program needs more virtual storage than the first 2-gigabyte address space provides. Before z/OS V1R2, a program's need for storage beyond what the former 2-gigabyte address space provided was sometimes met by creating one or more data spaces or hiperspaces and then designing a memory management schema to keep track of the data in those spaces. Sometimes programs written before z/OS V1R2 used complex algorithms to manage storage, reallocate and reuse areas, and check storage availability. With the 16-exabyte address space, these kinds of programming complexities are unnecessary. A program can potentially have as much virtual storage as it needs, while containing the data within the program's primary or home address space.

Virtual memory above 2 GB is organized as memory objects that a program creates. A memory object is a contiguous range of virtual addresses that are allocated by programs as a number of application pages which are 1 MB multiples on a 1 MB boundary. Programs continue to run and execute in the first 2 GB of the address space.

Dynamic address translation

Dynamic address translation is the process of translating a virtual address during a storage reference into the corresponding real address. The virtual address may be a primary virtual address, secondary virtual address, AR-specified virtual address, or home virtual address. These addresses are translated by means of the primary, the secondary, an AR-specified, or the home address-space-control element, respectively.

After selection of the appropriate address-space-control element, the translation process is the same for all of the four types of virtual address. An address-space-control element may be a segment-table designation specifying a 2-GB address space, a region-table designation specifying a 4-TB, 8-PB, or 16-EB space, or a real-space designation specifying a 16-EB space. The letters K, M, G, T, P, and E represent kilo, 2^{10} , mega, 2^{20} , giga, 2^{30} , tera, 2^{40} , peta, 2^{50} , and exa, 2^{60} , respectively. A segment-table designation or region-table designation causes translation to be performed by means of tables established by the operating system in real or absolute storage. A real-space designation causes the virtual address simply to be treated as a real address, without the use of tables in storage.

Is a dump 31-bit or 64-bit?

The easiest way to determine this is to use ISPF to browse the unformatted dump data set.

The header for each record in the dump will show DR1 for a system running in 31-bit mode and DR2 for a 64-bit system dump. Figure 10-6 shows an ISPF browse of the dump data set.

```

BROWSE      APSG.SC48TS.DUMP1
  Command ==>
*****
DR2 H .....
DR2 CV.....
DR2 CV.....]°.
DR2 CV.....]μ.
DR2 CV.....]^.
DR2 CV.....]{.

```

Figure 10-6 64-bit architecture dump header record

A slightly more complex method for those familiar with IPCS is as follows:

- ▶ 31-bit (2 GB) MVS address spaces have architected Prefix Save Areas starting at x'0' in low core. These start with the restart new PSW (which begins "040C..."). This is what you would expect to see in low core of dumps from systems that are not running on the new HW, or which are using the new 64-bit support hardware, but are not running in 64-bit mode.
- ▶ If an MVS image has been IPLed to exploit 64-bit architecture, the low core will look completely different. The PSA is now 2 KB in size, rather than 1 KB and the format of the PSA starting from x'0' is completely different. Only a few of the fields are retained (for compatibility purposes), for example, the CVT address, the current TCB address and current ASCB address.
- ▶ To quickly identify whether a dump was taken from an image exploiting the 64-bit architecture you can look at offset x'A3'. If the value x'01' is set, this dump comes from an MVS image running in 64-bit mode. If x'00' is set, it is running in 31-bit mode. Currently no other bits are used in this byte.

It must be said that apart from the historical significance, you will not see many non-64 bit dumps in most current z/OS environments.



A

IPCS tools and lab exercises

The interactive problem control system (IPCS) is a tool provided in MVS to aid in diagnosing software failures. IPCS provides formatting and analysis support for dumps and traces produced by MVS, other program products, and applications that run on MVS.

Dumps produced by MVS fall into two categories:

- ▶ Formatted dumps: SYSABEND and SYSUDUMP ABEND dumps and SNAP dumps. IPCS cannot be used with formatted dumps.
- ▶ Unformatted dumps: SVC dumps, SYSMDUMP ABEND dumps, and stand-alone dumps. IPCS formats and analyzes unformatted dumps.

When you submit unformatted dump data sets to IPCS, it simulates dynamic address translation (DAT) and other storage management functions to recreate the system environment at the time of the dump. IPCS reads the unformatted dump data and translates it into words. For example, IPCS can identify the following:

- ▶ Jobs with error return codes
- ▶ Resource contention in the system
- ▶ Control block overlays

The information here should guide you in how to use IPCS and get information from a dump. The dump can be downloaded.

A.1 IPCS lab exercise agenda

- ❑ Introduction to IPCS and dumps
- ❑ IPCS tools
- ❑ Diagnosing loops and hangs
- ❑ Downloading dumps using FTP
 - IPCS default settings
 - Commands to analyze dump

Figure A-1 IPCS and dump analysis

Introduction to IPCS and dumps

The following topics are described in this appendix:

- ▶ How the lab is presented
- ▶ How to get into IPCS and set up to view the first dump
- ▶ Other related sessions

IPCS commands

This appendix describes the use of the following IPCS commands:

- ▶ List Title/List SLIP trap
- ▶ Status worksheet
- ▶ Formatting the RTCT
- ▶ ST REGS
- ▶ SYSTRACE
- ▶ VERBX MTRACE
- ▶ Key fields in SUMMARY FORMAT
- ▶ ANALYZE RESOURCE

Diagnosing loops and hangs using these tools

You can download the following dumps from the Redbooks site:

- ▶ Dump of a job using excessive CPU time
- ▶ Dump of a hung TSO user
- ▶ Dump of a hung job that is a contention problem
 - Be warned there is a tremendous amount of material in this lab.
 - The on-page title indicates exercises on that page.
 - Each exercise details commands to be entered.

A flowchart available at the end of the presentation on diagnosing loops and hangs shows the methodology used to diagnose the dumps. Consider the following when using the dumps:

- ▶ Everybody develops their own method over time.
- ▶ Use them as a starting point in understanding how to look at dumps.

How to download dumps using FTP to locate the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. The additional Web material that accompanies this book includes the following files:

File name	Description
SG246988.zip	Zipped DUMPs - (13 DUMPs)

Attention: The dump data sets you are going to download are in tersed format.

Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246988>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the book form number, SG246988.

Text from the IBM Redbooks Web site

The directories on our FTP server contain additional materials such as code samples for specific Redbooks. If there is additional media, such as a diskette or CD-ROM included with the hardcopy book, it should be located in the directory with the same name as the IBM Redbooks form number (SG24xxxx). Just click the specific directory and you will find the text or binary files. Normally they are zipped to make file transfer faster and more reliable.

If your browser does not properly recognize the file extension, it may try to display the file rather than present a download window. If this happens, right click the file and select **Save Link as** or **Save Target as**, and your browser's normal download window will be presented.

[Click here](#) to get to the directory listing of additional materials to download.

(The save directory is one that you select.) The SG246988.zip file is now saved in a directory on your workstation.

How to use the downloaded material

Perform the following tasks:

1. Unzip the supplied SG246988.zip to a temporary directory. The dumps in this file are tersed.

The dump data set names are shown in Figure A-2.

Data sets	Tracks
ITS0.S2822.DUMP1.TERSE	90
ITS0.S2822.DUMP2.TERSE	75
ITS0.S2822.DUMP3.TERSE	90
ITS0.S2822.DUMP4.TERSE	75
ITS0.S2822.DUMP5.TERSE	705
ITS0.S2822.DUMP6.TERSE	45
ITS0.S2822.DUMP7.TERSE	75
ITS0.S2822.DUMP8.TERSE	90
ITS0.S2822.DUMP9.TERSE	135
ITS0.S2823.DUMP7.TERSE	915
ITS0.S2895.DUMP1.TERSE	255
ITS0.S2895.DUMP2.TERSE	255
ITS0.S2895.DUMP4.TERSE	120

Figure A-2 The dump data set names that are downloaded

2. Use the following commands, shown in Figure A-3 on page 247, from the PC to upload the dumps to your MVS system. In the following example, the c:\temp directory is used. You need to specify where you saved the zip file if you did not use the c:\temp directory.

```

cd c:\temp
ftp 'your MVS system IP address'
C:\temp>ftp wtsc43.itso.ibm.com (our MVS system IP address)
Connected to wtsc43.itso.ibm.com.
220-FTP Server (user 'paulroge@us.ibm.com')
220 User (wtsc43.itso.ibm.com:(none)): 'enter your MVS user ID
331 Send password please.
Password: 'enter your password'
230-220-FTPMVS1 IBM FTP CS V1R7 at wtsc43.itso.ibm.com, 21:29:14 on 2007-02-26.
230-ROGERS is logged on. Working directory is "ROGERS.".
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=90
200 Site command was accepted
ftp> binary
200 Representation type is IMAGE.
ftp> put ITS0.S2822.DUMP1.TERSE
200 Port request OK.
125 Storing data set ITS0.
250 Transfer completed successfully.
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=75
ftp> put ITS0.S2822.DUMP2.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=90
ftp> put ITS0.S2822.DUMP3.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=75
ftp> put ITS0.S2822.DUMP4.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=705
ftp> put ITS0.S2822.DUMP5.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=45
ftp> put ITS0.S2822.DUMP6.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=75
ftp> put ITS0.S2822.DUMP7.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=90
ftp> put ITS0.S2822.DUMP8.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=135
ftp> put ITS0.S2822.DUMP9.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=915
ftp> put ITS0.S2823.DUMP7.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=255
ftp> put ITS0.S2895.DUMP1.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=255
ftp> put ITS0.S2895.DUMP2.TERSE
ftp> quote site blk=6144 lrec1=1024 recfm=fb tracks unit=sysallda primary=120
ftp> put ITS0.S2895.DUMP3.TERSE

```

Figure A-3 Commands to FTP dumps to the MVS system

3. Once the dump data sets have been copied to the MVS system, they must be untertered. If you do not have the terse utility as part of your TSO environment, see the following note.

Note: Decompress *all* data sets using TRSMAIN, which can be downloaded from:

► <ftp://ftp.software.ibm.com/s390/mvs/tools/packlib/>

A.2 IPCS lab setup instructions

❑ Choose Option 0 to set defaults

- Specify processing to be performed by IPCS dialogs and subcommands

```
----- z/OS 01.08.00 IPCS PRIMARY OPTION MENU -----
OPTION ==>

0  DEFAULTS      - Specify default dump and options      * USERID   - ROGERS
1  BROWSE        - Browse dump data set                  * DATE      - 07/02/05
2  ANALYSIS      - Analyze dump contents                  * JULIAN    - 07.036
3  UTILITY       - Perform utility functions              * TIME      - 11:52
4  INVENTORY     - Inventory of problem data              * PREFIX    - ROGERS
5  SUBMIT        - Submit problem analysis job to batch   * TERMINAL  - 3278T
6  COMMAND       - Enter subcommand, CLIST or REXX exec   * PF KEYS   - 24
T  TUTORIAL      - Learn how to use the IPCS dialog       *****
X  EXIT          - Terminate using log and list defaults

Enter END command to terminate IPCS dialog
```

Figure A-4 IPCS PRIMARY OPTION MENU

IPCS primary options

At the IPCS primary options panel choose Option 0 for defaults, as shown in Figure A-4. When you press Enter, you receive the panel with the default settings. Add the dump data set name to the Source field to initialize the dump. Following are the IPCS default settings:

```
Scope    ==> BOTH      (LOCAL, GLOBAL, or BOTH)
Source    ==> DSNAME('xxx.yyy.dump')
Address Space ==>
Message Routing ==> NOPRINT TERMINAL
Message Control ==> CONFIRM VERIFY FLAG(WARNING)
Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL
```

A.3 Commands to analyze dumps

- ❑ Command to determine the dump type
 - IP LIST TITLE
- ❑ Command to determine what the dump represents
 - IP LIST SLIPTRAP
- ❑ Command to determine useful information
 - IP ST WORKSHEET

Figure A-5 IPVS commands to analyze a dump

IP LIST TITLE

Use the IP LIST TITLE command to get a first guess as to what the dump represents. Look for the following kinds of information:

- ▶ System generated dumps typically have a COMPID= and other system-generated information, depending on the recovery routine that takes the dump.
- ▶ 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 a SLIP trap ID in them.
- ▶ Any program can issue an SDUMP macro and generate a title of its choosing. For IBM products a dump title directory can be found in Chapter 10 of *z/OS MVS Diagnosis: Reference*, GA22-7588.

The IP LIST TITLE command can be used to get the title of the dump, as follows:

```
IP LIST TITLE
TITLE
LIST 00000000 LITERAL LENGTH(X'58') CHARACTER
COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPXFSCLS+16D6,ABEND=S00C4,REASON
=00000004
```

Lab exercise #1

Analyzing a SLIP trap dump.

Lab exercise #1:

- ▶ Enter IPCS.
- ▶ Specify the dump by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to ITSO.S2822.DUMP1.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose a SLIP trap dump.

Diagnosing the dump

Use the IPCS commands LIST TITLE and LIST SLIPTRAP to determine the type of dump being analyzed.

Questions:

1. Use the IP LIST TITLE command if you have reason to believe that a slip trap was used to produce the dump and you want to know what was set. _____

If a SLIP trap was used you will see the following type of output:

SLIP SET,C=0C4,A=SVCD

If a SLIPTRAP was not used you will get this message:

Symbol SLIPTRAP not found

2. Based on the title of the dump you can make a guess as to what type of dump this is. Choose one of the following by putting a circle around it:

- STANDALONE DUMP
- A CONSOLE DUMP
- SLIP TRAP GENERATED DUMP
- PROGRAM GENERATED DUMP

3. The IP LIST SLIPTRAP command can be used to show the SLIP trap used to obtain any dump, if a SLIP trap was used.

- Was a SLIP trap used? YES / NO (circle one)
- If a SLIP trap was used what was it? _____

Answers to questions: See Appendix A.15, "LIST TITLE and LIST SLIPTRAP - Answers" on page 275.

IP ST Worksheet

This command displays the MVS Diagnostic Worksheet. During the initial use, it is possible you may have to reply Y to get the displayed information the first time you use this command.

Issue the command to determine the useful information available in the dump. The following information is displayed, for example:

- ▶ Dump title
 - Temptable: COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,....
- ▶ Date and time dump was taken:
 - Date: 01/10/2002 Time: 21:23:40.675321 Local
- ▶ Original dump data set name (can be useful for reference with Systoles):
 - Original dump data set: SYS0.DUMPSA6F.S00447
- ▶ System name (useful verification tool if more than one system exists)
 - CVT SNAME (154) SA6F
- ▶ For SVCDUMPS the PSW and ASIDs in control at the time of the dump:
 - HASID 0006 PASID 0006 SASID 0006 PSW 070C1000 82467428
- ▶ Number of CPUs and their numbers, which is useful for looking for loops:
 - Alive CPU mask: C000 No. of active CPUs: 0002
 - The mask shows CPU numbers 0-16 thus C=1100... or CPU0 and CPU1

Figure A-6 Display of initial information in the MVS Diagnostic Worksheet

```

MVS Diagnostic Worksheet
Dump Title: ECB WAIT
CPU Model 9672 Version 84 Serial no. 220A83 Address 02
Date: 07/22/2002 Time: 13:41:19.105252 Local
Original dump dataset: JJ.DUMP.PS01.D020722.T133948.S00007
Information at time of entry to SVCDUMP:
HASID 0089 PASID 0089 SASID 0089 PSW 070C1000 8BE3F9CC
CML ASCB address 00000000 Trace Table Control Header address 7F742000

Dump ID: 007
Error ID: N/A
SDWA address N/A

SYSTEM RELATED DATA
CVT SNAME (154) PS01 VERID (-18)
      CUCB (64) 00FD00B0 PVTP (164) 00FF3548 GDA (230) 021C01A0
      RTMCT (23C) 00F47448 ASMT (2C0) 00FD6390 RCEP (490) 0167E468
CSD Available CPU mask: C000 Alive CPU mask: C000 No. of active CPUs: 0002

```

Figure A-7 IP ST WORKSHEET command example results

Questions: Using the IP ST WORKSHEET command answer the following questions. Refer to the previous page for information about what this information looks like in the output.

1. What is the dump title? _____
2. Does this agree with the list title output you saw before? _____
3. How many CPs are online in this dump? _____
4. What is the original dump data set name?

5. When was the dump taken? _____
6. What was the name of the system this dump was taken on? _____
7. What was the primary address space (PASID) in control at the time of the dump?

8. The IP SELECT ALL command provides a list of all the ASID numbers and the jobnames associated with them. Use this command to determine what the jobname is for the PASID found above _____

Answers to questions: See Appendix A.16, "IP ST WORKSHEET - Answers" on page 275.

A.4 The RTCT control block

```
RRTCT: 00F50B20
+0000 NAME..... RTCT      SAP..... 2FD0BE00  SUP..... 00100000
+000C SYD..... 4E800000  SDLA..... 0000      MECB..... 808DD1C8
+0018 FASB..... 00000000  NAS..... 00000001  EEDA..... 024C5040
+0024 SDDS..... 00000000  SDDC..... 0000      MTCT..... 0000
+002C DSV..... 00BAED98  SSTK..... 00000000  ADGL..... 00C8F490
+0038 ADG1..... 00C8F530  ADG2..... 00C8F540  ADG3..... 00C8F4AE
+0044 ADG4..... 00C8F5C0  ADG5..... 80C8FF0C  TABG..... 80CA90F0
+0050 TABQ..... 80CA910E  TABR..... 80CA9150  DSCA..... 00F93F28
+005C DIND..... 0263C5B0  DIRS..... 0263C980  SDAT..... 0263CD50
+0068 SMOD..... 024C8070  SCON..... 02436688  CPID..... 024B8100
+0074 RPAR..... 0175CEF8  BPXP..... 00000000  TABO..... 00C83AC0
+0080 SDSU..... 021E7000  SDPL..... 023D4E28  FMT..... 00000000
+00A4 MLCK..... 00000001  MSRB..... 00FA2260  TEST..... 00000000
+00B2 SEQ#..... 000B      SDSW..... 0261B000  TDCB..... 00000000
+00BC          00000000  00000000  00000000  00000000  00000000
+00D0          00000000  00000000  00000000  SDWK..... 00BAEE68
+00E0 ESEQ..... 0000      ECFU..... 0000      EASD..... 0000
+00E6 ETIM..... 00000000  SAO..... 2FD0BE00  SUO..... 00100000
+00F4 SYO..... 4E800000  SDO..... 0D000000  SDNA..... 01
+00FF INDX..... 01      SDPR..... 00      BUFV..... 00000000
+0108 SDF..... 6562      ZZZ3..... 0000
      ASTB
          SDAS  SDF4  SDF5
          ----  ----  ----
          001 0023 A0    00
          002 0000 00    00
          003 0000 00    00
```

Figure A-8 RTCT control block example (top part)

RTCT control block

The recovery termination control table (RTCT) contains information about what can be expected to be found in the dump. The RTCT provides a communication area between the various functions associated with dumping facilities, for SYSABEND, SYSMDUMP, SYSUDUMP, and SVC dumps. It is used for coordination of the dump-related processes of task and system recovery, the memory termination controller, installation- and operator-defined dump requirements.

IP CBF command

The IP CBF RTCT command shows what ASIDs were requested under the SDAS heading, as shown in Figure A-8.

The IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS) command shows what options were requested. This may be important to verify that the storage required to diagnose a problem was requested. Of the flags formatted, the most useful often is the SDUSDATA flag. For example, the output below would indicate that nucleus modules and LPA modules loaded at the time of the dump should be viewable.

```

SDUMP_PL: 00F40458

==> FLAGS SET IN SDUFLAG0:
    DCB specified.
    Dump 4K buffer.
    HDR/HDRADR specified.
    ECB specified.
    BRANCH=YES specified.

==> FLAGS SET IN SDUFLAG1:
    SVC dump request.
    SYSMDUMP request.
TSO user extension is present.
    48+ byte parameter list.

==> FLAGS SET IN SDUSDATA:
    Dump all PSAs.
    Dump current PSA.
    Dump LPA mod. for RGN.
    Dump trace data.
    Dump CSA.
    Dump SWA.
    Dump summary dump data.
Dump all nucleus.
    Dump all defaults.

```

Figure A-9 IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS) command results

Questions: The IP CBF RTCT command formats the RTCT control block, which gives information such as what ASIDs were dumped (use the SDAS field).

1. Use the CBF RTCT command to find the ASID(s) included in this dump and list them here (you can see an example of what the output may look like in Figure A-8 on page 253). _____

Additionally, the RTCT contains information about what SDATA options were used. To format this information, use the IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS) command. Try this command and determine:

2. Was LSQA requested on the dump? YES/NO (circle one).
3. Was RGN requested (shown as RGN-Private)? YES/NO (circle one).
The output will also indicate whether certain component exits receive control or not in the SDUEXIT flag.
4. Look at these flags to determine if GRSQ was specified. YES/NO (circle one).

Answers to questions: See Appendix A.17, "Using the RTCT control block - Answers" on page 275.

A.5 The IP ST REGS command

```
CPU STATUS:
PSW=070C1000 80FE5CFC (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
DISABLED FOR PER
ASID(X'001B') 00FE5CFC. IEANUC01.IEAVESVC+05FC IN READ ONLY NUCLEUS
ASCB27 at F3FA00, JOB(LLA), for the home ASID
ASXB27 at 9FDF00 for the home ASID. No block is dispatched
HOME ASID: 001B PRIMARY ASID: 001B SECONDARY ASID: 001B
GPR VALUES
  0-3 80000000 80FF0000 009FF5A0 00FC4E88
  4-7 009F8E88 009FD358 80FE5CD6 00F3FA00
  8-11 00000000 80FE579C 009FD418 7FFFE2C0
 12-15 7FFE0000 00006730 00FE6200 80014910
ACCESS REGISTER VALUES
  0-3 7FFEA5CC 00000000 00000000 00000000
  4-7 00000000 00000000 00000000 00000000
  8-11 00000000 00000000 00000000 00000000
 12-15 00000000 00000000 00005F60 8210532A
ALET TRANSLATION
AR 00 Not translatable
AR 14 Not translatable
AR 15 Not translatable
CONTROL REGISTER VALUES
  0-3 5EB1EE40 00A2007F 007CCDC0 8000001B
  4-7 0001001B 00C506C0 FE000000 00A2007F
  8-11 00000000 00000000 00000000 00000000
 12-15 0082E07B 00A2007F DF880C71 7FFE7008
```

Figure A-10 IP ST REGS command example

The IP ST REGS command

This command indicates what the registers were at the time of the dump for the following kinds of dumps:

- ▶ For SLIP dumps - REGS at the time SLIP matched.
- ▶ For console dumps - typically all zeros.
- ▶ For abend dumps - they are theoretically the REGS at the time of the abend.
- ▶ For standalone dumps - use the IP CPU REGS command to get the REGS from each CPU.

ST REGS example

These examples simply skim the surface of the wealth of technical information available with the IP ST REGS output. See the example shown in Figure A-10.

The example output in Figure A-10 shows that the address in the PSW is X'0FE5CFC', the ASID is X'1B', and the failing instruction is located in offset X'5FC' in the CSECT IEAVESVC in the module IEANUC01 in the nucleus. You can now browse the dump at this location and look at the specific failing instruction. You could also use the information about the registers to find out more about the error if the address in the PSW does not point to the failing instruction.

If the calling program is in AR mode, all addresses that it passes, whether they are in a GPR or in a parameter list, must be ALET-qualified. A parameter list can be in an address space

other than the calling program's primary address space or in a data space, but it cannot be in the calling program's secondary address space.

Note: You can use the IP ST FAILDATA command instead as it formats the SDWA *if it is present*. Generally it will give you a better overall picture but it may not always be there and may not be the same as IP ST REGS due to recovery actions. In AR mode, as is the case here, the General Purpose Registers will be qualified by the access registers (ARs). So to look at the storage pointed to by a GPR, you need to also determine what address space it refers to. An AR value of 00000000 means the Primary ASID; 00000001 means secondary ASI, and 00000002 means home ASID. For example, in this dump the value in R9 = 15756F00 would be browsed in ASID(x'105').

Information from IP ST REGS

The following questions can all be answered with the IP ST REGS command.

Questions:

1. Was this dump in AR mode at the time of the failure? _____
2. What was the failing PSW address? _____
3. What ASID is this failing code executing in? _____
4. What was the failing TCB address? _____

Now using the address portion of the PSW, you want to get more information about the module that was running. You also want to browse some of the register storage. Use IPCS browse, IPCS Option 1, as shown in Figure A-4 on page 248.

5. When you browse the PSW address and back up with PF7, what eyecatcher do you see? (Note: if an address begins with x'8' it must be changed to x'0'; if it begins with x'C' change it to x'4')

6. Browsing the code 2 bytes before the PSW can you determine the reason for the ABEND0C1? _____

Answers to questions: See Appendix A.18, "Information from IP ST REGS - Answers" on page 276.

A.6 Browsing storage

```

DSNAME('SHARE.S2822.DUMP1A') POINTERS
-----
ASID(X'0065') is the default address space
PTR   Address   Address space                               Data type
S0001 00000000 ASID(X'0065')                               AREA
Remarks:
***** END OF POINTER STACK *****
Fill in the 's' under pointer to select the address you'd like to browse or
fill in the 's' as shown to get to address zero where you can use the L
command to display the storage your interested in as demonstrated below:
ASID(X'0065') ADDRESS(00.) STORAGE
-----
00000000   040C0000   813FF440   FFFFFFFF   FFFFFFFF   | ....a.4 ..... |
00000010   00FC79B8   FFFFFFFF   070E0000   00000000   | ..`..... |
.
. (lines deleted for display purposes)
.
00000110.:011F.--Same as above
00000120.:01FF.--All bytes contain X'FF'
00000200   D7E2C140   00010041   00F44008   030A6008   | PSA .....4 ...-. |
00000210   00F83000   030E5000   00000000   00000000   | .8....&..... |
Command ==> L 07E00F04. asid(x'65')                               SCROLL ==> CSR

```

Figure A-11 Browsing storage example

Browsing storage using IPCS Option 1 (Browse)

To browse storage, on the IPCS primary panel, shown in Figure A-4 on page 248, select Option 1 or use =1 on any IPCS command line to obtain the panel shown in Figure A-12. Fill in the dump data set name and the source and when you press Enter, the panel shown in Figure A-11 is displayed.

```

----- IPCS - ENTRY PANEL -----

CURRENT DEFAULTS:
Source ==> DSNAME('ITSO.S2822.DUMP1')
Address space ==> ASID(X'0023')

OVERRIDE DEFAULTS:                                     (defaults used for blank fields)
Source ==> DSNAME('ITSO.S2822.DUMP1')
Address space ==>
Password      ==>

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

```

Figure A-12 IPCS panel to enter dump defaults

Address space

If you are interested in a particular address space, then specify as shown in Figure A-12 on page 257.

Browsing storage

Next, press Enter on the panel; Figure A-13 is displayed.

```
DSNAME('ITS0.S2822.DUMP1') POINTERS -----
Command ==>                                SCROLL ==> CSR
ASID(X'0023') is the default address space
PTR   Address      Address space          Data type
S0001 00.          ASID(X'0023')          AREA
Remarks:
***** END OF POINTER STACK *****
```

Figure A-13 Panel displayed after an Enter on the previous panel

Select storage location

Use the S line command, as shown in Figure A-11 on page 257, to choose a pointer from the address pointer stack on the pointer panel. IPCS then uses the pointer to display storage that is addressed by that pointer. The storage then displayed is shown in Figure A-11 on page 257 and Figure A-14. Notice that the storage selected in the example is at location 00000000.

```
ASID(X'0023') ADDRESS(00.) STORAGE -----
00000000 040C0000 813FF440 FFFFFFFF FFFFFFFF | ....a.4 ..... |
00000010 00FC79B8 FFFFFFFF 070E0000 00000000 | ..~..... |
.
. (lines deleted for display purposes)
.
00000110.:011F.--Same as above
00000120.:01FF.--All bytes contain X'FF'
00000200 D7E2C140 00010041 00F44008 030A6008 | PSA ....4 ...-. |
00000210 00F83000 030E5000 00000000 00000000 | .8....&..... |
Command ==> L 07E00F04. asid(x'23')          SCROLL ==> CSR
```

Figure A-14 Storage displayed when issuing the S command

Browse the PSW address

To browse the PSW address, issue the IP ST REGS command to obtain the PSW address, as shown in Figure A-15.

```
CPU STATUS:
PSW=070C0000 87E00F04
(Running in PRIMARY, key 0, AMODE 31, DAT ON)
DISABLED FOR PER
ASID(X'0023') 07E00F04. LOOPER+4C IN EXTENDED PRIVATE
ASID(X'0023') 07E00F04. AREA(Subpool251Key08)+0F04 IN EXTENDED PRIVATE
ASCB101 at F9B400, JOB(IBMUSER3), for the home ASID
ASXB101 at 5FDE88 and TCB101E at 5EC120 for the home ASID
HOME ASID: 0023 PRIMARY ASID: 0023 SECONDARY ASID: 0023
```

Figure A-15 IP ST REGS command displays the PSW address

Note: When you browse the PSW address, if an address begins with X'8' it must be changed to X'0' if it begins with X'C', change it to X'4'.

Select the PSW address

To browse the PSW address, insert an I in the previous address, as shown in Figure A-16; this creates another line where you enter the PSW address, as shown in Figure A-17.

```
DSNAME('ITS0.S2822.DUMP1') POINTERS -----
Command ==>                                SCROLL ==> CSR
ASID(X'0023') is the default address space
PTR   Address          Address space        Data type
I0001 00.              ASID(X'0023')        AREA
Remarks:
***** END OF POINTER STACK *****
```

Figure A-16 Panel displayed after an Enter on previous panel

To select the PSW address, place an S in the PTR field of the PSW address.

```
DSNAME('ITS0.S2822.DUMP1') POINTERS -----
Command ==>                                SCROLL ==> CSR
ASID(X'0023') is the default address space
PTR   Address          Address space        Data type
00001 00.              ASID(X'0023')        AREA
S0002 07E00F04.        ASID(X'0023')        AREA
Remarks:
```

Figure A-17 Selecting the PSW address to display the storage

Note: When the PSW address storage is displayed, browsing the storage 2 bytes before the PSW, you can you determine the reason for an ABEND0C1.

A.7 IPCS SYSTRACE subcommand

```
{ SYSTRACE [ TIME(HEX | GMT | LOCAL) ]
----- Data Selection Parameters -----
[ EXCLUDE(BR) ]
[ EXCLUDE(MODE) ]
[ START(mm/dd/yy, hh.mm.ss.dddddd) ]
[ STOP(mm/dd/yy, hh.mm.ss.dddddd) ]
[ CPU(cpu-address) ]
[ TCB(TCB-list) ]
[ TTCH(TTCH-address | LIST) ]
[ WEB(WEB-list) ]
----- Address Space Selection Parameters -----
[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR ]
[ ASIDLIST(asidlist) ]
[ JOBLIST(joblist) | JOBNAME(joblist) ]
----- SETDEF-Defined Parameters -----
Note: You can override the following SETDEF parameters.

[ ACTIVE | MAIN | STORAGE ]
[ DSNNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
```

Figure A-18 SYSTRACE subcommand parameters

System trace

System trace writes trace data in system trace tables in the trace address space. System trace maintains a trace table for each processor. Obtain the trace data in a dump that included option SDATA=TRT.

SYSTRACE subcommand

Use the SYSTRACE subcommand to format system trace entries for all address spaces. This command is used to determine what else was happening in the system at the time of the dump.

- Options:
 - IP SYSTRACE ALL - formats all ASIDS
 - IP SYSTRACE TIME(LOCAL) - converts the time to local time (readable)
 - IP SYSTRACE ASID(x'nn') - formats only trace records associated with the requested ASID
- If a WAIT entry is found in SYSTRACE, the system is not running 100% CPU.
- EXT 1005 entries for the same ASID may be indicative of a loop.
- The command only traces traceable events, for example, SVCs or PCs.

Chapter 8 of *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589 has lots of details about system trace.

Note: For formatted dumps, system trace formats the system trace data and the system prints it directly.

For unformatted dumps, use the IPCS SYSTRACE subcommand to format and print or view the trace data in the dump.

SYSTRACE definitions

Figure A-19 shows the beginning columns of the system trace (SYSTRACE), shown in bold text. An SSRV trace entry represents entry to a system service. The service can be entered by a PC instruction or a branch.

Note: For every entry in the trace there are different mappings for the entry. Figure A-19 is only an example of what an entry can contain.

PR	ASID	WU-ADDR-	IDENT	CD/D	PSW-----	ADDRESS-	UNIQUE-1 UNIQUE-4/UNIQUE-5/UNIQUE-6	UNIQUE-2	UNIQUE-3
01-0001	00000000	WAIT							
01-0028	01F5F200	SRB			070C0000	80FE1CD8	00000028 062817AC 86281780 007FF510 00		
01-0028	00000000	SSRV	78			80FE1E58	4060E552 00000058 007BEFA8 00280000		
01-0028	00000000	SSRV	78			80FE1E78	0000FD02 00000098 007F0780 00280000		

Figure A-19 First columns of the system trace

The remainder of the system trace columns are in Figure A-20.

PSACLHS- PSACLHSE	PSALOCAL	PASD	SASD	TIMESTAMP-RECORD	CP
				BF65FF0E4BA5E728	28
00		0028	0028	BF65FF0E4EA51F68	28
Getmain				BF65FF0E4EA58DE8	28
Getmain				BF65FF0E4EA5A3E8	28

Figure A-20 Remaining columns of the system trace

The columns are as follows:

- PR** pr: Identifier of the processor that produced the TTE.
- ASID** home: Home address space identifier (ASID) associated with the TTE.
- WU-ADDR** wu-addr: Address of the task control block (TCB) for the current task or the work element block (WEB).
- IDENT** The TTE identifier, as follows:
- ▶ DSP - Task dispatch
 - ▶ SRB - Initial service request dispatch
 - ▶ SSRB - Suspended service request dispatch

	<ul style="list-style-type: none"> ▶ WAIT - Wait task dispatch
CD/D	ssid
PSW-address	Address of the PSW: <ul style="list-style-type: none"> ▶ dsp-new- psw: Program status word (PSW) to be dispatched ▶ srb-new- psw: PSW to receive control on the SRB dispatch ▶ ssrb-new- psw: PSW to receive control on the SSRB redispach
UNIQUE-1-6	(6 values as follows:) <ul style="list-style-type: none"> ▶ gpr0----: General register 0 ▶ gpr1----: General register 1 ▶ psamodew: PSAMODEW field in the PSA ▶ safnasid: LCCASAFN field in the logical configuration communication area (LCCA) and the related ASID ▶ flg-srb: SRBFLGS field from the SRB ▶ purgetcb: TCB (located in address space of the scheduler of the SRB) that gets control if the SRB abends and percolates
PSACLHS	One of the following: <ul style="list-style-type: none"> ▶ psaclhs-: String for the current lock held, from the PSACLHS field of the PSA. ▶ psaclhs4: PSACLHS4 field of the PSA ▶ srbhlhi-: SRBHLHI field in the SRB <p>This field contains descriptive text for some SVC, SSRV, and PC trace entries. The descriptive text does not appear in SNAP, SYSUDUMP, or SYSABEND output.</p>
PSALocal	psalocal : Locally locked address space indicator, from the PSALocal field of the PSA. This field will contain descriptive text for some SVC, SSRV, and PC trace entries. The descriptive text will not appear in SNAP, SYSUDUMP, or SYSABEND output.
PASD	cpsd : Primary ASID (PASID) at trace entry. This field will contain descriptive text for some SVC, SSRV, and PC trace entries. The descriptive text will not appear in SNAP, SYSUDUMP, or SYSABEND output.
SASD	sasd : Secondary ASID (SASID) at trace entry. This field will contain descriptive text for some SVC, SSRV, and PC trace entries. The descriptive text will not appear in SNAP, SYSUDUMP, or SYSABEND output.
TIMESTAMP	timestamp----- : Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on logrec data set records.
CP	The CP column contains 2 hex digits of the processor model-dependent information, which is intended to identify the physical CP that made the trace entry. CP is only provided when formatting SYSTRACE under IPCS. CP is not provided for SYSUDUMP, SYSABEND, or SNAP.

Questions:

1. By using IP SYSTRACE ALL and looking in the output for the word WAIT you can find out if all CPUs were busy at the time of the dump. In this dump were all CPUs busy? YES/NO (circle one).
2. EXT 1005 entries and CLKC entries are indicative of possible loops, Use the FIND (F) command in the output to see if there are any EXT or CLKC entries.
3. What address spaces had EXT 1005 entries (hint: search to BOTTOM of the output using F 'EXT')? _____
4. A loop would most likely be indicated by EXT entries with the same PSW addresses over and over. Do the address spaces you found above appear to be in a loop?

5. Use the IP SYSTRACE TIME(LOCAL) ASID(x'23') command to determine what the last time stamp in the trace is. (Use the BOTTOM command then you may need to scroll right with PF11) _____
6. You saw previously from the IP CBF RTCT command that ASID X'23' was dumped. What is the last TCB that was active in the trace table for this ASID? You can use the IP SYSTRACE ASID(X'23') TIME(LOCAL) command and then go to the BOTTOM to find this out. _____
7. Sometimes it is useful to look for abends that show up in SYSTRACE in the output from above. Use the F '*R' PREV command to find the last *RCVY entry that shows entry to recovery. Press PF7 to find the *PGM 001 entry that appears above that shows the 0C1. What was the PSW address that the 0C1 occurred at from looking at the *PGM 001 entry? _____

Note: A PGM trace entry is for a program interrupt.

Answers to questions: See Appendix A.19, "IP SYSTRACE - Answers" on page 276.

A.8 IPCS VERBX MTRACE subcommand

- ❑ Provides a snapshot of what happens just before the dump in the system log, as follows:

- If a job was started
- If a message was issued
- If a command was issued just prior to the problem

Sample output:

```
13:41:16.48 STC00761 00000210 DUMP COMM(DUMP OF JOE0400S)
13:41:16.88 P1      00000010 IEE311I DUMP      PARAMETER MISSING
13:41:16.89 P1      00000010 IEE711I SYSTEM DUMP NOT TAKEN. DUMP SPECIFICATION NOT VALID
13:41:18.27 P1 S8738 00000014 B092I- VSAM      01/30/98 08.40 STARTED
13:41:18.27 P1 S8738 00000014 B092I- KSDS      01/30/98 08.41 STARTED
13:41:18.27 P1 S8738 00000014 B054I- SESSION LIMIT SET TO 2048
13:41:18.29 P1 S8738 00000014 B015I- VTAMAPPL VERSION 6.1 TAPE LC2681 INITIALIZATION
complete
```

Figure A-21 VERBX MTRACE subcommand

VERBX MTRACE subcommand

This command displays the following:

- ▶ The master trace table entries for the dumped system. This table is a wraparound data area that holds the most recently issued console messages in a first-in, first-out order.
- ▶ The NIP hard-copy message buffer.
- ▶ The branch entry and NIP time messages on the delayed issue queue.

This trace gives you a snapshot of what is taking place just before the dump in the system log and is useful to see if a job was started, a message was issued or a command was issued just prior to the problem.

In the example, shown in Figure A-22 on page 265, the operator apparently was trying to capture a console dump and entered DUMP COMM(DUMP OF JOE0400S) instead of the correct syntax, which would have been DUMP COMM=(DUMP OF JOE0400S). Note also that this is a JES2 log. A JES3 log looks quite different.

```

13:41:16.48 STC00761 00000210 DUMP COMM(DUMP OF JOE0400S)
13:41:16.88 P1      00000010 IEE311I DUMP      PARAMETER MISSING
13:41:16.89 P1      00000010 IEE711I SYSTEM DUMP NOT TAKEN. DUMP SPECIFICATION NOT VALID
13:41:18.27 P1 S8738 00000014 B092I- VSAM      01/30/98 08.40 STARTED
13:41:18.27 P1 S8738 00000014 B092I- KSDS      01/30/98 08.41 STARTED
13:41:18.27 P1 S8738 00000014 B054I- SESSION LIMIT SET TO 2048
13:41:18.29 P1 S8738 00000014 B015I- VTAMAPPL VERSION 6.1 TAPE LC2681 INITIALIZATION complete

```

Figure A-22 Sample MTRACE output

Questions: The D GRS,C console command can be used to determine whether there is any resource contention on the active system. Looking at the IP VERBX MTRACE output, determine whether there were any GRS displays recently.

1. If so, what resource was the contention on? _____

Use F 'GRS,C' to find the message below and fill in the blanks in the message below:

- ISG343I 18.36.16 GRS STATUS
- S=SYSTEMS _____

The first blank represents the major name; the second represents the minor resource name.

Answers to questions: See Appendix A.20, "IP VERBX MTRACE - Answers" on page 277.

A.9 IP SUMMARY FORMAT subcommand

- ❑ Use the SUMMARY subcommand to:
 - Display or print dump data associated with one or more specified address spaces.
- ❑ Specify different parameters to selectively display the information you want to see.
 - SUMMARY produces different diagnostic reports depending on the report type parameter:
 - FORMAT, KEYFIELD, JOBSUMMARY, or TCBSUMMARY,
 - Address space selection parameters
 - ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, or JOBLIST

Figure A-23 SUMMARY subcommand and parameters

The SUMMARY subcommand

Use the SUMMARY subcommand to display or print dump data associated with one or more specified address spaces.

SUMMARY produces different diagnostic reports depending on the report type parameter, FORMAT, KEYFIELD, JOBSUMMARY, and TCBSUMMARY, and the address space selection parameters, ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, and JOBLIST. Specify parameters to selectively display the information you want to see.

Question information: The IP SUMM FORMAT ASID(x'nn') command will format lots of data about the specified address space. In this lab you are interested in the following fields:

- | | |
|----------------|---|
| ASCBDPH | Use the command F DPH from the top in the output. This is the dispatching priority of the address space. The range is 00-FF with FF being the highest. |
| RBOPSW | This field is contained in the RB under the TCB of interest. It can be found by going to the BOTTOM and issuing the find command F 'TCB: 00nnnnnn' PREV, then F ACTIVE to find the most recently active RB. This field shows the last running PSW address at the time the dump was taken or the address where the TCB entered a wait. |
| RBLINK | This field is found the same way as the RBOPSW above. The first byte (two digits) will indicate that RB (task for the first RB) is in a WAIT... 00 means not waiting, >00 (typically 01) means waiting. See RBOPSW to determine where it entered the wait. |

The result of the IP SUMMARY FORMAT ASID(x'7') followed by a F 'DPH' shows GRS has dispatching priority of X'FF' as expected, as shown in Figure A-24.

```

JOB GRS
  SELECTED BY: ASIDLIST

ASCB: 00F42D00
+0000  ASCB..... ASCB      FWDP..... 00FA7E80  BWDP..... 00F42E80
        LTCS..... 00000000  SVRB..... 005FD598  SYNC..... 000000B9
+0018  IOSP..... 00000000  R01C..... 0000      WQID..... 0000
        SAWQ..... 00000000  ASID..... 0007      R026..... 0000
+0028  LL5..... 00      HLHI..... 03      DPH..... 00FF
        TCBE..... 00000000  LDA..... 7FF12EA0  RSMF..... C0
  
```

Figure A-24 Result on the FIND command, F 'DPH'

Figure A-25 is the result of issuing the command to get to the bottom (BOTTOM command and press F8) followed by the F 'ACTIVE' PREV command to locate the top RB of the last task in the address space. This task is in a WAIT, which was issued at 87E44BD8.

Note: The WLIC field shows 00020001, which means the last SVC this task issued was SVC 1 (Wait). The first byte in the LINK field shows 01 if we are in a wait scenario.

```

ACTIVE RBS
PRB: 005F51D8
-0020  XSB..... 7FFFE10  FLAGS2... 80 RTPSW1... 00000000 00000000 RTPSW2... 00000000 2436D000
-0008  FLAGS1... 42800008 WLIC..... 00020001
+0000  RSV..... 00000000 00000000  SZSTAB... 00110083 CDE..... 005F5638  OPSW..... 070C1000 87E44BD8
+0018  SQE..... 00000000 LINK..... 015CCE88
+0020  GPRO-3... 00000002  005FF710  005DE244  00000000
+0030  GPR4-7... 00000048  00000000  00FC3CC0  005DE000
+0040  GPR8-11.. 005F9640  07E1E7B5  07E1D7B6  07E1C7B7
+0050  GPR12-15. 07E1B7B8  00016C28  00000001  005F9640
+0060  RSV..... C9E2C7E6  C4D9E5D9
  
```

Figure A-25 IP SUMM FORMAT

SUMM FORMAT subcommand questions

Questions:

1. Use the IP SUMM FORMAT ASID(X'23') command to determine what the dispatching priority of ASID x'23' is (Use the F 'DPH' command to find the dispatching priority):

Use the F 'TCB: 008D1E88' command to find the TCB that took the ABEND0C1, then issue F 'ACTIVE' to find the top RB....

2. From that RB, what are the values of OPSW _____
3. And the first byte of LINK _____
4. Is this TCB in a detected wait (hint: If RBLINK is >00 then TCB is in a wait, which was entered at the OPSW address recorded earlier)? YES/NO (circle one).

Answers to questions: See Appendix A.21, "SUMMARY FORMAT - Answers" on page 277.

A.10 The IP ANALYZE RESOURCE subcommand

- ❑ **ANALYZE** produces different diagnostic reports depending on the report type parameter
 - **EXCEPTION** displays contention information when a unit of work holds at least one resource for which contention exists and that unit of work is not waiting for another resource
 - **RESOURCE** displays contention information organized by resource name
 - **ASID** displays contention information organized by ASID
 - **ALL** displays all contention information

Figure A-26 IPCS ANALYZE subcommand

ANALYZE subcommand

Use the ANALYZE subcommand to gather contention information from component analysis exits and format the data to show where contention exists in the dump. ANALYZE obtains contention information for I/O, ENQs, suspend locks, allocatable devices, real frames, global resource serialization latches, and other resources.

The command is used to detect resource contention. Specifying GRSQ in the SDATA options makes the information more reliable. Generally the most useful information is found at the bottom of this report. The top is generally I/O device contention and isn't usually relevant. Figure A-27 on page 269 is an example of some contention, as follows:

- ▶ NAME=MAJOR=IGDCDSXS MINOR=SYSD.DFSMS.COMMDS is the resource name in contention.
- ▶ Note that the scope of the resource name is scope=systems.

Contention analysis

IPCS gathers contention information once for each dump. ANALYZE invokes each ANALYZE exit routine specified by parmlib members embedded in the BLSCECT parmlib member. When contention information has not been previously gathered, IPCS issues this message:

```
BLS01000I Contention data initialization is in progress
```

The amount of time required to gather contention information depends on the size of the dump, how many address spaces it contains, the number of I/O devices, and the amount of

contention in the dump. IPCS recommends that you run the ANALYZE subcommand in the background as part of a preliminary screening report.

In the event that no contention information is detected, IPCS issues:

BLS01002I No resource contention detected. Undetected contention is possible.

```
RESOURCE #0011:
  NAME=MAJOR=IGDCDSXS MINOR=SYSD.DFSMS.COMMDS
SCOPE=SYSTEMS
RESOURCE #0011 IS HELD BY:
  JOBNAME=SMS      ASID=0025  TCB=009EB0F0  SYSNAME=CM01
RESOURCE #0011 IS REQUIRED BY:
  JOBNAME=SMS      ASID=0026  TCB=009EB0F0  SYSNAME=PR02
  JOBNAME=SMS      ASID=0026  TCB=009EB0F0  SYSNAME=PR03
  JOBNAME=SMS      ASID=0028  TCB=009EC660  SYSNAME=SP02
  JOBNAME=SMS      ASID=0027  TCB=009EB0F0  SYSNAME=TS01
```

Figure A-27 IP ANALYZE RESOURCE subcommand

Note: Holders and waiters are identified in the output. ASID and TCB (where appropriate) are provided and whether a scope=systems resource is the holding system name.

Questions:

1. Use the IP ANALYZE RESOURCE command to identify any contention that exists in the dump. Record the resource name represented by RESOURCE #0001 (found at the bottom of the report) _____
2. How many address spaces are waiting for this resource: _____

Answers to questions: See Appendix A.22, "ANALYZE RESOURCE - Answers" on page 277.

Note: The ANALYZE RESOURCE output matches that of the D GRS,C command seen earlier.

A.11 Diagnosing excessive CPU time

Lab exercise #2:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNNAME to ITSO.S2822.DUMP2.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: A customer reports that jobname EXSCPU1 is using excessive CPU time.

Diagnosing the dump

To diagnosis this dump, the questions that follow will walk you through the relevant questions that will lead you to the diagnosis.

Questions:

1. Issue IP SYSTRACE ALL. Does IP SYSTRACE ALL show SVC entries? _____
2. Issue IP SELECT ALL. What is the ASID of EXSCPU1? _____
3. Issue IP SYSTRACE ASID(X'nn') where nn is the ASID number found above. Does this show a pattern? _____

Answers to questions: See Appendix A.23, “Diagnosing excessive CPU time - Answers” on page 278.

The CBFORMAT (CBF) command

Knowing that an SVC 6B is a modeset, we conclude that this code is in a loop. In this case the program storage is dumped, so it is not possible to browse the code. This is because RGN-private was requested. This can be verified with the following command:

```
IPCS CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS)
```

Figure A-28 on page 281 shows a flowchart that describes the process used to diagnose this problem.

A.12 TSO user hung

Lab exercise #3:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNNAME to ITSO.S2822.DUMP3.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: TSO user ID IBMUSER seems to be hung.

Diagnosing the dump

To diagnose this dump, the questions that follow will walk you through the relevant questions that will lead you to the diagnosis.

Questions:

1. Issue IP SYSTRACE ALL. Are there waits found in SYSTRACE ALL? _____
2. Is there an obvious pattern in the output above? _____
3. What address space number is IBMUSER? (IP SELECT ALL) _____
4. Are there any entries in SYSTRACE for this ASID? (IP SYSTRACE ASID(X'nn'))

5. Identify some other ASIDs that are running in IP SYSTRACE ALL. What are they?

6. Check the dispatching priority of these address spaces and compare them with the dispatching priority of IBMUSER (recall that the dispatching priority can be found by issuing IP SUMM FORMAT ASID(x'nn'), then F 'DPH'):
ASID 22 DPH? ASID 20 DPH? ASID 1F DPH? _____
7. Based on the above information we may assume that ASID x'1B' does not have a high enough dispatching priority. However, the only other non-system address space that was found was x'20', so perhaps we should try IP SYSTRACE ASID(x'20') ALL. Now do you see a pattern? _____
8. What diagnosis would you make based on this information?

Answers to questions: See Appendix A.24, "TSO user hung - Answers" on page 278.

A.13 Job IBMUSER3 hung (contention problem?)

Lab exercise #4:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNNAME to ITSO.S2822.DUMP4. Use any ASID from (0 to 20).
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Job IBMUSER3 appears to be hung.

Diagnosing the dump

To diagnose this dump, the questions that follow will walk you through the relevant questions that will lead you to the diagnosis.

Questions:

1. Use IP SYSTRACE ALL to determine if there are waits. YES/NO.
2. What ASID number is IBMUSER3 (IP SELECT ALL)? _____
3. Issue IP SYSTRACE ASID(X'nn'). Is there a pattern? YES/NO.
4. Are there any entries at all in SYSTRACE for ASID(X'nn')? YES/NO.
5. Issue IP ANALYZE RESOURCE, go to the BOTTOM and look for contention. Is IBMUSER3 waiting on an ENQUEUE? If so, what ASID is holding it? _____
6. Calling the ASID of the holder yy, is there a pattern for IP SYSTRACE ASID(X'yy')? YES/NO.
7. Are there any entries at all for ASID(yy)? YES/NO.
8. Is there any indication in IP ANALYZE RESOURCE that ASID yy is waiting for another resource? YES/NO.
9. Note the TCB address holding the enqueue. _____
10. Issue IP SUMM FORMAT ASID(x'yy') and look for the status of the top RB of TCB found above. Issue F 'TCB: 00zzzzzz' where zzzzzz is the address found above. Then issue F 'ACTIVE'.
11. What is the value of the first byte of the link field? _____
12. Does this indicate that the task is in a wait? See Figure A-24 on page 267. YES/NO.
13. In this case the next course of action would be to find out why the program issued a wait while holding the enqueue. However, once again the program was not dumped (RGN not requested). So try looking in IP VERBX MTRACE to see if you can find the answer to this problem. In the output go to the BOTTOM and find STARTJQ (ASID x'20' corresponds to STARTJQ as shown in IP SELECT ALL) to see what the job did after it started).
14. Is there anything that could be done on the console to relieve this problem?

Answers to questions: See Appendix A.25, "Job IBMUSER3 hung (contention problem?) - Answers" on page 278.

A.14 A standalone dump example

Lab exercise #5:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to ITSO.S2822.DUMP5. Use any ASID from (0 to 20).
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: The system crashed, ending up in a non-restartable wait state. A standalone dump was taken.

Diagnosing the dump

To diagnose this dump, the questions that follow will walk you through the relevant questions that will lead you to the diagnosis.

Questions:

1. First issue IP ST WORKSHEET and note the title:

2. Page down and look at the PSW address. For a wait state the last three digits of the PSW address will contain the wait state code and the two digits proceeding that will be the reason code. What was the WAITSTATE and reason code?

3. If you look up the meaning of this wait state you will note that it means the FRR stack is corrupted. The FRR stacks are located at PSA+380. Since the PSA starts at virtual address 0 for each processor we can simply browse address 380 to see what's there. Note if there were more than one CPU we'd need to browse each PSA separately as they are processor dependent. So on the L (locate) command use the keyword CP(1). Browse the storage address 00000380, using the IPCS browse function and write down the eyecatcher that appears there:

4. Try issuing the command L 380. cpu(0) while still looking at storage. What does this indicate? _____

The next step is to find out what caused the overlay you found above. To do that we can look in logdata, MTRACE or SYSTRACE in the hope that whatever overlaid the storage left some "footprints." In this case let us start with IP VERBX MTRACE. Whatever did this overlay would have been one of the last things running before the wait state (the system would not survive long from an overlay of this magnitude).

5. What was the last entry in VERBX MTRACE?

6. Use IP SELECT ALL to find the ASID of the item found above. _____
7. Use IP SYSTRACE ALL. What was the last SVC issued?

SVC 6B is a modeset SVC that will allow a program to be in supervisor state, allowing it to write to locations such as PSA+380.

Browse the PSW shown in the SYSTRACE entry found above. The PSW address is 25400E46. Page back (and sometimes you may have to page forward and then page back to get the correct address, which is 25400DDA) and record the eyecatcher found there:

Answers to questions: See Appendix A.26, "A standalone dump example - Answers" on page 279.

Conclusion

You have now completed the lab exercises. If you wish to go back and use any of the dumps to try some other IPCS items you may have learned, feel free to do so.

Remember that the dumps used are available via FTP for download as stated in the introduction.

The answers to the labs are on the following pages.

A.15 LIST TITLE and LIST SLIPTRAP - Answers

1. IP LIST TITLE can be used to get the title of the dump.
 - Record the tile here: __SLIP DUMP ID=0002__
2. Based on the title of the dump you can make a guess as to what type of dump this is. Is this a (answers are highlighted):
 - STANDALONE DUMP
 - A CONSOLE DUMP
 - SLIP TRAP GENERATED DUMP
 - PROGRAM GENERATED DUMP
3. IP LIST SLIPTRAP can be used to show the slip trap used to obtain any dump, if a slip trap was used.
 - Was a slip trap used? YES/ NO (circle one)
 - If a slip trap was used, what was it?
 - __SLIP SET,
C=0C1,A=SVCD,SDATA=(GRSQ,XESDATA,COUPLE,NUC,LPA,LSQA,RGN,TRT,S
UM,SQA,PSA)____

A.16 IP ST WORKSHEET - Answers

Using the IP ST WORKSHEET command answer the following questions. Refer to the previous page for information on what this information looks like in the output.

1. What is the dump title? __SLIP DUMP ID=0002__
2. Does this agree with the list title output you saw before? __YES__
3. How many CPs are online in this dump? __2__
4. What is the original dump data set name?
__LOWRYE.SHARE.D00006__
5. When was the dump taken? __02/13/2006 18:41:12.217103__

Note: You could use the above information to prove that everyone in this lab session is using a copy of the same dump.

6. What was the name of the system this dump was taken on? __SY1__
7. What was the primary address space (PASID) in control at the time of the dump? __23__
8. The IP SELECT ALL command provides a list of all the ASID numbers and the jobnames associated with them. Use this command to determine what the jobname is for the PASID found above __BADPROG2__

A.17 Using the RTCT control block - Answers

IP CBF RTCT formats the RTCT control block, which gives information such as what ASIDs where dumped.

1. Use the CBF RTCT command to find the ASID(s) included in this dump and list them here:

_____23_____

Additionally, the RTCT contains information on what SDATA options were used. To format this information, use the IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS) command. Try this command and determine:

2. Was LSQA requested on the dump? YES/NO
3. Was RGN requested (shown as RGN-Private)? YES/NO

The output above will also indicate whether certain component exits receive control or not in the SDUEXIT flag.

4. Look at these flags to determine if GRSQ was specified. YES/NO

A.18 Information from IP ST REGS - Answers

The following questions can all be answered by using the IP ST REGS command (as before, refer to the previous page for an example):

1. Was this dump in AR mode at the time of the failure? _NO_ (AR mode not indicated and HASID=PASID=SASID)
2. What was the failing PSW address? _25400F1C _
3. What ASID is this failing code executing in? __23__(PASID)
4. What was the failing TCB address? __8D1E88__

Now using the address portion of the PSW, you want to get more information about the module that was running. You also want to browse some of the register storage. Use IPCS browse IPCS Option 1.

5. When you browse the PSW address and back up with PF7, what eyecatcher do you see? __COMPLETED ENQ REPLY TO CONTINUE__
6. Browsing the code 2 bytes before the PSW, can you determine the reason for the ABEND0C1?
 - _Two bytes before the PSW address contains x'0000' which is an invalid opcode and will result in an ABEND0C1__.

A.19 IP SYSTRACE - Answers

1. By using IP SYSTRACE ALL and looking in the output for the word WAIT you can find out whether all CPUs were busy at the time of the dump. In this dump were all CPUs busy? YES/NO (highlighted)
2. EXT 1005 entries and CLCK entries are indicative of possible loops, Use the FIND (F) command in the output to see if there are any EXT or CLCK entries.
3. What address spaces had EXT 1005 entries (hint: search to BOTTOM of output using F 'EXT')? __1, 9, and 23__
4. A loop would most likely be indicated by EXT entries with the same PSW addresses over and over. Do the address spaces you found above appear to be in a loop? __No__

5. Use the IP SYSTRACE TIME(LOCAL) ASID(x'23') and determine what the last time stamp in the trace is: (use the BOTTOM command; then you may need to scroll right with PF11.)
_18:41:12.138602__
6. You saw previously from the IP CBF RTCT command that ASID X'23' was dumped. What is the last TCB that was active in the trace table for this ASID? You can use the IP SYSTRACE ASID(X'23') TIME(LOCAL) command and then BOTTOM to find this out.
008D1E88
7. Sometimes it is useful to look for ABENDs that show up in SYSTRACE in the output from above. Use the F '*R' PREV command to find the last *RCVY entry that shows entry to recovery. Issue PF7 to find The *PGM 001 entry that appears above that shows the 0C1. What was the PSW address that the 0C1 occurred at from looking at the *PGM 001 entry?
A5400F1C or 25400F1C

Note: A PGM trace entry is for a program interrupt.

A.20 IP VERBX MTRACE - Answers

The D GRS,C console command can be used to determine whether there is any resource contention on the active system. Looking at IP VERBX MTRACE output, determine if there were any GRS displays recently.

1. If so, what resource was the contention on? (Use F 'GRS,C' to find the message below and fill in the blanks in the message):

ISG343I 18.36.16 GRS STATUS

S=SYSTEMS SYSZTEST CLSSHR01____

The first blank represents the major name; the second represents the minor resource name.

A.21 SUMMARY FORMAT - Answers

1. Use the IP SUMM FORMAT ASID(X'23') command to determine what the dispatching priority of ASID x'23' is. (Use the F 'DPH' command to find the dispatching priority):
___D0___

Use F 'TCB: 008D1E88' command to find the TCB that took the ABEND0C1, then issue F 'ACTIVE' to find the top RB.

2. From that RB, what are the values of OPSW? ___070C0000 A5400F1C _____
3. And the first byte of LINK ___00_
4. Is this TCB in a detected wait (hint: If RBLINK is >00, then TCB is in a wait, which was entered at the OPSW address recorded earlier)? YES/NO (highlighted)

A.22 ANALYZE RESOURCE - Answers

1. Use IP ANALYZE RESOURCE to identify any contention in the dump. Record the resource name represented by RESOURCE #0001 (found at the bottom of the report)
___NAME=MAJOR=SYSZTEST MINOR=CLSSHR01___
2. How many address spaces are waiting for this resource: ___2___

A.23 Diagnosing excessive CPU time - Answers

1. Issue IP SYSTRACE ALL. Does IP SYSTRACE ALL show SVC entries? __ Yes__
2. Issue IP SELECT ALL. What is the ASID of EXSCPU1? ____ 1D__
3. Issue IP SYSTRACE ASID(X'nn'), where nn is the ASID number found above. Does this show a pattern? ____ Yes_____

A.24 TSO user hung - Answers

1. Issue IP SYSTRACE ALL. Are there waits in SYSTRACE all? ____ NO ____
2. Is there an obvious pattern in the output above? _NO (Not obvious, although as seen later one does exist)____
3. What address space number is IBMUSER? (IP SELECT ALL) ____ 1B_____
4. Are there any entries in SYSTRACE for this ASID? (IP SYSTRACE ASID(X'nn'))
____ NO _____
5. Identify some other ASIDs that are running in IP SYSTRACE ALL. What are they _Any of the following 1, 20, 1F, 5, 9_____
6. Check the dispatching priority of these address spaces and compare them with the dispatching priority of IBMUSER (recall that the dispatching priority can be found by issuing IP SUMMFORMAT ASID(x'nn'), then F 'DPH':
ASID=22 DPH=C1, __ ASID=1 DPH=FF __ ASID=20 DPH=FE__, ASID=1F DPH=FE
ASID=5 DPH=FF __ ASID=9 DPH=FF__
7. Based on the above information we may assume that ASID x'22' does not have a high enough dispatching priority. However, the only other non-system address spaces that were found are X'20' and X'1F', so perhaps we should try IP SYSTRACE ASID(x'20') ALL. Now do you see a pattern? _YES (SVC and SVCR)____
8. What diagnosis would you make based on this information? _Loop in ASID X'20' and ASID X'1F', which run in a higher dispatching priority than ASID X'22' (IBMUSER) resulting in IBMUSER getting starved for CPU._

Note: We cannot see what ASID(x'20') is doing because local storage for ASID X'20' was not dumped in this case.

A.25 Job IBMUSER3 hung (contention problem?) - Answers

1. Use IP SYSTRACE ALL to determine if there are waits. **Yes/No**
2. What ASID number is IBMUSER3 (IP SELECT ALL) ____ 16_____
3. Issue IP SYSTRACE ASID(X'nn'). Is there a pattern? Yes/**No**
4. Are there any entries at all in SYSTRACE for ASID(x'nn')? Yes/**No**
5. Issue IP ANALYZE RESOURCE go to the BOTTOM and look for contention. Is IBMUSER3 waiting on an ENQUEUE? If so, what ASID is holding it? ____ x'20' _____
6. Calling the ASID of the holder yy, is there a pattern for IP SYSTRACE ASID(X'yy')? Yes/**No**
7. Are there any entries at all for ASID(yy) Yes/**No**

8. Is there any indication in IP ANALYZE RESOURCE that ASID yy is waiting for another resource? Yes/**No**
9. Note the TCB address holding the enqueue ____008FF2A0__
10. Issue IP SUMM FORMAT ASID(x'yy'). Look for the status of the top RB of TCB found above. Issue F 'TCB: 00zzzzzz' where zzzzzz is the TCB address found above. Then issue F 'ACTIVE'.
11. What is the value of the first byte of the link field? _01_
12. Does this indicate that the task is in a wait? See Figure A-24 on page 267. __Yes/No__
13. In this case the next course of action would be to find out why the program issued a wait while holding the enqueue. However, once again the program was not dumped (RGN not requested). So try looking in IP VERBX MTRACE to see if you can find the answer to this problem. In the output go to the BOTTOM and find STARTJQ (ASID x'20' corresponds to STARTJQ as shown in IP SELECT ALL), to see what the job did after it started).

Is there anything that could be done on the console to relieve this problem? __

```
$HASP100 PK100LP ON STCINRDR
IEF695I START STARTJQ WITH JOBNAME PK100LP IS ASSIGNED
$HASP373 PK100LP STARTED
*13 COMPLETED ENQ REPLY TO CONTINUE
$HASP100 IBMUSER3 ON INTRDR SUB VIA TSO
IRRO10I USERID IBMUSER IS ASSIGNED TO THIS JOB.
ICH70001I IBMUSER LAST ACCESS AT 17:25:19 ON WEDNESDAY
$HASP373 IBMUSER3 STARTED - INIT 1 - CLASS A - SYS
```

_____STARTJQ issued a WTOR indicating it is stuck waiting for a reply. It is reasonable to assume that a reply would eliminate the hang and release the enqueue that IBMUSER3 is waiting on.____

```
D GRS,C
ISG343I 17.31.36 GRS STATUS 239
S=SYSTEMS SYSZTEST CLSSHR04
SYSNAME      JOBNAME      ASID      TCBADDR      EXC/SHR
SY1          PK100LP      0023      008FF2A0     EXCLUSIVE
SY1          IBMUSER3     0016      008D1E88     EXCLUSIVE
```

A.26 A standalone dump example - Answers

1. First issue IP ST WORKSHEET. Note the title: ____WAIT 084 SYSTEM CRASH
STANDALONE DUMP_____
2. Page down and look at the PSW address. For a wait state the last three digits of the PSW address will contain the wait state code and the two digits proceeding that will be the reason code. What was the WAITSTATE and reason code? ____WAIT 084-04__
3. If you look up the meaning of this wait state you will note that it means the FRR stack is corrupted. The FRR stacks are located at PSA+380. Since the PSA starts at virtual address 0 for each processor we can simply browse address 380 to see what is there. Note that if there were more than one CP, we would need to browse each PSA separately as they are processor dependent. So on the L (locate) command we would use the keyword CP(1). Browse the storage address 00000380 using the IPCS browse function and write down the eyecatcher that appears there: __Y OF IS A BIG BAD OVERLAY OF THIS PSA STACK__

Try issuing command L 380. cpu(0) while still looking at storage. What does this indicate:

```
__ __BLS19003I PCCAVT indicates CPU(0) is not online
____BLS18104I Symbol PCCA0 not found
____BLS18104I Symbol PSA0 not found _____
```

The next step is to find out what caused the overlay you found above. To do that we can look in logdata, MTRACE or SYSTRACE in the hope that whatever overlaid the storage left some “footprints.” In this case let us start with IP VERBX MTRACE. Whatever did this overlay would have been one of the last things running before the wait state (the system would not survive long from an overlay of this magnitude).

4. What was the last entry in VERBX MTRACE? _____\$HASP373 KILLER
STARTED_____
 5. Use IP SELECT ALL to find the ASID of the item found above _____x'19'_____
 6. Use IP SYSTRACE ALL what was the last SVC issued? _____SVC 6B_____
- SVC 6B is a modeset SVC that will allow a program to become supervisor state, allowing it to write to locations such as PSA+380.
7. Browse the PSW shown in the SYSTRACE entry found above. The PSW address is 25400E46. Page back (and sometime you may have to page forward and then page back to get the correct address, which is 25400DDA) and record the eyecatcher found there:
__THIS PROGRAM DESIGNED TO CAUSE WAIT084 __

A.27 Diagnosing loops and hangs

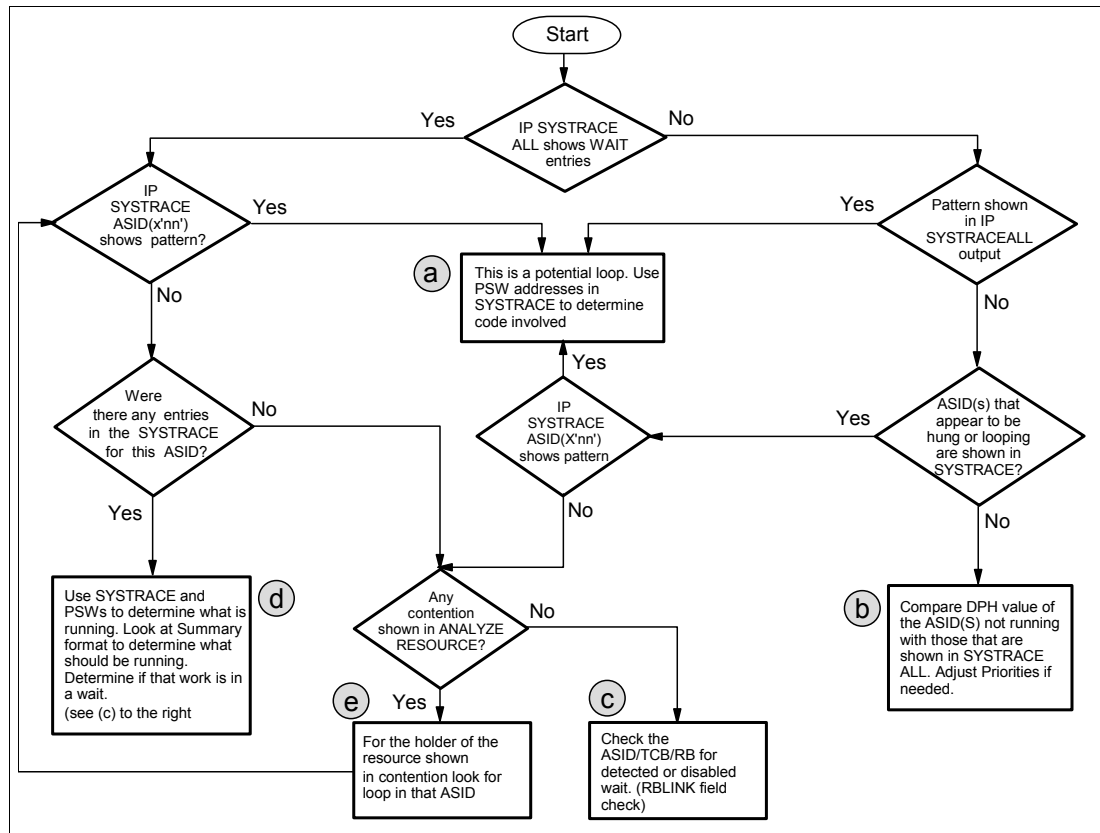


Figure A-28 Flowchart for loops and hangs

Loops and hangs

The flowchart shown in Figure A-28 can be used to diagnose possible loops and system hangs that may occur during processing.

Step a

To get to this point a pattern of entries has been found in SYSTRACE. Use the PSW address in the SYSTRACE entries to determine what modules may be involved in the potential loop that has been found. If EXT 1005 entries have been found, this indicates that the code running is not executing traceable events. Even a couple of these entries can be significant if the PSW is in the same area of code on each entry.

Step b

At this point a dispatching priority problem should be suspected. While this is not the only possible reason for the ASIDs being hung, it should be checked. Pick a couple of the ASIDs that occur frequently in the SYSTRACE ALL and look at the DPH values. Compare this to the DPH value of the job that should be running. If the DPH values in SYSTRACE ALL are higher, then suspect that perhaps the job or ASID simply cannot get the processor. If this does not work out, analyze resource for contention and look to see if the job that should be running is in a detected wait. (See steps c and e.)

Step c

In A.9, “IP SUMMARY FORMAT subcommand” on page 266, some of the key fields in the SUMMARY FORMAT were described. To check to see if the address space is in a detected wait, the RBLINK field of the TCB that should be running (assume this is the last TCB unless there is a specific reason to believe that another TCB may be involved). Also, the ASCBENST field can be used to check the last time this ASID went into a wait (compare with time stamps in system trace).

Step d

If the ASID in question is running in SYSTRACE then the goal is to determine what is supposed to be running in the address space that is not. To accomplish this requires some knowledge of the address space, looking at SUMMARY FORMAT with all individual TCBs to be examined, to determine what ones should be running. If these TCBs are not in detected waits then SYSTRACE ASID(x'nn') can be checked to see if those tasks are looping.

Step e

If contention is noted in the ANALYZE RESOURCE command (more than a couple tasks waiting for a resource), then the goal becomes finding out why the task holding the resource is not releasing it. This can be treated as though the resource holder is a hung address space, which means going back to look for patterns in IP SYSTRACE ASID(x'nn') where nn is the ASID number holding the resource.



Using IPCS to diagnose abends

This appendix describes certain abend types and how to analyze them. Following are the procedures to analyze the dumps:

- First symptoms

Messages indicate a system or user abend. For example, message IEA995I has been issued to the operator console. A dump was produced. An error was recorded in the logrec data set.

- Information and tools needed for analysis

- IPCS installed
- SVC dump, SYSUDUMP, SYSMDUMP, or SYSABEND dump
- Logrec error record
- Master trace
- Job log

- Types of dumps to be analyzed:

- Abend0C1: PSW, REGS and some basics
- Abend0c4: Exploring standard save areas
- ABEND138: Diagnosing abends via RBs and the parmlist to SVCs.
- An ABEND878-8: Entering the world of VSM figuring out storage related ABENDs.
- An ABEND878-10 Local Storage shortages
- WAIT 0A2 - General system problems
- Language Environment - System dump diagnosis

B.1 Lab exercises

There are eight dumps that you can work on. You do not need to go through each sequentially. An index to the dumps follows:

- ▶ An “Introduction to IPCS tools” dump.
- ▶ An abend0C1: PSW, REGS and some basics
- ▶ An Abend0c4: Exploring Standard save areas
- ▶ An ABEND138: Diagnosing ABENDs via RBs and the parmlist to SVCs.
- ▶ An ABEND878-8: Entering the world of VSM figuring out storage related ABENDs.
- ▶ An ABEND878-10: Local Storage shortages
- ▶ WAIT 0A2 - General system problems
- ▶ Language Environment - System dump diagnosis

Lab setup instructions

At the IPCS primary options panel, shown in Figure B-1, choose Option 0 for defaults.

```
----- z/OS 01.08.00 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

          * USERID - ROGERS
          * DATE   - 07/02/05
          * JULIAN  - 07.036
          * TIME   - 11:52
          * PREFIX  - ROGERS
          * TERMINAL- 3278T
          * PF KEYS - 24
          *****

Enter END command to terminate IPCS dialog
```

Figure B-1 IPCS PRIMARY OPTION MENU

Lab exercise #1:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to ITS0E.S2895.DUMP1.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose a SLIP dump.

When selecting Option 0, Figure B-2 on page 285 is displayed. Add the dump data set name to the Source field to initialize the dump, as follows:

```
Source ==> DSNAME('ITS0.S2895.DUMP1')
```

```

----- IPCS Default Values -----
Command ==>

You may change any of the defaults listed below. The defaults shown before
any changes are LOCAL. Change scope to GLOBAL to display global defaults.

Scope ==> LOCAL (LOCAL, GLOBAL, or BOTH)

If you change the Source default, IPCS will display the current default
Address Space for the new source and will ignore any data entered in
the Address Space field.

Source ==> DSNAME('ITS0.S2895.DUMP1')
Address Space ==>
Message Routing ==> NOPRINT TERMINAL
Message Control ==> CONFIRM VERIFY FLAG(WARNING)
Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL

Press ENTER to update defaults.

Use the END command to exit without an update.

```

Figure B-2 Default panel after selecting Option 0

Use IPCS commands

The IP ST REGS command tells you what the registers were at the time of the dump, as follows:

- ▶ For SLIP dump REGS at time slip matched.
- ▶ For console dump - typically all zeros.
- ▶ For abend dump - they are theoretically the REGS at time of abend.
- ▶ For standalone dump use IP CPU REGs to get REGS from each CPU.
- ▶ The IP ST FAILDATA command formats the SDWA if it is present. Generally it will give you a better overall picture but may not always be there and may not be the same as ST REGS due to recovery actions.

Information from the IP ST REGS command

If the calling program is in AR mode, all addresses that it passes, whether they are in a GPR or in a parameter list, must be ALET-qualified. A parameter list can be in an address space other than the calling program's primary address space or in a data space, but it cannot be in the calling program's secondary address space.

What does an AR contain? An AR contains a token, an access list entry token (ALET). An ALET is an index to an entry on the access list. An access list is a table of entries, each of which points to an address space, data space, or hiperspace to which a program has access.

The following questions can all be answered by using the IP ST REGS command.

Questions:

1. Was this dump in AR mode at the time of the failure? _____
2. What was the failing PSW address? _____
3. What ASID is this failing code executing in? _____
4. What was the failing TCB address? _____
5. What is the value in R14? _____

Answers to questions: See “Lab exercise #1 - Answers IP ST REGS” on page 302.

IP SYSTRACE

Use to determine what else was happening on the system at the time of the dump.

- ▶ Options to use:
 - IP SYSTRACE ALL - formats all ASIDs.
 - IP SYSTRACE TIME(LOCAL) - converts the time to local time (readable).
 - IP SYSTRACE ASID(X'nn') - formats only trace records associated with the requested ASID.

Things to look for in the SYSTRACE:

- ▶ If a WAIT entry is found in SYSTRACE the system is not running 100% CPU.
- ▶ EXT 1005 entries for the same ASID may be indicative of a loop.
- ▶ Only traces traceable events such as SVCs, PCs.

Note: See Chapter 8 in *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085 for examples and details of SYSTRACE. See “SYSTRACE definitions” on page 261 for sample output.

Questions:

1. By using IP SYSTRACE ALL and looking in the output for the word WAIT you can find out if all CPUs were busy at the time of the dump. In this dump were all CPUs busy? YES/NO (circle one)
2. EXT 1005 entries and CLKC entries are indicative of possible loops. Use the FIND command in the output to see if there are any EXT or CLKC entries.
 - What address spaces had EXT 1005 entries (Use: F 'EXT')?

 - A loop would most likely be indicated by EXT entries with the same PSW addresses over and over. Do the address spaces you found above appear to be in a loop?

3. Use IP SYSTRACE TIME(LOCAL) ASID(x'20') and determine what the last time stamp in the trace is. Use the BOTTOM command, then you may need to scroll right with PF11. _____
4. You saw previously from the IP CBF RTCT command that ASID X'20' was dumped. What is the last TCB that was active in the trace table for this ASID? You can use the IP SYSTRACE ASID(X'20') TIME(LOCAL) command and then BOTTOM to find this out.

5. Sometimes it is useful to look for abends that show up in SYSTRACE in the output from above. Use the F '*R' PREV command to find the last *RCVY entry that shows entry to recovery. Press PF7 to find the *PGM 001 entry that appears above that shows the 0C1. What was the PSW address that the 0C1 occurred at from looking at the *PGM 001 entry? _____
6. Issue the F 'B' PREV command to find the SVC B entry (repeat find once to get past the SVCR), use the mapping below to answer the following questions:
 - What address will the time be returned to _____
 - What format will it be returned in _____

Answers to questions: See “Lab exercise #1 - Answers IP SYSTRACE” on page 302.

Some Key Fields in IP SUMM FORMAT

The IP SUMM FORMAT ASID(X'nn') command will format lots of data about the specified address space. In this lab we will be interested in the following fields:

- ▶ RBOPSW - (contained in the RB under the TCB: of interest) - Can be found by going to the bottom and issuing F 'TCB: 00nnnnnn' PREV, then F ACTIVE to find the most recently active RB. This field shows the last running PSW address at the time the dump was taken or the address that the TCB entered a wait at.
- ▶ WLIC - (found in the same manner as RBOPSW above) shows the last interrupt that occurred on a given RB.
- ▶ GPR values - Show the register values at the time of the interrupt in the *previous* RB. That means that the RB with the WLIC value stores its registers in the next RB, or in the TCB if there is not a following RB.
- ▶ TCB summary at the very bottom of the output - Contains a CMP field that shows the last completion code issued for a TCB.

Figure B-3 on page 288 shows an example of a TCB summary where the last TCB shows a completion code of ABEND 0C9.

```

JOB IBMUSER3 ASID 0023 ASCB 00F9B400 FWDP 00FA3300 BWDP 00FA3480 PAGE 00000005
TCB AT      CMP      NTC      OTC      LTC      TCB      BACK      PAGE
005FE0A8 00000000 00000000 00000000 005FFBF8 005FFE88 00000000 00000020
005FFE88 00000000 00000000 005FE0A8 00000000 005FFBF8 005FE0A8 00000024
005FFBF8 00000000 005FFE88 005FE0A8 005EC8B0 005EC8B0 005FFE88 00000026
005EC8B0 00000000 00000000 005FFBF8 005EC120 005EC120 005FFBF8 00000029
005EC120 940C9000 00000000 005EC8B0 00000000 00000000 005EC8B0 00000032

```

Figure B-3 TCB summary

Figure B-4 shows the result of issuing the BOTTOM command followed by the F 'ACTIVE' previous command to locate the TOP RB of the Last Task in the address space. Note this task is in a WAIT that was issued at: 87E44BD8.

Note: The WLIC field shows 00020001, which means the last SVC this task issued was SVC 1 (Wait).

```

ACTIVE RBS
PRB: 005F51D8
-0020 XSB..... 7FFFE10  FLAGS2... 80          RTPSW1... 00000000 00000000
RTPSW2... 00000000 2436D000
-0008 FLAGS1... 42800008 WLIC..... 00020001
+0000 RSV..... 00000000 00000000          SZSTAB... 00110083 CDE.....
005F5638 OPSW..... 070C1000 87E44BD8
+0018 SQE..... 00000000 LINK..... 015CCE88
+0020 GPR0-3... 00000002 005FF710 005DE244 00000000
+0030 GPR4-7... 00000048 00000000 00FC3CC0 005DE000
+0040 GPR8-11.. 005F9640 07E1E7B5 07E1D7B6 07E1C7B7
+0050 GPR12-15. 07E1B7B8 00016C28 00000001 005F9640
+0060 RSV..... C9E2C7E6 C4D9E5D9

```

Figure B-4 PRB layout

SUMMARY FORMAT exercises

Questions:

1. Use IP SUMM FORMAT ASID(X'20') followed by the BOTTOM command. Looking at the TCB summary, what is the TCB address that ended in a non-zero completion code:

2. Use F 'TCB: 00' PREV command to find the TCB that took the ABEND0C1 then issue F 'ACTIVE' to find the top RB.
 - From that RB what are the values of OPSW _____
 - And the WLIC value _____
 - What is the value in R14 at the time the SVC 1 was issued? (Hint: obtain R14 from the following SVRB since the registers at the time the SVC was entered can be found in the following RB) _____
 - Browse the storage at R14 above. What is the offset into the “Module” above assuming that the “I” at the beginning of the name represents the start of the module

Answers to questions: See “Lab exercise #1 - Answers Summary Format” on page 302.

Diagnosing an ABEND0C1 dump

The exercises on the following pages are designed to demonstrate how to diagnose an ABEND0C1. An ABEND0C1 is an attempt by the processor to execute an instruction that is not valid or not coded correctly.

Typically the abend will occur when a program executes a bad branch. Thus, often the PSW where the abend occurs is less important than where the last valid instruction was executed. There are a couple of ways to determine that.

- ▶ Find a base register. Many programs use a base register to establish addressability. This may be one or more registers but typically R12 is chosen. Thus looking at R12 may point to code that was last in control.
- ▶ Find the source of the branch. By convention often the BALR 14,15 instruction is used to get from one program to another. If this is the case, R14 will point to the source of the call.
- ▶ Look at the TCB/RBs of the abending task. In some cases the previous RB can give a clue as to what program was to get control next. For instance, perhaps the previous RB has a WLIC of 00010006 which would be a LINK SVC and will enable you to look at the parmlist for the link to find the information about what program got control as a result.
- ▶ Examine SYSTRACE for the ASID/TCB that abended. Perhaps there was a traceable event that occurred prior to the abend that will give you a clue as to what program was in control leading up to the abend.

Use any details you get from the above to search problem databases for a known fix for a vendor problem or to feed back to the programmer for a customer-written program.

Lab exercise #2:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to ITSO.S2822.DUMP6.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose an ABEND0C1 ABEND dump.

Questions:

1. Determine what this dump is all about: Issue the IP LIST TITLE command. From the output is there any indication that this dump was the result of an abend? What abend? _____.

Normally we could confirm this with the IP ST WORKSHEET command. However, in this case the dump was captured by adding a SYSMDUMP DD card in the job so no SDWA was captured, so in this case ST WORKSHEET does not help.

2. Using the IP SYSTRACE ALL command and issuing a F '*PGM', what PSW address was the PGM 001 (a.k.a. ABEND0C1) taken at? _____
3. Fill in the abend code in the *RCVY entry below based on the *RCVY entry that immediately follows the *PGM 001:
 - *RCVY PROG 94 _ _ _ 000 (fill in the 3 missing characters)
4. Use the IP ST REGS command to get the relevant information about the abend 0C1. Record the following:
 - PSW _____
 - R14 _____
 - Primary ASID (PASN) _____
 - Abending JOBNAME _____
5. Use the =1 command to get into IPCS browse:
 - Browse the PSW address what 'instruction' does the PSW point to? _____
6. Often, branches are accomplished with BALR 14,15, making R14 point to the caller. Check R14 in this dump and see what the instruction before R14 is: Browse the address in R14 and record the 4-byte instruction below:

7. The instruction above was written in Assembler as BAL 14,LALALAND. What program issued this bad branch? Use PF7 to locate an eyecatcher and record what you find:

8. What is the offset from the beginning of the program (assume the 47F0 instruction to branch around the program is the beginning of the program) of the BAL 14,LALALAND found above _____

Answers to questions: See “Lab exercise #2 - Answers diagnosing an ABEND0C1” on page 303.

Diagnosing an ABEND0C4

This exercise is designed to show how to diagnose an ABEND0C4. An ABEND0C4 is a data exception which typically means that an attempt was made by the executing instruction to either read or write data in an area of storage that was either not GETMAINed or that the program is not authorized (by PSW key) to access.

To diagnose this problem it is necessary to determine what instruction caused the abend to occur and what program was running at the time that contained that instruction. To accomplish this:

- ▶ Find the PSW address from the registers at the time of the abend.
- ▶ Browse this PSW address in storage and back up until an eyecatcher is found.
- ▶ Determine the offset of the abend in the code by subtracting the PSW address from the beginning of the program name.
- ▶ Additional information that may be useful includes the program that called the program that abended.
- ▶ Additionally, it may be of value to determine what data was attempting to be written or read at the time of the abend.

Standard save area mapping and use

Standard save areas are a convention that provides linkage between called programs. By using standard save areas, you can find the caller of a program or subroutine as well as the values of the registers at the time of the call.

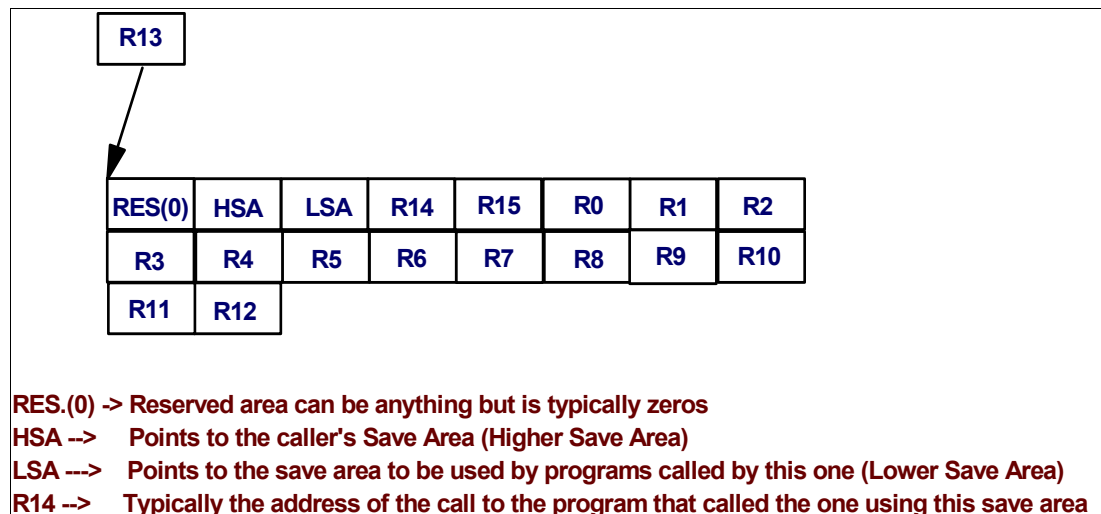


Figure B-5 Save area layout

Diagnosing an ABEND0C4

Lab exercise #3:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to ITSO.S2822.DUMP7.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose an ABEND0C4 ABEND dump.

Questions:

1. Using the IP SYSTRACE ALL command and issuing a F '*RCVY', what PSW address was the preceding PGM 004 (ABEND0C4) taken at? _____
2. Use the IP ST REGS command to get the relevant information about the ABEND 0C4 (ignore and page past any error messages). Record the following:
 - PSW _____
 - R3 _____
 - R4 _____
 - R13 _____
 - R14 _____
 - Primary ASID (PASN) _____
3. Abending JOBNAME _____
4. Use the =1 command to get into IPCS browse. Browse the PSW address and back up 4 bytes (L x-4 when looking at the PSW). What instruction does the PSW point to? _____ (record the 4 bytes of hex data)
 - The above instruction represents a ST 3,20(,4)
 - What caused the ABEND0C4 _____
5. What value was to be stored _____
6. Determine the name of the abending module by backing up with PF7 and record the eyecatcher you find there _____
7. What is the offset of the ABEND0C4 from the above program (assume the program beginning is the 47F0): _____
8. Based on the above the ABEND 0C4 occurs because R4 is bad. Use R14 value to determine what program called this one (browse the storage in R14 and back up to find the eyecatcher; record it here): _____
9. See Figure B-5 on page 291 for a description of standard save areas. Based on the failing instruction above, find what value R3 had on entry to S0C4RTN (the instruction prior to the abend was SR 3,3, which zeroes it). Use the R13 value you recorded above to map the standard save area.
 - RESERVE HSA
 - _____
10. In this case the program has set up R13 to point to a save area that will be used to call another program. Find the higher save area pointed to by the HSA value above. Browse that address and record the value of R3 and R4 from our caller's save area. These are R3 and R4 at the time of the call to S0C4RTN: _____

Answers to questions: See "Lab exercise #3 - Answers diagnosing an ABEND0C4" on page 303.

Diagnosing ABEND138 errors

Some types of abends are issued because the caller of the system service provided bad input. When you encounter such an abend, the key to diagnosing it is to find the issuer of that abend as well as the parameters passed on. This exercise demonstrates how to do both.

Essentially the steps are to find the PSW where the SVC was issued from. You can do this from by using the following:

- ▶ Use the SYSTRACE to find the ABEND and then backing up to the corresponding SVC entry in the SYSTRACE and extracting the PSW address from there.
- ▶ The RBOPSW will also point to the issuing PSW. The RB of interest will likely have a WLIC value that contains the SVC number. For instance, WLIC of 00010038 would be an ENQUEUE.
- ▶ Once the PSW has been obtained then you can use the same methods to obtain registers at time of issuance.
 - In the SYSTRACE, the SVC entry will have R15, R0, and R1 recorded.
 - In the RBs, the registers at the time of the SVC will be contained in the RB (or SVRB, PRB, or IRB) that follows the one with the OPSW and WLIC values. If there is no following RB then the registers can be found in the TCB.
- ▶ Once the PSW and registers have been found, browse the storage at the PSW to find the issuer of the SVC. Also use the *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085 to map the parameter list passed to the SVC.

Lab exercise #4:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to ITSO.S2822.DUMP8.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose an ABEND138 ABEND dump.

Questions:

1. Issue the IP SUMM FORMAT command to format the failing ASID. Max to the bottom using PF8. What is the TCB address with a completion code of 138?

2. An ABEND138 indicates that a second ENQUEUE was issued for the same resource twice without an intervening DEQUEUE. Determine the resource name that caused the abend, as follows:
 - Find the TCB above with the command F 'TCB: NNNNNNNN' PREV (address must be 8 digits). Then find the first active RB with F 'ACTIVE'. This should be a PRB. Find and record the WLIC value _____ and the OPSW _____

 - The WLIC value represents the last interrupt that occurred on that RB (in this case an ENQUEUE), while the OPSW represents the address where it was issued from.
 - To find the resource name requested on the ENQUEUE, the parameter list must be found. To accomplish this, find R1 from the next RB (an SVRB) and record the value _____

 - The registers associated with the RB are saved in the next RB. If there is not a next RB then those registers will be saved in the TCB.
 - Go to IPCS browse and locate the R1 value you recorded above and record the first 12 bytes here _____
3. The ABEND138 occurred because the ENQUEUE above was the second such ENQUEUE for the resource without a DEQUEUE in between. Issue IP SYSTRACE and max to the bottom of the output. Issue F 'SVC 3' PREV (use 4 spaces between the C and the 3). This is the failing SVC 38 request. Note the PSW address and the R1 values _____

- Note that the format of SYSTRACE entries is SVC#, 2nd word of PSW, R15, R0, and R1.
- Notice these values are the same as those obtained from the RBs above.
4. Using the F command above, look at the preceding ENQUEUEs and DEQUEUEs (SVC 30) to find a preceding ENQUEUE or DEQUEUE for the same resource. What are the PSW and R1 values for the preceding entry with a matching resource name?

5. You will have to look through a couple of entries. The parmlist pointed to by a DEQUEUE has the resource names in the same places as the ENQUEUE parmlist. Hint: The parmlist address you are interested in will be in the same general area as R1 above.
 - Record the name of the module that issued the ENQUEUEs _____

Answers to questions: See “Lab exercise #4 - Answers diagnosing an ABEND138” on page 304.

Diagnosing storage problems - ABEND878

To diagnose storage problems with a dump, it is best to use the VERBX VSMDATA 'SUMMARY' command in IPCS. There is a wealth of information about the output of this command. Chapter 29 of *z/OS MVS Diagnosis: Reference*, GA22-7588 provides details.

In general the approach is to determine whether this is a common or local storage problem. The exercise that follows details a common storage shortage problem. The steps for diagnosing a local (ASID) storage problem are similar.

To answer the questions, question # 2, for this abend, use the following information.

SSRV trace entries

For virtual storage management, use the following information:

- For SSRV 132 (Storage Obtain)
- SSRV 133 (Storage Release)

SSRV requests for VSM

For an SSRV request to virtual storage management, the data is:

Under UNIQUE-1: Information input to the VSM storage service, the bytes are as follows:

0	Flags:
	X... RESERVED
	.1.. KEY was specified
	..1. AR 15 is in use
	..0. AR 15 is not in use
	...1 LOC=(nnn,64) was specified. Storage can be backed above the bar
 1... CHECKZERO=YES was specified
 0... CHECKZERO=NO was specified explicitly, or by default
1.. TCBADDR was specified on STORAGE OBTAIN or RELEASE
00 OWNER=HOME was specified explicitly, or by default
01 OWNER=PRIMARY was specified
10 OWNER=SECONDARY was specified
11 OWNER=SYSTEM was specified
1	Storage key (bits 8 through 11)
2	Subpool number
3	Request flags:
	1... ALET operand specified
	.1.. Storage can be backed anywhere
	..00 Storage must have callers residency
	..01 Storage must have a 24-bit address
	..10 The request is for an explicit address
	..11 Storage can have a 24- or 31-bit address
 1... Maximum and minimum request
1.. Storage must be on a page boundary
1. Unconditional request
0 OBTAIN request
1 FREEMAIN request

SSRV storage size

Under UNIQUE-2, the following information is needed for question # 2:

In an SSRV trace entry for a VSM STORAGE OBTAIN or GETMAIN, one of the following:

- The length of the storage successfully obtained
- The minimum storage requested, if the storage was not obtained

ABEND878 - finding the request

Lab exercise #5:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNNAME to ITSO.S2895.DUMP4.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose an ABEND878 ABEND dump.

Questions:

1. Since this dump was captured by the system, use the ST FAILDATA command to find the abending request. What was the abend code _____ and Reason code _____
2. Issue the IP SYSTRACE command and then F '*S' to find the failing SVC. The failing request is the SSRV 132 above it. Use the mappings provided in *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085 (SSRV trace entries) to fill in the following information and for this exercise, check the SSRV trace entries above.
 - What request does SSRV 132 represent? _____
 - Note: had this request been an SVC entered, the mapping you need to use is found in *z/OS MVS Diagnosis: Reference*, GA22-7588 under SVC 10 (0A0A) or SVC 132 (0A78).
 - What was the PSW address of the request? _____
 - Note: if it was PC entered as this one was you will need to get the PSW address from the PC entry, which in this case is a 30B (storage obtain) and use the information provided above.
 - What subpool was requested? _____
 - Was storage requested above or below the line? _____
 - What was the size requested for the storage? _____
3. Looking backward in the system trace, is there an apparent pattern? To do this, issue F '132' prev . _____

Answers to questions: See “Lab exercise #5 - Answers diagnosing storage - ABEND878” on page 305.

ABEND878 - analyzing storage use

Using the same dump, issue the VERBX VSMDATA 'SUMMARY' command.

Questions:

1. Issue the F 'GLOBAL DATA' command. Using the table found, fill in the following information from the Global Data Area:

- SIZE OF:
- CSA _____
- SQA _____
- ECSA _____
- ESQA _____
- Was any of CSA or ECSA converted to SQA in this dump? _____

If large amounts of CSA have been converted to SQA, suspect an SQA problem.

2. Use the F 'CSA TOTAL' command to find the total current usage of CSA/ECSA (note: CSA is the lower number and ECSA is the upper number). Use SQA Total to get the SQA information and fill in the information below:

- Current usage of:
- CSA _____
- SQA _____

3. SQA can overflow into CSA. Since this did not happen, we assume that this is a CSA problem (only one page of CSA is left). Find the CSA total line again with the F 'CSA TOTAL' command. Prior to this line is the information for each subpool in CSA. Use the F '*****' PREV command and PF5 to find the CSA subpool with the largest amount of storage below the line. Stop looking when you get to the SQA total line. Fill in the following line:

- ***** SUBPOOL _____, KEY ____ TOTAL ALLOC: 0052C000 (_____ BELOW, 00098000 ABOVE)

4. Does the Subpool number here match that found on the previous page? _____

Answers to questions: See "Lab exercise #5 - Answers ABEND878 - Analyzing storage use" on page 306.

ABEND878 - CSA/SQA tracker

Enter the VERBEXIT VSMDATA OWNCOMM command to display information about jobs or address spaces that hold storage in the common service area (CSA), extended CSA, system queue area (SQA), or extended SQA. The dump being analyzed with VERBEXIT VSMDATA OWNCOMM must contain the SQA and ESQA subpools. If you use the SDUMP or SDUMPX macro or the DUMP command to obtain the dump, make sure to specify the SQA option of the SDATA parameter.

Enter the VERBEXIT VSMDATA 'OWNCOMM DETAIL' command to obtain a report that displays a list of storage ranges owned by one or more jobs.

Lab exercise #5:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNNAME to ITSO.S2895.DUMP2.
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose an ABEND878 ABEND dump.

Questions:

Use *z/OS MVS Diagnosis: Reference*, GA22-7588, which describes the output of the VERBX VSMDATA OWNCOMM command to complete this exercise.

1. . Using the same dump as on the previous page, issue the IP VERBX VSMDATA 'OWNCOMM SUMMARY' command
 - What jobname consumed the most SQA? _____
 - How much SQA was allocated to that jobname? _____
2. Issue the IPVERBX VSMDATA 'OWNCOMM DETAIL ASIDLIST(32)' command. Answer the following questions about the storage:
 - What jobname allocated this storage? _____
 - What was the length of the storage requested? _____
 - What was the return_address of the storage request? _____
 - What were the first 16 bytes of the storage area in question?

 - Is there an obvious pattern here? _____
3. Since private storage was not dumped in this dump, it will not be possible to browse storage to look for an eyecatcher for the program represented by the return address. However, use the IP W command (IP W return_address) to determine what the name of the program was that issued this request: _____

Answers to questions: See “Lab exercise #5 - Answers ABEND878 - CSA/SQA tracker” on page 306.

Diagnosing local storage shortage

This exercise will abbreviate the process by:

- ▶ Understanding the failing request
- ▶ Getting a picture of current local storage usage
- ▶ Using that picture to evaluate where (high private or user region) the problem lies.
- ▶ Using VSM control blocks to specifically identify the problem pattern
- ▶ Using IPCS tools to identify the problem program

Lab exercise #6:

- ▶ Switch dumps by typing =0 (zero) on the IPCS command line.
- ▶ Change the DSNAME to **ITSO.S2822.DUMP9**
- ▶ Press Enter and proceed back to IPCS Option 6 (commands) by typing =6 on the command line. Proceed with the exercise.
- ▶ The Problem: Diagnose local storage shortages.

SSRV trace entries

For this exercise, in the SYSTRACE, use the following information. An example of a SYSTRACE entry is shown in Figure A-19 on page 261 and Figure A-20 on page 261.

Under UNIQUE-1:

- ▶ Byte 2:
Contains the subpool number.
- ▶ Byte 3 Request flags:

1...	ALET operand specified
.1..	Storage can be backed anywhere
..00	Storage must have callers residency
..01	Storage must have a 24-bit address
..10	The request is for an explicit address
..11	Storage can have a 24- or 31-bit address
....	1..	Maximum and minimum request
....	.1..	Storage must be on a page boundary
....	..1.	Unconditional request
....	...0	OBTAIN request
....	...1	FREEMAIN request

Under UNIQUE-2:

- ▶ In an SSRV trace entry for a VSM STORAGE OBTAIN or GETMAIN, one of the following:
 - The length of the storage successfully obtained
 - The minimum storage requested, if the storage was not obtained

Under UNIQUE-3:

- ▶ In an SSRV trace entry for a VSM STORAGE OBTAIN or GETMAIN, one of the following:
 - The address of the storage successfully obtained, if you specified address; otherwise, zero.

- The maximum storage requested, if the storage was not obtained
- ▶ In an SSRV trace entry for a VSM STORAGE RELEASE or FREEMAIN:
The address of the storage to be released.

Under UNIQUE-4:

- ▶ Left 2 bytes: ASID of the target address space
- ▶ Next byte4: Reserved
- ▶ Right byte:

If the GETMAIN/FREEMAIN/STORAGE OBTAIN/STORAGE RELEASE is unconditional, an abend will be issued and the SSRV trace entry 3rd byte of UNIQUE-4 will contain X'FF'. If the GETMAIN/FREEMAIN/STORAGE OBTAIN/STORAGE RELEASE is conditional, no abend will be issued and the SSRV trace entry 3rd byte of UNIQUE4 will contain the actual return code from the storage service.

Questions:

1. Issue the IP SYSTRACE ALL followed by the F *SVC command to find the SVC D request for this error. Back up a couple of lines with the UP 5 command. Use the mappings provided in *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085 (SSRV trace entries) to fill in the following information:
 - What was the ASID where the failure occurred? _____
 - What was the size requested of the failing GETMAIN? _____
 - Does this seem excessive? _____
 - What was the requested subpool? _____
 - Based on the Subpool requested, is this a global or local problem? _____
 - SP 0-127 are low private (Region) subpools.
2. Issue the IP VERBX VSMDATA 'SUMMARY NOG ASID(101)' command, go to the bottom of this output and find the local storage map. Fill in the following values from the map:
 - _____ <- Max Ext. User Region Address
 - _____ <- Ext. User Region Top
 - _____ <- Ext. User Region Start
3. Extended private storage grows down until it reaches the current top of region; subsequent local storage may then fail as a result. Based on the storage map, did this happen? _____
4. The user region grows up until the current top of the region approaches the maximum user region. Subsequent region requests that would push the current top of the region over the max will fail. Did this occur in this case? _____
5. At this point we can assume that the problem is with the user region. This is not always as obvious when REGION and PRIVATE storage “collide.” To determine whether the problem is that the user region is exhausted or whether instead it is somehow fragmented, look for FBQEs that describe storage in the USER REGION range if there are any.
 - _____
 - Does this suggest fragmentation or storage exhaustion? _____

Questions (Continued):

6. Find a pattern in the user region subpools. Look at the Local Subpool Summary near the bottom of the report. What subpool and key has the largest storage allocation: SP: _____ Key: _____
7. Look for a pattern in the subpool found previously. Issue F ***** PREV until you find the subpool summary line for the subpool of interest. Page back and see if there is a pattern. Based on what you see what would be the size of the problematic GETMAIN?

8. Pick any one of the addresses you browsed in the previous question and record the eyecatcher that you find: Hint: Use address 273F3000; did you find it?
— _____
9. Go back to SYSTRACE ALL and determine the PSW address where the GETMAIN was issued from. Browse that storage and record the eyecatcher of the offending module:
► _____
10. Use the SUMMARY FORMAT ASID(X'1D') command to find the EP name (under the RB that took the abend). Max to the bottom using PF8. Select the TCB address with a completion code. Find the TCB above with the command F 'TCB: NNNNNNNN' PREV (address must be 8 digits). Then find the first active RB with F 'ACTIVE'. What is the EP..... name under the RB. _____

Answers to questions: See “Lab exercise #6 - Answers diagnosing local storage shortages” on page 307.

Lab exercise #1 - Answers IP ST REGS

The following questions can all be answered by using the IP ST REGS command.

1. Was this dump in AR mode at the time of the failure? __NO__
2. What was the failing PSW address? _8141359C__
3. What ASID is this failing code executing in? _20__
4. What was the failing TCB address? _8FF2A0__
5. What is the value in R14 _A5400F74__

Lab exercise #1 - Answers IP SYSTRACE

1. By using IP SYSTRACE ALL and looking in the output for the word WAIT, you can determine whether all CPUs were busy at the time of the dump. In this dump, were all CPUs busy? **YES/NO** (highlighted)
2. EXT 1005 entries and CLCK entries are indicative of possible loops. Use the FIND command in the output to see if there are any EXT or CLCK entries.
 - What address spaces had EXT 1005 entries (hint: F 'EXT ')? _1, 9, and 20__
 - A loop would most likely be indicated by EXT entries with the same PSW addresses over and over. Do the address spaces you found above appear to be in a loop?
__NO__
3. Use IP SYSTRACE TIME(LOCAL) ASID(x'20') and determine what the last time stamp in the trace is. Use the BOTTOM command; you may need to scroll right with PF11.
__18:58:04.459546__
4. You saw previously from the IP CBF RTCT command that ASID X'20' was dumped. What is the last TCB that was active in the trace table for this ASID? You can use the IP SYSTRACE ASID(X'20') TIME(LOCAL) command and then bottom to find this out.
__008FF2A0__
5. Sometimes it is useful to look for abends that show up in SYSTRACE in the output from above. Use the F '*P' PREV command to find the last *PGM 001 entry that shows the 0C1. What was the PSW address that the 0C1 occurred at from looking at the *PGM 001 entry?
__A47BE8BA__
6. Issue F 'B ' PREV command to find the SVC B entry and repeat find once to get past the SVCR. Use the mapping below to answer the following questions:
 - What address will the time be returned to? __00D392C8__
 - What format will it be returned in? __Elapsed time in hundredths of a second__

Lab exercise #1 - Answers Summary Format

1. Use IP SUMM FORMAT ASID(X'20') followed by the BOTTOM command. Looking at the TCB summary, what is the TCB address that ended in a non-zero completion code:
__008FF2A0__
2. Use the F 'TCB: 00' PREV command to find the TCB that took the ABEND0C1, then issue F 'ACTIVE' to find the top RB.
 - From that RB what are the values of OPSW __070C1000 8141359C__
 - and the WLIC value __0002000D__

- What is the value in R14 at the time the SVC 1 was issued? (Hint: obtain R14 from the following SVRB since the registers at the time the SVC was entered can be found in the following RB:) __A5400F74__
- Browse the storage at the PSW above. What is the offset into the “Module” above assuming that the “I” at the beginning of the name represents the start of the module. __x'95C'__ Note: The module name is IGVVSEND.

Lab exercise #2 - Answers diagnosing an ABEND0C1

1. Determine what this dump is all about: Issue the IP LIST TITLE command. From the output is there any indication that this dump was the result of an abend? What abend? __ABEND0C1__.

Normally we could confirm this with IP ST WORKSHEET. However in this case the dump was captured by adding a SYSMDUMP DD card in the job, so no SDWA was captured so in this case ST WORKSHEET doesn't help.
2. Using IP SYSTRACE ALL command and issuing a F '*PGM', what PSW address was the PGM 001 (a.k.a. ABEND0C1) taken at? __ 078D0000 00007FD2__
3. Fill in the abend code in the *RCVY entry below based on the *RCVY entry that immediately follows the *PGM 001:
 - *RCVY PROG 94 _0_ _C_ _1_ 000 (fill in the 3 missing characters)
4. Use the IP ST REGS command to get the relevant information about the abend 0C1. Record the following:
 - PSW __078D0000 00007FD2__
 - R14 __80007F5C__
 - Primary ASID (PASN) __1D__
 - ABENDING JOBNAME __BADPROG1__
5. Use the =1 command to get into IPCS browse:
 - Browse the PSW address what “instruction” does the PSW point to? __x'00'__
6. Often branches are accomplished with BALR 14,15 making R14 often point to the caller. Check R14 in this dump and see what the instruction before R14 is: Browse the address in R14 and record the 4-byte instruction below: __45E0F060__
7. The instruction above was written in Assembler as BAL 14,LALALAND. What program issued this bad branch? Use PF7 to locate an eyecatcher and record what you find: __BAD_BRANCH 2/16/2006 BAD BRANCH PROGRAM FOR S2823__
8. What is the offset from the beginning of the program (assume the 47F0 instruction to branch around the program is the beginning of the program) of the BAL 14,LALALAND found above __x'34'__

Lab exercise #3 - Answers diagnosing an ABEND0C4

1. Using IP SYSTRACE ALL command and issuing a F '*RCVY', what PSW address was the preceding PGM 004 (a.k.a. ABEND0C4) taken at? __078D0000 00007F60__
2. Use IP ST REGS command to get the relevant information about the abend 0C4 (ignore and page past any error messages). Record the following:
 - PSW __078D0000 00007F60__

- R3 ____ 00000000 ____
 - R4 ____ 00000000 ____
 - R13 ____ 00007F64 ____
 - R14 ____ 00007F36 ____
 - Primary ASID (PASN) ____ x'1D' ____
3. ABENDING JOBNAME ____ ABEND0C4 ____
 4. Use the =1 command to get into IPCS browse: Browse the PSW address and back up 4 bytes (L x-4 when looking at the PSW); what instruction does the PSW point to? ____ 50304014 ____ (record the 4 bytes of hex data)
 - The above instruction represents an ST 3,20(,4).
 - What caused the ABEND0C4 ____ Attempt to store into address x'14' (20 off R4 = 0) ____
 - What value was to be stored ____ 0 (Contents of R3) ____
 5. Determine the name of the abending module by backing up with PF7 and record the eyecatcher you find here ____ S0C4RTN 03206 UWXXXX ____
 6. What is the offset of the ABEND0C4 from the above program (assume the program beginning is the 47F0): ____ x'2C' ____
 7. Based on the above, the ABEND 0C4 occurs because R4 is bad. Use the R14 value to determine what program called this one (browse the storage in R14 and back up to find the eyecatcher; record it here): _BADPROG 2/16/2006 DEVELOPED FOR SHARE ____
 8. See Figure B-5 on page 291 for a description of Standard save areas. Based on the failing instruction above, find what value R3 had on entry to S0C4RTN (the instruction prior to the abend was SR 3,3, which zeroes it). Use R13 to map the standard save areas. Browse the storage at R13 and fill in the following:
 - RESERVE HSA
 - ____ 0 ____ 6F60 ____
 9. In this case the program has set up R13 to point to a save area that will be used to call another program. We need to find the higher save area pointed to by the HSA value above. Browse that address and record the value of R3 and R4 from our caller's save area. These are R3 and R4 at the time of the call to S0C4RTN: _F2F8F2F2
 ____ R3=2822 ____ R4=0 ____

Lab exercise #4 - Answers diagnosing an ABEND138

1. Issue the IP SUMM FORMAT command to format the failing ASID. Max to the bottom using PF8. What is the TCB address with a completion code of 138? ____ 008D1888 ____
2. An ABEND138 indicates that a second ENQUEUE was issued for the same resource twice without an intervening DEQUEUE. Determine the resource name that caused the abend, as follows:
 - Find the TCB above with the command F 'TCB: NNNNNNNN' PREV. Then find the first active RB with F 'ACTIVE'. This should be a PRB. Find and record the WLIC value ____ 00020038 ____ and the OPSW ____ 078D0000 00007F7E ____
 - The WLIC value represents the last interrupt that occurred on that RB (in this case an ENQUEUE), while the OPSW represents the address where it was issued from.

- To find the resource name requested on the ENQUEUE, the parameter list must be found. To accomplish this, find R1 from the next RB (an SVRB) and record the value _80007F70_
The registers associated with the RB are saved in the next RB. If there is not a next RB, then those registers will be saved in the TCB.
 - Go to IPCS browse and locate the R1 value you recorded above. Record the first 12 bytes here ___C0084800___ ___00007FEC___ ___00007FF4___
3. The ABEND138 occurred because the ENQUEUE above was the second such ENQUEUE for the resource without a DEQUEUE in between. Issue IP SYSTRACE ALL and max to the bottom of the output. Issue F 'SVC 3' PREV (note: use 5 spaces between the C and the 3). This is the failing SVC 38 request. Note the PSW address and the R1 values ___078D0000 00007E46 _ ___80007E38_____
The format of SYSTRACE entries is SVC#, 2nd word of PSW, R15, R0, and R1.
Notice that these values are the same as those obtained from the RBs above.
4. Using the FIND command above look at the preceding ENQUEUEs and DEQUEUEs (SVC 30) to find a preceding ENQUEUE or DEQUEUE for the same resource. What are the PSW and R1 values for the preceding entry with a matching resource name?
For an SVC entry, the UNIQUE entries are as follows:
UNIQUE-1/UNIQUE-2/UNIQUE-3
gpr15--- gpr0---- gpr1----: General registers 15, 0, and 1
? ___078D0000 00007F7E ___80007F70___
5. You will have to look through a couple of entries. The parmlist pointed to by a DEQUEUE has the resource names in the same places as the ENQUEUE parmlist. Hint: The parmlist address you are interested in will be in the same general area as R1 above.
- Record the name of the module that issued the ENQUEUEs _____
GENS013802/17/06 _____

Lab exercise #5 - Answers diagnosing storage - ABEND878

1. Since this dump was captured by the system, use the ST FAILDATA command to find the abending request. What was the abend code _878_ and Reason code ___08___
2. Issue the IP SYSTRACE ALL command and then F '*S' to find the failing SVC. The failing request is the SSRV 132 above it.
 - What request does SSRV 132 represent? _____STORAGE OBTAIN_____
 - Note: had this request been SVC entered, the mapping we would have used would be found in z/OS Diagnosis: Reference, GA22-7588 under SVC 10 (0A0A) or SVC 132 (0A78)
 - What was the PSW address of the request? ___070C0000 A5400F42 _____
 - Note that if it was PC-entered as this one was, you will need to get the PSW address from the PC entry, which in this case is a 30B (storage obtain)
 - What subpool was requested? ___x'F1'___ Decimal 241 _____
 - Was storage requested above or below the line? _____Below_____
 - What was the size requested for the storage? _____00001000 _____
3. Looking backward in the system trace is there an apparent pattern? ___Yes repeated GETMAINS for same length_____

Lab exercise #5 - Answers ABEND878 - Analyzing storage use

Using the same dump as on the previous page. issue the VERBX VSMDATA 'SUMMARY' command.

1. Issue the F 'GLOBAL DATA' command. Using the table found, fill in the following information from the Global Data Area:

- SIZE OF:
 - CSA _____2AF000_____
 - SQA _____263000_____
 - ECSA _____1F4A1000_____
 - ESQA _____D88000_____
- Was any of CSA or ECSA converted to SQA in this dump? __YES__

If large amounts of CSA have been converted to SQA, suspect an SQA problem.

2. Use the F 'CSA TOTAL' command to find the total current usage of CSA/ECSA (note CSA is the below number and ECSA is the above number). Use SQA Total to get the SQA information and fill in the information below:

Current usage of:

- CSA _____D9B000_____
- SQA _____1001000_____

3. SQA can overflow into CSA. Since this did happen, we assume that this is a CSA problem (only one 4k page of CSA is left). Find the CSA total line again with F 'CSA TOTAL'. Prior to this line is the information for each subpool in CSA. Use the F '*****' PREV command and PF5 to find the CSA subpool with the largest amount of storage below the line. Stop looking when you get to the SQA total line. Fill in the following line:

- ***** SUBPOOL _241_, KEY _0_ TOTAL ALLOC: 0056F000 (_14000_ BELOW, 3B3000 ABOVE)

4. Does the Subpool number here match that found on the previous questions?
__Yes__

Lab exercise #5 - Answers ABEND878 - CSA/SQA tracker

Use the output of the VERBX VSMDATA 'OWNCOMM' command to complete this exercise.

1. Using the same dump as on the previous page, issue the IP VERBX VSMDATA 'OWNCOMM SUMMARY' command.

- What jobname consumed the most CSA? __S2895B__
- How much CSA was allocated to that jobname? __270000__

2. Issue the IPVERBX VSMDATA 'OWNCOMM DETAIL ASIDLIST(32)' command. Answer the following questions about the storage:

- What jobname allocated this storage? __S2895B__
- What was the length of the storage requested? __1000__
- What was the return address of the storage request? __25400F6C__
- What were the first 16 bytes of the storage area in question? __DUMBPRG2 - EATS COMMON STORAGE__
- Is there an obvious pattern here? __YES__

3. Since private storage was not dumped in this dump, it will not be possible to browse storage to look for an eyecatcher for the program represented by the return address. However, use the IP W command (IP W return address) to determine what the name of the program was that issued this request__ASID(X'0020') 25400F6C. STRGHOG2+1C IN EXTENDED PRIVATE_____ and _____ASID(X'0020') 25400F6C. AREA(Subpool251Key08)+0F3C IN EXTENDED PRIVATE ____

Lab exercise #6 - Answers diagnosing local storage shortages

1. Issue IP SYSTRACE ALL followed by the F *SVC command to find the SVC D request for this error. Back up a couple of lines with the UP 5 command. Use the mappings provided in z/OS MVS Diagnosis: Tools and Service Aids, SY28-1085 (SSRV trace entries) to fill in the following information.
 - What was the ASID where the failure occurred? ____1D_____
 - What was the size requested of the failing GETMAIN? ____C8_____
 - Does this seem excessive? ____NO_____
 - What was the requested subpool? ____6_____
 - Based on the Subpool requested, is this a global or local problem? __Local____
SP 0-127 are low private (Region) subpools.
2. Issue IP VERBX VSMDATA 'SUMMARY NOG ASID(29)' go to the bottom of this output and find the local storage map. Fill in the following values from the map:
 - ____27400000_____ <- Max Ext. User Region Address
 - ____273FF000_____ <- Ext. User Region Top
 - ____25400000_____ <- Ext. User Region Start
3. Extended private storage grows down until it reaches the current top of the region and subsequent local storage may then fail as a result. Based on the storage map, did this happen? __NO__
4. The user region grows up until the current top of the region approaches the Max user region. Subsequent region requests that would push the current top of the region over the Max will fail. Did this occur in this case? ____YES____
5. At this point we can assume that the problem is with the user region. This isn't always as obvious when REGION and PRIVATE Storage “collide”. To determine if the problem is that the user region is exhausted, or if instead it is somehow fragmented, look for FBQEs that describe storage in the USER REGION range; are there any?
 - ____NO_____
 - Does this suggest fragmentation or storage exhaustion? ____Exhaustion_____
6. Find a pattern in the user region subpools. Look at the Local Subpool Summary near the bottom of the report. What subpool and key has the largest storage allocation:
SP: ____6____ Key: ____8____
7. Now we look for a pattern in the subpool found previously. Issue F ***** PREV until you find the subpool summary line for the subpool of interest. Page back and see whether there is a pattern. Based on what you see, what would be the size of the problematic GETMAIN? ____5FB8_____
8. Pick any one of the addresses to browse and record the eyecatcher that you find:
 - ____DUMBPRG2 - EATS LOCAL STORAGE_____

9. Go back to SYSTRACE ALL and determine the PSW address where the GETMAIN was issued from. Browse that storage and record the eyecatcher of the offending module:
 - __This module actually doesn't have an eyecatcher__
10. Use the SUMMARY FORMAT ASID(X'1D') command to find the EP name (under the RB: that took the abend). Max to the bottom using PF8. Select the TCB address with a completion code. Find the TCB above with the command F 'TCB: NNNNNNNN' PREV (address must be 8 digits). Then find the first active RB with F 'ACTIVE'. What is the EP..... name under the RB? _____STRGHOG1_____



z/OS trace processing data

This appendix contains trace processing data information and information related to z/OS trace capabilities.

The trace output data sets must be specific to each instance of GTF and can be defined in the cataloged procedure. Each instance of GTF to be started must have a separate cataloged procedure, or if the same cataloged procedure is used, then a different trace data set must be supplied with the GTF START command.

C.1 GFS trace information

GFS trace is a diagnostic tool that collects information about the use of the GETMAIN, FREEMAIN, or STORAGE macro. You can use GFS trace to analyze the allocation of virtual storage and identify users of large amounts of virtual storage.

You must use the generalized trace facility (GTF) to get the GFS trace data output.

C.1.1 DIAGxx parmlib member syntax

When creating a DIAGxx parmlib member, enter data only in columns 1 through 71. Do not enter data in columns 72 through 80; the system ignores these columns.

If the system finds a syntax error in DIAGxx, it issues an error message, and then attempts to continue processing the next keyword.

The syntax for the DIAGxx parmlib member is as follows:

```
[VSM TRACE                                     ]
[ {GETFREE(ON)|GET(ON|OFF) FREE(ON|OFF)       } ]
[   [ASID({asid1|asid1-asidx}[,{asid2|asid2-asidx}]...)] ]
[   [DATA(data1[,data2]...)] ]
[   [KEY({key1|key1-keyx}[,{key2|key2-keyx}]...)] ]
[   [LENGTH({len1|len1-lenx}[,{len2|len2-lenx}]...)] ]
[   [SUBPOOL({sub1|sub1-subx}[,{sub2|sub2-subx}]...)] ]
[   [JOBNAME([job1,job2...])] ]
[   [ADDRESS([addr1|addr1-addrx][,addr2-|addr2-addrx..)] ]
[   [LOCREAL(1oc1[,1oc2]...)] ]
[ {GETFREE (OFF)                               } ]
[VSM TRACK                                     ]
[ {CSA (ON|OFF)                                } ]
[ {SQA (ON|OFF)                                } ]
[ {CSA (ON|OFF) SQA (ON|OFF)                   } ]
[VSM CHECKREGIONLOSS(bbb{K|M},aaa{K|M})       ]
```

C.1.2 GFS trace data

When GTF writes trace data in a data set, format and print the trace data with the IPSCS GTFTRACE subcommand.

When GTF writes trace data only in the GTF address space, use a dump to see the data. Request the GTF trace data in the dump through the SDATA=TRT dump option.

Issue the IPSCS GTFTRACE subcommand to format and see the trace in an unformatted dump. An example of formatted GETMAIN/FREEMAIN (GFS) trace data, which is output from IPSCS in GTFTRACE format is shown in Figure C-1, "Output from IPSCS GTFTRACE" on page 311.

```

IPCS
GTFTRACE DA('MY.GTF.TRACE')   USR(F65)
IKJ56650I TIME-03:42:20 PM. CPU-00:00:01 SERVICE-52291 SESSION-00:00:20
BLS18122I Initialization in progress for DSNAME(¢ MY.GTF.TRACE¢ )
IKJ56650I TIME-03:42:21 PM. CPU-00:00:01 SERVICE-54062 SESSION-00:00:20
**** GTFTRACE DISPLAY OPTIONS IN EFFECT ****
USR=SEL

**** GTF DATA COLLECTION OPTIONS IN EFFECT: ****
USRP option

**** GTF TRACING ENVIRONMENT ****
Release: SP6.0.6 FMID: HBB6606 System name: CMN
CPU Model: 9672 Version: FF Serial no. 270067

USRDA F65 ASCB 00FA0800 JOBN MYGTF2
Getmain SVC(120) Cond=Yes
Loc=(Below,Below) Bndry=Db1wd
Return address=849CA064 Asid=001A Jobname=MYGTF2
Subpool=229 Key=0 Asid=001A Jobname=MYGTF2 TCB=008DCA70 Retcode=0
Storage address=008D6768 Length=10392 X¢2898¢
GPR Values
0-3 00002898 00000000 7FFFC918 0B601E88
4-7 01FE3240 008FF830 849CA000 00FA0800
8-11 00000000 00000DE8 049CBFFE 849CA000
12-15 049CAFFF 0B601A9C 00FE9500 0000E510
GMT-01/06/1998 21:15:43.111628 LOC-01/06/1998 21:15:43.1

USRDA F65 ASCB 00FA0800 JOBN MYGTF2
Freemain SVC(120) Cond=No
Return address=8B2D608A Asid=001A Jobname=MYGTF2
Subpool=230 Key=0 Asid=001A Jobname=MYGTF2 TCB=008DCA70 Retcode=0
Storage address=7F73DFF8 Length=8 X¢ 8 ¢
GPR Values
0-3 00000000 7F73DFF8 008D82D8 008D7BC0
4-7 008D8958 008D6B08 008D85C8 0B335000
8-11 00000002 00000000 7F73DFF8 008D862C
12-15 8B2D6044 008D8C98 849D242A 0000E603
GMT-01/06/1998 21:15:43.111984 LOC-01/06/1998 21:15:43.1

```

Figure C-1 Output from IPCS GTFTRACE

C.1.3 IPCS MVS dump component data analysis panel

This panel is displayed by entering Option 2.6 on the IPCS primary option menu panel. You then see displayed the Dump Component Data Analysis panel, bypassing the Analysis of Dump Contents Menu panel. The panel is shown in Figure C-2 on page 312.

To display information, specify “S option name” or enter S to the left of the option desired. Enter ? to the left of an option to display help regarding the component support.

S	Name	Abstract

_	IMSDUMP	IMS analysis
_	IOSCHECK	Active input/output requests
_	IPCSDATA	IPCS control data
_	IPHDR	TCP/IP IP Header Formatter
_	IRLM	IMS Resource Lock Manager analysis
_	JESXCF	JESXCF Address Space Analysis
_	JES2	JES2 analysis
_	JES3	JES3 analysis
_	LEDATA	Language Environment formatter
_	LISTEDT	Format eligible device table
_	LLATRACE	Library Lookaside trace
_	LOGDATA	LOGREC formatter
_	LOGGER	System logger formatter
_	LPAMAP	Map link pack area
_	MERGE	Merge GTF/CTRACE output
_	MMSDATA	MMS control block analysis
_	MTRACE	Master TRACE formatter
_	NUCMAP	Nucleus CSECT Map
_	OAMDATA	OAM Control Block Analysis
_	OMVSDATA	OpenMVS analysis
_	RESOLVER	TCP/IP Resolver Analysis
_	RMMDATA	RMM Control Block Analysis
_	RMMPDA	RMM PDA Trace Analysis
_	RSMDATA	Real storage manager summary
_	SADMPMSG	Format SADMP console messages
_	SKMSG	TCP/IP Streams Message Formatter
_	SMSDATA	SMS control block analysis
_	SMSXDATA	SMSX Control Block Formatter
_	SRMDATA	SRM control block analysis
_	SSIDATA	Subsystem Interface analysis
_	STRDATA	Coupling Facility Structure Data
_	SUMDUMP	Format summary dump data
_	SYMDEF	Static Symbol Table Formatter
_	SYMPTOMS	Format symptoms
_	SYSTRACE	Format system trace
_	TCAMMAP	TCAM control block analysis
_	TCPHDR	TCP/IP TCP Header Formatter
_	TCPIPCS	TCP/IP Analysis
_	TSODATA	TSO analysis
_	UDPHDR	TCP/IP UDP Header Formatter
_	VLFDATA	Virtual Lookaside Facility data
_	VLFTRACE	Virtual Lookaside Facility trace
_	VSMDATA	VSM control block analysis
_	VTAMMAP	VTAM control block analysis
_	XESDATA	XES analysis

Figure C-2 Dump Component Data Analysis panel

C.1.4 SUMMARY subcommand parameters

Use the SUMMARY subcommand, shown in Figure C-3 on page 313, to display or print dump data associated with one or more specified address spaces.

SUMMARY produces different diagnostic reports depending on the report type parameter, FORMAT, KEYFIELD, JOBSUMMARY, and TCBSUMMARY, and the address space selection parameters, ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, and JOBLIST. Specify different parameters to selectively display the information you want to see.

```
{ SUMMARY }
{ SUMM    }

----- Report Type Parameters -----

    [ KEYFIELD [REGISTERS | NOREGISTERS] ]
    [ FORMAT ]
    [ TCBSUMMARY ]
    [ JOBSUMMARY ]

----- Address Space Selection Parameters -----

    [ ALL ]
    [ CURRENT ]
    [ ERROR ]
    [ TCBERROR | ANOMALY ]
    [ ASIDLIST(asidlist) ]
    [ JOBLIST(joblist) | JOBNAME(joblist) ]

----- SETDEF-Defined Parameters -----

    [ ACTIVE | MAIN | STORAGE ]
    [ DSNNAME(dsname) | DATASET(dsname) ]
    [ FILE(ddname) | DDNAME(ddname) ]
    [ PATH(path-name) ]

    [ FLAG(severity) ]
    [ PRINT | NOPRINT ]
    [ TERMINAL | NOTERMINAL ]
    [ TEST | NOTEST ]
```

Figure C-3 SUMMARY command parameters

C.1.5 VERBEXIT subcommand

Use the VERBEXIT subcommand, shown in Figure C-4 on page 314, to run an installation-supplied or IBM-supplied verb exit routine.

```

{ VERBEXIT } { pgmname }
{ VERBX     } { verbname }
               [ 'parameter [,parameter]...' ]
               [ AMASK(mask) ]
               [ SYNTAX | NOSYNTAX ]
               [ TOC | NOTOC ]

----- SETDEF-Defined Parameters -----
               [ ACTIVE | MAIN | STORAGE ]
               [ DSNAME(dsname) | DATASET(dsname) ]
               [ FILE(ddname) | DDNAME(ddname) ]
               [ PATH(path-name) ]

               [ PRINT | NOPRINT ]
               [ TERMINAL | NOTERMINAL ]
               [ TEST | NOTEST ]

```

Figure C-4 VERBEXIT subcommand parameters

C.1.6 VERBX VSMDATA subcommand

Specify the VSMDATA verb name and optional parameters on the VERBEXIT subcommand, shown in Figure C-5, to format diagnostic data from virtual storage management (VSM).

```

VERBEXIT VSMDATA [ 'parameter [,parameter]...' ]
The parameters are:
[CONTROLBLOCKS] [ALL] [DETAIL]
                  [SUMMARY]
                  [CURRENT]
                  [ERROR]
                  [TCBERROR]
                  [NOASIDS]
                  [ASIDLIST(asidlist)]
                  [JOBNAME(joblist) | JOBLIST(joblist)]
                  [GLOBAL|NOGLOBAL]
[OWNCOMM [[CSA] [SQA]]]
          [SUMMARY ]
          [DETAIL ]
          [ALL]
          [ASIDLIST(asidlist)]
          [SYSTEM]
          [SORTBY(ASIDADDR|ASIDLEN|ADDRESS|TIME|LENGTH)]
          [CONTENTS(YES|NO)]

```

Figure C-5 VERBX VSMDATA subcommand parameters

C.1.7 STATUS FAILDATA subcommand

Figure C-6 on page 315 and Figure C-7 on page 316 show the output from a STATUS FAILDATA subcommand described in 6.21, “Using IPCS to find the failing instruction” on page 174.

* * * DIAGNOSTIC DATA REPORT * * *
 SEARCH ARGUMENT ABSTRACT

PIDS/5752SC1B6 RIDS/IEFSD060#L RIDS/IEFSD060 AB/S023E PRCS/00000008 REGS/OE0
 REGS/06110 RIDS/IEFIB620#R

Symptom	Description
-----	-----
PIDS/5752SC1B6	Program id: 5752SC1B6
RIDS/IEFSD060#L	Load module name: IEFSD060
RIDS/IEFSD060	Csect name: IEFSD060
AB/S023E	System abend code: 023E
PRCS/00000008	Abend reason code: 00000008
REGS/OE018	Register/PSW difference for R0E: 018
REGS/06110	Register/PSW difference for R06: 110
RIDS/IEFIB620#R	Recovery routine csect name: IEFIB620

OTHER SERVICEABILITY INFORMATION

Recovery Routine Label: IEFIB620
 Date Assembled: 04328
 Module Level: HBB7720
 Subfunction: INITIATOR JOB PROCESS

Time of Error Information

PSW: 070C1000 81329D48 Instruction length: 02 Interrupt code: 000D
Failing instruction text: 00181610 0A0D50E0 D0049180

Registers 0-7

GR: 84000000 8423E000 7FFEFF0C 007FF3D0 007FF448 00000BE0 81329C38 007C9E1C
 AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Registers 8-15

GR: 00000000 0167BFA0 7F775A68 00000B80 00000BD8 007FDB28 81329D30 00000008
 AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000C05

Home ASID: 008A Primary ASID: 008A Secondary ASID: 008A
 PKM: 8040 AX: 0000 EAX: 0000

RTM was entered because an SVC was issued in an improper mode.
 The error occurred while a locked or disabled routine was in control.
 No locks were held.
 No super bits were set.

STATUS FROM THE RB WHICH ESTABLISHED THE ESTAE EXIT

PSW: 070C3000 840F55D8 Instruction length: 02 Interrupt code: 003E

Registers 0-7

GR: 840F48A2 007F0B18 00000020 007FF708 007CA9B0 007FF448 007F0DC4 007EAE1
 AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Registers 8-15

GR: 007F0B18 840F5372 00FC0200 007FF448 840F48A2 007CAA18 840F55C0 00000000
 AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000C05

Figure C-6 Output from a STATUS FAILDATA subcommand

RECOVERY ENVIRONMENT

Recovery routine type: ESTAE recovery routine

Recovery routine entry point: 040EC0F0

The RB associated with this exit was not in control at the time of error.

User requested no I/O processing.

VARIABLE RECORDING AREA (SDWAVRA)

+000	Key: 22	Length: 14			
+002	C9D5C9E3	C9C1E3D6	D940C6D6	D6E3D7D9	INITIATOR FOOTPR
+012	C9D5E3E2				INTS
+016	Key: 23	Length: 50			
+018	C9D5C9E3	C6D7D5E3	80BF00C0	00CCE3F0	INITFPNT...{..TO
+028	E0000EEC	80000080	C000F0C0	EE3FC000	\.....{.0{..{.
+038	00000000	E63C0300	000000E0	E068009FW.....\ ...
+048	EF FE8000	00A0B87F	E3003C00	0017FFFF "T.....
+058	80000000	FF E0A000	00000000	00000000 \.....
+068	Key: 50	Length: 02			
+06A	0023				..
+06C	Key: 53	Length: 00			

IEA24013I FORMATTING COMPLETED SUCCESSFULLY.

Figure C-7 Completion of STATUS FAILDATA output



IPCS commands

This appendix discusses IPCS commands. Use the IPCS primary command to invoke an IPCS subcommand, CLIST, or REXX EXEC from any of the panels of the IPCS dialog. The subcommand, CLIST, or REXX EXEC is entered exactly as though it was being invoked under IPCS inline mode. If the subcommand, CLIST, or REXX EXEC sends a report to the terminal, you view the report using the dump display reporter panel.

Note: Do not use the IPCS primary command to invoke a CLIST that contains a combination of a TSO/E CLIST function, such as SYSOUTTRAP, and an authorized TSO/E command, such as LISTD. Such a CLIST should be invoked only in IPCS line or batch mode or in a TSO/E environment.

D.1 IPCS commands

There are two ways to enter subcommands from the IPCS dialog:

- Choose option 4 (COMMAND) and enter the subcommand on the command line:

```
====> ANALYZE EXCEPTION
```

- Use the IPCS primary command to prefix the subcommand invocation from any command or option line of the IPCS dialog. For example:

```
COMMAND ====> IPCS ANALYZE EXCEPTION
```

```
COMMAND ====> IP ANALYZE EXCEPTION
```

Example D-1 shows a list of IPCS subcommands.

Example: D-1 IPCS commands and subcommands

ip analyze exception	: shows lock contention
ip asmk	: ASM info
ip verbx asmdata	: formats asm cb's (part, parte etc)
ip cbstat @ str(ascb)	: ascb status.
ip cbstat @ str(srb) asid(1)	: SSRB and SRB status.
ip cbstat @ str(tcb) asid(1)	: tcb status from TCBFLGS8.
ip cbstat str(storestatus)	: data about failing NIP RIMs
ip cbf (xxx)	: any CB in the symbol table, see ip listsym)
ip cbf cvt	: Communications Vector Table
ip cbf gda	: +D8=VSM tracker flags 08=CSA on 04=SQA on.
ip cbf lcca1	: first lcca for wait064 rc9 at IPL
ip cbf rtct	: Recov. Term. Cont. Tbl. (asids dumped)
ip cbf rtct str(sdump) view(flags)	: shows SDUMP flag output
ip cbf ucbxxx	
ip cbf @ str(asid ucb)	
ip cbf @. str(tcb)	
ip ctrace comp(syslla) full	: LLA @ space must be in the dump
ip ctrace comp(sysrsm) full	
ip ctrace comp(sysrsm) asid(x'nn') full	
	: abenda78 rc18, 'f defer' for fixed pg.
	: See VSMHELP file for output explanation.
ip ctrace query(sysrsm)	: Start & end times if RSM tracing is on.
ip divdata	: Exception report is the default.
ip divdata detail	: DIB DIBX DOA WCB for curr error asid.
ip divdata exception	: exception
ip divdata summ	: DOA queue summary.
ip divdata trace	: Output of RSM DIV component trace.
ip divdata trace full	: RSM DIV component trace for all ASIDs.
ip divdata trace asid(x'ff')	: output for rsm div comp. trace in asid ff
ip eq labelname @@. str(control block type EX; svrb, rb ascb)	
ip eq labelname @@. l(x'20')	: ex: eq gxl 1F1888E0 L(X'20')
find ' SSRV 78'	
ip gtf svc	: formats SVCs in an SVC trace.
ip gtf svc(6,9)	: formats SVC 6 and 9's in an SVC trace.
ip gtf usr(f65)	: formats get/free trace data in the dump
	see GTF info above
ip gtf usr(f65) startloc(ddd,hh.mm.ss) ddd=date	
ip ioscheck smgrblks	: SMGR blks from IECVEXSM sp226 do a find
on smgr:, then look at nopages, repeat, one smgr blk should have a lot of pages	
(mine had DF), do a find on '0076 ' and asid doing the EXCP is previous to the	

0076, keep track of them and see whom is doing the most, there will be a pattern or type of job doing the majority.

```

ip iosk actvucbs          : active UCB's
ip iosk capture           : max to bott to see captured UCB @'s
ip ipcdata                : IPCS r10+ cmd. See what rel of IPCS used.
ip list @ 1(x'nnn')       : list storage at virt. location x'nnn'
ip list E0. 1(16) block(0) : Partial dump rsn codes per iea611/iea911
                           also mapped by SDRSN control block.
IP L 208?+4 L(2)          : If is a FF (x'C6C6') we are under VM.
                           : PSA+208->PCCA+4 = CPU id (serial number)
ip l cvt+8c?+188?         : lists where IPLINFO get its info from.
ip l cvt+128?+68?         : lists out the PPT. see section above on PPT
ip l cvt+2AC?+8?          : Processor type & model(see also below).
ip l cvt+3e0?             : what CVTRAC pts 2. RACF/ACF/TSS used?
ip l CVT+24c?+2c? 1(4096) : reg6= Fetch work area in a IEWFETCH,10 slip
ip l CVT+4c0?+10 1(4)     : word with: if DFSMS is used (byte 1>0) and
                           3 bytes w/SMS version,release,mod lvl
ip l CVT+42C?+1A 1(26)    : Type & model of processor. See CPU DIAG
                           CVT+42C->HID+1A = CPU related data
                           ** may not be filled in too early in IPL?
ip l CVT+340?+50 L(1)     : If x'80' bit is on we are in LPAR mode.
                           CVT+340->SCCB+50 = byte w/BFY bit
ip l ecvt+150 1(8)        : hardware name, ECVTHDNM, HWNAME
ip l ecvt+158 1(8)        : ECVTLPMN, LPARNAME
ip l ecvt+160 1(8)        : ECVTMNMN, VMUSERID X'40's if not used.
ip l init tcb@.+b4?+15C?+148?: +x'10'=SCT, +x'50'(byte 1)= jobstep#
                           +148= SCT header, SCT starts at +158,
                           +40=step # for restart.
ip list 0 dspname(nnn) nnn : the dataspace name dump directory LD cmd
ip list 0 dspname(nnn) asid(x'nn') nn if asid is other than your own
ip list 0 dspname(nnn) asid(x'nn') len(x'1000')
ip E0. 1(16) block(0)     : partial dump reasons, formats SDRSN cb.
ip list cvt+24c? len(800) : slip sdump buffer PSW & regs when slip hit.
ip l cvt+10C? len(400)    : shows the qmsg area, see iea705I in notes
ip list cvt+7c%+D0?+8?    : V5.2+ shows the ucb lookup @'s.
ip list IEAVESLA          : system resources lock list.
ip list sliptrap          : shows slip trap creating dump if so.
ip list title
ip list PSAn len(4000)    : N is the CPU PSA you want to look at.
ip list sliptrap
ip listsym                : lists all symbols.
ip list ucbxxx             : lists ucb #xxx.
ip listu xxx              : lists ucb storage for device #xxx.
ip listucb xxx            : lists ucb storage for device #xxx.
ip lpamap                 : lists lpa mods and locations
ip rsmdata all             : for all asid's.
ip rsmdata                : see ASKQ RTA000153245 for output explan.
                           - with rsmdata options, asid(x'n'), all, jobname, can be used
ip rsmdata divmap          : divmapped areas within the address space
ip rsmdata summ            : Expanded, In configuration # / 256 =
                           total meg of estor defined to system.<
                           each gig defined requires 8 meg of ESQA
                           storage pg tables to be getmained.
ip rsmdata addrspac { Diag reference chapter 21 RSM explains outputs }
ip rsmdata addrspac all

```

```

ip rsmdata realframe { Diag reference chapter 21 RSM explains outputs }
ip rsmdata realframe ra(nnn) : nn is real rame address of storage.
ip rsmdata virtpage { Diag reference chapter 21 RSM explains outputs }
ip rsmdata virtpage ra(virt @ of pg of storage) asid(x'nnnn')

```

Note: if stat = fref (first reference) then storage was never
referenced (used) and will not be available in the dump
G = is it getmained, K = key of storage,
F = is pg fetch protected, PSW key must match pg key
P = is pg pg protected, pg cannot be storage to

```

ip select : tells which asid dump is taken of...
ip select all : gives all asid assignments
ip stack @. remark('comment') : puts @ on IPCS dump pointers page
ip status : time and error psw
ip status cpu
ip status faildata : format SDWA (none for a slip dump)
ip status faildata cpu reg5
ip status system : IEAVTSDT=scheduled, SVCDUMP=sync.dump
ip status worksheet : SDUMP SDATA tells what storage is dumped.
ip summ format : format out RTM2WA
ip summ format asid(x'nn')
ip summ jobsummary all : lists all CPU and job status info.
ip summ tcberror : format the tcb in error.
ip symdef : Displays symbols symbolics
ip systrace time(local) : verbx systrace if mvs 5.1 or older.
ip systrace exc(br) : excludes branch entries in systrace.
ip tcbexit iecdafmt tcb@. asid(x'nn')
ip verbx cnmipcs 'cpool' : Netview cell pool cellpool stats
ip verbx csvllipc 'stats,lib=SYS1.SCEERUN,member=*,fetched'
: shows dasd and VLF fetch stats for lib
ip verbx DFHPDxxx 'SM=3' : shows CICS storage (x is release ex:410)
: & lvls of cics mods

ip verbx iplstats :
ip verbx ledata : LE envirement data, there heap options.
ip verbx ledata 'all' :
ip verbx ledata 'heap' : f @, tells if user/anywhere/below CB's
ip verbx logdata : formatted sys1.logrec records
ip verbx mtrace :
ip verbx nucmap :
ip verbx sadmpmsg : SAD messages not yet avilable??
ip verbx srmdata :
ip verbx smsdata : DFSMS and PDSE data for the dump
ip verbx smsxdata : See DFSMS Diagnosis Reference
ip verbx smsxdata active : SMSXDATA against active storage.
ip verbx smsxdata 'comp(pml)' : See DFSMS Diag Ref pg378
ip verbx smsxdata 'nog map asid(nn)' : for these and PDSE MVS commands.
ip verbx trace :systrace info for >r520 systems, default: all
ip verbx utrace : shows systrace all for problem inhouse SADs.
ip verbx vtammap : find 'BPCB' each bpcp +2C is size of that
: vtam buffer, large one (>1000) is suspect.
: output same as MVS 'd net,bfruse' command.

ip verbx vsmdata 'noa summ' :
ip verbx vsmdata 'nog summ' :
ip verbx vsmdata 'asidlist(NN) nog' :

```

```

ip verbx vsmdata 'o d' :
ip verbx vsmdata 'o d conte(no) so(time)'
ip verbx vsmdata 'o d sortby(address)' :
ip verbx vsmdata 'o d sortby(asidaddr)':
ip verbx vsmdata 'o d sortby(asidlen)' :
ip verbx vsmdata 'o d sortby(length)' : storage length
ip verbx vsmdata 'o d sortby(time)'
ip verbx vsmdata 'o summ' : lists asids total global storage usage
ip verbx wtlb : unwritten syslog buffer
ip webcount all : inhouse only, produces global web report
ip where @. : shows mod in which @ resides, done via a
               LPDE CB (LPA Dir. Entry) or CDE for a
               local storage address.
ip w structurename : any structure in symbol table EX:ascb1>

```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 324. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *z/OS Diagnostic Data Collection and Analysis*, SG24-7110

Other publications

These publications are also relevant as further information sources:

- ▶ *Environmental Record Editing and Printing Program (EREP) User's Guide*, GC35-0151
- ▶ *z/OS MVS Diagnosis: Tools and Service Aids*, SY28-1085
- ▶ *z/OS MVS Diagnosis: Reference*, GA22-7588
- ▶ *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- ▶ *z/OS XL C/C++ User's Guide*, SC09-4767
- ▶ *z/OS MVS IPCS Commands*, SA22-7594
- ▶ *z/OS MVS System Codes*, SA22-7626
- ▶ *z/OS MVS System Messages, Vol 1 (ABA-AOM)*, SA22-7631
- ▶ *z/OS MVS System Messages, Vol 2 (ARC-ASA)*, SA22-7632
- ▶ *z/OS MVS System Messages, Vol 3 (ASB-BPX)*, SA22-7633
- ▶ *z/OS MVS System Messages, Vol 4 (CBD-DMO)*, SA22-7634
- ▶ *z/OS MVS System Messages, Vol 5 (EDG-GFS)*, SA22-7635
- ▶ *z/OS MVS System Messages, Vol 6 (GOS-IEA)*, SA22-7636
- ▶ *z/OS MVS System Messages, Vol 7 (IEB-IEE)*, SA22-7637
- ▶ *z/OS MVS System Messages, Vol 8 (IEF-IGD)*, SA22-7638
- ▶ *z/OS MVS System Messages, Vol 9 (IGF-IWM)*, SA22-7639
- ▶ *z/OS MVS System Messages, Vol 10 (IXC-IZP)*, SA22-7640
- ▶ *z/OS UNIX System Services Messages and Codes*, SA22-7807
- ▶ *z/OS MVS Data Areas, Volume 1 (ABEP - DALT)*, GA22-7581
- ▶ *z/OS MVS Data Areas, Volume 2 (DCCB - ITZYRETC)*, GA22-7582
- ▶ *z/OS MVS Data Areas, Volume 3 (IVT - RCWK)*, GA22-7583
- ▶ *z/OS MVS Data Areas, Volume 4 (RD - SRRA)*, GA22-7584
- ▶ *z/OS MVS Data Areas, Volume 5 (SSAG - XTLST)*, GA22-7585

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



ABCS of z/OS System Programming Volume 8

(0.5" spine)
0.475" <-> 0.873"
250 <-> 459 pages



Redbooks®

ABCs of z/OS System Programming

Volume 8

Diagnosis fundamentals, IPCS

Dump analysis, problem diagnosis

Diagnostic procedures

The ABCs of z/OS System Programming is an 11-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information you need to start your research into z/OS and related subjects. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection serves as a powerful technical tool.

This publication, Volume 8, shows you how to:

- ▶ Adopt a systematic and thorough approach to dealing with problems and identifying the different types of problems
- ▶ Determine where to look for diagnostic information and how to obtain it
- ▶ Interpret and analyze the diagnostic data collected
- ▶ Escalate problems to the IBM Support Center when necessary
- ▶ Collect and analyze diagnostic data—a dynamic and complex process
- ▶ Identify and document problems, collect and analyze pertinent diagnostic data and obtain help as needed, to speed you on your way to problem resolution

The content of the volumes is as follows:

Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation

Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKST, authorized libraries, SMP/E, Language Environment

Volume 3: Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMSStvs

Volume 4: Communication Server, TCP/IP, and VTAM/E

Volume 5: Base and Parallel Sysplex/E, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex (GDPS/E)

Volume 6: Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries/E firewall technologies, LDAP, and Enterprise identity mapping (EIM)

Volume 7: Printing in a z/OS environment, Infoprint/E Server and Infoprint Central

Volume 8: An introduction to z/OS problem diagnosis

Volume 9: z/OS UNIX System Services

Volume 10: Introduction to z/Architecture, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC

Volume 11: Capacity planning, performance management, WLM, RMF, and SMF

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-6988-00

ISBN 0738486167