Problem Management

Version 1  Release 11
Seventh Edition, October 2009

This edition applies to Version 1 Release 11 of z/OS (5694-A01), and to subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
   Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
• Title and order number of this document
• Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Tables ........................................... vii

Figures ........................................... ix

About this information .............................. xi
Who should use this information .................... xi
How to use this information ......................... xi
Where to find more information ..................... xii
Information updates on the web ...................... xii
The z/OS Basic Skills Information Center .......... xii

Summary of Changes .............................. xiii

Part 1. Problem management overview ................ 1

Chapter 1. Introduction ............................ 3
Overview of problem resolution ...................... 3
Steps for diagnosing problems on z/OS ............... 4
Gathering diagnosis data ............................ 5
Problem categories ................................ 6
Searching problem reporting databases ............... 8
  Extracting problem symptoms and search arguments .................. 8
  Formats for symptoms ................................ 9
  Searching for a known problem ...................... 10
  Steps for searching problem reporting databases ................. 12
Determining the level of z/OS ...................... 13

Chapter 2. Common tools for problem determination ........ 15
Messages .......................................... 15
  BPXMTXT for z/OS UNIX reason codes ............... 16
IBM z/OS Management Facility ...................... 16
IPCS ............................................. 16
Logs ............................................. 17
Traces ........................................... 19
Dumps ........................................... 21
IBM Omegamon for z/OS Management Console .......... 22
AMATERSE for FTP ................................ 22
IBM documentation ................................ 22

Chapter 3. Best practices for large stand-alone dump ........ 25
Planning a multivolume stand-alone dump data set ........ 25
Creating the multivolume SADUMP .................... 26
Defining a dump directory for large stand-alone and SVC dumps .... 26
Preparing the dump for further processing with IPCS COPYDUMP .... 27
Compressing data for faster transmission and analysis .......... 28
Using PUTDOC to transmit the data to IBM ................ 28
Setting up remote access ............................ 28
Testing your stand-alone dump operations .............. 29
Automating the SADMP process ...................... 29
  Sample JCL for post-processing ...................... 29
IBM System Test example ........................... 31

Part 2. Predictive Failure Analysis .................... 33

© Copyright IBM Corp. 2006, 2009
Determining the status of a hung job or subsystem .............................................. 123
  Steps for determining the status of a hung job or subsystem .............................. 123
Determining if a job is waiting for resources ....................................................... 124
  Steps for determining if a job is waiting for resources ..................................... 124
Determining address space dispatchability ............................................................ 124
  Steps for examining address space dispatchability ............................................. 124
Examining the SRB status ....................................................................................... 125
  Steps for examining the SRB status ..................................................................... 125
Examining the TCB status ....................................................................................... 126
  Steps for examining the TCB status ..................................................................... 126
Examining why a job is not running ........................................................................ 128
  Steps for examining why a job is not running ..................................................... 128

Chapter 11. Diagnosing a loop ................................................................................. 129
Overview of a loop ................................................................................................... 129
  Steps for diagnosing a loop ................................................................................. 131
Gathering additional data for a loop ....................................................................... 132
  Steps for gathering loop data ............................................................................. 132
Analyzing the dump to determine the type of loop ................................................. 133
  Step for analyzing the dump for loop type ......................................................... 133
Diagnosing a disabled loop ..................................................................................... 134
  Steps for diagnosing a disabled loop .................................................................. 134
Diagnosing an enabled loop .................................................................................... 135
  Steps for diagnosing an enabled loop .................................................................. 135
Diagnosing an excessive spin (spin loop) ............................................................... 138
  Steps for diagnosing an excessive spin ................................................................ 138
Analyzing a logrec error record ............................................................................. 140
  Steps for analyzing a logrec error record ............................................................ 140
Searching the problem reporting databases ........................................................... 141
  Steps for searching the problem reporting databases ........................................ 141

Chapter 12. Diagnosing an output problem .............................................................. 143
Overview of analyzing output problems .................................................................. 143
  Steps for diagnosing output problems ................................................................. 143
Collecting problem data for an output problem ..................................................... 145
  Step for collecting problem data ....................................................................... 145
Analyzing data set allocation for an output problem .............................................. 145
  Steps for analyzing data set allocation ............................................................... 145
Analyzing the inputs and outputs ......................................................................... 145
  Steps for analyzing the inputs and outputs ......................................................... 146
Analyzing installation exits for an output problem ............................................... 146
  Steps for analyzing installation exits ................................................................. 146
Identifying the program or component ................................................................... 147
  Steps for identifying the program or component ................................................. 147
Searching the problem reporting databases for an output problem ....................... 148
  Step for searching the problem reporting database ............................................ 148
Gathering additional data for output problems ..................................................... 149
  Steps for gathering additional information for output problems ....................... 149
Reporting output problems to IBM ........................................................................ 150

Chapter 13. Diagnosing a performance problem ...................................................... 153
Overview of a performance problem ...................................................................... 153
  Steps for diagnosing a performance problem ..................................................... 153
Collecting data using commands ......................................................................... 154
  Steps for collecting data using DISPLAY .......................................................... 154
  Steps for using JES2 commands to collect data .................................................. 156
Checking for resource contention or loop ............................................................. 157
  Steps for checking resource contention ............................................................. 157
Searching the problem reporting database ........................................................... 158
  Steps for searching the problem reporting databases ........................................ 158
Part 4. Diagnosis reference material ............................................. 163

Chapter 14. Diagnosis information for z/OS base elements and features ........................................ 165

Chapter 15. Reporting problems to IBM ........................................ 169
Software support service checklist .................................................. 169
Automatic problem reporting ......................................................... 171
Invoking IPCS as a background job ................................................ 171
Step for invoking IPCS as a background job .................................. 171

Chapter 16. Problem diagnostic worksheet ................................... 173

Appendix. Accessibility ............................................................... 175
Using assistive technologies ......................................................... 175
Keyboard navigation of the user interface ....................................... 175
z/OS information ....................................................................... 175

Notices ....................................................................................... 177
Policy for unsupported hardware .................................................. 179
Programming Interfaces Information .......................................... 179
Trademarks ................................................................................. 179

Index ......................................................................................... 181
## Tables

1. Problem indicators by type ........................................ 7
2. Obtaining search arguments from SVC dump, stand-alone dump or SYSMDUMP using IPCS commands ........................................ 9
3. Standardized symptom keyword list ......................... 10
4. PFA\_COMMON\_STORAGE\_USAGE check parameters ................................. 50
5. PFA\_LOGREC\_ARRIVAL\_RATE check parameters ................................. 56
6. PFA\_FRAMES\_AND\_SLOTS\_USAGE check parameters ................................. 61
7. PFA\_MESSAGE\_ARRIVAL\_RATE check parameters ................................. 67
8. Summary of IPCS dump subcommands by problem ........... 102
9. Common wait states that occur during IPL ................. 111
10. Common disabled wait states ................................. 116
11. Selecting the problem type for **STATUS CPU** output ........................................ 118
12. Example of using the message prefix to identify the component ........................................ 147
13. Where to find diagnosis information for z/OS base elements and features ................................. 165
14. Severity levels and examples ................................. 170
15. What is the impact of your problem? ................................. 173
16. How is your system configured? ................................. 173
17. What are the external symptoms ................................. 173
18. What symptom information did you collect? ..................... 173
19. Which type of documentation did you obtain? ................................. 174
20. What recovery actions did you attempt? ................................. 174
## Figures

1. LPAR view of the PFA components ................................... 36  
2. Defining the file system ............................................ 40  
3. Example of the Java 5.0 path ....................................... 42  
4. Common storage usage output report .......................... 52  
5. LOGREC Arrival Rate Prediction Report ...................... 57  
6. Frames and slots usage prediction report ....................... 63  
7. Flowchart for abend analysis .................................... 89  
8. RSMDATA SUMMARY report ........................................ 103  
9. Flowchart for system hang analysis ........................... 108  
10. IPL statistics example ........................................... 110  
11. IPCS STATUS WORKSHEET report from a stand-alone dump ............................................ 115  
12. Flowchart for job or subsystem hang analysis ........... 122  
13. Flowchart for diagnosis of a loop ............................... 132  
14. System trace table entry .......................................... 135  
15. Flowchart for output problem analysis ....................... 144  
16. Flowchart for performance problem analysis ............... 154  
17. JCL for invoking IPCS as a background job ................. 172
About this information

This information is designed to help you avoid potential problems and diagnose problems on z/OS, its subsystems, its components, and problems in applications running under the system. Using this information, you can:

- Identify a potential problem
- Identify the problem type
- Determine the failing subsystem, component, job, or application
- Collect the correct data needed to diagnose the problem
- Develop a search argument and use it to search problem reporting databases
- Know the correct problem data to collect before reporting the problem to IBM or the independent software vendor.

This information can help you determine why a problem occurred and where a problem occurred; it does not describe how to fix program instructions in your own code.

Who should use this information

This information is for anyone who diagnoses software problems that occur while running the operating system. This person is typically a system programmer for the installation. This information is also for application programmers who are testing their programs.

The level of detail at which this information is written assumes that the reader:

- Understands basic system concepts and the use of system services
- Codes in Assembler language, and reads Assembler and linkage editor output
- Codes Job Control Language (JCL) statements for batch jobs and cataloged procedures
- Understands the commonly used diagnostic tasks and aids, such as message logs, dumps, and Interactive Problem Control System (IPCS)

How to use this information

Use the procedures in this information to properly collect problem data, avoid potential problems, and diagnose failures.

If your installation does not want to debug the problem or does not have the source code involved in the problem, use the diagnosis procedures to collect the problem data needed for reporting the problem to IBM or other software vendors. The techniques described in this information are also relevant to non-IBM problems.

If your installation wants to debug the problem and has the source code, use the procedures to collect problem data and debug the problem. If the problem is in IBM code, report the problem to IBM. Where possible, IBM will debug the problem and provide a fix.
Where to find more information

Where necessary, this information references information in other documents, using cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see z/OS Information Roadmap.

This information also references diagnosis books for specific components, see Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165.

Information updates on the web

For the latest information updates that have been provided in PTF cover letters and Documentation APARs for z/OS® and z/OS.e, see the online information at: http://publib.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/ZDOCAPAR.

This information is updated weekly and lists documentation changes before they are incorporated into z/OS publications.

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required): http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp
Summary of Changes

Summary of Changes for G325-2564-06
z/OS Version 1 Release 11
as updated October 2009

This document contains information previously presented in z/OS Problem Management, which supports z/OS Version 1 Release 11.

New information

- PFA has a new default Java™ version when you apply APAR OA30358 to V1R11. See "What's new for PFA installation in V1R11?" on page 37.
- Information about the z/OS Management Facility. See "IBM z/OS Management Facility" on page 16.

Changed information

- The PFA_LOGREC_ARRIVAL_RATE check:
  - This check can now issue a subset of messages within the LOGREC arrival rate prediction report.
  See "PFA_LOGREC_ARRIVAL_RATE" on page 55 for complete details.
- The PFA_VIRTUAL_STORAGE_USAGE check:
  - This check is renamed to be the PFA_FRAMES_AND_SLOTS_USAGE check
  - The maximum value for the STDDEV parameter is now 100.
  See "PFA_FRAMES_AND_SLOTS_USAGE" on page 61 for complete details.
- The PFA_MESSAGE_ARRIVAL_RATE check:
  - The default value for the STDDEV parameter is now 10 and the maximum value is now 100.
  - This check does not process certain address spaces.
  See "PFA_MESSAGE_ARRIVAL_RATE" on page 66 for complete details.
- The following messages have changes:
  - AIRH148E on page 77
  - AIRH149I on page 77
  - AIRH150I on page 78
  - AIRH151I on page 78

You might notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.
This document contains information previously presented in z/OS Problem Management, which supports z/OS Version 1 Release 10.

New information:
- There is new information that explains "How PFA interacts with IBM Health Checker for z/OS" on page 37.
- PFA checks are self-managed. For information about how model data is handled, see "installation guidelines" on page 37 and check-specific information in "PFA_LOGREC_ARRIVAL_RATE/data/" on page 59 and "PFA_MESSAGE_ARRIVAL_RATE/data/" on page 71.
- The following Predictive Failure Analysis® checks are new:
  - PFA_VIRTUAL_STORAGE USAGE
  - "PFA_MESSAGE_ARRIVAL_RATE" on page 66.

New messages
- AIRH148E - AIRH173I

Changed information
- All AIR system messages are now located in z/OS MVS System Messages, Vol 1 (ABA-AOM).

Changed messages
- AIRH101E
- AIRH109E
- AIRH110E

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of Changes
for G325-2564-04
z/OS Version 1 Release 10
as updated April 2009

This document contains information previously presented in z/OS Problem Management, which supports z/OS Version 1 Release 10.

New information:
- New information about Predictive Failure Analysis system messages is located in "AIR system messages".

Summary of Changes
for G325-2564-03
z/OS Version 1 Release 10
as updated April 2009

This document contains information previously presented in z/OS Problem Management, which supports z/OS Version 1 Release 10.

New information:
- New information about Predictive Failure Analysis and its available checks and messages:
  - Part 2, "Predictive Failure Analysis," on page 33
Summary of Changes
for G325-2564-02
z/OS Version 1 Release 10

This document contains information previously presented in z/OS Problem Management, which supports z/OS Version 1 Release 9.

New information
• A new tip for using the AutoIPL function is in "Automating the SADMP process" on page 29.

Changed information
• The RSM summary report, Figure 8 on page 103, is updated.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.
Part 1. Problem management overview

Before you begin diagnosing problems or using PFA, it is important to understand the basics and best practices of problem management. This section covers the following topics:

- Chapter 1, “Introduction,” on page 3
- Chapter 2, “Common tools for problem determination,” on page 15
- Chapter 3, “Best practices for large stand-alone dump,” on page 25
Chapter 1. Introduction

If a problem occurs, this information can help you determine what happened, why it happened, and how to find the fix or report the problem to IBM®.

This information can also help you avoid problems and soft failures, find specific information about tools and service aids, and locate diagnosis information in the z/OS library. For details, see:

- Chapter 2, “Common tools for problem determination,” on page 15
- Part 2, “Predictive Failure Analysis,” on page 33
- Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165.

This chapter covers these topics:

- “Overview of problem resolution”
- “Gathering diagnosis data” on page 5
- “Problem categories” on page 6
- “Searching problem reporting databases” on page 8
- “Extracting problem symptoms and search arguments” on page 8
- “Formats for symptoms” on page 9
- “Determining the level of z/OS” on page 13

Overview of problem resolution

Typical z/OS problems are classified by the following symptoms:

- **Abend** – an error or abnormal end of a program or job.
- **Wait or Hang** – a coded wait state is loaded or the system or a job appears hung or does not complete.
- **Loop** – the system or program executes infinitely typically using large or higher amounts of processor resource.
- **Incorrout** – there is incorrect or missing output from a program or job.
- **Performance** – processing is using too much system resource and impacting other parts or users of the system, or processes are taking too long.
- **Message** – an error is reported through a message to the operator or in a log.

When an error occurs, z/OS provides various forms of diagnosis information that contains symptoms. These symptoms can help you with diagnosis and be used in problem source identification (PSI). PSI is the determination of what caused the error based on answers to these questions:

- Why was an abend issued?
- What caused storage to be exhausted?
- Why is the job hung?
- What is causing the job to loop?

This document is designed to help answer these questions and others and make efficient use of your time when diagnosing, searching, and reporting a problem to IBM.

PSI is useful even if the root cause of the problem is not identified. During the process, information and symptoms are gathered to check for a known problem or
report a new problem. To ease and expedite problem identification, it is important to provide all the background information available. This includes information about:

- Hardware involved
- System and application software levels
- External symptoms
- Problem impact
- Diagnostic data produced

By providing sufficient information during the first call to IBM or the individual software vendor, you might avoid having to re-create the problem.

The problem diagnostic worksheet contains key information needed to expedite problem resolution. If you are an experienced z/OS system programmer, use the Chapter 16, “Problem diagnostic worksheet,” on page 173 as a reminder of the important information to gather and report. For example:

- Extract the diagnostic data and symptoms
- Build a search argument
- Search for a known problem
- Gather available diagnostic information

### Steps for diagnosing problems on z/OS

To diagnose a problem, follow these steps:

1. When the problem occurs, gather all the available diagnosis information for the problem. Use the Chapter 16, “Problem diagnostic worksheet,” on page 173 as a template for recording data. This might also include your internal problem report describing external symptoms, what might have triggered the problem, and what was done to recover, including the following types of diagnostic information:
   - Dumps
   - Traces
   - Error messages
   - SYS1.LOGREC entries
   - External symptoms
   - Hardware devices
   - Processor models
   - Any other information

   These topics can help you collect the data more effectively:
   - “Gathering diagnosis data” on page 5
   - “Problem categories” on page 6

2. After the problem type is identified, see these diagnosis procedures to identify the source and extract symptoms:
   - Chapter 8, “Diagnosing an abend,” on page 85
   - Chapter 9, “Diagnosing a system hang or wait state,” on page 105
   - Chapter 10, “Diagnosing a job or subsystem hang,” on page 121
   - Chapter 11, “Diagnosing a loop,” on page 129
   - Chapter 12, “Diagnosing an output problem,” on page 143
3. While using the procedure, build a search argument from the data collected. See “Extracting problem symptoms and search arguments” on page 8 for more information.

4. Perform the search. Keep in mind that you might refine your search with more data from the problem. See “Searching problem reporting databases” on page 8 for more information.

5. If the problem is not found in a database, report it as a new problem providing the documentation and information collected in Chapter 16, “Problem diagnostic worksheet,” on page 173. See Chapter 15, “Reporting problems to IBM,” on page 169 for more information.

Tip: Sometimes information is found that is useful in routing the problem to the right place. For example, if the problem is an ABEND0C4 in module XYZ and your search shows multiple hits for ABEND0C4 and XYC and information about product from another company, contact that company or search that company’s problem reporting databases.

Gathering diagnosis data

It is important to gather the external symptoms and know the impact to the system or sysplex to define the scope of the problem. There can be many symptoms.

For example: Shortly after JOB A started, a dump was produced for an ABEND0C4, the system went into a WAIT064, and was partitioned from the sysplex.

1. Start with diagnosis of the ABEND0C4, which appears to be the trigger, but also understand the cause of the WAIT064 and why the job failure resulted in a system outage. It is important to check for known problems for both symptoms.

2. Next, gather all the diagnosis data available from the time frame the problem or failure occurred.

To identify a system problem, look at the diagnostic data such as:

- External symptoms and the initial problem report. Look for indications, which can include:
  - Messages
  - Job hang
  - System hang
  - High processor usage
  - Incorrect output
  - Dumps produced
  - System slowdown
  - Jobs not starting
- SVC dumps produced as indicated by these messages:

  IEA794I
  IEA794I SVC DUMP HAS CAPTURED; DUMPID=dumpid REQUESTED BY JOB (*MASTER*)
  DUMP TITLE=dump-title

  IEA911E
  IEA911E {COMPLETE|PARTIAL} DUMP ON SYS1.DUMPnn
  DUMPID=dumpid REQUESTED BY
JOB (jobname)
FOR ASIDS(id,id,...)
[REMOTE DUMPS REQUESTED | REMOTE DUMP FOR SYSNAME: sysname]
INCIDENT TOKEN: incident-token
[SRSN = vvvvvvvv wwww www xxxxxxx
zzzzzzzz]
[reason-text]
[ERRORID = SEQyyyyy
CPUzz ASIDasid
TIMEhh.mm.ss.f]
[TSOID = tsoid]
[ID = uuuuuuuuu]

IEA611I
IEA611I {COMPLETE|PARTIAL) DUMP ON dsname
text

• SYS1.LOGREC data set, which is a repository for information about hardware and system-level software errors.
• Logs from the time frame the problem occurred. This can include SYSLOG, OPERLOG, job log(s), and others.
• Traces associated with the problem.

Tip: After a problem has been isolated to a particular component, query using the TRACE command to see if detailed component traces or GTF was active at the time. For example: If the error is announced by ISGxxxx messages, then check for SYSGRS CTRACE. The message prefix (three or more characters) will determine the component owner. See the topic on identifying modules, components, and products in z/OS MVS Diagnosis: Tools and Service Aids.

Problem categories

The problem indicator table contains examples of indicators. Some problems might need to be investigated using more than one diagnostic procedure to find the cause. If there are several indicators, look for the earliest problem that caused the other problems.

Example: There are several abends and a wait state, look for the earliest abend code and begin diagnosis there.
<table>
<thead>
<tr>
<th>Problem type</th>
<th>Indicator</th>
<th>System action</th>
<th>System programmer action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abend</strong></td>
<td>SVC dump taken and a record of the error (in Logrec)</td>
<td>Produces dump</td>
<td>Review dump to determine if further diagnosis is required</td>
</tr>
<tr>
<td></td>
<td>Message received indicating a system or user abend</td>
<td>Produces record of error</td>
<td>Review response of system message to determine the impact of the abend on the installation.</td>
</tr>
<tr>
<td></td>
<td>An ABEND dump is produced</td>
<td>Continue processing</td>
<td>Review the dump to determine if further diagnosis is required.</td>
</tr>
<tr>
<td></td>
<td>• SVC dump produced</td>
<td></td>
<td>1. Use IPCS to do problem diagnosis on the dump.</td>
</tr>
<tr>
<td></td>
<td>• Error recorded to SYS1.LOGREC</td>
<td></td>
<td>2. Look up abend and reason code recorded for more information about error.</td>
</tr>
<tr>
<td></td>
<td>• Error message issued</td>
<td></td>
<td>3. Look up the message to gather more information about cause of the error and the system programmer action to correct.</td>
</tr>
<tr>
<td></td>
<td>• SYSUDUMP, SYSABEND or CEEDUMP produced</td>
<td></td>
<td>4. Review the dump to do problem diagnosis.</td>
</tr>
<tr>
<td></td>
<td>System actions are the same as the indicators listed above.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The system might also initiate recovery actions. See SYSLOG and component trace to determine what these recovery actions were. Some recovery actions can cause data to be lost, requiring the installation to resubmit jobs or transactions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Job hang/wait or loop</strong></td>
<td>Job does not end, no further output is produced, and the job can or cannot be CANCEL’ed or FORCE’ed</td>
<td>No response</td>
<td>Use the DUMP command to obtain an SVC dump of the hung job. If the DUMP is not successful, consider taking a stand-alone dump.</td>
</tr>
<tr>
<td></td>
<td>Disabled wait indicated on the HMC and wait state message issued</td>
<td>The system issues a wait state message and loads a disable wait state PSW. The system might load the following into the PSW: X’070E0000 00000000’</td>
<td>Take a stand-alone dump.</td>
</tr>
<tr>
<td></td>
<td>Many jobs are hung in the system</td>
<td>Resource contention</td>
<td>Enter the DISPLAY GRS,C command to check for ENQ resource and latch contention and take a dump of the holder of the resource including SDATA=GRSQ. <strong>Note:</strong> Use the DISPLAY GRS,ANALYZE command to aid in the discovery of blockers in the system.</td>
</tr>
<tr>
<td></td>
<td>No response to system or subsystem commands entered</td>
<td>No response</td>
<td>Partition the system from the sysplex and take a stand-alone dump.</td>
</tr>
</tbody>
</table>
### Table 1. Problem indicators by type (continued)

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Indicator</th>
<th>System action</th>
<th>System programmer action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loop</strong></td>
<td>High processor resource being consumed locking out other work; Excessive spin detected with IEE178I or ABEND071 issued, or both.</td>
<td>ABEND071 issued in an attempt to stop the looping program</td>
<td>Use an online monitor, such as RMF™ or IBM OMEGAMON® z/OS Management Console, to determine whether the problem originates from a high priority job in normal processing or from a problem.</td>
</tr>
<tr>
<td></td>
<td>A job is using a high percentage of central processor storage</td>
<td>Processing degrades</td>
<td>Use an online monitor, such as RMF, to determine whether the problem originates from a high priority job in normal processing or from a problem.</td>
</tr>
<tr>
<td><strong>Enabled wait or performance degradation</strong></td>
<td>System processing slows.</td>
<td>Processing degrades</td>
<td>Use an online monitor, such as RMF, to determine where the problem originates.</td>
</tr>
<tr>
<td><strong>Output problem</strong></td>
<td>Job output is missing or is incorrect.</td>
<td>Processing continues</td>
<td>Use GTF or SLIP to trace input and output.</td>
</tr>
</tbody>
</table>

#### Searching problem reporting databases

While you are diagnosing a system problem, you will collect data about that problem:
- What was the abend code?
- What did the registers and PSW contain at the time of error?
- What is the failing module or CSECT?
- What components or products were involved with the error?

The answers to these questions are the material for a search argument. A search argument is a list of symptoms for a problem. A search argument is also called a symptom string.

This section contains these topics:
- “Extracting problem symptoms and search arguments” describes how to develop a search argument while you are performing diagnosis.
- “Formats for symptoms” on page 9 distinguishes between the types of symptom formats.
- “Searching for a known problem” on page 10 lists the symptoms used in search arguments.
- “Steps for searching problem reporting databases” on page 12 explains the steps to begin your search.

#### Extracting problem symptoms and search arguments

Obtain search arguments from an SVC dump, SYSMDUMP dump, or stand-alone dump by using IPCS subcommands.
For most problems, use three to five symptoms in the search argument. If the first search produces no matches, remove some symptoms and search again. If the first search produces too many matches, add one or more symptoms and search again. Also, try different symptoms. Searching is an iterative process.

The following are suggestions for selecting symptoms:

- Start with the symptom keyword, for example ABEND0C4, WAIT or LOOP and the module or CSECT name and component ID. Add or remove symptoms from the search argument depending on the number of matches produced.
- Symptoms about data areas are useful for identifying a problem. Use the names of a data area and the incorrect field in the data area as symptoms.
- If searching does not produce a match, remove some symptoms or use different symptoms and try again.

Table 2. Obtaining search arguments from SVC dump, stand-alone dump or SYSMDUMP using IPCS commands

<table>
<thead>
<tr>
<th>IPCS subcommand</th>
<th>Dump output heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS FAILDATA</td>
<td>Search Argument Abstract</td>
</tr>
<tr>
<td>VERBEXIT DAEDATA</td>
<td>DUMP ANALYSIS AND ELIMINATION (DAE)</td>
</tr>
<tr>
<td>VERBEXIT LOGDATA</td>
<td>SEARCH ARGUMENT ABSTRACT</td>
</tr>
<tr>
<td>VERBEXIT SYMPTOM</td>
<td>Primary Symptom String</td>
</tr>
</tbody>
</table>

Build a free-format search from the IPCS reports by extracting:

- CSECT name
- Abend code
- Reason code
- Component id

Ensure use of the standardized symptom keyword. For example, system abend code presented as S005C in the IPCS ST FAILDATA report is converted to ABEND05C and reason code PRCS/00000214 is converted to RSN00000214. Table 3 on page 10 contains a list of the standardized symptoms.

Formats for symptoms

Symptom strings or search arguments are presented in several different formats. They include:

- **Free-Format symptom**: is commonly used to search on the Internet and in IBMLINK for a known problem. The symptoms in the freely formatted string are standardized (see Table 3 on page 10).
  
  For example:
  - ABEND0C4 5752SCXCF IXCS2STB
  - A module CSECT name: IEEA8CD

- **RETAINTM symptom string**: Use RETAIN® symptoms:
  - With a tool such as Info/Management to search the RETAIN database
  - When reporting a problem to IBM
  - In descriptions of problems in APARs and program temporary fixes (PTF)

RETAINTM symptoms are also called **structured symptoms** and **failure keywords**. An example of a module CSECT name as a RETAIN symptom is: RIDS/IEEA8CD
The table of RETAIN and MVS™ symptoms is in the topic on specifying symptoms in z/OS MVS Diagnosis: Reference.

- MVS symptom, is used by dump analysis and elimination (DAE) when determining if a dump is a duplicate of a previous dump; MVS symptoms are not used for searching problem databases. These symptoms are contained in the DAE data set. An example of a module CSECT name as an MVS symptom is: CSECT/IEAABCD. For a complete example, see the topic on dump analysis and elimination (DAE) in z/OS MVS Diagnosis: Tools and Service Aids.

Searching for a known problem

Use the following search argument of standardized symptoms when performing a search for a known problem in the Technical Support database, IBMLink™ or when reporting a problem to IBM:

- Always concatenate a number to a word when it modifies that word (for example, SVC13, ABEND0C4)
- Use the word “missing” whenever messages do not appear as expected
- Never abbreviate system commands
- Include = in a search argument with no blanks on either side (for example, DISP=MOD is correct)
- Do not use hyphens in search arguments (for example, SC231234 is the correct way to enter a publication number).

The standardized symptom table describes common symptom keywords to use while doing a search for a known problem or reporting a problem to IBM:

<table>
<thead>
<tr>
<th>Free-format symptom</th>
<th>Problem data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABENDxxx</td>
<td>Any system abend except JES3; where xxx is the hexadecimal value of the abend code, always 3 digits including leading zeros</td>
</tr>
<tr>
<td>ABENDDMxxx</td>
<td>JES3 abend; where xxx is the hexadecimal value of the abend code, always 3 digits including leading zeros</td>
</tr>
<tr>
<td>ABENDUxxxx</td>
<td>User abend; where xxxx is the user abend code</td>
</tr>
<tr>
<td>AMODE31</td>
<td>Program running in AMODE 31 (31-bit mode)</td>
</tr>
<tr>
<td>AMODE64</td>
<td>Program running in AMODE 64 (64-bit mode)</td>
</tr>
<tr>
<td>ARnn</td>
<td>Access register; where nn is the decimal register number without leading zeros</td>
</tr>
<tr>
<td>CRnn</td>
<td>Control register; where nn is the decimal register number without leading zeros</td>
</tr>
<tr>
<td>D/Txxxx</td>
<td>Device type; where xxxx is the device number</td>
</tr>
<tr>
<td>DATASET</td>
<td>Data set</td>
</tr>
<tr>
<td>DEQ</td>
<td>Dequeue</td>
</tr>
<tr>
<td>DESCODEnn</td>
<td>WTO descriptor code; where nn is the decimal value of the code, 1-13, without leading zeros</td>
</tr>
<tr>
<td>ENQ</td>
<td>Enqueue</td>
</tr>
<tr>
<td>ERRNO2n...n</td>
<td>Where n...n is the 4 byte hexadecimal value of the errno2</td>
</tr>
<tr>
<td>ERRNOJRn...n</td>
<td>Where n...n is the 4 byte hexadecimal value of the errnojr</td>
</tr>
<tr>
<td>ERRNOnnn</td>
<td>Where nnn is the errno in decimal</td>
</tr>
<tr>
<td>HANG</td>
<td>Always include this form of the word</td>
</tr>
</tbody>
</table>
Table 3. Standardized symptom keyword list (continued)

<table>
<thead>
<tr>
<th>Free-format symptom</th>
<th>Problem data</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>Input Output</td>
</tr>
<tr>
<td>KEYn</td>
<td>PSW Key or Storage Key (in hex)</td>
</tr>
<tr>
<td>KEYnn</td>
<td>PSW Key or Storage Key (in dec)</td>
</tr>
<tr>
<td>LATCH#nn</td>
<td>Where nnn is decimal latch number without leading zeros (for example: LATCH#2)</td>
</tr>
<tr>
<td>LOOP</td>
<td>Always include this form of the word</td>
</tr>
<tr>
<td>LPAR</td>
<td>Logical Partition (PR/SM™)</td>
</tr>
<tr>
<td>LU62</td>
<td>Logical Unit 6.2 protocol</td>
</tr>
<tr>
<td>MIH</td>
<td>Missing interrupt handler</td>
</tr>
<tr>
<td>MSGxxxx</td>
<td>Any message except JES2 messages; where xxxx is the complete message id of any length</td>
</tr>
<tr>
<td>MSGHASPxxx</td>
<td>JES2 messages; note that the ‘$’ prefix has been removed and xxx is the message id of any length</td>
</tr>
<tr>
<td>OVERLAY</td>
<td>Storage overlay; always include this form of the word</td>
</tr>
<tr>
<td>PAGEFIX</td>
<td>Page-Fix</td>
</tr>
<tr>
<td>PICxx</td>
<td>Program Interrupt Code associated with ABENDOCx; where xx is the interrupt code, always 2 digits with leading zeros</td>
</tr>
<tr>
<td>Rxxx</td>
<td>Release level; where xxx is the product release level</td>
</tr>
<tr>
<td>RCnn</td>
<td>Return code; where nn is decimal or hexadecimal and at least two digits</td>
</tr>
<tr>
<td>REGnn</td>
<td>General purpose register; where nn is the decimal register number without leading zeros</td>
</tr>
<tr>
<td>ROUTCODEnnn</td>
<td>WTO route code; where nnn is the decimal value of the code, 1-128, without leading zeros</td>
</tr>
<tr>
<td>RSNxxx</td>
<td>Reason code; where xxx is the hexadecimal reason code of any length</td>
</tr>
<tr>
<td>SADMP</td>
<td>Stand-alone dump</td>
</tr>
<tr>
<td>SIGxxxx</td>
<td>Where xxxx is the name of the signal (for example: SIGTERM)</td>
</tr>
<tr>
<td>SIO</td>
<td>Start Input Output</td>
</tr>
<tr>
<td>SMFTYPEennn</td>
<td>SMF type records; where nnn is the decimal value of the record, 0-255, without leading zeros</td>
</tr>
<tr>
<td>SUBTYPEennn</td>
<td>SMF subtype records; where nnn is the decimal value of subtype, 0-255, without leading zeros. Also make sure the SMFTYPEennn is included</td>
</tr>
<tr>
<td>SPnnn</td>
<td>Subpool number; where nnn is the decimal value of subpool, 0-255, with no leading zeros</td>
</tr>
<tr>
<td>SVCnnn</td>
<td>Supervisor Call; where nnn is the decimal value of the SVC, 0-255, with no leading zeros</td>
</tr>
<tr>
<td>VOLSER</td>
<td>Volume serial</td>
</tr>
<tr>
<td>WAIT</td>
<td>Always use this form of the word</td>
</tr>
<tr>
<td>WAITxxx</td>
<td>System wait state; where xxx is the hex value of the wait code, always 3 digits including leading zeros</td>
</tr>
<tr>
<td>Z/ARCHITECTURE</td>
<td>64-bit mode</td>
</tr>
</tbody>
</table>
Steps for searching problem reporting databases

Often the problem has already been reported and fixed. Using the symptom string or search argument extracted, you can do a search of the technical database associated with the product identified.

1. Go to one of these Web sites:
   • IBMLink at ibm.com/ibmlink/link2
   • Support for z/OS Web site contains troubleshooting, fixes and tools.
2. Select the documents you want to search for problem related information. For example, select APARs, FAQs, Technotes, or Flashes.
3. If the Internet is not available at your installation, call the IBM support center and ask them to do the search for you.

Search arguments are used to search problem reporting databases. If the problem being diagnosed was already reported and the symptoms entered into the database, the search will produce a match.

References

Example of a IBMLink search

If your installation has access to IBMLink, an interactive online database program, you can:

• Search for an existing authorized program analysis report (APAR) that is similar to your problem.
• Search for an available program temporary fix (PTF) for the existing APAR.
• Order the PTF if it is available.
• Create an Electronic Technical Response (ETR) problem report to get assistance from a service representative.

This example shows a search argument using the free-format symptom ABEND0C4 IEFJRA):
Determining the level of z/OS

When you report problems to the IBM Support Center, you must provide the name and level of the operating system or systems. If you have communication with your console, you can use the DISPLAY command or you can query the dump using IPCS.

Use the console command DISPLAY IPLINFO or the IPCS command IPLINFO in a dump to obtain the following information:

- The date and time of the IPL
- The release level of the system
- The license value for the system
- The contents of parmlib members IEASYSxx and IEASYMxx
- LOADxx information used for the IPL
- The architecture level of the IPL
- The IODF (input/output definition file) device
- The IPL device and volume serial
- The status of MTL (manual tape library) tape devices.

For example:

```
D IPLINFO
IEE254I 11.14.07 IPLINFO DISPLAY 350
SYSTEM IPLED AT 01.15.39 ON 11/01/2007
RELEASE z/OS 01.09.00 LICENSE = z/OS
USED LOAD08 IN SYSO.IPLPARM ON ACB2
ARCHLVL = 2 MTLSHARE = N
```
IEASYM LIST = (X6,U6,0L,R8)
IEASYS LIST = (ST, LN) (OP)
IODEF DEVICE ACB2
IPL DEVICE 3C2A VOLUME D83EL

If you cannot communicate through the console, use IPCS to perform the following steps to determine which system or systems you are using:

1. Use the IPCS subcommand **CBFORMAT** with the communications vector table (CVT) control block to determine the product level.

   In the **CBFORMAT CVT** example output, the **PRODN** field indicates an MVS operating system level of **SP7.0.7** and the **PRODI** field indicates the FMID as **HBB7720**.

   CVT: 00FD48A0
   -0028 PRODN.... SP7.0.7  PRODI.... HBB7720  VERID....
   +0006 MCL....... 2084  RELNO.... 038
   +0000 TCBP....... 00000218  OEF00.... 00FEA3EC  LINK..... 00FD481C
   +000C AUSCB.... 00FD57E8  BUF...... 00000000  XAPG..... 00FE0380

2. **Determine if the system is running as a uniprocessor or multiprocessor.** In the IPCS **STATUS WORKSHEET** output, just above the heading **PROCESSOR RELATED DATA**, find the MVS Diagnostic Worksheet:

   **MVS Diagnostic Worksheet**

   Dump Title: W059 SLIP TRAP

   CPU Model 2084 Version 00 Serial no. 220CBE Address 00
   Date: 09/15/2006    Time: 09:33:32.124515 Local

   CSD Available CPU mask: FFC0 Alive CPU mask: FFC00000 00000000
   Number of active CPUs: 000000nn

   The **nn** in **000000nn** indicates the number of processors running.
   In this output example, there are 10 active processors.

   CSD Available CPU mask: FFC0 Alive CPU mask: FFC00000 00000000
   No. of active CPUs: 000000A

   **Related information**

   - See **SMP/E User's Guide** for using SMP/E.
   - See **z/OS MVS IPCS Commands** for more information about the **CBFORMAT** subcommand.
   - See **z/OS MVS Data Areas** in the z/OS library at [http://www.ibm.com/systems/z/os/zos/bkserv/](http://www.ibm.com/systems/z/os/zos/bkserv/) for the format of the SCCB.
Chapter 2. Common tools for problem determination

z/OS contains many tools and service aids to assist you when a problem does occur. The more you know about these tools and service aids, the easier it is for you to diagnose problems and send data to IBM. This chapter provides an overview of the some commonly used tools and where to find more information about each of them.

This chapter covers these topics:
- “Messages”
  - “BPXMTEXT for z/OS UNIX reason codes” on page 16
- “IPCS” on page 16
- “Logs” on page 17
- “Traces” on page 19
- “Dumps” on page 21
- “IBM Omegamon for z/OS Management Console” on page 22
- “AMATERSE for FTP” on page 22
- “IBM documentation” on page 22

Messages

z/OS issues messages from z/OS elements, features, program products, and application programs running on the system. The system issues messages in different ways and to different locations:
- Automated messaging services automatically react to certain messages.
- Most messages are issued through WTO and WTOR macros to one of these locations:
  - Console
  - Hard-copy log
  - Job log
  - SYSOUT data set
Routing codes determine where the messages are displayed or printed. The routing codes for messages issued by the operating system are included with each message.
- Unless specified otherwise, messages, in general, go to the system log (SYSLOG).
- Dump messages are issued through the dumping services routines and are found in:
  - SVC dumps, stand-alone dumps, or SYSMDUMP ABEND dumps formatted by the interactive problem control system (IPCS)
  - Trace data sets formatted by the interactive problem control system (IPCS)
  - ABEND dumps or SNAP dumps produced by the dumping services
In dump or trace data sets formatted by IPCS, the messages are shown on a terminal or in a printed dump.
- Some messages are issued through DFSMS/MVS™ access methods directly to one of these locations:
  - Output data set
  - Display terminal
BPXMTEXT for z/OS UNIX reason codes

BPXMTEXT is shipped in SYS1.SBPXEXEC and must be in SYSEXEC or SYSPROC to use. It can run from TSO, IPCS, or the z/OS UNIX Shell. In z/OS V1R8 and above, you can use BPXMTEXT to interpret errnojr values from zFS (reason code qualifier=EFxx) and TCP/IP (reason code qualifier=7xxx). In z/OS V1R9 and above, you can interpret errnojr values from the C/C++ run-time library (reason code qualifier=Cxxx).

To determine the meaning of reason codes for z/OS UNIX® and zSeries® File System (zFS), use BPXMTEXT.

From TSO, enter TSO BPXMTEXT xxxxxxxx, where xxxxxxxx is the reason code.

Here’s an example:
EQQPH35I: EQQPH351 BPX1ATX FAILED WITH RC=0157, RSN=0B1B03AC

To find the meaning of RSN=0B1B03AC from TSO, enter:
BPXMTEXT 0B1B03AC

You get this result:
BPXPREXC date JRAuthCaller: The caller of this service is authorized. Authorized callers are not permitted to load or call unauthorized programs.
Action: System key, supervisor state, or APF authorized callers cannot load or call unauthorized programs.

IBM z/OS Management Facility

IBM z/OS Management Facility (z/OSMF) provides a framework for managing various aspects of a z/OS system through a task oriented, Web browser interface. By streamlining some traditional tasks and automating others, z/OSMF can help to simplify some areas of system management and reduce the level of expertise needed for managing a system. The initial release of z/OSMF includes functions for configuring TCP/IP policy-based networking functions and performing problem data management tasks through the Incident Log, which consolidates abend dump related problems and data and simplifies the process of sending the data for further diagnosis. For more information, see z/OSMF at www.ibm.com/systems/z/os/zos/zosmf/.

IPCS

Interactive Program Control System (IPCS) is a powerful diagnostic tool in the MVS system that aids the diagnosis of 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 (SVC dump, stand-alone dump, SYSMDUMP) and traces (system trace, GTF trace, and CTRACE) need to be formatted before 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 allow 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.

Using dump and trace data sets and, in some cases, active storage as a source IPCS analyzes information and produces reports that can be viewed at a Time Sharing Option Extensions (TSO/E) terminal or can be printed.

Related information: For complete information about IPCS, see these procedures and documents:
- "Invoking IPCS as a background job" on page 171
- z/OS MVS IPCS User’s Guide
- z/OS MVS IPCS Commands
- z/OS MVS IPCS Customization
- Using IPCS to format component dump data in z/OS MVS Diagnosis: Reference

Logs

Do not overlook log data — it should be the first place to look when reviewing a problem. z/OS communicates problems through messages that it writes to logs. Six logs contain the primary sources of problem data:

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

View SYSLOG through 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 from IPCS using the VERBX MTRACE command.

This example shows the MVS SYSLOG without time stamps.

```
STC18213 00000090 SHASPF00 BXPAXS ON STCINOR
STC18213 00000090 SHASPF73 BXPAXS STARTED
STC18213 00000010 IE4031 BXPAXS = STARTED - TME=13.36.36 - ASID=001F - SC53
STC18213 00000090 IE4031 BXPAXS = STARTED - TME=13.36.36 - ASID=001F - SC53
STC18213 00000090 IE4031 BXPAXS = STARTED - TME=13.36.36 - ASID=001F - SC53
STC18213 00000090 IE4031 BXPAXS = STARTED - TME=13.36.36 - ASID=001F - SC53
STC18213 00000090 IE4031 BXPAXS = STARTED - TME=13.36.36 - ASID=001F - SC53
STC18213 00000090 IE4031 BXPAXS = STARTED - TME=13.36.36 - ASID=001F - SC53
```

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.

In SDSF the OPERLOG panel displays the merged, sysplex-wide system message log. You can use the parameters of the LOG command to select the OPERLOG panel or the single-system SYSLOG panel. 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.

An individual product has its own log file. These log files might contain data that is valuable when diagnosing a problem. It is particularly important to look for events that precede an actual abend or failure because the problem, in many cases, will have been caused by a previous action.

This example shows the SYSOUT data sets that might be associated with a CICS® address space:

<table>
<thead>
<tr>
<th>NP DDNAME</th>
<th>StepName</th>
<th>ProcStep</th>
<th>DSID</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>JESJCLIN</td>
<td></td>
<td></td>
<td>1</td>
<td>CICSTS</td>
</tr>
<tr>
<td>JESMGLG</td>
<td>JES2</td>
<td></td>
<td>2</td>
<td>CICSTS</td>
</tr>
<tr>
<td>JESJCL</td>
<td>JES2</td>
<td></td>
<td>3</td>
<td>CICSTS</td>
</tr>
<tr>
<td>JESYSMSG</td>
<td>JES2</td>
<td></td>
<td>4</td>
<td>CICSTS</td>
</tr>
<tr>
<td>$INTTEXT</td>
<td>JES2</td>
<td></td>
<td>5</td>
<td>CICSTS</td>
</tr>
<tr>
<td>CAFF</td>
<td>SCSCPA1</td>
<td></td>
<td>101</td>
<td>CICSTS</td>
</tr>
<tr>
<td>CINT</td>
<td>SCSCPA1</td>
<td></td>
<td>103</td>
<td>CICSTS</td>
</tr>
<tr>
<td>DFHCXRF</td>
<td>SCSCPA1</td>
<td></td>
<td>104</td>
<td>CICSTS</td>
</tr>
<tr>
<td>COUT</td>
<td>SCSCPA1</td>
<td></td>
<td>105</td>
<td>CICSTS</td>
</tr>
<tr>
<td>CEEMSG</td>
<td>SCSCPA1</td>
<td></td>
<td>106</td>
<td>CICSTS</td>
</tr>
<tr>
<td>CEEOUT</td>
<td>SCSCPA1</td>
<td></td>
<td>107</td>
<td>CICSTS</td>
</tr>
<tr>
<td>PLIMSG</td>
<td>SCSCPA1</td>
<td></td>
<td>108</td>
<td>CICSTS</td>
</tr>
<tr>
<td>CRPO</td>
<td>SCSCPA1</td>
<td></td>
<td>109</td>
<td>CICSTS</td>
</tr>
<tr>
<td>MSGUSR</td>
<td>SCSCPA1</td>
<td></td>
<td>110</td>
<td>CICSTS</td>
</tr>
</tbody>
</table>

The key SYSOUT data sets to review for problem determination data are the JESMGLG and MSGUSR data sets. The CEEMSG and CEEOUT data sets will contain Language Environment® (LE) problem data typically associated with application problems.

The CICS JESMGLG SYSOUT data set includes information related to CICS startup and errors related to system problems, not specifically transaction related.

**Logrec Error Recording**

Log recording (logrec) log stream is an MVS System Logger application that records hardware errors, selected software errors, and symptom records across the sysplex.

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 can point you in the right direction while supplying you with symptom data about the failure. Use the Environmental Record, Editing, and Printing program (EREP) to:

- Print reports about the system records
- Determine the history of the system
- Learn about a particular error
Logrec 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 ICFEREPI program, or if the abend has triggered a dump, the EREP data can be reviewed using the IPCS VERBX LOGDATA command. Generally, the error log entries at the end of the display, if they have an influence on the problem being reviewed, have time stamps that relate to or immediately precede the actual abend; although there is no guarantee the error records will be written in the order they occurred. The error log entries are also written to an internal storage buffer that is included in the dump.

Using a logrec log stream rather than a logrec data set (SYS1.LOGREC, by default) for each system can streamline logrec error recording.

Console log
Console data that the installation chooses to log.

Hardcopy log
The hardcopy log is a record of the system message traffic that the installation chooses to log, such as messages to and from all consoles, commands and replies entered by the operator. In a dump, these messages are in the master trace. With JES3, the hardcopy log is always written to the SYSLOG. With JES2, the hardcopy log is typically written to the SYSLOG, but can also be written to a console printer, if your installation chooses.

Related information:
- z/OS MVS Planning: Operations contains information about OPERLOG and SYSLOG.
- Recording logrec error records in z/OS MVS Diagnosis: Tools and Service Aids contains complete information about EREP.
- Error recording on the logrec data set in z/OS MVS Diagnosis: Reference lists the incidents and the types of records that can be recorded on the logrec data set for each incident.

Traces

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

For complete information, see System trace in z/OS MVS Diagnosis: Tools and Service Aids.

Master trace
Master trace maintains a table of all recently issued system messages. This creates a log of external system activity; 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 is formatted in a dump using the VERBX MTRACE command.
Traces

**Component trace**

The component trace service provides a way for z/OS components to collect problem data about events that occur in the component. 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. Component trace is queried and formatted using the **IPCS CTRACE** command. Trace data is commonly used by the IBM Support Center to:

- Diagnose problems in the component
- Check how the component is running

The IBM support center might direct you to use specific component trace options when you need to re-create a problem to gather more diagnostic data.

For complete information, see Component trace in **z/OS MVS Diagnosis: Tools and Service Aids**.

**Transaction trace**

Transaction trace enables you to debug problems by tracing the path of a work unit running in a single system or across systems in a sysplex environment. Transaction trace provides a consolidated trace of key events for the execution path of application or transaction type work units running in a multi-system application environment.

The essential task of transaction trace is to aggregate data showing the flow of work between components in the sysplex that combine to service a transaction. Transaction trace traces events such as component entry, exit, exceptions and major events such as COMMIT, and ROLLBACK.

**Restriction:** Do not use transaction trace as a component tracing facility.

For complete information, see Transaction trace in **z/OS MVS Diagnosis: Tools and Service Aids**.

**Generalized trace facility (GTF)**

GTF traces system and hardware events similar to those in system trace, but also offers the option of an external writer and to write user defined trace events. GTF trace records can be formatted in a dump or trace data set using the IPCS **GTFTRACE** command.

For complete information, see The Generalized Trace Facility (GTF) in **z/OS MVS Diagnosis: Tools and Service Aids**.

**GFS trace (GFS)**

GFS trace is a 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.

For complete information, see GETMAIN, FREEMAIN, STORAGE (GFS) trace in **z/OS MVS Diagnosis: Tools and Service Aids**.

**Related information:**

- **z/OS DFSMSdfp Diagnosis**
- **z/OS Infoprint Server Messages and Diagnosis**
• For a comprehensive overview of tools and service aids, see the topic on Selecting tools and service aids in z/OS MVS Diagnosis: Tools and Service Aids.
• z/OS Communications Server: IP Configuration Reference

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. Complete details are found in SVC dump in z/OS MVS Diagnosis: Tools and Service Aids.

Transaction dump
A transaction dump provides a representation of the virtual storage for an address space when an error occurs. Typically, an application requests the dump from a recovery routine when an unexpected error occurs. Complete details are found in Transaction dump in z/OS MVS Diagnosis: Tools and Service Aids.

Abend dump
An ABEND dump shows the virtual storage predominately for an unauthorized program. To produce a dump when one is requested for an error, a JCL DD statement of SYSUDUMP, SYSABEND or SYSMDUMP must be included in the input job stream. See z/OS MVS JCL Reference for more information. An operator can also request an ABEND dump while ending a program, an address space, or canceling a job. There are three types of abend dumps:
• SYSMDUMP – is an unformatted dump that requires IPCS to view and format. Unformatted dumping is sometimes more efficient because only the storage requested is written to the data set, which means the application can capture diagnostic data and be brought back online faster.
• SYSABEND – The largest of the ABEND dumps, is a pre-formatted dump containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program.
• SYSUDUMP – The smallest of the ABEND dumps, containing data and areas only about the failing program.

SNAP dump
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 one 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. Complete details are found in SNAP dump in z/OS MVS Diagnosis: Tools and Service Aids.

Stand-Alone dump
The other tools discussed in this chapter are used to collect data for individual work units on a system or a subset of components on a system. A stand-alone dump is used to collect diagnostic information about the
Dumps

entire system. Stand-alone dumps are not produced by z/OS but by an either the IPCS SADMP dump data set utility or the AMDSADD REXX utility. After a stand-alone dump is taken, because the system cannot resume usual processing, the IPL is of the stand-alone dump instead of z/OS.

The stand-alone dump program produces a stand-alone dump of storage that is occupied by either:

- A system that is stopped. For example, your installation has a wait state with no processing, so you must capture a stand-alone dump to diagnosis it.
- A stand-alone dump program that failed. Either the stand-alone dump program dumped itself — a self-dump —, or the operator loaded another stand-alone dump program to dump the failed stand-alone dump program.

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 usual system operations and does not require the system to be in a condition for normal operation. It is essential to perform a store status before taking a stand-alone dump because the program gets loaded over storage that might be needed in the dump.

For more information:

- See the topics on Chapter 3, “Best practices for large stand-alone dump,” on page 25.
- See the complete details in Stand-Alone dump in z/OS MVS Diagnosis: Tools and Service Aids and in z/OS MVS IPCS User’s Guide.

IBM Omegamon for z/OS Management Console

The OMEGAMON z/OS Management Console is a monitoring product that includes an interface for z/OS management and is designed to help eliminate, and simplify many z/OS management tasks. The OMEGAMON z/OS Management Console helps deliver real-time, check information provided by the IBM Health Checker for z/OS, and configuration status information for z/OS systems and sysplex resources.

For more information, see IBM OMEGAMON for z/OS Management Console at www.ibm.com/systems/z/os/zos/zmc/.

AMATERSE for FTP

In z/OS V1R9 and above, use AMATERSE to compress and extract problem documentation you send to IBM. There are differences between AMATERSE and the TRSMAIN utility. AMATERSE is the preferred, supported program. For complete details, see the AMATERSE topic in z/OS MVS Diagnosis: Tools and Service Aids.

IBM documentation

There are many types of documentation to aid problem determination. Here are some of the categories:

- Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165, which contains diagnosis material by element or feature name.
The z/OS Internet Library, which contains complete, updated information about all z/OS elements and features:

www.ibm.com/servers/eserver/zseries/zos/bkserv/

Many people find it helpful to search on an individual element or feature using the z/OS elements and features search engine:


z/OS Hot Topics Newsletter, written by leading z/OS experts, contains hands-on, technical information about z/OS that is not contained in the traditional product libraries:

www.ibm.com/servers/eserver/zseries/zos/bkserv/hot_topics.html

The Techdocs Library, which includes:

- Flashes that alert you to significant new technical developments and provide guidance on the installation, use and management of z/OS:
  www.ibm.com/support/techdocs/atsmastr.nsf/Web/Flashes
- FAQs to assist you with the installation, use, and management of z/OS:
  www.ibm.com/support/techdocs/atsmastr.nsf/Web/FAQs
- White papers, presentations, and more at www.ibm.com/support/techdocs/atsmastr.nsf/Web/Techdocs
- Technotes that includes best practices, performance evaluations, recent enhancements and helpful hints and tips:
  www.ibm.com/support/techdocs/atsmastr.nsf/Web/Technotes

IBM Redbooks® provide positioning and value guidance, installation and implementation experiences, typical solution scenarios, and step-by-step “how-to” guidelines:

www.redbooks.ibm.com/
Dumps
Chapter 3. Best practices for large stand-alone dump

This information describes a set of best practices for optimizing stand-alone dump (SADMP) data capture, optimizing problem analysis time, and ensuring that the stand-alone dump is successful at capturing the necessary information for use by IBM Support. In particular, the following areas:

- “Planning a multivolume stand-alone dump data set”
- “Creating the multivolume SADUMP” on page 26
- “Defining a dump directory for large stand-alone and SVC dumps” on page 26
- “Preparing the dump for further processing with IPCS COPYDUMP” on page 27
- “Compressing data for faster transmission and analysis” on page 28
- “Using PUTDOC to transmit the data to IBM” on page 28
- “Setting up remote access” on page 28
- “Testing your stand-alone dump operations” on page 29
- “Automating the SADMP process” on page 29, which includes “Sample JCL for post-processing” on page 29
- “IBM System Test example” on page 31

This information updates the existing stand-alone dump best practices information previously documented in:

- “z/OS Best Practices: Large stand-alone dump handling,” found by searching for TD103286 at Techdocs www.ibm.com/support.techdocs/atmsr.nsf/Web/Technotes

Planning a multivolume stand-alone dump data set

Plan a multivolume stand-alone dump data set that places each volume on a separate DASD volume on a separate control unit. You can achieve the best dump performance when the dump is taken to a multivolume DASD stand-alone dump data set. Stand-alone dump exploits multiple, independent volume paths to accelerate data recording. The dump data set is actually spread across all of the specified volumes, not each volume in succession. They should not be treated as multiple single data sets. See the topic on “Creating the multivolume SADUMP” on page 26.

One of the key performance elements of stand-alone dump is the rate at which data writes to DASD. Modern DASD uses cache in the control unit to improve the performance of write operations. The placement of the multivolume stand-alone dump data set across logical subsystems (LSS) needs to avoid filling the bus or cache within the DASD with the data to be written. When the bus or cache is full of data to be written, the speed of the DASD is reduced to the (slower) speed at which the data can be written to the physical media.

There are significant performance improvements when writing the data to a multivolume stand-alone dump data set, or to specific types of DASD. For more
Best practices for large dumps

When defining your placement of a multivolume stand-alone dump data set, use the following guidelines:

1. Configure each volume on a separate logical subsystem (LSS) to ensure maximum parallel operation. You can achieve the best performance of stand-alone dump when the multivolume data sets have the most separation. That is, separate physical control units and separate channel paths.

2. Configure, if possible, the control units to minimize the occurrence of other activity at the time of the stand-alone dump. For example, DB2 database recovery writing to a local database volume on the same control unit as the stand-alone dump volume can result in slower dump speed and might affect the elapsed time needed to restart an alternative DB2 on an LPAR that is still running.

3. Use FICON-attached DASD volumes, when possible, to yield the best data rates. FICON channels can deliver much better performance than ESCON channels. However, with sufficient infrastructure and I/O tuning, an ESCON configuration can still deliver high performance.

4. Dedicate more DASD volumes to SADUMP, up to the maximum of 32 volumes, for better overall performance. IPCS offers a SADMP Dump Data Set Utility, available from the IPCS Utility menu. From the data set utility panel, you can specify whether to define, clear, or reallocate your stand-alone dump data set, specify its name and the volume serial numbers for the SADMP “stripes”. This panel will then start the SADMP allocation program to define the data set that you requested. The volume names, device type, and allocated space are also confirmed. While it is not recommended by IBM, stand-alone dump can also be written to a fast tape subsystem. When directing the SADUMP to a tape drive, the dump only uses a single device and does not prompt for another device, so you cannot switch back to using a DASD device for that stand-alone dump.

Creating the multivolume SADUMP

Use the AMDSADDD utility to define a stand-alone dump data set. Specify a volume list (VOLLIST) in AMDSADDD to designate a list of VOLSERS corresponding to each DASD volume making up the data set. You can allocate a multivolume data set using the specified list of volumes. The device number of the first volume is used to specify the data set to stand-alone dump. Again, each volume should be on a different LSS to ensure parallelism when writing the dump. For a sample job that uses AMDSADDD to generate the SADMP data set, see "IBM System Test example" on page 31. Be sure to catalog your SADMP data set to prevent the possibility of accessing the wrong version of the data set when using IPCS COPYDUMP later.

For additional details, see the topic on Using the AMDSADDD utility in z/OS MVS Diagnosis: Tools and Service Aids.

Defining a dump directory for large stand-alone and SVC dumps

Choosing the right attributes is the key to facilitating post-processing of large stand-alone dumps and large SVC dumps. IPCS is used to consolidate and extract ASIDs from the dump, format, and analyze the dump. IPCS uses a dump directory to maintain information about the layout and content of the dump. The dump directory is a VSAM data set that you can tune for optimal performance.
You can improve IPCS performance by reducing the number of control interval (CI) splits during initialization and analysis of dumps. To do this, specify the RECORDSIZE parameter in BLSCDDIR (shipped in SYS1.SBLSCLI0). In z/OS V1R7 and above, the RECORDSIZE parameter in BLSCDDIR is 'RECORDSIZE (2560 3072)' and has proven to yield well performing CISIZEs for the data portion of the data set. IBM recommends using this CI size specification prior to z/OS V1R7. To allow IPCS to be more efficient in its processing, it is recommended that you delete old dump references from the directory periodically (especially stand-alone dumps).

Notes:
1. When IBM System Test uses BLSCDDIR, they specify a CI size of 24,576 and a BUFSIZE of X'100000'.
2. You can tune the RECORDSIZE parameter by observing the number of CI splits using standard VSAM data set analysis techniques, such as the LISTCAT command.

Preparing the dump for further processing with IPCS COPYDUMP

Before you begin: If you are using IPCS on z/OS V1R7, apply PTF UA26080 to fix a problem that causes long IPCS initialization time.

After a stand-alone dump is taken to a multivolume data set, it needs to be post-processed before IPCS or other tools can view it. The IPCS COPYDUMP utility reads and processes the multivolume stand-alone dump faster than IEBGENER, and produces a merged dump ordered by ASID. The COPYDUMP utility processes the dump volumes in parallel, allowing the original dump to be read faster. Using COPYDUMP also helps IBM process the dump more quickly because it eliminates the need to reprocess the dump at IBM. Here is how it works:

1. Use IPCS COPYDUMP to produce a merged dump data set from the multivolume stand-alone dump, and a subset of the original stand-alone dump (ASIDs 1-20). "Sample JCL for post-processing" on page 29 contains a sample batch job.
2. Ensure that the output data set specified to COPYDUMP is DFSMS-striped with at least eight stripes.
3. Catalog the output dump data set to allow IPCS to access it properly.
4. Send the subset dump to IBM using a program like "AMATERSE for FTP" on page 22. This is a smaller data set, which takes less time to send through the FTP program to IBM.
5. Keep the merged dump for later use by IBM Support, if necessary.

You can run COPYDUMP to produce a merged version of the entire multivolume stand-alone dump, or to extract a subset of the address spaces contained in the original dump and written to the merged output data set. IBM recommends that you use two COPYDUMP jobs in parallel to produce a full merged dump and a subset merged dump. The subset dump will contain ASIDs 1-20 with the primary system components of the operating system.

IPCS performance is improved when the dump being processed (the COPYDUMP output) is DFSMS-striped. Placing the dump into a data set with at least eight stripes has shown marked improvement in IPCS response (when IBM is analyzing the problem).

A subset of ASIDs can be extracted from the full stand-alone dump into a separate data set and sent to IBM using COPYDUMP. This has been shown to reduce the
Best practices for large dumps

data transferred by roughly 30% to 40% compared to transmitting a full
stand-alone dump. Include the first 20 system ASIDs in the COPYDUMP and the
CATALOG and JES ASIDs, if known. This allows the base system address spaces to
be extracted from the dump. Also included any other ASIDs known to be involved
in the problem or requested by IBM Support.

The syntax of the IPCS COPYDUMP command is:

COPYDUMP ODS('OUTPUT DATASET NAME')
ASIDLIST(1:20,JES_ASID,Catalog_ASID,Problem_ASID)
IDS('INPUT DATASET NAME') NOCONFIRM

If the JES, Catalog, and problem ASIDs are not known, the syntax is:

COPYDUMP ODS('OUTPUT DATASET NAME')
ASIDLIST(1:20)
IDS('INPUT DATASET NAME') NOCONFIRM

Again, ensure that the specified output data set name supports DFSMS striping.

Compressing data for faster transmission and analysis

Compress dumps before sending the data to IBM using FTP. In z/OS V1R9 and
above, use AMATERSE. Otherwise, use TRSMAIN. You might need to encrypt the
resulting data set, so that the data is secure when it arrives at one of the IBM
Support staging points (TESTCASE or ECUREP).

- For z/OS V1R9 and above, see the topic on AMATERSE in z/OS MVS Diagnosis:
  Tools and Service Aids
- For z/OS V1R8 and earlier, download TRSMAIN from the IBM support Web
  site: https://service.software.ibm.com/s390/support
- If you require that the data be encrypted prior to sending it to IBM, place the
  necessary decryption information in the PMR for IBM Support to use. For more
  information on using IBM Encryption Facility for z/OS, see
  www.ibm.com/de/support/ecurep/mvs_encryption.html

Using PUTDOC to transmit the data to IBM

When sending large dumps to IBM, use the PUTDOC facility to allow the dump to
be split into multiple files and then sent using FTP. Sending a large dump in
smaller segments allows recovery of the data in the event of a network failure is
limited to the segment in process.

- For complete instructions on the use of the PUTDOC facility, see:
  techsupport.services.ibm.com/server/nav/zSeries/putdoc/putdoc.html
- See “Sample JCL for post-processing” on page 29 for the JCL example.

Setting up remote access

Set up remote access through Remote Screen Viewing Support Facility (RSVSF) or
Assist On-Site (AOS) to allow IBM to remotely view your dump in time-critical
situations. Remote access products allow you to permit IBM Support personnel to
immediately log into an IPCS session and view available documentation with no
initial data transfer. Choices include:

- Assist On Site (AOS) available for use worldwide at www.ibm.com/support/
  assistonsite).
- Remote Screen Viewing Support Facility (RSVSF), contact your service
  representative for details.
Best practices for large dumps

- OnTop available for use in Europe, Northeast Europe, and Southwest Europe, contact your service representative for details.

This should always be the first option in a time-critical situation. Rapid viewing of the documentation has the advantage of allowing IBM Support to itemize or customize any additional documentation they may want to send to IBM for the given situation. If documentation is required to be sent through FTP, the analysis of the current documentation can continue while the requested documentation is in transit. In many cases, sending a subset of the stand-alone dump to IBM can prove sufficient for problem resolution as the complete stand-alone dump is not always required for diagnosis of a given problem.

Testing your stand-alone dump operations

It is critical for your Operations staff to train and practice taking a stand-alone dump so that they are familiar with the procedure, and to ensure that all data sets are set up properly before you run into a critical situation. This includes the process and set up for:

- Taking a SADUMP as part of the standard scheduled shutdown of an LPAR
- Using COPYDUMP to obtain the merged dump and the subset dump.

If the dump resides in a DASD dump data set, IBM recommends that you copy the dump to another data set for IPCS processing and clear (re-initialize) the dump data set using the AMDSADDD or IPCS SADMP dump data set utilities. For more information, see the topic on “Using the AMDSADDD utility” in Using the IPCS Dialog in z/OS MVS IPCS User’s Guide.

The best practice is to rehearse taking a stand-alone dump during scheduled disaster recovery drills. You can also consider practicing when migrating to a new z/OS release, or when moving to a new processor. If you have a test LPAR that you use to train your operations staff, one part of that training might be to take a stand-alone dump following your local procedures and prepare how to react to stand-alone dump messages.

Automating the SADMP process

The following sample JCL can help you automate several best practices. The result is two “steps” that can be run as background jobs:

1. Use IPCS COPYDUMP to merge the data and produce a single data set to send to IBM. Because the JCL requires invoking IPCS in a background TSO environment, it is not possible to obtain condition code information from the COPYDUMP “step” to determine whether to invoke the preparation step. That means you must manually examine the results of the COPYDUMP step.

2. Use the “Preparation” job, which will compress the output data set produced by COPYDUMP, encrypt the compressed version, and send the final result through FTP to IBM using PUTDOC.

Tip: Beginning with z/OS V1R10, you can use the AutoIPL function to instruct z/OS to take a stand-alone dump in lieu of loading a disabled wait state. See z/OS MVS Planning: Operations for details.

Sample JCL for post-processing

Post-processing of a stand-alone dump needs to occur in two steps:
Best practices for large dumps

1. Run IPCS COPYDUMP to merge the data and produce a single data set to send to IBM. Examine the output from the run step to ensure that the COPYDUMP ran correctly. This JCL is identified below as *** IPCS COPYDUMP ****.

2. Run the following JCL, which will terse the resulting (striped) dump data set, encrypt the tersed version, and send it through FTP to IBM using PUTDOC. This JCL is identified below as *** TERSE, ENCRYPT and FTP ****.

You can tailor the following JCL to process the data sets to be transmitted to the FTP server. Turn off the LINE NUMBERING in following job.

*** IPCS COPYDUMP ****
//IPCSCPYD JOB MSGLEVEL=(2,1),....
// CLASS=V,NOTIFY=&SYSUID.,MSGCLASS=H
//*********************************************************************
//* IN DD IS USED TO POINT TO THE SOURCE OF INPUT WHICH WOULD BE
//* THE SYS1.SADMP... DATASET
//* OUT DD IS USED TO POINT TO THE OUTPUT OF THE COPYDUMP
//* WHERE PPPPPP SHOULD BE THE NUMBER OF CYLINDERS FOR PRIMARY
//* SSSS SHOULD BE THE NUMBER OF CYLINDERS FOR SECONDARY
//* &DATACLAS SHOULD BE THE DATACLAS
//* &MGMTCLAS SHOULD BE THE MGMTCLAS
//* &STORCLAS SHOULD BE THE STORCLAS
//* IPCSDDIR DD DEFINING &SYUID..COPYDUMP.DDIR WITH NON-COMPRESS
//* DATACLAS
//* COPYDUMP SUBCOMMAND TO REQUEST FIRST 20 ADDRESS SPACES
//* IF JES OR CATALOG WERE NOT AMONG THE FIRST 20 ADDRESS SPACES
//* XXX AND YYY SHOULD BE USED FOR THESE TWO SUBSYSTEM ASIDS
//*********************************************************************
//RUN EXEC PGM=IKJEFT01,REGION=200096K,DYNAMNBR=50
//IPCSPRNT DD SYSOUT=H
//IPCSTOC DD SYSOUT=H
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSTSPRT DD SYSOUT=H
//IN DD DISP=SHR,DSN=SYS1.SADMP.....
//OUT DD DISP=(NEW,CATLOG),DSN=OUTPUT.DATASET.NAME
//SPACE=(CYL,(PPPPP,SSSS),RLSE),DATACLAS=&DATACLAS,
// MGMTCLAS=&MGMTCLAS,STORCLAS=&STORCLAS
//SYSTSIN DD *
EX 'SYS1.SBLSCLI0(BLSCDDIR)' 'DSN(&SYSUID..COPYDUMP.DDIR)+RECORDS(90000)DATACLAS(NOCOMP)MGMTCLAS(DMGDEBUG)'
IPCS NOPARM
COPYDUMP IFILE(IN) OFILE(OUT) ASIDLIST(1:20,XXX,YYY) NOCONFIRM END
/*

==== TERSE, ENCRYPT and FTP ====
//TRENCFTP JOB CLASS=I,.....
// NOTIFY=&SYSUID.
//JOBLIB DD DISP=SHR,DSN=PS_WITH_TERSE_ENCRY_PGM
//TERSE EXEC PGM=TRSMAIN,PARM=PACK
//SYSPRINT DD SYSOUT=H
//INFILE DD DISP=SHR,DSN=SOURCE OF DUMP
//OUTFILE DD DISP=(NEW,CATLG),
// DSN=&SYSUID..PMR....TERSE.OUTFILE
// UNIT=SYSDAL,
// DATACLAS=COMPRESS,
// SPACE=(CYL,(PPPPP,SSSS),RLSE)
//DECRIPT EXEC PGM=FTPENCRD,PARM='PASSCODE',COND=(0,NE)
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//FIN DD DISP=SHR,DSN=*.TERSE.OUTFILE
//OUT DD DSN=&SYSUID..PMR.....TERSENCRP,
// DCB=(DSORG=PS,RECFM=FB,LECL=1024),
// DISP=(NEW,CATLG),UNIT=SYSDAL,
// DATACLAS=COMPRESS,
// SPACE=(CYL,(PPPPP,SSSS),RLSE)
IBM System Test example

In a recent set of tests performed by the IBM System z Product Evaluation Test team, a 12-volume configuration was set up to support a stand-alone dump of a 152 GB real memory system:

- Three Enterprise Storage Server (ESS) subsystems were used:
  - ESS 2105 F20 - 2 FICON 2 GB CHP, 8 GB cache
  - ESS 2105 mod. 800 - 8 FICON 2 GB CHP, 8 GB cache
  - DS6000 1750 mod. 511 - 6 FICON 2GB CHP, 1.3 GB cache.

- Four volumes per CP were defined:
  - Each volume on a unique LSS
  - Each volume as 14902 cylinders
  - The DSTYPE=LARGE attribute was used.

Here is an example of the AMDSADDD JCL that used the DASD configuration:

```bash
cd mvs/toibm
bin
PUT PMR......TRSENCRP PMR......TRS.ENCRP64
quit
/*

Best practices for large dumps

In a recent set of tests performed by the IBM System z Product Evaluation Test team, a 12-volume configuration was set up to support a stand-alone dump of a 152 GB real memory system:

- Three Enterprise Storage Server (ESS) subsystems were used:
  - ESS 2105 F20 - 2 FICON 2 GB CHP, 8 GB cache
  - ESS 2105 mod. 800 - 8 FICON 2 GB CHP, 8 GB cache
  - DS6000 1750 mod. 511 - 6 FICON 2GB CHP, 1.3 GB cache.

- Four volumes per CP were defined:
  - Each volume on a unique LSS
  - Each volume as 14902 cylinders
  - The DSTYPE=LARGE attribute was used.

Here is an example of the AMDSADDD JCL that used the DASD configuration:

```
Best practices for large dumps
Part 2. Predictive Failure Analysis

Soft failures are abnormal yet allowable behaviors that can slowly lead to the degradation of the operating system. To help eliminate soft failures, z/OS has developed Predictive Failure Analysis (PFA). PFA is designed to predict if a soft failure will occur sometime in the future and to identify the cause while keeping the base operating system components stateless. PFA is intended to detect abnormal behavior early enough to allow you to correct the problem before it affects your business. PFA uses remote checks from IBM Health Checker for z/OS to collect data about your installation. Next, PFA uses machine learning to analyze this historical data to identify abnormal behavior. It warns you by issuing an exception message when a system trend might cause a problem. To help you correct the problem, it identifies a list of potential issues. This section covers the following topics:

- Chapter 4, “Predictive Failure Analysis overview and installation,” on page 35
- Chapter 5, “Managing PFA checks,” on page 43
- Chapter 6, “Predictive Failure Analysis checks,” on page 49
- Chapter 7, “AIRH Messages,” on page 73
Chapter 4. Predictive Failure Analysis overview and installation

This chapter contains the following information:
- “Avoiding soft failures”
- “Overview of Predictive Failure Analysis”
  - “How PFA works with a typical remote check” on page 36
  - “How PFA interacts with IBM Health Checker for z/OS” on page 37
- “Installing PFA” on page 37
  - “Installing PFA in a z/OS UNIX shared file system environment” on page 39
  - “Updating the Java path” on page 41

Avoiding soft failures

Unlike typical problems or hard failures that have a clear start and a clear cause, soft failures are caused by abnormal, but allowable behavior. Because the cause of the problem is dependent on a certain sequence or combination of events that are unique and infrequent, a solution is often difficult to determine. Multiple atypical, but legal actions performed by components on the z/OS image cause most soft failures. By design, most components of z/OS are stateless and are therefore unable to detect soft failures caused by atypical behavior.

A classic example is the exhaustion of common storage usage. A low priority, authorized task obtains common storage, but obtains significantly more common storage than usual. Then, a critical authorized system component fails while attempting to obtain a normal amount of common storage. Although the problem occurs in the second critical component, this second component is actually the victim. The first component caused the problem and is considered the villain. Soft failures usually occur in four generic areas:

- Exhaustion of shared resources
- Recurring or recursive failures often caused by damage to critical control structures
- Serialization problems such as classic deadlocks and priority inversions
- Unexpected state transition

z/OS has developed Predictive Failure Analysis (PFA) to help eliminate these soft failures.

Overview of Predictive Failure Analysis

Predictive Failure Analysis (PFA) is designed to predict potential problems with your systems. PFA extends availability by going beyond failure detection to predict problems before they occur. PFA provides this support using remote checks from IBM Health Checker for z/OS to collect data about your installation. Using this data, PFA constructs a model of the expected (future) behavior of the z/OS images, compares the actual behavior with the expected behavior, and if the behavior is abnormal, PFA issues a health check exception. PFA uses a z/OS UNIX System Services (z/OS UNIX) file system to manage the historical and problem data that it collects.
Here is an LPAR view of the PFA components:

**Figure 1. LPAR view of the PFA components**

PFA creates report output in the following ways:

- In a z/OS UNIX file that stores the list of suspect tasks. See the individual check for a description of the directory and file names.
- In an IBM Health Checker for z/OS report that is displayed by z/OS System Display and Search Facility (SDSF) and the message buffer.
- Your installation can also set up IBM Health Checker for z/OS to send output to a log stream. After you set it up, you can use the HZSPRINT utility to view PFA check output in the message buffer or in the log stream. For complete details, see [Using the HZSPRINT utility](#) in [IBM Health Checker for z/OS: User’s Guide](#).

**How PFA works with a typical remote check**

PFA_COMMON_STORAGE_USAGE is a remote check that evaluates the common storage use of each system. PFA, running in its own address space, periodically collects common storage area (CSA + SQA) data from the system on which the check is running. The check writes the CSA usage data, at intervals, to a z/OS UNIX file. The check identifies a list of common storage users that are abnormal and that might contribute to exhausting common storage. PFA issues an exception message to alert you if there is a potential common storage problem and provides a list of suspect tasks. You can then examine the list and stop the cause of the potential problem or move critical work off the LPAR.
How PFA interacts with IBM Health Checker for z/OS

When PFA issues an exception, the PFA check’s WTOTYPE parameter changes to NONE in IBM Health Checker for z/OS so that the check does not continue to issue exceptions to the console until more data is collected and new predictions are made. The check continues to run at the defined interval so that the latest exception report data is available using the CK panel in SDSF. If the latest comparisons indicate the exception is no longer occurring, the check’s WTOTYPE is reset to its original value to enable future exceptions. After modeling has successfully created new predictions, if the latest comparisons still indicate an exception is occurring, the check’s WTOTYPE is reset to its original value and the check is immediately run to make new comparisons and issue a new exception based on the original WTOTYPE, if needed. If the WTOTYPE is changed from NONE to another type during this time, PFA does not overlay this new type.

What’s new for PFA installation in V1R11?

APAR OA30358 changes the Java level required for Predictive Failure Analysis from Java 1.4 to Java 5.0 or later with IBM 31-bit SDK for z/OS. If you start PFA before applying APAR OA30358, you can continue using Java 1.4 until you have a convenient time to move it to Java 5.0 or higher. When you move to Java 5.0 or higher, you must edit the .ini file for each check. See “Updating the Java path” on page 41.

Installing PFA

Before you begin: Before installing PFA in your environment, you must initialize z/OS UNIX and install the following products on your system:

- z/OS V1R10 and later
- Java 5.0 or later with IBM 31-bit SDK for z/OS. For the appropriate Java version for your release, see the z/OS Java home page at [www.ibm.com/servers/eserver/zseries/software/java/](http://www.ibm.com/servers/eserver/zseries/software/java/)

Restriction: PFA does not support the IBM 64-bit SDK for z/OS.

- IBM Health Checker for z/OS. You must be familiar with the set up for IBM Health Checker for z/OS. Most of the setup for PFA involves security definitions that are similar to the setup for any other started task and remote check. You must ensure that both PFA and IBM Health Checker for z/OS have access to the necessary resources including z/OS UNIX. For IBM Health Checker for z/OS details, see Setting up IBM Health Checker for z/OS in [IBM Health Checker for z/OS: User’s Guide](http://www.ibm.com/servers/eserver/zseries/software/hc/zos/matsprod3490a.pdf).

Guidelines:

1. The examples this procedure shows are for illustrative purposes only. Replace the example parameters with the correct specifications for your environment.
2. The Distributed File Service zSeries File System (zFS) is a z/OS UNIX file system that contains files and directories that can be accessed with z/OS UNIX application programming interfaces (APIs) and that support access control lists (ACLs). In this documentation, all references to the z/OS UNIX file system assume that you are using zFS. For complete zFS details, see [z/OS Distributed File Service zSeries File System Administration](http://www.ibm.com/servers/eserver/zseries/software/fs/zfs/). If z/OS UNIX is shut down for any reason, restart PFA.
4. Each PFA check manages its own data by capturing and modeling the appropriate diagnostic information and storing it in a check-specific data
directory. When that information is no longer useful, PFA deletes it. Any additional check-specific file handling information is found in the check documentation.

If your installation is migrating from z/OS V1R10 to V1R11 and previously started PFA, the directories for PFA_COMMON_STORAGE_USAGE and PFA_LOGREC_ARRIVAL_RATE checks already exist and contain history data. If you want to preserve the history data, use the AIRSHKP.sh script. If you want to start with clean directories, use the AIRSHREP.sh script.

**Steps for installing PFA**

Use the following steps to set up PFA with RACF® and z/OS UNIX:

1. Create a user ID to define the location in the z/OS UNIX file system that stores the PFA data and connects the PFA user ID to an existing or new RACF group. The home directory of the user ID that owns the PFA started task must match where the install script is run.

   If you are using PFA in a sysplex that shares file systems for z/OS UNIX, use a unique directory for each LPAR so that the event data that PFA writes to the file system is stored separately for each system. For details, see “Installing PFA in a z/OS UNIX shared file system environment” on page 39.

   a. Create a new userid (UID) to own the PFA. For example, pfauser. The PFA user ID must be unique; do not use the same user ID that is assigned to the IBM Health Checker for z/OS.

   b. Define the PFA started task by creating a RACF profile for the pfauser with the following items:

      - OMVS segment with a UID parameter (for example, omvs(uid(7)))
      - Home directory (for example, home(/pfa))
      - PROGRAM pathname of /bin/sh (for example, program(/bin/sh))

   **Examples:**

   This example shows how you can define and connect a new user to RACF. Ensure that you replace the parameters with the correct settings for your installation.

   ```
   ADDUSER pfauser
   OMVS(UID(7) HOME(/pfa) PROGRAM(/bin/sh)) PASSWORD(sys1)
   ADDGROUP OMVSGRP OMVS(GID(46))
   CONNECT pfauser GROUP(OMVSGRP)
   ```

   This example shows how you can change the information in a user’s RACF profile:

   ```
   altuser pfauser omvs(uid(7) shared home(/pfa) program(/bin/sh))
   ```

   For information about Security Server RACF, see the ADDUSER and ADDGROUP sections in z/OS Security Server RACF Command Language Reference.

2. Add the PFA task to the STARTED class table in RACF and refresh, if necessary. For example:

   ```
   SETROPTS GENERIC(STARTED)
   RDEFINE STARTED PFA.* STDATA(USER(pfauser) GROUP(OMVSGRP))
   SETROPTS CLASSACT(STARTED)
   SETROPTS RACLIST(STARTED)
   ```

   If you have already activated RACLST in the STARTED class, the last statement is:

   ```
   SETROPTS RACLST(STARTED) REFRESH
   ```

   For more information, see the following information:
3. Copy the sample PFA procedure, AIRPROC, from SYS1.SAMPLIB to the PFA member of SYS1.PROCLIB data set. If SMP/E does not write the executable code in the z/OS UNIX file system to `PARM='path=(/usr/lpp/bcp)'`, change the PARM value in AIRPROC to the path in which you store the executable code.

4. Use the AIRSHREP.sh install script to name the new members and create the z/OS UNIX directories that store the PFA data.

   **Guideline:** The directories and data that are created by the install process must be in the home directory of the user ID that owns the started task. PFA writes the historical data necessary for predictions to the directories created by the installation script.

   Perform the following steps to create the PFA directories using the shell script from the OMVS command line.
   
   **a.** Make the home directory of the pfauser the current directory. For example: `cd/<homedir>`
   
   **b.** If necessary, edit the Java PATH= line and the LIBPATH= statement with the appropriate Java library, before or after running the script, to point to the correct location for the Java code. See “before running airshrep” on page 41 and “after running airshrep” on page 42.
   
   **c.** Run the shell script using the following command:

       `/usr/lpp/bcp/AIRSHREP.sh`

5. Allow the appropriate people access to the PFA results in SDSF and the z/OS UNIX file system. Both systems use standard security controls.

6. Update the COMMNDxx parmlib member with the PFA procedure to ensure that PFA restarts on IPL as shown here:

   ```
   START pfa_procnme
   ```


7. Update your WLM Service Class policy for PFA to be the same priority that your installation uses for monitoring products like RMF. For more information about defining service classes, see “Defining service classes and performance goals” in [z/OS MVS Planning: Workload Management](#).

8. Customize your system settings for PFA:
   
   **a.** Update your system automation to look for PFA exception messages. For complete details, see the topic about Approaches to automation with IBM Health Checker for z/OS in [IBM Health Checker for z/OS: User’s Guide](#).
   
   **b.** Follow the guidelines for correcting problems by reviewing the check-specific best practice.
   
   **c.** After the checks have been running for a while, you might decide that the default parameters are not appropriate in your environment. You can customize the PFA checks using the check-specific parameters. For details, see the check-specific parameters.

**Installing PFA in a z/OS UNIX shared file system environment**

In this procedure you create a z/OS UNIX file system that is shared among members of the sysplex with directories that are local to the LPAR. This procedure
uses zSeries File System (zFS) because it is the strategic file system. This procedure enables you to define one started task and user ID that permits PFA to write files to a system-unique directory.

1. Define the file systems as one for each LPAR. You can use the TSO ISHELL (ISPF shell) panel to define and format the zFS file system. In this example, the Integration Test team used OMVSSPT.Z1.PFA.ZFS as the example system.

For more information about using ISHELL, see the topic “Invoking the ISPF shell” in z/OS UNIX System Services User’s Guide.

2. After defining the file systems for each LPAR, define a symbolic link (also called a symlink) to the sysplex root. From the root directory, enter the following command using the UID you assigned to pfauser:

```
cd
ln -s \$SYSNAME/pfa pfa
```

This results in a home directory of /systemname/pfa.

**Guideline:** The home directory of user ID that owns the started task is where PFA expects the directories and data created by the install process to reside. PFA writes the historical data it needs to make predictions to the directories created by the installation process.

3. Create the PFA directory in each of the system directories by entering the following commands for each of your system names. For example, the command for system Z1 is:

```
mkdir /Z1/pfa
mkdir /Z2/pfa
mkdir /Z3/pfa
mkdir /Z4/pfa
```

4. Create the new file systems (one for each system) and mount them at the appropriate system mount point. For example, for OMVSSPT.Z1.PFA.ZFS, the mount point is Z1/pfa:

```
OMVSSPT.Z1.PFA.ZFS
OMVSSPT.Z2.PFA.ZFS
OMVSSPT.Z3.PFA.ZFS
OMVSSPT.Z4.PFA.ZFS
```

Here is an example of the file system attributes:
**File System Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system name:</td>
<td>OMVSSPT.Z1.PFA.ZFS</td>
</tr>
<tr>
<td>Aggregate name:</td>
<td>OMVSSPT.Z1.PFA.ZFS</td>
</tr>
<tr>
<td>Mount point:</td>
<td>/Z1/pfa</td>
</tr>
<tr>
<td>Status:</td>
<td>Available</td>
</tr>
<tr>
<td>File system type:</td>
<td>ZFS</td>
</tr>
<tr>
<td>Mount mode:</td>
<td>R/W</td>
</tr>
<tr>
<td>Device number:</td>
<td>28412</td>
</tr>
<tr>
<td>Type number:</td>
<td>1</td>
</tr>
<tr>
<td>DD name:</td>
<td></td>
</tr>
<tr>
<td>Block size:</td>
<td>1024</td>
</tr>
<tr>
<td>Total blocks:</td>
<td>720000</td>
</tr>
<tr>
<td>Available blocks:</td>
<td>561716</td>
</tr>
</tbody>
</table>

See the topic [Managing the z/OS UNIX file system](#) in [z/OS UNIX System Services Planning](#).

5. Place an entry in the SYS1.PARMLIB(BPXPRMXX) member to mount the new file systems during IPL. (If you do not want to wait until the next IPL, you can manually mount these file systems.) Use the UNMOUNT attribute on the BPXPRMXX parmlib member to unmount the file system when OMVS or the LPAR is taken down. The file system mount point is `/sysname/pfa` for example:

```bash
SYS1.PARMLIB(BPXPRM00)
    MOUNT FILESYSTEM('OMVSSPT.&SYSNAME..PFA.ZFS') TYPE(ZFS)
    MODE(RDWR) MOUNTPOINT('/&SYSNAME./pfa') UNMOUNT
```

### Updating the Java path

Before you update the Java path, back up AIRSHREP.sh. If the path to the JVM is not correct, PFA stops running and issues AIR022I.

**Note:** The update to the PATH= and LIBPATH= lines is dependent of the level of Java that your installation uses. In Java 5.0, these lines typically point to:

`$JAVA_HOME/bin/classic`

In Java 6.0, these lines typically point to:

`$JAVA_HOME/lib/s390/classic`

Therefore, the value for the `$JAVA_HOME` variable is typically as follows:

- Java 5.0 - `/usr/lpp/java/J5.0` (/usr/lpp/java/J5.0/bin/classic)
- Java 6.0 - `/usr/lpp/java/J6.0` (/usr/lpp/java/J6.0/lib/s390/classic)

**Java update before running AIRSHREP.sh:** The JAVAPATH= line must be set to the SMP/E install path for the Java code for PFA. If your Java code is installed elsewhere, you must change this line to the path where the Java code is installed. The default is `/usr/lpp/bcp`. For example:

```
JAVAPATH= /usr/lpp/bcp
```
Update the PATH= and LIBPATH= statements in the .ini file for each check to point to the executable code needed for JNI calls. The following example shows typical paths:

```
echo 'PATH= /bin:/usr/lpp/java/J5.0/bin:/usr/lpp/java/J5.0/bin' >> ./PFA_COMMON_STORAGE_USAGE/ini
echo 'LIBPATH= /usr/lpp/java/J5.0/bin:/usr/lpp/java/J5.0/bin/classic:/lib:/usr/lib:'
```

*Figure 3. Example of the Java 5.0 path*

**Java update after running AIRSHREP.sh:** After you run AIRSHREP.sh from the OMVS command line, the PFA process creates a Java .ini file in the home directory of each of the PFA checks (in the example above, PFA_COMMON_STORAGE_USAGE). The .ini files contain a line that points to the path for the Java code. If the path to the JDK for your system is not the same as the default path in the .ini file, edit the line in each check to point to the appropriate path for your system.
Chapter 5. Managing PFA checks

You can display PFA checks using the `MODIFY pfa,DISPLAY` command as well as the IBM Health Checker for z/OS commands. To help you understand how to manage these differences, this section contains the following topics:

- “DISPLAY” on page 44 describes the `MODIFY pfa,DISPLAY` command.
- “Modifying PFA checks” on page 47 describes the unique differences PFA checks have from other checks.

For the IBM Health Checker for z/OS commands, see IBM Health Checker for z/OS: User’s Guide.

You can make installation updates to PFA checks that persist across check refreshes and restart IBM Health Checker for z/OS by activating IBM Health Checker for z/OS policies. You might do this if some check default values are not suitable for your environment or configuration. For complete details, see the topic on Creating IBM Health Checker for z/OS policies in IBM Health Checker for z/OS: User’s Guide.

Restriction: The IBM Health Checker for z/OS debug commands are not the same debug parameter that PFA checks use. For details, see “Modifying PFA checks” on page 47.
Purpose

MODIFY pfa,DISPLAY (f pfa,display) issues messages with information specified as different options (listed in “Options”).

When displaying PFA checks using the IBM Health Checker for z/OS commands, the user-supplied parameters listed contain only the parameters that were specified on the last update command not the cumulative set of modified parameters. Therefore, you must use the MODIFY pfa,DISPLAY command to display the check-specific parameters that are currently used by a PFA check. See “Modifying PFA checks” on page 47 for more information.

Format

DISPLAY
{
  [CHECKS [,filters] [,SUMMARY] | ,DETAIL]]
 |
  [filters [,SUMMARY] | ,DETAIL]]
 |
  [STATUS]
}

Options

CHECKS CHECKS displays information about PFA checks.
filters Filters specify which check or checks you want to take an action against. You can specify the wildcard character * for filters in the last position of the filter. An asterisk (*) represents any string having a length of zero or more characters.

Filters must be specified in one of the following formats:
CHECKS,CHECK=(check_name)
or
CHECK=(check_name)

check_name specifies the 1- through 32-character check name.

SUMMARY PFA issues message AIR013I with summary information about the specified checks. See “Example of DISPLAY SUMMARY message output” on page 45. For each check matching the specified filter, the following information is returned:
• Check name
• Indicator of whether the check is eligible to run at the next interval (ACTIVE(ENABLED)) in IBM Health Checker for z/OS
• The last successful collection time
• The last successful model time

Because SUMMARY is the default value, it does not need to be specified in the command.

DETAIL PFA issues message AIR018I with detailed information about the specified checks. See “Example of DISPLAY DETAIL message output” on page 46. For each check matching the specified filter, the following information is returned:
• Check name
DISPLAY

- Indicator of whether the check is eligible to run at the next interval (ACTIVE(ENABLED)) in IBM Health Checker for z/OS
- Total number of collections attempted
- Total number of successful collections
- The last time the collection ran
- The last successful collection time
- The next collection time
- Total number of models attempted
- Total number of successful models
- The last time the model ran
- The last successful model time
- The next model time
- The current settings of the check-specific parameters for this check:
  - The collection interval in minutes
  - The model interval in minutes
  - Indicator of whether to collect data and model data even if the check is not eligible to run (ACTIVE(ENABLED)) in IBM Health Checker for z/OS
  - Indicator if the check is generating additional diagnostic information
  - Any other parameters that are supported for this check

STATUS

PFA issues message AIR017I with general status information for PFA. See "Example of DISPLAY STATUS message output" on page 45. The following information is returned:
- The number of checks registered to PFA
- The number of PFA checks eligible to run (ACTIVE(ENABLED)) in IBM Health Checker for z/OS
- The number of collections currently queued
- The number of models currently queued
- The number of JVM terminations that have occurred since PFA started

Examples

Example of DISPLAY STATUS message output

<table>
<thead>
<tr>
<th>IR017I 10.31.32 PFA STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER OF CHECKS REGISTERED: 2</td>
</tr>
<tr>
<td>NUMBER OF CHECKS ACTIVE: 2</td>
</tr>
<tr>
<td>COUNT OF COLLECT QUEUE ELEMENTS: 0</td>
</tr>
<tr>
<td>COUNT OF MODEL QUEUE ELEMENTS: 0</td>
</tr>
<tr>
<td>COUNT OF JVM TERMINATIONS: 0</td>
</tr>
</tbody>
</table>

The DISPLAY STATUS message output displays in response to the following commands:
- f pfa,display
- f pfa,display,status

Example of DISPLAY SUMMARY message output
The DISPLAY SUMMARY message output displays in response to the following commands:

- `f pfa,display,checks` - results in all checks
- `f pfa,display,checks,check=(check_name)` - results in one check
- `f pfa,display,checks,check=(PFA_*)` - can result in > 1 check
- `f pfa,display,checks,check=(*),summary` - results in all checks
- `f pfa,display,check(check_name)` - results in one check
- `f pfa,display,check(PFA_*)` - can result in > 1 check
- `f pfa,display,check(*),summary` - results in all checks

Example of DISPLAY DETAIL message output

The DISPLAY DETAIL message output displays in response to the following commands:

- `f pfa,display,checks,detail` - results in all checks
- `f pfa,display,checks,check=(name),detail` - results in one check
- `f pfa,display,checks,check=(check_na*),detail` - can result in > 1 check
- `f pfa,display,checks,check=(*),detail` - results in all checks
- `f pfa,display,check(check_name),detail` - results in one check
- `f pfa,display,check(check_na*),detail` - can result in > 1 check
- `f pfa,display,check(*),detail` - results in all checks
Modifying PFA checks

PFA checks work differently than traditional checks. Although you can modify PFA checks using the IBM Health Checker for z/OS commands and policies (as described in [IBM Health Checker for z/OS: User’s Guide](#)), modifications to the PFA checks are unique in the following ways:

- You can use the MODIFY hzsproc command to modify individual parameters of PFA checks.

When specifying the PARM parameter on `f hzsproc,update` for PFA checks, you do not have to specify all check-specific parameters. The parameters that are not specified are not changed. If the parameters were not previously modified, the values remain the default values. If the parameters were previously modified, the previously specified values remain.

The following example of the MODIFY `hzsproc` command sets the debug parameter for PFA to ON.

`f hzsproc,update,check(IBMPFA,PFA_COMMON_STORAGE_USAGE),parm('debug(1)')`

The rest of the check-specific parameters retain their default values as follows:

**CHECK SPECIFIC PARAMETERS:**

```
COLLECTINT : 15
MODELINT : 360
COLLECTINACTIVE : 1=ON
DEBUG : 1=ON
THRESHOLD : 2
```

If the check is displayed using IBM Health Checker for z/OS interfaces, the user supplied parameters are displayed as either:

**USER SUPPLIED PARAMETERS:** debug(1)

or

**CHECK PARM:** debug(1)

Then, when the following command is entered:

`f hzsproc,update,check(IBMPFA,PFA_COMMON_STORAGE_USAGE),parm('collectint(10) modelint(60)')`

the values for the check change to:

**CHECK SPECIFIC PARAMETERS:**

```
collectint : 10
modelint : 60
collectinactive : 1=ON
debug : 1=ON
threshold : 5
```

If the check is displayed again using IBM Health Checker for z/OS interfaces, the user-supplied parameters that are displayed are the last ones specified on the MODIFY command as follows:

**USER SUPPLIED PARAMETERS:** collectint(10) modelint(60)

or

**CHECK PARM:** collectint(10) modelint(60)

This is not the cumulative list of parameters currently in use by the check. Therefore, to display the current parameter values being used by the check, use the `f pfa,display,checks,detail` command as listed in “DISPLAY” on page 44.

- **The debug parameter is a PFA check-specific parameter.**
The debug parameter is a check-specific parameter and not the same debug parameter as the one in IBM Health Checker for z/OS. The debug parameter within IBM Health Checker for z/OS applies only to the phase of performing the check when the interval is reached and not to all other phases, such as data collection and modeling that are done internally within the PFA checks. Therefore, the debug parameter within IBM Health Checker for z/OS has no meaning to PFA and is ignored.

To set the debug parameter for PFA checks, specify it as a parm as in the following example:

\[
f \text{hzsproc,update,check(IBMPFA,PFA\_CHECK\_NAME),parm('debug(1)')} \]

Each PFA check contains more information about its check-specific parameter.
Chapter 6. Predictive Failure Analysis checks

Predictive Failure Analysis (PFA) provides the following remote checks:

- "PFA_COMMON_STORAGE_USAGE"
- "PFA_LOGREC_ARRIVAL_RATE" on page 55
- "PFA_FRAMES_AND_SLOTS_USAGE" on page 61
- "PFA_MESSAGE_ARRIVAL_RATE" on page 66

PFA_COMMON_STORAGE_USAGE

Description:
The check is looking to see if there is a potential for storage in the common storage area (CSA) plus system queue area (SQA) to be exhausted in the upcoming predictive failure analysis (PFA) model interval. The PFA_COMMON_STORAGE_USAGE check detects three classes of common storage exhaustion:

- Spike
- Leak
- Creep

If PFA detects that there is a potential for the exhaustion of common storage, PFA issues exception message AIRH101E or AIRH109E and provides a list of suspect tasks. During the analysis, this check writes the CSA + SQA usage data at intervals to a z/OS UNIX System Services file in comma-separated value (.csv) format. The check identifies a list of users of common storage that might contribute to exhausting common storage. You can then examine the list and stop the cause of the potential problem or move critical work off the LPAR. See steps for preventing common storage failures. PFA also issues the following informational messages:

- AIRH102I
- AIRH103I
- AIRH111I

Reason for check:
If the system runs out of common storage, jobs and started tasks experience abends.

Best practice:
The best practice is to predict common storage problems before they occur, determine the cause of the problem, and take the appropriate action.

When IBM Health Checker for z/OS issues exception message AIRH101E or AIRH109E, PFA has predicted that the amount of storage allocated to the common storage area is in jeopardy of being exhausted. Use the following steps to determine the appropriate action:

1. Examine the Common Storage Usage Prediction Report issued with the exception message. This report contains the total current usage and predictions for storage below the line and above the line. It also contains up to fifteen “users” of common storage that have been predicted to use the most CSA + SQA in the next model interval. The cause of the problem is most likely within this list of the top predicted users. See Common storage usage output report on page 52 for the example report.

2. If the cause of the problem is not obvious from the common storage usage report, you can obtain additional information in the csadata report. The csadata report is a text file in comma-separated value (.csv) format and contains the historical data on the usage for each interval. You can export the csadata report into any spreadsheet-type program.
3. Determine which type of common storage problem is occurring by examining the symptoms, and then correct the behavior:

  - **Spike:** A piece of code uses more and more of the common storage area with usage growing linearly or exponentially over time. If the problem is caused by a spike, the csadata report contains one or more users that are in the last few intervals and that consume a significant and measurable amount of common storage.
    
    Determine if the job causing the spike can be stopped, canceled, or slowed without affecting the overall system behavior.

  - **Leak:** A piece of code returns some but not all of the storage, which results in more usage of the common storage area over time. If the problem is caused by a leak, look for the contributor that is on the list multiple times, but not in every interval.
    
    Determine if the job causing the leak can be stopped, canceled, or slowed down without affecting the overall system behavior.

  - **Creep:** The common storage area usage grows slowly reflecting the overall system usage, which means there is no individual user of CSA responsible for the storage exhaustion. If there is no job or address space that appears to be using an excessive or unusual amount of common storage, the amount of work being done by the LPAR is probably causing the usage of common storage to creep.
    
    Determine if the amount of work being sent to this LPAR can be reduced.

*Note:* Because of the random variation in common storage usage that typically occurs, PFA is sometimes unable to differentiate a leak or creep that is less than 750 bytes per second.

**z/OS releases the check applies to:**

z/OS V1R10 and later.

**Type of check:**

Remote

**Parameters accepted:**

Yes, as follows:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectint</td>
<td>15 Minutes</td>
<td>1</td>
<td>360</td>
<td>This parameter determines the time (in minutes) to run the data collector that determines the amount of common storage being used. The default is 15 minutes (15).</td>
</tr>
<tr>
<td>modelint</td>
<td>360 Minutes</td>
<td>4</td>
<td>1440</td>
<td>This parameter determines how often (in minutes) you want the system to analyze the data and construct a new common storage usage model or prediction. By default, PFA analyzes the data and constructs a new model every 6 hours (360 minutes). The model interval must be at least four times larger than the collection interval. If necessary modeling can occur more frequently.</td>
</tr>
<tr>
<td>threshold</td>
<td>2 Percent</td>
<td>1</td>
<td>100</td>
<td>The percentage of CSA + SQA capacity to add to the CSA + SQA capacity to produce the capacity value to use in comparisons. The threshold can be used to reduce false positive comparisons. Setting the threshold too high might cause exhaustion problems to be undetected. The default is 2 percent (2).</td>
</tr>
<tr>
<td>collectinactive</td>
<td>1 (on)</td>
<td>0 (off)</td>
<td>1 (on)</td>
<td>Defines whether data is collected and modeled even if the check is not eligible to run (is not ACTIVE(ENABLED)) in IBM Health Checker for z/OS.</td>
</tr>
</tbody>
</table>
Table 4. PFA_COMMON_STORAGE_USAGE check parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>0 (off)</td>
<td>0 (off)</td>
<td>1 (on)</td>
<td>This parameter (an integer of 0 or 1) is used at the direction of IBM service to generate additional diagnosis information for the IBM Support Center. This debug parameter is used in place of the IBM Health Checker for z/OS policy. See <a href="#">debug support</a>. The default is off (0).</td>
</tr>
</tbody>
</table>

To determine the status of the common storage usage check, issue `f pfa,display,check(pfa_common_storage_usage),detail` See ["Example of DISPLAY DETAIL message output" on page 46](#) for the complete command example. The following is an example of the output written to message AIR018I in SDSF user log (ULOG):

Ф PFA,DISPLAY,CHECK(PFA_COMMON_STORAGE_USAGE),DETAIL
AIR018I 16:20:21 PFA CHECK DETAIL
CHECK NAME: PFA_COMMON_STORAGE_USAGE
ACTIVE: YES
TOTAL COLLECTION COUNT: 5
SUCCESSFUL COLLECTION COUNT: 5
LAST COLLECTION TIME: 09/01/2008 10.18.22
LAST SUCCESSFUL COLLECTION TIME: 09/01/2008 10.18.22
NEXT COLLECTION TIME: 09/01/2008 10.33.22
TOTAL MODEL COUNT: 1
SUCCESSFUL MODEL COUNT: 1
LAST MODEL TIME: 09/01/2008 10.18.24
LAST SUCCESSFUL MODEL TIME: 09/01/2008 10.18.24
NEXT MODEL TIME: 09/01/2008 11.18.24
CHECK SPECIFIC PARAMETERS:
  COLLECTINT: 15
  MODELINT: 60
  COLLECTINACTIVE: 1=ON
  DEBUG: 0=OFF
  THRESHOLD: 5

User override of IBM values:

The following example shows keywords you can use to override check values either on a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. See Chapter 5, "Managing PFA checks," on page 43. You can copy and modify this statement to override the check defaults:

UPDATE CHECK(IBMPFA,PFA_COMMON_STORAGE_USAGE)
  ACTIVE
  SEVERITY(MEDIUM)
  INTERVAL(00:01)
  PARMS=('COLLECTINT(15)', 'MODELINT(360)', 'THRESHOLD(2)', 'COLLECTINACTIVE(1)', 'DEBUG(0)')
  DATE(20071101)
  REASON('Common storage usage is nearing the user defined threshold.')

Verbose support:

The check provides additional details in verbose mode. You can put a check into verbose mode either using the UPDATE,filters,VERBOSE=ON parameters on the MODIFY command or on a POLICY statement on an HZSPRMxx parmlib member.

Debug support:

The DEBUG parameter in IBM Health Checker for z/OS is ignored by this check. Rather, the debug parameter is a PFA check specific parameter. The IBM Health Checker for z/OS debug commands are not the same debug parameter that PFA checks use. For details, see "Modifying PFA checks" on page 47.
PFA_COMMON_STORAGE_USAGE

Reference:
For more information about PFA, see the topic on "Overview of Predictive Failure Analysis" on page 35.

Messages:
This check issues the following exception messages:
- AIRH101E
- AIRH109E

For additional message information, see the topics:
- Chapter 7, "AIRH Messages," on page 73
- AIR messages in z/OS MVS System Messages, Vol 1 (ABA-AOM)

SECLABEL recommended for MLS users:
SYSLOW

Output:
The common storage usage output report:

Common Storage Usage Prediction Report

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Last successful model time</td>
<td>11/06/2008 17:32:44</td>
<td></td>
</tr>
<tr>
<td>Next model time</td>
<td>11/06/2008 23:33:44</td>
<td></td>
</tr>
<tr>
<td>Model interval</td>
<td>360</td>
<td></td>
</tr>
<tr>
<td>Last successful collection time</td>
<td>11/06/2008 18:33:49</td>
<td></td>
</tr>
<tr>
<td>Next collection time</td>
<td>11/06/2008 18:48:49</td>
<td></td>
</tr>
<tr>
<td>Collection interval</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Below line CSA+SQA (in kilobytes):
- Current usage : 750
- Future prediction : 613
- Capacity when predicted: 5212

Above line CSA+SQA (in kilobytes):
- Current usage : 205555
- Future prediction : 235408
- Capacity when predicted: 526112

Top predicted users:

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Current Usage (in kilobytes)</th>
<th>Predicted Usage (in kilobytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSATST4</td>
<td>ABOVE</td>
<td>35002</td>
<td>40023</td>
</tr>
<tr>
<td>CSATST3</td>
<td>ABOVE</td>
<td>32364</td>
<td>33530</td>
</tr>
<tr>
<td>CSATST2</td>
<td>ABOVE</td>
<td>23647</td>
<td>23467</td>
</tr>
<tr>
<td>CSATST8</td>
<td>ABOVE</td>
<td>12367</td>
<td>13479</td>
</tr>
<tr>
<td>CSATST1</td>
<td>ABOVE</td>
<td>12456</td>
<td>12478</td>
</tr>
<tr>
<td>ZTTLARM0</td>
<td>ABOVE</td>
<td>3102</td>
<td>3110</td>
</tr>
<tr>
<td><em>SYSTEM</em></td>
<td>ABOVE</td>
<td>UNAVAILABLE</td>
<td>190</td>
</tr>
</tbody>
</table>

Figure 4. Common storage usage output report

- **Last successful model time**: The date and time of the last successful model for this check. The predictions on this report were generated at that time.
- **Next model time**: The date and time of the next model. The next model will recalculate the predictions.
- **Model interval**: The value in the configured MODELINT parameter for this check. If PFA determines new prediction calculations are necessary, modeling can occur earlier.
- **Last successful collection time**: The date and time of the last successful data collection for this check.
- **Next collection time**: The date and time of the next collection.
• Collection interval: The value in the configured COLLECTINT parameter for this check.
• Current® usage: The amount of CSA + SQA used in kilobytes when the check was run.
• Future prediction: The prediction of CSA + SQA usage for the end of the model interval.
• Capacity when predicted: The total defined CSA + SQA capacity (both used and unused) at the time the prediction was made. This value does not include the threshold percentage.
• Top predicted users: The jobs that were predicted to use the most CSA + SQA by the end of the model interval. This list is only printed if the check issues an exception or the debug parameter is on. The number of jobs printed can vary.
• Job name: The name of the job that is a top predicted user.
• Storage location: The storage location for the values in the row of the report.
  – ABOVE - above the line storage.
  – BELOW - below the line storage.
• Current usage (for job): The amount of CSA + SQA used in kilobytes when the check was run.
  The predicted usage for "SYSTEM" jobs is accumulated, but no attempt is made to accumulate the current usage for "SYSTEM" jobs. Therefore, UNAVAILABLE is printed for the current usage of "SYSTEM" jobs.
• Predicted usage (for job): The amount of CSA + SQA in kilobytes predicted to be used by this job at the end of the model interval.

Directories
When you install PFA_COMMON_STORAGE_USAGE, the shell script creates the following directories that hold the executable program, log, error, data store, intermediate, and results files.

Note: The content and names for these files are subject to change and cannot be used as programming interfaces; these files are documented only to provide help in diagnosing problems with PFA.

pfa_directory
This directory contains all the PFA checks and is pointed to by the home directory of the started task. The following files only contain data if errors are generated by the JVM:
• java.stderr (generated by JVM)
• java.stdout (generated by JVM)

pfa_directory/PFA_COMMON_STORAGE_USAGE/data
The directory for common storage usage that holds data and modeling results.

Results files:
• system_name.prediction - This file lists of the top users of common storage. The following example shows the common storage usage prediction report in .csv format, which is written to the system_name.prediction file:

```
A/*MASTER*/0001/EPRV+498C,2421.48
A/*MASTER*/0001/ICHSEC06+90CE,4469.76
A/*SYSTEM*/0000/CSAALLOC+21BE,5043.76
A/*SYSTEM*/0000/EPRV+16CE,4969.44
A/*SYSTEM*/0000/EPRV+23511A,6479.28
A/*SYSTEM*/0000/EPRV+4100C,11141.12
A/*SYSTEM*/0000/EPRV+4A00,4151.36
A/*SYSTEM*/0000/EPRV+5779.15
A/*SYSTEM*/0000/ERON+470AB6,7311.36
A/*SYSTEM*/0000/ESQA+C31140,5365.76
A/ABOVE /0001/RESIDUAL,68740.49
A/JES2/001B/HASJES20+114C88,5118.48
A/VTAMOSR3/0018/2518E968,44230.68
A/VTAMOSR3/0018/ISTORFPO+798,8601.60
B/BELLOW /0001/RESIDUAL,3128.88
```
The values found in the common storage usage prediction report in .csv format are as follows:
- **A** or **B**: The location in storage as either above or below the line.
- **MASTER**: The address space name.
- **0001**: The ASID.
- **EPRV+498C**: The program status word (PSW) of the location in storage from which the amount was requested.
- **2421.48**: The amount of storage predicted to be used by the address space in kilobytes.

The values found in the common storage usage prediction report in .html format are as follows:
- **User of Common Storage**: This is the identification of the user of common storage. It consists of the address space name, ASID, and PSW.
- **Instance Count**: The number of records with this user that were factored into the prediction model.
- **Current Estimated Common Storage Used**: The current amount of common storage used by this user in the last collection interval included in this model.
- **Prediction Look Forward Seconds**: The number of seconds the prediction should project into the future.
- **Predicted Common Storage Usage**: The predicted amount of common storage usage for this user.

Data store files:
- **system_name.prediction.html**: This file lists the top users of common storage in an .html report format. The values found in the common storage usage prediction report in .html format are as follows:
  - **User of Common Storage**: This is the identification of the user of common storage. It consists of the address space name, ASID, and PSW.
  - **Instance Count**: The number of records with this user that were factored into the prediction model.
  - **Current Estimated Common Storage Used**: The current amount of common storage used by this user in the last collection interval included in this model.
  - **Prediction Look Forward Seconds**: The number of seconds the prediction should project into the future.
  - **Predicted Common Storage Usage**: The predicted amount of common storage usage for this user.

Intermediate files:
- **system_name.csadata**: The input to modeling in CSV format. csadata saves one entry “top contributor” for each csaSum file and csaAll file. There are two entries for residuals: one for above the bar and one for below the bar.
- **system_name.mapmvs**: Convert PSW execution address to module name.
- **system_nameCSATMON.OUT**: Contains common storage usage for the current collection interval.
- **system_nameMAPREQF.OUT**: Contains the location of the module.
- **system_name.cart.log**: The log file generated by modeling code that contains the execution details of modeling code.
- **system_nameCSATMON.LOG**: The log file generated by data collection.
- **system_name.mapcsa.log**: The log file that contains the mapping of the requester of common storage to the module name.
- **system_name.launcher.log**: The log file generated by launcher code.
- **system_name.tree**: This file is generated by the modeling code. It contains information about the model tree that is built based on collected common storage usage data.
PFA_LOGREC_ARRIVAL_RATE

Description:
The check is looking at the arrival frequency of selected software logrec entries. By monitoring the arrival rate of these logrec entries over time, PFA can detect when the rate of logrec entries exceeds what is considered normal for a system. An unusually high rate of logrec entries can be indicative of recurring failures on the system. PFA can identify when these rates exceed the normal frequency and make an accurate prediction of when you need to take corrective action.

When PFA creates the model of expected logrec arrivals, categorized by the PSW key at time of failure, it discards the most recent segment identified in the collected data and uses the remaining data to develop the predictions. This elimination avoids any sudden and possibly short-lived trends in the data when determining the normal expected logrec arrival activity on the system. Analyzing the arrival rate by category prevents an expected, normal, but large number of logrecs in the key 8-15 category from masking an unexpected, critical, but small number of logrecs in the key 0 category.

Tip: z/OS logrec provides two options for the logrec recording medium. Your installation either uses System Logger to produce a logrec log stream or writes logrec to a data set. When a logrec is produced, PFA is notified through an ENF listener. If your installation is set up to write to a data set and that data set fills up, PFA will stop getting notification when a logrec is produced. Therefore, for the best reliability, it is recommended that you use the log stream method with PFA_LOGREC_ARRIVAL_RATE check.

Reason for check:
By detecting recurring failures on the system early, you are able to take corrective action for the failure before a system outage results.

Best practice:
The best practice is to:
1. To determine if the number of software logrec entries are excessive, look at the Logrec Arrival Rate Prediction Report. See "LOGREC arrival rate prediction output" on page 57 for an example.
2. If the number of software logrec entries are excessive, look at the system logrec entries and try to identify any trend that might exist. That is, for example, many LOGRECs are associated with a single job (address space) or component. For IBM code, the first three letters of the module name identify the component. For component identification, see the module identification chart in z/OS MVS Diagnosis: Reference.
   a. If a particular job (address space) is causing the problem, look at the job logs in SDSF.
      • If the job is issuing messages, follow the directions in the message text.
   b. If the job supports commands to evaluate its status, issue those commands.
      • If the job is responsive, if possible, schedule a recycle for the time that has the lowest business impact.
      • If the job is nonresponsive, if possible, capture diagnostic information and recycle the job.
   c. Look in SYSLOG for messages issued by the job or about the job.
      • If the job is issuing message, follow the directions in the message text.
3. If a particular component can be identified as causing the problem, follow the standard approach to diagnosing a problem with the component. (For example, have recent changes been made that impact the component?)
   • Look for messages issued by that component in the SYSLOG. If the component is issuing message, follow the directions in the message text.
   • Some components have monitors or operator commands to further evaluate the health of the component.
4. When in a parallel sysplex, you have the option of moving work from the LPAR experiencing the problem (excessive number of logrec entries) to a different LPAR.
**PFA_LOGREC_ARRIVAL_RATE**

*z/OS releases the check applies to:*

*z/OS V1R10 and later.*

**Type of check:**

Remote

**Parameters accepted:**

Yes, as follows:

| Table 5. PFA_LOGREC_ARRIVAL_RATE check parameters |
|----------------------------------|-----------------|----------------|-----------------|-----------------|
| **Parameter name** | **Default value** | **Minimum Value** | **Maximum Value** | **Description** |
| collectint | 60 Minutes | 1 | 360 | This parameter determines the time (in minutes) to run the data collector that determines the amount of logrec entries. The default is 15 minutes (15). |
| modelint | 360 Minutes | 4 | 1440 | This parameter determines how often (in minutes) you want the system to analyze the system and construct a new model. Note that, by default, PFA analyzes the data and constructs a new model every 6 hours (360 minutes). |
| stddev | 2 | 1 | 10 | The standard deviation used to determine if the logrec entry rate falls outside of the expected range. |
| collectinactive | 1 (off) | 1 (on) | Defines whether data will be collected and modeled even if the check is not eligible to run (is not ACTIVE(ENABLED)) in IBM Health Checker for z/OS. |
| debug | 0 (off) | 1 (on) | This parameter (an integer of 0 or 1) is used at the direction of IBM service to generate additional diagnostic information for the IBM Support Center. This debug parameter is used in place of the IBM Health Checker for z/OS policy. See “debug support” on page 57. The default is off (0). |

To determine the status of the logrec arrival usage check, issue `pfadisplay,check(pfa_logrec_arrival_rate),detail`. For the command example and more details, see “Example of DISPLAY DETAIL message output” on page 46. The following is an example of the output written to message AIR018I in SDSF user log (ULOG):

```
AIR018I 02.22.54 PFA CHECK DETAIL
CHECK NAME: PFA_LOGREC_ARRIVAL_RATE
 ACTIVE : YES
 TOTAL COLLECTION COUNT : 5
 SUCCESSFUL COLLECTION COUNT : 5
 LAST COLLECTION TIME : 04/05/2008 10.18.22
 LAST SUCCESSFUL COLLECTION TIME: 04/05/2008 10.18.22
 NEXT COLLECTION TIME : 04/05/2008 10.33.22
 TOTAL MODEL COUNT : 1
 SUCCESSFUL MODEL COUNT : 1
 LAST MODEL TIME : 04/05/2008 10.18.24
 LAST SUCCESSFUL MODEL TIME : 04/05/2008 10.18.24
 NEXT MODEL TIME : 04/05/2008 11.18.24
CHECK SPECIFIC PARAMETERS:
 COLLECTINT : 15
 MODELINT : 60
 COLLECTINACTIVE : 1=YES
 DEBUG : 0=NO
 STDDEV : 2
```

**User override of IBM values:**

The following shows keywords you can use to override check values on either a POLICY statement in the HZSPRMsxx parmlib member or on a MODIFY command. This statement can be copied and modified to override the check defaults:

```
``
PFA_LOGREC_ARRIVAL_RATE

UPDATE CHECK(IBMPFA,PFA_LOGREC_ARRIVAL_RATE)
ACTIVE
SEVERITY(MEDIUM)
INTERVAL(00:01)
PARMS=('COLLECTINT(60)', 'MODELINT(360)', 'STDDEV(2)',
       'DEBUG(0)', 'COLLECTINACTIVE(1)')
DATE(20080330)
REASON('LOGREC entry arrival rate is approaching the user defined standard deviation.')

If you change the COLLECTINT parameter or the INTERVAL parameter, the minutes set in the larger of the two parameters must elapse before the reports will be accurate.

Verbose support:
The check provides additional detail in verbose mode. You can put a check into verbose mode using the UPDATE, filters, VERBOSE=ON parameters on either the MODIFY command or in a POLICY statement in an HZSPRMxx parmlib member.

Debug support:
The DEBUG parameter in IBM Health Checker for z/OS is ignored by this check. Rather, the debug parameter is a PFA check specific parameter. For details, see "Modifying PFA checks" on page 47.

Reference:
For more information about PFA, see the topic on "Overview of Predictive Failure Analysis" on page 35.

Messages:
This check issues the following exception messages:
• AIRH110E

For additional message information, see the topics on:
• Chapter 7, “AIRH Messages,” on page 73
• AIR message z/OS MVS System Messages, Vol 1 (ABA-AOM)

SECLABEL recommended for MLS users:
SYSLOW

Output:
The LOGREC Arrival Rate Prediction Report:

Figure 5. LOGREC Arrival Rate Prediction Report

LOGREC Arrival Rate Prediction Report
Last successful model time : 11/06/2008 17:32:44
Next model time : 11/06/2008 23:33:44
Model interval : 360
Last successful collection time: 11/06/2008 18:33:49
Next collection time : 11/06/2008 19:34:49
Collection interval : 60

Key 0   Key 1-7   Key 8-15
Arrivals in last collection interval:
1   0   2

Predicted rates based on...
1 hour of data: 1   0   1
24 hours of data: 0   0   1
7 days of data: 0   0   1
30 days of data: 0   0   1

Jobs having LOGREC arrivals in last collection interval:
**PFA_LOGREC_ARRIVAL_RATE**

<table>
<thead>
<tr>
<th>Job Name</th>
<th>ASID</th>
<th>Arrivals</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGREC08</td>
<td>0029</td>
<td>2</td>
</tr>
<tr>
<td>LOGREC09</td>
<td>0027</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Last successful model time:** The date and time of the last successful model for this check. The predictions on this report were generated at that time.
- **Next model time:** The date and time of the next model. The next model will recalculate the predictions.
- **Model interval:** The value in the configured MODELINT parameter for this check. If PFA determines new prediction calculations are necessary, modeling can occur earlier.
- **Last successful collection time:** The date and time of the last successful data collection for this check.
- **Next collection time:** The date and time of the next collection.
- **Collection interval:** The value in the configured COLLECTINT parameter for this check.
- **Key column headings:** The program key of the issuer of the logrec entry.
- **Arrivals in last collection interval:** The value is one of the following:
  - `nnnn`  
    The actual count of the logrec entries received in the last collection interval period.
  - `****`  
    The number of arrivals in the last collection interval is not available because the system generated a very large number of logrec entries during this period. For example:
    
    | Key 0 | Key 1-7 | Key 8-15 |
    |-------|---------|----------|
    |       | ****    | ****     |

  If the arrival count is unavailable (****), the following message also appears on the report immediately following the predicted rates:

  **** Arrivals in last collection interval unavailable due to high level of system activity.

  In addition, if the check was unable to report all the logrec entries found in the last collection interval, it can also display the following message before the job listings:

  Job list may not include all arrivals in last collection interval due to high level of system activity.

  This message typically appears when the arrival count is unavailable (with ****) however it can appear when a logrec entry is skipped because of a high level of system activity.

- **Predicted rates based on...:** The logrec entry counts based on different ranges of historical data. If the required amount of data is not available, the line in this report is not generated. For example, if PFA has been running for a week, it has not yet collected 30 days of historical data and therefore the "30 days of data" line is not generated in the report.

- **Jobs having LOGREC arrivals in last collection interval:** The jobs that contributed to the logrec arrivals in the last collection interval.
- **Job name:** The name of the job that had logrec arrivals in the last collection interval.
- **ASID:** The ASID of the job that had logrec arrivals in the last collection interval.
- **Arrivals:** The actual count of the logrec arrivals for the job.

**Directories:**

When you install the PFA_LOGREC_ARRIVAL_RATE check, the shell script creates the following directories:

- **pfa_directory**
  This directory contains all the PFA checks and is pointed to by the home directory of the started task. The following files only contain data if errors are generated by the JVM:
    - `java.stderr` (generated by JVM)
java.stdout (generated by JVM).

pfa_directory/PFA_LOGREC_ARRIVAL_RATE/data/
The directory for logrec arrival rate that holds data and modeling results.

Guideline: If the use of the z/OS image is radically different after an IPL (for instance, the change from a test system to a production system), delete the files from PFA_LOGREC_ARRIVAL_RATE/data directory to enable the check to collect the most accurate modeling information. After the PFA_LOGREC_ARRIVAL_RATE check finds a problem and you correct it, delete the files in PFA_LOGREC_ARRIVAL_RATE/data to prevent that previous error from hiding any new errors.

Results files:

- system_name.1hr.prediction - This file is generated by the modeling code for the predictions made for one hour of historical data. It contains the list of program keys with their corresponding predictions.
- system_name.24hr.prediction - This file is generated by the modeling code for the predictions made for 24 hours of historical data. It contains the list of program keys with their corresponding predictions.
- system_name.7day.prediction - This file is generated by the modeling code for the predictions made for seven days of historical data. It contains the list of program keys with their corresponding predictions.
- system_name.30day.prediction - This file is generated by the modeling code for the predictions made for 30 days of historical data. It contains the list of program keys with their corresponding predictions.
- system_name.1hr.prediction.html - This file lists the program keys in an .html report format for the predictions made for one hour of historical data.
- system_name.24hr.prediction.html - This file lists the program keys in an .html report format for the predictions made for 24 hours of historical data.
- system_name.7day.prediction.html - This file lists the program keys in an .html report format for the predictions made for seven days of historical data.
- system_name.30day.prediction.html - This file lists the program keys in an .html report format for the predictions made for 30 days of historical data.
- system_name.1hr.diag - The predictions generated by modeling when one hour of historical data was used. This file is used as input to the code that compares the predicted logrec arrivals to the actual logrec arrivals.
- system_name.24hr.diag - The predictions generated by modeling when 24 hours of historical data was used. This file is used as input to the code that compares the predicted logrec arrivals to the actual logrec arrivals.
- system_name.7day.diag - The predictions generated by modeling when seven days of historical data was used. This file is used as input to the code that compares the predicted logrec arrivals to the actual logrec arrivals.
- system_name.30day.diag - The predictions generated by modeling when 30 days of historical data was used. This file is used as input to the code that compares the predicted logrec arrivals to the actual logrec arrivals.

The values for the logrec arrival prediction file in .html (prediction.html) files are as follows:

- Program Key: The program key when the error described by the logrec entry occurred.
- Instance Count: The number of records with this program key that were factored into the prediction model.
- Current Logrec Arrivals: The number of logrec arrivals for this program key in the last collection interval included in this model.
- Prediction Look Forward Seconds: The number of seconds the prediction should project into the future.
PFA_LOGREC_ARRIVAL_RATE

- **Predicted Logrec Arrivals**: The predicted logrec arrivals for this program key.

**Intermediate files:**
- `system_nameLAR.OUT` - The data collection file.
- `system_name.1hr.lardata` - The file used as input to modeling code. It contains one hour of historical data.
- `system_name.24hr.lardata` - The file used as input to modeling code. It contains 24 hours of historical data.
- `system_name.7day.lardata` - The file used as input to modeling code. It contains seven days of historical data.
- `system_name.30day.lardata` - The file is used as input to modeling code. It contains 30 days of historical data.

This directory holds the following log files. Additional information is written to these log files when DEBUG(1).
- `system_nameLAR.LOG` - The log file for data collection.
- `system_name.1hr.cart.log` - The log file generated by modeling code with details about code execution while one hour of historical data was being modeled.
- `system_name.24hr.cart.log` - The log file generated by modeling code with details about code execution while 24 hours of historical data was being modeled.
- `system_name.7day.cart.log` - The log file generated by modeling code with details about code execution while seven days of historical data was being modeled.
- `system_name.30day.cart.log` - The log file generated by modeling code with details about code execution while 30 days of historical data was being modeled.
- `system_name.buildLar.log` - The log file generated by intermediate code that builds the files that are input to modeling with details about code execution.
- `system_name.launcher.log` - The log file generated by launcher code.
- `system_name.1hr.tree` - This file is generated by the modeling code. It contains information about the model tree which was built based on the last one hour of collected data.
- `system_name.24hr.tree` - This file is generated by the modeling code. It contains information about the model tree which was built based on the last 24 hours of collected data.
- `system_name.7day.tree` - This file is generated by the modeling code. It contains information about the model tree which was built based on the last seven days of collected data.
- `system_name.30day.tree` - This file is generated by the modeling code. It contains information about the model tree which was built based on the last 30 days of collected data.
PFA_FRAMES_AND_SLOTS_USAGE

Description:
This check is looking for a persistent job that is using an abnormal number of frames and slots, which often indicates that the job is leaking virtual storage. A job is considered persistent if it starts within the first hour after IPL.

A unit of work can acquire a range of virtual storage addresses for its use. When the memory represented by the virtual storage address range is referenced or updated, the operating system backs the memory with real frames. The operating system can move the data from the real frames to auxiliary storage. This check detects abnormal usage (a larger than expected number) of virtual storage by monitoring the usage of real frames and auxiliary storage slots of persistent jobs. When this usage is unusual, this check generates an exception message.

This check is not designed to detect the following problems:
- leaks that are:
  - small and inconsequential
  - known to exhaust the internal constraints of the virtual storage manager (such as when the storage manager attempts to use more data space than was allocated)
- fragmentation of space managed by the virtual storage manager
- space requests that are large and random, which exhaust the internal constraints of the virtual storage manager

Reason for check:
The objective of this check is to determine when a started task or started job is leaking virtual storage to prevent an outage caused by exhausting auxiliary storage slots.

Best practice:
The best practice is to add page packs to keep the system running until the next IPL, perform normal diagnostic actions to determine the cause of the problem, or restart the task or job.

z/OS releases the check applies to:
z/OS V1R11 and later.

Type of check:
Remote

Parameters accepted:
Yes, as follows:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectint</td>
<td>15 Minutes</td>
<td>1</td>
<td>360</td>
<td>This parameter determines the time (in minutes) to run the data collector that determines the number of frames and slots usage. The default is 15 minutes (15).</td>
</tr>
<tr>
<td>modelint</td>
<td>360 Minutes</td>
<td>4</td>
<td>1440</td>
<td>This parameter determines how often (in minutes) you want the system to analyze the system and construct a new model. Note that, by default, PFA analyzes the data and constructs a new model every 6 hours (360 minutes).</td>
</tr>
<tr>
<td>stddev</td>
<td>3</td>
<td>2</td>
<td>100</td>
<td>The standard deviation used to determine if the frames and slots rate falls outside of the expected range.</td>
</tr>
<tr>
<td>collectinactive</td>
<td>0 (off)</td>
<td>1 (on)</td>
<td></td>
<td>Defines whether data will be collected and modeled even if the check is not eligible to run (is not ACTIVE(ENABLED)) in IBM Health Checker for z/OS.</td>
</tr>
</tbody>
</table>
### Table 6. PFA_FRAMES_AND_SLOTS_USAGE check parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>0</td>
<td>0 (off)</td>
<td>1 (on)</td>
<td>This parameter (an integer of 0 or 1) is used at the direction of IBM service to generate additional diagnostic information for the IBM Support Center. This debug parameter is used in place of the IBM Health Checker for z/OS policy. See <a href="#">debug support</a> on page 57. The default is off (0).</td>
</tr>
</tbody>
</table>

To determine the status of the frames and slots usage check, issue `fpfa,display,check(pfa_frames_and_slots_usage),detail`. For the command example and more details, see [“Example of DISPLAY DETAIL message output”](#) on page 46. The following example shows the output written to message AIR018I in SDSF:

```
AIR018I 02.22.54 PFA CHECK DETAIL
CHECK NAME: PFA_FRAMES_AND_SLOTS_USAGE
ACTIVE : YES
TOTAL COLLECTION COUNT : 5
SUCCESSFUL COLLECTION COUNT : 5
LAST COLLECTION TIME : 02/05/2009 10.18.22
LAST SUCCESSFUL COLLECTION TIME : 02/05/2009 10.18.22
NEXT COLLECTION TIME : 02/05/2009 10.33.22
TOTAL MODEL COUNT : 1
SUCCESSFUL MODEL COUNT : 1
LAST MODEL TIME : 02/05/2009 10.18.24
LAST SUCCESSFUL MODEL TIME : 02/05/2009 10.18.24
NEXT MODEL TIME : 02/05/2009 11.18.24
CHECK SPECIFIC PARAMETERS:
  COLLECTINT : 15
  MODELINT : 60
  COLLECTINACTIVE : 1=YES
  DEBUG : 0=NO
  STDDEV : 3
```

#### User override of IBM values:

The following shows keywords you can use to override check values on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. This statement can be copied and modified to override the check defaults:

```
UPDATE CHECK(IBMPFA,PFA_FRAMES_AND_SLOTS_USAGE)
  ACTIVE
  SEVERITY(MEDIUM)
  INTERVAL(00:01)
  PARMS=('COLLECTINT(15)', 'MODELINT(60)', 'STDDEV(3)',
              'DEBUG(0)', 'COLLECTINACTIVE(1)')
  DATE(20080330)
  REASON('Virtual storage rate is approaching the user defined standard deviation.')
```

If you change the COLLECTINT parameter or the INTERVAL parameter, the minutes set in the larger of the two parameters must elapse before the reports will be accurate.

#### Verbose support:

The check provides additional detail in verbose mode. You can put a check into verbose mode using the UPDATE,filters,VERBOSE=ON parameters on either the MODIFY command or in a POLICY statement in an HZSPRMxx parmlib member.

#### Debug support:

The DEBUG parameter in IBM Health Checker for z/OS is ignored by this check. Rather, the debug parameter is a PFA check specific parameter. For details, see [“Modifying PFA checks”](#) on page 47.
Reference:
For more information about PFA, see the topic on “Overview of Predictive Failure Analysis” on page 35.

Messages:
This check issues the following exception messages:
- AIRH148E

For additional message information, see the topics on:
- Chapter 7, “AIRH Messages,” on page 73
- AIR messages in z/OS MVS System Messages, Vol 1 (ABA-AOM)

SECLABEL recommended for MLS users:
- SYSLOW

Output:
The frames and slots usage prediction report:

Frames and Slots Usage Prediction Report

<table>
<thead>
<tr>
<th>Last successful model time</th>
<th>Next model time</th>
<th>Model interval</th>
</tr>
</thead>
</table>

<p>| Address spaces with the highest increased usage: |</p>
<table>
<thead>
<tr>
<th>Job Name</th>
<th>ASID</th>
<th>Current Frames</th>
<th>Expected Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZFS</td>
<td>001D</td>
<td>12920</td>
<td>12891</td>
</tr>
<tr>
<td>Hzsproc</td>
<td>0028</td>
<td>3515</td>
<td>3186</td>
</tr>
<tr>
<td>CEA</td>
<td>0032</td>
<td>3014</td>
<td>3012</td>
</tr>
<tr>
<td>Hwibpci1</td>
<td>0043</td>
<td>2302</td>
<td>2302</td>
</tr>
<tr>
<td>Vtamosr3</td>
<td>001F</td>
<td>1902</td>
<td>1904</td>
</tr>
<tr>
<td>GRS</td>
<td>0035</td>
<td>1748</td>
<td>1706</td>
</tr>
<tr>
<td>Xcfas</td>
<td>0022</td>
<td>1634</td>
<td>1639</td>
</tr>
<tr>
<td>Allocas</td>
<td>0023</td>
<td>2145</td>
<td>1535</td>
</tr>
<tr>
<td>Sms</td>
<td>0018</td>
<td>695</td>
<td>674</td>
</tr>
<tr>
<td>Jesxcf</td>
<td>001E</td>
<td>611</td>
<td>612</td>
</tr>
<tr>
<td>Iosas</td>
<td>0025</td>
<td>383</td>
<td>379</td>
</tr>
<tr>
<td>Resolver</td>
<td>003B</td>
<td>276</td>
<td>282</td>
</tr>
<tr>
<td>JES2mon</td>
<td>0042</td>
<td>289</td>
<td>206</td>
</tr>
<tr>
<td>JES2aux</td>
<td>0039</td>
<td>211</td>
<td>201</td>
</tr>
</tbody>
</table>

Figure 6. Frames and slots usage prediction report

- Last successful model time: The date and time of the last successful model for this check. The predictions on this report were generated at that time.
- Next model time: The date and time of the next model. The next model will recalculate the predictions.
- Model interval: The value in the configured MODELINT parameter for this check. If PFA determines new prediction calculations are necessary, modeling can occur earlier.
- Last successful collection time: The date and time of the last successful data collection for this check.
- Next collection time: The date and time of the next collection.
- Collection interval: The value in the configured COLLECTINT parameter for this check.
Address spaces with the highest usage: The address spaces whose usage of frames and slots recently increased the most. The number of jobs printed can vary. If PFA issues the report for the AIRH148E exception, the list of jobs contains only those that caused the exception to be issued.

Job name: The name of the persistent job.

ASID: The ASID of the job that is a top predicted user.

Current Frames and Slots Usage: The number of frames and slots used by the job represented in whole numbers.

Expected Frames and Slots Usage: The number of frames and slots expected to be used by this job at the end of the model interval.

Directories:
When you install the PFA_FRAMES_AND_SLOTS_USAGE check, the following directories are created:

pfa_directory
This directory contains all the PFA checks and is pointed to by the home directory of the started task. The following files only contain data if errors are generated by the JVM:
- java.stderr (generated by JVM)
- java.stdout (generated by JVM).

pfa_directory/PFA_FRAMES_AND_SLOTS_USAGE
This is the check specific directory for frames and slots usage.

pfa_directory/PFA_FRAMES_AND_SLOTS_USAGE/data/
The directory for frames and slots usage that holds data and modeling results.

Results files:
- system_name.prediction - This file is generated by the modeling code. It lists the jobs that have the highest recent growth in use of frames and slots. For each job in the list, it records the expected frames and slots usage and the standard deviation of the data used to generate the expected usage.

Results files:
- system_name.prediction.html - This file lists the top users of frames and slots in an .html report format. The values found in the frames and slots usage prediction report in .html format are as follows:
  - User of Virtual Storage: This is the identification of the job name using of frames and slots.
  - Instance Count: The number of records with this user that were factored into the prediction model.
  - Current Estimated Virtual Storage Used: The current number of frames and slots used by this job in the last collection interval included in this model.
  - Prediction Look Forward Seconds: The number of seconds the prediction should project into the future.
  - Predicted Virtual Storage Usage: The predicted number of frames and slots usage for this user.
  - Standard Deviation: The standard deviation of the set of data used to generate the predicted frames and slots usage value.

Data store files:
- system_name.vsuAll.timestamp - The vsuAll files contain the frames and slots usage in a collection interval.
**PFA_FRAMES_AND_SLOTS_USAGE**

- system_name.vsuSum.timestamp - The vsuSum files summarize the data in the vsuAll files. After five days, the vsuAll data is averaged and compressed to one file each day, and then time-stamped with the start of that day. The data then moves to a vsuSum file and vsuAll files are deleted.

**Intermediate files:**
- system_name.vsudata - The input to modeling in CSV format. vsudata saves one entry "top contributor" for each vsuSum file and vsuAll file.

This directory also holds the following log files. Additional information is written to these log files when DEBUG(1).
- system_name.cart.log - The log file generated by modeling code that contains the execution details of modeling code.
- system_nameVSU.LOG - The log file generated by data collection.
- system_name.launcher.log - The log file generated by launcher code.
- system_name.tree - This file is generated by the modeling code. It contains information about the model tree that is built based on collected frames and slots usage data.
**PFA_MESSAGE_ARRIVAL_RATE**

**Description:**
This check is determining if an LPAR is damaged by checking the arrival rate of abnormal messages per number of CPU seconds used. If the arrival rate is too high, it might cause damage to the LPAR sometime in the near future.

To avoid skewing the message arrival rate, PFA ignores the first hour of message data after IPL and the last hour of message data prior to shutdown. In addition, PFA attempts to track the same persistent jobs that it had tracked prior to IPL or PFA restart as long as the same persistent job is still active.

This check is not designed to detect performance problems caused by insufficient resources, faulty WLM policy, spikes in work, or stalled work. However, it might help to determine if a performance problem detected by a performance monitor or WLM is caused by a damaged system.

The message arrival rate check issues an exception using the following four types of comparisons. After PFA issues an exception, the next comparison type is not performed.

**Top persistent jobs**
PFA tracks the top persistent jobs individually. Jobs are considered persistent if they start within an hour after IPL. PFA determines which jobs to track individually based on the following criteria:

- If PFA previously ran on this system and the same jobs that were previously tracked are active, PFA tracks the same jobs.
- If PFA did not previously run on this system or the same jobs previously tracked are not all active, PFA will collect data for a period of time to use in determining which jobs have the highest arrival rates. After this time passes, PFA individually tracks the jobs that have the highest arrival rates for that period.
  - During the first hour after IPL and during the time PFA is determining the jobs to track individually, normal data collection and modeling are suspended.
  - Changing the COLLECTINT or the MODELINT parameters during these times is allowed, but the changes are not used until after these times have passed.
  - Next collection and model times change automatically during these times to reflect the most accurate times known at each phase of the initial processing.

**Other persistent jobs**
The persistent jobs that PFA does not track individually are the other persistent jobs. The predictions are done using the totals for this group and the comparisons are done individually using a mathematical formula.

**Non-persistent jobs**
The jobs that start over an hour after IPL are the non-persistent jobs. The predictions and comparisons are done using the totals for this group.

**Total system**
This group includes all jobs. The predictions and the comparisons are done using the totals for the entire system.

**Reason for check:**
The objective of this check is to determine if an LPAR is damaged by checking the arrival rate of abnormal messages per number of CPU seconds used.

**Best practice:**
The best practice is to analyze the messages being sent by the address spaces identified on the report by examining the system log to determine what is causing this burst of message activity. Establish which messages were issued around the time of the activity and review the message details. Follow the directions provided by the message to continue to diagnose and fix the problem.
z/OS releases the check applies to:
  
  z/OS V1R11 and later.

**Type of check:**
  
  Remote

**Parameters accepted:**
  
  Yes, as follows:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectint</td>
<td>15 Minutes</td>
<td>15</td>
<td>360</td>
<td>This parameter determines how often (in minutes) to run the data collector that retrieves the current message arrival rate.</td>
</tr>
<tr>
<td>modelint</td>
<td>360 Minutes</td>
<td>60</td>
<td>1440</td>
<td>This parameter determines how often (in minutes) you want the system to analyze the system and construct a new message arrival rate model or prediction. By default, PFA analyzes the data and constructs a new model every “default value” minutes. The model interval must be at least four times larger than the collection interval.</td>
</tr>
<tr>
<td>stddev</td>
<td>10</td>
<td>2</td>
<td>100</td>
<td>The number by which to multiply the predicted message arrival rate per amount of CPU used to determine if the actual rate has increased beyond the allowable limit.</td>
</tr>
<tr>
<td>collectinactive</td>
<td>1 (on)</td>
<td>0 (off)</td>
<td>1 (on)</td>
<td>Defines whether data will be collected and modeled even if the check is not eligible to run, not ACTIVE(ENABLED), in IBM Health Checker for z/OS.</td>
</tr>
<tr>
<td>trackedmin</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>This parameter defines the minimum message arrival rate required for a persistent job in order for it to be considered a top persistent job that should be tracked individually.</td>
</tr>
<tr>
<td>exceptionmin</td>
<td>1</td>
<td>0</td>
<td>1000</td>
<td>This parameter defines the minimum message arrival rate and the minimum predicted message arrival rate required to cause an exception.</td>
</tr>
<tr>
<td>debug</td>
<td>0 (off)</td>
<td>0 (off)</td>
<td>1 (on)</td>
<td>This parameter (an integer of 0 or 1) is used at the direction of IBM service to generate additional diagnostic information for the IBM Support Center. This debug parameter is used in place of the IBM Health Checker for z/OS policy. See “message arrival rate log files” on page 72. The default is off (0).</td>
</tr>
</tbody>
</table>

To determine the status of the message arrival rate check, issue \`f pfa,display,check(pfa_message_arrival_rate),detail\`. For the command example and more details, see “Example of DISPLAY DETAIL message output” on page 46. The following example shows the output written to message AIR018I in SDSF:

```
AIR018I 02.22.54 PFA CHECK DETAIL
CHECK NAME: PFA_MESSAGE_ARRIVAL_RATE
  ACTIVE : YES
  TOTAL COLLECTION COUNT : 5
  SUCCESSFUL COLLECTION COUNT : 5
  LAST COLLECTION TIME : 02/05/2009 10.18.22
  LAST SUCCESSFUL COLLECTION TIME : 02/05/2009 10.18.22
  NEXT COLLECTION TIME : 02/05/2009 10.33.22
  TOTAL MODEL COUNT : 1
  SUCCESSFUL MODEL COUNT : 1
  LAST MODEL TIME : 02/05/2009 10.18.24
  LAST SUCCESSFUL MODEL TIME : 02/05/2009 10.18.24
  NEXT MODEL TIME : 02/05/2009 11.18.24
  CHECK SPECIFIC PARAMETERS:
```
**PFA_MESSAGE_ARRIVAL_RATE**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLECTINT</td>
<td>15</td>
</tr>
<tr>
<td>MODELINT</td>
<td>60</td>
</tr>
<tr>
<td>COLLECTINACTIVE</td>
<td>1=NO</td>
</tr>
<tr>
<td>DEBUG</td>
<td>0=NO</td>
</tr>
<tr>
<td>STDDEV</td>
<td>10</td>
</tr>
</tbody>
</table>

**User override of IBM values:**

The following shows keywords you can use to override check values on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. This statement can be copied and modified to override the check defaults:

```plaintext
UPDATE CHECK(IBMPFA,PFA_MESSAGE_ARRIVAL_RATE)
  ACTIVE
  SEVERITY(MEDIUM)
  INTERVAL(ONETIME)
  PARMS=('COLLECTINT(15)','MODELINT(360)','STDDEV(10)',
     'DEBUG(0)','COLLECTINACTIVE(1)')
  DATE(20080330)
  REASON('The message arrival rate is too high which can indicate a system that is damaged.')
```

The message arrival rate check is designed to run automatically after every data collection. Do not change the INTERVAL parameter.

**Verbose support:**

The check provides additional detail in verbose mode. You can put a check into verbose mode using the UPDATE, filters, VERBOSE=ON parameters on either the MODIFY command or in a POLICY statement in an HZSPRMxx parmlib member.

**Debug support:**

The DEBUG parameter in IBM Health Checker for z/OS is ignored by this check. Rather, the debug parameter is a PFA check specific parameter. For details, see "Modifying PFA checks" on page 47.

**Reference:**

For more information about PFA, see the topic on "Overview of Predictive Failure Analysis" on page 35.

**Messages:**

The output is a message arrival rate prediction report that corresponds to the message issued. One of the following reports is generated:

- AIRH152E
- AIRH165E
- AIRH166E
- AIRH169E

For additional message information, see the topics on:

- Chapter 7, "AIRH Messages," on page 73
- AIR messages, z/OS MVS System Messages, Vol I (ABA-AOM)

**SECLABEL recommended for MLS users:**

SYSLOW

**Output:**

The output is a variation of the message arrival rate prediction report. MAR does not process the CONSOLE address space and all address spaces that begin with JES. The values found in the message arrival prediction file are as follows:

**Tracked top persistent jobs exception report:** PFA issues the message arrival rate tracked jobs exception report (for AIRH165E) when any one or more tracked, persistent jobs cause an exception. Only the tracked jobs that caused the exception are included in the list of jobs on the report. The following example is the message arrival rate tracked jobs exception report (for AIRH165E):
Message Arrival Rate Prediction Report

Last successful model time : 01/27/2009 17:08:01
Next model time : 01/27/2009 23:08:01
Model interval : 360
Last successful collection time: 01/27/2009 17:41:38
Next collection time : 01/27/2009 17:56:38
Collection interval : 15

Persistent address spaces with high rates:

<table>
<thead>
<tr>
<th>Job Name</th>
<th>ASID</th>
<th>Rate 1 Hour</th>
<th>Rate 24 Hour</th>
<th>Rate 7 Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBS1</td>
<td>001D</td>
<td>58.00</td>
<td>23.88</td>
<td>15.82</td>
</tr>
<tr>
<td>JOBS2</td>
<td>0028</td>
<td>11.00</td>
<td>0.34</td>
<td>11.11</td>
</tr>
<tr>
<td>JOBS3</td>
<td>0029</td>
<td>11.00</td>
<td>12.43</td>
<td>2.36</td>
</tr>
</tbody>
</table>

Other persistent jobs exception report: PFA issues the message arrival rate other persistent jobs exception report (for AIRH166E) when a comparison of a persistent job (that is not being individually tracked) causes an exception when compared to the totals of the other persistent jobs. The list of jobs is only those persistent jobs (not tracked individually) that have a problem. No predictions are given for these jobs because PFA does not model individual predictions for jobs that are not tracked individually. If there is more than one job with the same name, four asterisks **** are printed for the ASID in the report. The following example is the message arrival rate other persistent jobs exception report:

Message Arrival Rate Prediction Report

Last successful model time : 01/27/2009 17:08:01
Next model time : 01/27/2009 23:08:01
Model interval : 360
Last successful collection time: 01/27/2009 17:41:38
Next collection time : 01/27/2009 17:56:38
Collection interval : 15

Persistent address spaces with high rates:

<table>
<thead>
<tr>
<th>Job Name</th>
<th>ASID</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERS1</td>
<td>001E</td>
<td>83.22</td>
</tr>
<tr>
<td>PERS2</td>
<td>0038</td>
<td>75.52</td>
</tr>
<tr>
<td>PERS3</td>
<td>0039</td>
<td>47.47</td>
</tr>
</tbody>
</table>

Non-persistent jobs exception report: PFA issues the message arrival rate non-persistent jobs exception report (for AIRH169E) when the non-persistent jobs as a group can cause an exception. The list of jobs contains only three non-persistent jobs that have high arrival counts. No predictions are given for these jobs because PFA does not model individual predictions for jobs that are not tracked individually. The following example is the message arrival rate non-persistent jobs exception report (for AIRH169E):

Message Arrival Rate Prediction Report

Last successful model time : 01/27/2009 17:08:01
Next model time : 01/27/2009 23:08:01
Model interval : 360
Last successful collection time: 01/27/2009 17:41:38
Next collection time : 01/27/2009 17:56:38
Collection interval : 15

Address spaces with high arrivals:
No problem and total system exception report: When no exception is issued or when the total message arrival rate exception (AIRH152E) is issued, the following report is generated. The list of jobs contains all of the jobs being tracked individually. The following example is the message arrival rate no problem report and total system exception report:

Message Arrival Rate Prediction Report

Last successful model time : 01/27/2009 17:08:01
Next model time : 01/27/2009 23:08:01
Model interval : 360
Last successful collection time: 01/27/2009 17:41:38
Next collection time : 01/27/2009 17:56:38
Collection interval : 15

Message arrival rate at last collection interval : 83.52
Prediction based on 1 hour of data : 98.27
Prediction based on 24 hours of data: 85.98
Prediction based on 7 days of data : 100.22

Top persistent users:

<table>
<thead>
<tr>
<th>Job Name</th>
<th>ASID</th>
<th>Message Arrival Rate</th>
<th>Predicted Message Arrival Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB1</td>
<td>001D</td>
<td>58.00</td>
<td>23.88</td>
</tr>
<tr>
<td>JOB2</td>
<td>0028</td>
<td>11.00</td>
<td>0.34</td>
</tr>
<tr>
<td>JOB3</td>
<td>0029</td>
<td>11.00</td>
<td>12.43</td>
</tr>
</tbody>
</table>

The following fields apply to all reports:

- **Last successful model time**: The date and time of the last successful model for this check. The predictions on this report were generated at that time.
- **Next model time**: The date and time of the next model. The next model will recalculate the predictions.
- **Model interval**: The value in the configured MODELINT parameter for this check. If PFA determines new prediction calculations are necessary, modeling can occur earlier.
- **Last successful collection time**: The date and time of the last successful data collection for this check.
- **Next collection time**: The date and time of the next collection.
- **Collection interval**: The value in the configured COLLECTINT parameter for this check.
- **Message arrival rate in last collection interval**: The actual message arrival rate in the last collection interval where the rate is defined to be the number of messages divided by the CPU seconds.
- **Predicted rates based on...**: The message arrival rates based on one hour, 24 hours, and seven days. If no prediction is available for a given time range, the line is not printed. For example, if the check has been running for two days, seven days of data is not available and the “Prediction based on 7 days of data” line is not printed.
- **Job Name**: The name of the job that has message arrivals in the last collection interval.
- **ASID**: The ASID for the job that has message arrivals in the last collection interval.
**Message Arrival Rate:** The current message arrival rate for the persistent job.

**Message Arrival Counts:** The message arrival count for the non-persistent job.

**Note:** The "Message Arrival Count" field is unique to the non-persistent jobs exception report.

**Predicted Message Arrival Rate:** The predicted message arrival rate based on one hour, 24 hours, and seven days of data. **UNKNOWN** might print for the predicted rate at any point from the time an IPL occurred until at least seven days after the IPL depending on how the data collected prior to the IPL is used. When **UNKNOWN** is printed, no comparisons are made for that category.

**Directories**

**Note:** The content and names for these files and directories are subject to change and cannot be used as programming interfaces; these files are documented only to provide help in diagnosing problems with PFA.

**pfa_directory**

This directory contains all the PFA checks and is pointed to by the home directory of the started task. The following files only contain data if errors are generated by the JVM:

- java.stderr (generated by JVM)
- java.stdout (generated by JVM)

**pfa_directory/PFA_MESSAGE_ARRIVAL_RATE/data**

The directory for message arrival rate that holds data and modeling results.

**Guideline:** If the use of the z/OS image is radically different after an IPL (for instance, the change from a test system to a production system), delete the files in the **PFA_MESSAGE_ARRIVAL_RATE/data** directory to enable the check to collect the most accurate modeling information.

- system_name.1hr.prediction - This file is generated by the modeling code for the predictions made for one hour of historical data. It contains the list of persistent address spaces with their corresponding predictions.
- system_name.24hr.prediction - This file is generated by the modeling code for the predictions made for 24 hours of historical data. It contains the list of persistent address spaces with their corresponding predictions.
- system_name.7day.prediction - This file is generated by the modeling code for the predictions made for seven days of historical data. It contains the list of persistent address spaces with their corresponding predictions.
- system_name.1hr.prediction.html - This file lists the persistent address spaces in an .html report format for the predictions made for one hour of historical data.
- system_name.24hr.prediction.html - This file lists the persistent address spaces in an .html report format for the predictions made for 24 hours of historical data.
- system_name.7day.prediction.html - This file lists the persistent address spaces in an .html report format for the predictions made for seven days of historical data.

**Intermediate files:**

- system_nameMAR.OUT - The data collection file.
- system_name.mardata - The file is used as input to the modeling to track if enough data is available to model.
- system_name.1hr.mardata - The file used as input to modeling code. It contains one hour of historical data.
- system_name.24hr.mardata - The file used as input to modeling code. It contains 24 hours of historical data.
- system_name.7day.mardata - The file used as input to modeling code. It contains seven days of historical data.
This directory holds the following log files. Additional information is written to these log files when DEBUG(1).

- **system_nameMAR.LOG** - The log file for data collection.
- **system_name.1hr.cart.log** - The log file generated by modeling code with details about code execution while one hour of historical data was being modeled.
- **system_name.24hr.cart.log** - The log file generated by modeling code with details about code execution while 24 hours of historical data was being modeled.
- **system_name.7day.cart.log** - The log file generated by modeling code with details about code execution while seven days of historical data was being modeled.
- **system_name.buildMar.log** - The log file generated by intermediate code that builds the files that are input to modeling with details about code execution.
- **system_name.launcher.log** - The log file generated by launcher code.
- **system_name.1hr.tree** - This file is generated by the modeling code. It contains information about the model tree which was built based on the last one hour of collected data.
- **system_name.24hr.tree** - This file is generated by the modeling code. It contains information about the model tree which was built based on the last 24 hours of collected data.
- **system_name.7day.tree** - This file is generated by the modeling code. It contains information about the model tree which was built based on the last seven days of collected data.
Chapter 7. AIRH Messages

The following messages are IBM Health Checker for z/OS messages that are issued by PFA. The check exception messages are issued both as WTOs and also to the message buffer. The WTO version contains only the message text, while the exception message in the message buffer includes both the text and explanation of the potential problem found, including the severity, and information about what to do to fix the potential problem.

AIRH101E

The common storage usage (CSA and SQA) below the line has been predicted to exceed the capacity in the future.

Explanation: The model of common storage utilization for this LPAR has predicted that usage of common storage (CSA and SQA) below the line will exceed the threshold set by the user before the current model interval ends at nextmodeltime. The prediction was modeled at lastsuccmodeltime. The threshold used in the calculations is controlled by the user of predictive failure analysis by modifying the THRESHOLD parameter for this check.

System action: The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

Operator response: A list of candidates for the cause of the problem has been produced which may contain the job or address space which is causing common storage to be exhausted. Review the list of candidates and follow your standard diagnostic procedures to determine what actions are needed to prevent common storage exhaustion from occurring.

System programmer response: See operator response to determine the cause of the common storage shortage.

Problem determination: See operator response to determine the cause of the common storage shortage.

Source: Predictive Failure Analysis (SCPFA)

Reference Documentation: None.

Automation: This message should trigger an alert to system programmers or operator to determine the cause of the common storage shortage.

Module: AIRHMCHK

Routing code: N/A

Descriptor code: N/A

AIRH102I No problem with common storage usage (CSA and SQA) is predicted to occur before nextmodeltime.

The current usage is within the expected range based on comparisons between the total capacity plus the threshold, the current usage, the current prediction, and the future prediction modeled at lastsuccmodeltime.

Explanation: The model of common storage usage for this LPAR has predicted that usage of common storage (CSA and SQA) will be below the total common storage capacity plus additional common storage defined by the user threshold until after the end of the current model interval.

System action: The check processing continues.

Operator response: None.

System programmer response: None.

Problem determination: None.

Source: Predictive Failure Analysis (SCPFA)
AIRH103I  Comparisons of predictions and arrivals will occur when the check is run after modeling has run and succeeded. Modeling is scheduled for nextmodeltime.

Explanation: Sufficient data needs to be collected to issue a prediction of future common storage usage. Sufficient data will be collected after a successful model occurs. The first modeling will occur after MODELINT minutes from the start of PFA. Once enough data has been collected to build a model of common storage usage trends and a successful model has occurred, common storage usage predictions will start to be generated.

System action: The check processing continues.

Operator response: None.

System programmer response: None.

Problem determination: None.

Source: Predictive Failure Analysis (SCPFA)

Reference Documentation: None.

Automation: None.

Module: AIRHMCHK

Routing code: N/A

Descriptor code: N/A

AIRH109E

The common storage usage (CSA and SQA) above the line has been predicted to exceed the capacity in the future.

Explanation: The model of common storage utilization for this LPAR has predicted that usage of common storage (CSA and SQA) above the line will exceed the threshold set by the user before the current model interval ends at nextmodeltime. The prediction was modeled at lastsuccmodeltime. The threshold used in the calculations is controlled by the user of predictive failure analysis by modifying the THRESHOLD parameter for this check.

System action: The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

Operator response: A list of candidates for the cause of the problem has been produced which may contain the job or address space which is causing common storage to be exhausted. Review the list of candidates and follow your standard diagnostic procedures to determine what actions are needed to prevent common storage exhaustion from occurring.

System programmer response: See operator response to determine the cause of the common storage shortage.

Problem determination: See operator response to determine the cause of the common storage shortage.

Source: Predictive Failure Analysis (SCPFA)

Reference Documentation: None.

Automation: This message should trigger an alert to system programmers or operator to determine the cause of the common storage shortage.

Module: AIRHMCHK
AIRH110E

The LOGREC entry arrival rate is higher than expected and could cause a system problem by nextmodeltime.

When the LOGREC entry rate is higher than expected, it can indicate recurring failures on the system which can eventually lead to system hang. The prediction used in the comparison was modeled at lastsuccmodeltime.

Explanation: The model of the LOGREC entry arrival rate has detected that the arrival rate is higher than expected based on the previous history of this system. An unusually high LOGREC entry rate can indicate a series of recurring failures on the system which can lead to a system hang or may be indicative of a damaged system.

System action: The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

Operator response: Examine LOGREC entries to determine the source of the higher than expected entry arrival rate.

System programmer response: See operator response to determine the cause of the higher than expected LOGREC entry arrival rate.

Problem determination: See operator response to determine the cause of the higher than expected LOGREC entry arrival rate.

Source: Predictive Failure Analysis (SCPFA)

Reference Documentation: None.

Automation: This message should trigger an alert to system programmers or operator to determine the cause of the high LOGREC entry arrival rate.

Module: AIRHMCHK

Routing code: N/A

Descriptor code: N/A

AIRH111I

No problems based on LOGREC entry arrival rate are expected to occur before nextmodeltime. The current LOGREC entry arrival rate is within the predicted range modeled at lastsuccmodeltime.

Explanation: The LOGREC entry arrival rate for the system is within the normal boundaries for this system.

System action: The check processing continues.

Operator response: None.

System programmer response: None.

Problem determination: None.

Source: Predictive Failure Analysis (SCPFA)

Reference Documentation: None.

Automation: None.

Module: AIRHMCHK

Routing code: N/A

Descriptor code: N/A
AIRH112I Comparisons of predictions and arrivals will occur when the check is run after 24 hours of data has been collected and modeling has run and succeeded. Modeling is scheduled for `nextmodeltime`.

Explanation: Insufficient data has been collected to issue a prediction on future LOGREC entry arrival rates. Once enough data has been gathered to build a model of LOGREC entry arrival rate trends, arrival rate predictions will start to be generated.

System action: The check processing continues.
Operator response: None.
System programmer response: None.
Problem determination: None.
Source: Predictive Failure Analysis (SCPFA)
Reference Documentation: None.
Automation: None.
Module: AIRHMCHK
Routing code: N/A
Descriptor code: N/A

AIRH132I Last model of future common storage usage attempted at `lastmodeltime` was unsuccessful. Modeling is next scheduled to occur at `nextmodeltime`.

Explanation: PFA modeling runs at a frequency of `MODELINT` minutes. When the last model was attempted, it was unsuccessful. Modeling will run again at the next scheduled model interval.

System action: The check processing continues.
Operator response: None.
System programmer response: None.
Problem determination: None.
Source: Predictive Failure Analysis (SCPFA)
Reference Documentation: None.
Automation: None.
Module: AIRHMCHK
Routing code: N/A
Descriptor code: N/A

AIRH133I Last model of future LOGREC arrival rate attempted at `lastmodeltime` was unsuccessful. Modeling is next scheduled to occur at `nextmodeltime`.

Explanation: PFA modeling runs at a frequency of `MODELINT` minutes. When the last model was attempted, it was unsuccessful. Modeling will run again at the next scheduled model interval.

System action: The check processing continues.
Operator response: None.
System programmer response: None.
Problem determination: None.
Source: Predictive Failure Analysis (SCPFA)
Reference Documentation: None.
Automation: None.
Module: AIRHMCHK
AIRH148E

The current frames and slots used by one or more persistent address spaces is higher than expected.

**Explanation:** The current number of frames and slots used is higher than expected when compared to the model of the expected number of frames and slots used for one or more persistent address spaces. The result of this comparison exceeds the limits set by the user. An unusually high number of frames and slots used may indicate that the address space is leaking virtual storage. The next model will occur at `nextmodeltime`. The expected usage value used in the comparison was modeled at `lastsuccmodeltime`.

**System action:** The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

**Operator response:** A list of candidates for the cause of the problem has been produced which contains the jobs or address spaces which are using more frames and slots than expected. Review the list of candidates and follow your standard diagnostic procedures to determine what actions should be taken.

**System programmer response:** See the operator response to determine the cause of the frames and slots usage problem.

**Problem determination:** See the operator response to determine the cause of the frames and slots usage problem.

**Source:** Predictive Failure Analysis (SCPFA)

**Reference Documentation:** None.

**Automation:** This message should trigger an alert to system programmers or operator to determine the cause of the frames and slots usage problem.

**Module:** AIRHMCHK

Routing code: N/A

Descriptor code: N/A

---

AIRH149I

No problem with the number of frames and slots used by persistent address spaces is occurring.
The next model will occur at `nextmodeltime`. The expected usage value used in the comparison was modeled at `lastsuccmodeltime`.

**Explanation:** The model of frames and slots used by persistent address spaces for this LPAR has predicted that the usage of frames and slots is in the expected range for all persistent address spaces.

**System action:** The check processing continues.

**Operator response:** None.

**System programmer response:** None.

**Problem determination:** None.

**Source:** Predictive Failure Analysis (SCPFA)

**Reference Documentation:** None.

**Automation:** None.

**Module:** AIRHMCHK

Routing code: N/A

Descriptor code: N/A
AIRH150I Comparisons of the model of the frames and slots usage with the current frames and slots usage will occur when the check is run after modeling has run and succeeded. Modeling is scheduled for nextmodeltime.

Explanation: Sufficient data needs to be collected to build a model of frames and slots usage. Once sufficient data has been collected and a model of frames and slots usage trends has been built, comparisons can be made.

System action: The check processing continues.
Operator response: None.
System programmer response: None.
Problem determination: None.
Source: Predictive Failure Analysis (SCPFA)
Reference Documentation: None.
Automation: None.
Module: AIRHMCHK
Routing code: N/A
Descriptor code: N/A

AIRH151I Last model of frames and slots usage attempted at lastmodeltime was unsuccessful. Modeling is next scheduled to occur at nextmodeltime.

Explanation: PFA modeling runs at a frequency of MODELINT minutes. When the last model was attempted, it was unsuccessful. Modeling will run again at the next scheduled model interval.

System action: The check processing continues.
Operator response: None.
System programmer response: None.
Problem determination: None.
Source: Predictive Failure Analysis (SCPFA)
Reference Documentation: None.
Automation: None.
Module: AIRHMCHK
Routing code: N/A
Descriptor code: N/A

AIRH152E The total system message arrival rate is higher than expected and may indicate a damaged system.

Explanation: The model of the total system message arrival rate has detected that the arrival rate is higher than expected based on the previous history of this system. An unusually high message arrival rate can indicate a series of recurring failures on the system which can lead to a system hang or may be indicative of a damaged system. The message arrival rate is the ratio of the arrivals and the CPU used in the last collection interval. The prediction used in the comparison was modeled at lastsuccmodeltime and predicted the arrival rate from that time to nextmodeltime.

System action: The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

Operator response: Examine message arrivals to determine the source of the higher than expected entry arrival rate.
**System programmer response:** See operator response to determine the cause of the higher than expected message arrival rate.

**Problem determination:** See operator response to determine the cause of the higher than expected message arrival rate.

**Source:** Predictive Failure Analysis (SCPFA)

**Reference Documentation:** None.

**Automation:** This message should trigger an alert to system programmers or operator to determine the cause of the high message arrival rate.

**Module:** AIRHMCHK

**Routing code:** N/A

**Descriptor code:** N/A

**AIRH154I** No problems based on message arrival rate are expected to occur before nextmodeltime. The current message arrival rate is within the predicted range modeled at lastsuccmodeltime.

**Explanation:** The message arrival rate for the system is within the normal boundaries for this system.

**System action:** The check processing continues.

**Operator response:** None.

**System programmer response:** None.

**Problem determination:** None.

**Source:** Predictive Failure Analysis (SCPFA)

**Reference Documentation:** None.

**Automation:** None.

**Module:** AIRHMCHK

**Routing code:** N/A

**Descriptor code:** N/A

**AIRH155I** Comparisons of predictions and arrivals will occur after data has been collected and modeling has run and succeeded. Data collection waits until the system has stabilized after IPL. Modeling is scheduled for nextmodeltime.

**Explanation:** Sufficient data needs to be collected to issue a prediction of future message arrival rates. Sufficient data will be collected after at least one hour of usable data exists and after a successful model occurs. Once enough data has been collected to build a model of message arrival rate trends and a successful model has occurred, message arrival rate predictions will start to be generated.

**System action:** The check processing continues.

**Operator response:** None.

**System programmer response:** None.

**Problem determination:** None.

**Source:** Predictive Failure Analysis (SCPFA)

**Reference Documentation:** None.

**Automation:** None.

**Module:** AIRHMCHK

**Routing code:** N/A

**Descriptor code:** N/A
The message arrival rate for one or more persistent address spaces is higher than expected and may indicate a system problem.

Explanation: The model of the tracked persistent address spaces when compared to the arrival rate for each is higher than expected based on the previous history of each tracked address space. An unusually high message arrival rate can indicate a series of recurring failures for the address space which can lead to a system hang or may be indicative of a damaged system. The message arrival rate is the ratio of the arrivals and the CPU used in the last collection interval. The prediction used in the comparison was modeled at lastsuccmodeltime and predicted the arrival rate from that time to nextmodeltime.

System action: The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

Operator response: Examine message arrivals to determine the source of the higher than expected entry arrival rate.

System programmer response: See operator response to determine the cause of the higher than expected message arrival rate.

Problem determination: See operator response to determine the cause of the higher than expected message arrival rate.

Source: Predictive Failure Analysis (SCPFA)

Reference Documentation: None.

Automation: This message should trigger an alert to system programmers or operator to determine the cause of the high message arrival rate.

Module: AIRHMCHK

Routing code: N/A

Descriptor code: N/A
AIRH169E

The message arrival rate for one or more non-persistent address spaces is higher than expected and may indicate a system problem.

Explanation: The model of the non-persistent address spaces when compared to the arrival rate for the group is higher than expected. An unusually high message arrival rate can indicate a series of recurring failures for the address space which can lead to a system hang or may be indicative of a damaged system. The message arrival rate is the ratio of the arrivals and the CPU used in the last collection interval. The predictions used in the comparison were modeled at `lastsuccmodeltime` and predicted the arrival rate from that time to `nextmodeltime`.

System action: The WTOTYPE of the check has been changed to NONE so that subsequent runs of this check will not produce an operator message until new data is available. The check will collect data and model new predictions at the next collection interval. After modeling is successful, if the current setting of WTOTYPE is still NONE, the previous setting will be restored so that the check will issue future exceptions based on that setting.

Operator response: Examine message arrivals to determine the source of the higher than expected entry arrival rate.

System programmer response: See operator response to determine the cause of the higher than expected message arrival rate.

Problem determination: See operator response to determine the cause of the higher than expected message arrival rate.

Source: Predictive Failure Analysis (SCPFA)

AIRH172I The check parameters were changed successfully. The check will run after the next collection which is scheduled for `nextcolltime` if enough data has been collected and a successful model has occurred.

Explanation: The check is designed to run after a data collection. It will not run immediately after parameters were changed. The parameters have been changed successfully.

System action: The check processing continues.

Operator response: None.

System programmer response: None.

Problem determination: None.

Source: Predictive Failure Analysis (SCPFA)
AIRH173I  Last model of future message arrival rates attempted at lastmodeltime was unsuccessful. Modeling is next scheduled to occur at nextmodeltime.

Explanation:  PFA modeling runs at a frequency of MODELINT minutes. When the last model was attempted, it was unsuccessful. Modeling will run again at the next scheduled model interval.

System action:  The check processing continues.

Operator response:  None.

System programmer response:  None.

Problem determination:  None.

Source:  Predictive Failure Analysis (SCPFA)

Reference Documentation:  None.

Automation:  None.

Module:  AIRHMCHK

Routing code:  N/A

Descriptor code:  N/A
Part 3. Diagnosing a problem

After you identify the problem type, use the following diagnosis procedures to identify the source and extract symptoms to build a search argument. This section covers the following topics:

- Chapter 8, “Diagnosing an abend,” on page 85
- Chapter 9, “Diagnosing a system hang or wait state,” on page 105
- Chapter 10, “Diagnosing a job or subsystem hang,” on page 121
- Chapter 11, “Diagnosing a loop,” on page 129
- Chapter 12, “Diagnosing an output problem,” on page 143
- Chapter 13, “Diagnosing a performance problem,” on page 153
Chapter 8. Diagnosing an abend

Overview of an abend

The purpose of this chapter is to guide the diagnosis of an abnormal end (abend). Abends have an associated system completion code to describe the error and most have a reason code to further explain the error. These codes can be found by searching:

- LookAt http://www.ibm.com/systems/z/os/zos/bkserv/lookat/
- z/OS MVS System Codes
- The documentation for the particular application that failed. For example:
  - For Language Environment completion codes, see z/OS Language Environment Run-Time Messages
  - For RMF completion codes, see z/OS RMF Messages and Codes

An abend is classified as follows:

- Software-detected:
  - A system code in the form of three hexadecimal digits, possibly with a four byte reason code. For example, ABEND075. A system abend code is issued with the ABEND or CALLRTM macros used to terminate a task or address space when a system service or function detects an error.
  - A user code in the form of a four decimal digits, possibly with a four byte reason code. For example, ABENDU4094. A user code is issued using the ABEND macro to terminate a task or the entire job step. When the highest-level task in a job step ends abnormally, all related tasks or subtasks also terminate. When a subtask terminates, only work running on behalf of the subtask is affected, unless STEP=YES is specified.

- Hardware-detected:
  Hardware might present a program interrupt or machine check on the execution of an instruction. The operating system detects these hardware problems and presents them as an abend.

  Example: An instruction in an application running in storage key 7 references storage assigned to key 0. The difference in storage key causes a protection exception. This exception results in hardware presenting a program interruption code of 0004 to the operating system, which is externalized as ABEND0C4.

Related information:
- z/Architecture® Principles of Operation, SA22-7832

Symptoms of an abend: You can identify an abend by one or more of the following indicators:

- A symptom dump message on the console, in the system log, or job log can indicate a system or user abend.

  For example, message IEA9951 is issued to the console:

  **System message indicating an abend**

  Notice the indication of a system completion code.

  IEA9951 SYMPTOM DUMP OUTPUT 731
  SYSTEM COMPLETION CODE=EC6  REASON CODE=0000FD18
  TIME=13.58.26  SEQ=00724  CPU=0000  ASID=0147
  PSW AT TIME OF ERROR  070C4400  A90B111A  ILC 2  INTC 7B
  NO ACTIVE MODULE FOUND - PRIMARY NOT EQUAL TO HOME
Abend analysis

NAME=UNKNOWN
DATA AT PSW  290B1114 - 1F001F11  05EFEBEC  D2640096
AR/GR  0: 00000000/00000000  1: 00000000/00000000
  2: 00000000/2ED9E4D0  3: 00000007/820A33B8
  4: 00000000/29AF3180  5: 00000000/2ED9E200
  6: 00000000/28AADD08  7: 00000002/00000100
  8: 00000000/294F31D8  9: 00000000/2ED9E200
 A: 00000000/00FD8B28  B: 00000000/000000E3
 C: 00000000/006FF390  D: 00000000/00F9C5B8
 E: 00000000/A90B111A  F: 00000000/00FFFB4C

- A system message indicating an SVC dump was requested for an error:
  For example, here are some messages that might be issued when a SVC dump is
taken for an error:

    IEA794I SVC DUMP HAS CAPTURED: 357
    DUMPID=002 REQUESTED BY JOB (OMVS)
    DUMP TITLE=COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPXLK
    LCP+LDC2,ABEND=S0422,REASON=083A01A5
    IEA6111 COMPLETE DUMP ON DUMP.MVS06.D060320.T162245.S00034
    DUMPID=034 REQUESTED BY JOB (ZFS)
    FOR ASID (1001)
    INCIDENT TOKEN: ORACLE MVS06 03/21/2006 00:22:45
    ID = IOEDFS
    IEA911E COMPLETE DUMP ON SYS1.DUMP08
    DUMPID=001 REQUESTED BY JOB (RESOLVER)
    FOR ASIDS(003B,0001)
    INCIDENT TOKEN: CWYPLEX1 CPUX 04/21/2006 14:38:56
    ERROR ID = SEQ00046 CPU00 ASID003B TIME09.38.56.1

- An application detects an error. One example is the following ISMF panel:

<table>
<thead>
<tr>
<th>ISMF ABEND PANEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND ===&gt;</td>
</tr>
<tr>
<td>*******************************************************</td>
</tr>
<tr>
<td>**            **</td>
</tr>
<tr>
<td>** AN ABEND OCCURRED WHILE EXECUTING ISMF **</td>
</tr>
<tr>
<td>**            **</td>
</tr>
<tr>
<td>** SYSTEM ABEND CODE: OC4 **</td>
</tr>
<tr>
<td>**            **</td>
</tr>
<tr>
<td>** ISMF CANNOT CONTINUE **</td>
</tr>
<tr>
<td>**            **</td>
</tr>
<tr>
<td>** PRESS THE ENTER KEY OR USE END TO TERMINATE ISMF **</td>
</tr>
<tr>
<td>** USE HELP TO DISPLAY A LIST OF COMMON ABEND CODES **</td>
</tr>
<tr>
<td>**            **</td>
</tr>
<tr>
<td>*******************************************************</td>
</tr>
</tbody>
</table>

Another example is the ISPF panel:

| ---------------------ERROR RECOVERY------------------------------- |
| COMMAND ===>          |
| *******************************************************     |
| ** ISPF PROCESSOR ENDED ABNORMALLY **                       |
| **            **                                      |
| **            **                                      |
| ** Task ABEND code OC1 **                                  |
| **            **                                      |
| **            **                                      |
| Press ENTER to display primary option menu.                |
| Enter HELP command for list of common ABEND CODES.         |
| **            **                                      |
| *******************************************************     |

- A component, function, subsystem or application message indicating an abend
occurred through a message. For example, TSO/E message INMR030I that
identifies an abend condition:
Abend analysis

**TSO/E message**

INMR030I RECEIVE command terminated. ABEND abend_code.

- An error is recorded in SYS1.LOGREC record.
  
  For example:
  
  ERRORID: SEQ=11696 CPU=0040 ASID=00A1 TIME=12:48:20.3

**SEARCH ARGUMENT ABSTRACT**

PIDS/5752SCXMS RIDS/IEANUC01#:L RIDS/IEAVXALA AB/S013E REGS/00000 REGS/C009C RIDS/IEAVXALR#:R

**SYMPTOM DESCRIPTION**

<table>
<thead>
<tr>
<th>PROGRAM ID: 5752SCXMS</th>
<th>LOAD MODULE NAME: IEANUC01</th>
<th>CSECT NAME: IEAVXALA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM ABEND CODE: 013E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGISTER/PSW DIFFERENCE FOR R0D: 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGISTER/PSW DIFFERENCE FOR R0C:-009C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECOVERY ROUTINE CSECT NAME: IEAVXALR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OTHER SERVICEABILITY INFORMATION**

- **RECOVERY ROUTINE LABEL**: IEAVXALR
- **DATE ASSEMBLED**: 96270
- **MODULE LEVEL**: HBB6603
- **SUBFUNCTION**: ACCESS LIST ADD

**TIME OF ERROR INFORMATION**

- **PSW**: 070C0000 80FF5D00
- **INSTRUCTION LENGTH**: 02
- **INTERRUPT CODE**: 0078
- **FAILING INSTRUCTION TEXT**: 5DB88140 174458C0 022856C0

**REGISTERS 0-7**

- **GR**: 01381495 00000000 00000002 7FFFBF00 00000000 7FFFBF10 00000000
- **AR**: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

**REGISTERS 8-15**

- **GR**: 00FF3D88 01382494 81380496 00F9B700 80FF5D9C 80FF5D00 81380D58 01BFD620
- **AR**: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

**HOME ASID**: 00A1  **PRIMARY ASID**: 0002  **SECONDARY ASID**: 00A1

**PKM**: 80C0  **AX**: 0001  **EAX**: FFFF

Often, a system completion code or a wait state code indicate an abend. However, there are some exceptions, so use the following table to help guide your diagnosis of an abend:

<table>
<thead>
<tr>
<th>If you receive wait state code</th>
<th>Code represents</th>
<th>Diagnose with</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'071'</td>
<td>System failure or the operator initiated a restart.</td>
<td>Chapter 11, “Diagnosing a loop,” on page 129</td>
<td>Find complete explanations of wait state codes in MVS System Codes.</td>
</tr>
<tr>
<td>Abend X'122'</td>
<td>Operator canceled the job, requesting a dump.</td>
<td>Chapter 10, “Diagnosing a job or subsystem hang,” on page 121</td>
<td>This abend might also indicate a loop, see Chapter 11, “Diagnosing a loop,” on page 129. Find complete explanations of wait state codes in MVS System Codes.</td>
</tr>
</tbody>
</table>
## Abend analysis

<table>
<thead>
<tr>
<th>If you receive wait state code</th>
<th>Code represents</th>
<th>Diagnose with</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abend X'222'</td>
<td>Operator canceled the job, without requesting a dump.</td>
<td>Chapter 10, “Diagnosing a job or subsystem hang,” on page 121</td>
<td>This abend might also indicate a loop; see Chapter 11, “Diagnosing a loop,” on page 129. Find complete explanations of wait state codes in <a href="z/OS_MVS_System_Codes">z/OS MVS System Codes</a>.</td>
</tr>
<tr>
<td>X'322'</td>
<td>Job exceeded the time limit specified by the TIME option.</td>
<td>Chapter 11, “Diagnosing a loop,” on page 129</td>
<td>Find complete explanations of wait state codes in <a href="z/OS_MVS_System_Codes">z/OS MVS System Codes</a>.</td>
</tr>
<tr>
<td>All others</td>
<td>&quot;Steps for diagnosing an abend.&quot;</td>
<td>To find the abend code and reason code, see “Obtaining the abend and reason code” on page 89.</td>
<td></td>
</tr>
</tbody>
</table>

### Steps for diagnosing an abend

**Before you begin:** You need to know how to use IPCS and have access to the following:
- SVC dump, SYSMDUMP, or stand-alone dump
- EREP of software records (TYPE=S) from SYS1.LOGREC or IPCS VERBX LOGDATA report from a dump.
- OPERLOG, SYSLOG, Job log, other message log for the time frame of the error

You should also be able to locate and use the following:
- IBM product documentation; see Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165.

Use the following flowchart to guide diagnosis of an abend.
Obtaining the abend and reason code

The abend code indicates the nature of the problem. If you have the abend code, move on to "Identifying the module and component" on page 93.

Steps for obtaining the abend code

**Before you begin**: You need to know how to access the following:

- SVC dump, SYMDUMP, and transaction dump
- Software record in SYS1.LOGREC from the time frame of the error.

If an SVC dump was taken for the error as indicated by IEA794I, IEA611I or IEE911E, use the following IPCS commands to format information and extract the abend and reason code (when available).

- IPCS STATUS FAILDATA
Abend analysis

For the abend code, look for AB/S0hhh in the symptoms under the heading SEARCH ARGUMENT ABSTRACT, if present. For the reason code, look for PRCS/hhhhhhh in the symptoms, or look in the register indicated in the abend code explanation.

Not every dump has error information associated with it. Some dumps, like those requested through the SLIP or DUMP command will not have STATUS FAILDATA output. There are occasions when STATUS FAILDATA will not provide the information you need. Check SYSTRACE and SUMMARY FORMAT (look for RTM2WA SUMMARY) for the abend. If you cannot find it, the dump was probably not requested for an abend. Check the dump title (L Title) and determine why it was not requested. Also, check SYS1.LOGREC, and the SYSLOG and job log for the time frame of the dump.

In the following example, fields AB/S005C and PRCS/00000214 indicate what to extract for the abend and reason code. The free-format search argument is: ABEND05C and RSN00000214.

* * * DIAGNOSTIC DATA REPORT * * *

SEARCH ARGUMENT ABSTRACT

RIDS/IEFW21SD#L RIDS/#UNKNOWN AB/S005C PRCS/00000214 REGS/0E01E REGS/0C6D8

Symptom Description
--- -----------
RIDS/IEFW21SD#L Load module name: IEFW21SD
RIDS/#UNKNOWN Csect name: #UNKNOWN
AB/S005C System abend code: 005C
PRCS/00000214 Abend reason code: 00000214
REGS/0E01E Register/PSW difference for R0E: 01E
REGS/0C6D8 Register/PSW difference for ROC: 6D8

SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

Program id
Recovery routine csect name
Recovery Routine Label
Date Assembled
Module Level
Subfunction

Time of Error Information

PSW: 071C2000 83AA2110 Instruction length: 02 Interrupt code: 000D
Failing instruction text: BFFFB148 0A0D98EC B08807FE

Breaking event address: 00000000_00000000
AR/GR 0-1 00000000/00000000_7F10C128 00000000/00000000_7F363028
AR/GR 2-3 00000000/00000000_066A336 00000000/00000000_066A354B
AR/GR 4-5 00000000/00000000_066A2D64 00000000/00000000_7F363028
AR/GR 6-7 00000000/00000000_066A336 00000000/00000000_00000000
AR/GR 8-9 00000000/00000000_00000000_00000000_00000000

In the following STATUS FAILDATA output, the symptom string indicates a system abend code of X'0C4' with a reason code of X'00000010'

SEARCH ARGUMENT ABSTRACT

PIDS/5752SCPX1 RIDS/BPXINPVTL#L RIDS/BPXVPFXC AB/S00C4 PRCS/00000010
REGS/C4E10 RIDS/BPXMPCE#R

Symptom Description
--- -----------
PIDS/5752SCPX1 Program id: 5752SCPX1
RIDS/BPXINPVTL#L Load module name: BPXINPVVT
Abend analysis

RIDS/BPXVFPT Csect name: BPXVFPT
AB/S00C4 System abend code: 00C4
PRCS/00000010 Abend reason code: 00000010
REGS/C4E10 Register/PSW difference for R0C:-4E10
RIDS/BPXMPCE#R Recovery routine csect name: BPXMPCE

OTHER SERVICEABILITY INFORMATION

Recovery Routine Label: BPXMPCE
Date Assembled: 12/19/04
Module Level: HBB7720
Subfunction: OpenMVS

Time of Error Information

PSW: 47043000 80000000 00000000 27EA624C
Instruction length: 02 Interrupt code: 0010
Failing instruction text: 58E0D56C 18F10E0E B2190200
Translation exception address: 00000000_66831000

• EREP or VERBX LOGDATA

Use EREP to format software records (TYPE=S) recorded to SYS1.LOGREC for the time frame of the failure or error or format SYS1.LOGREC records from the in storage buffer using the VERBX LOGDATA command. For information about EREP, see EREP Reference.

To find software records that might be associated with the failure, follow these suggestions:
– Look for the general time frame
– Search on the job name or ASID involved
– Search for the failing component id.

For the abend code, look for AB/S00hhh in the symptoms under the heading SEARCH ARGUMENT ABSTRACT, if present.

For the reason code, look for PRCS/hhhhhh in the symptoms, or look in the register indicated in the abend code explanation.

Here is an example of a software logrec entry formatted by EREP or the IPCS VERBX LOGDATA command against an SVC dump. The free-format search argument generated for this entry is: ABEND05C RSN00000214.

TYPE: SOFTWARE RECORD REPORT: SOFTWARE EDIT REPORT DAY.YEAR (SVC 13) REPORT DATE: 115.06 FORMATTED BY: IEAVTFDE HBB7703 ERROR DATE: 096.06 MODEL: 2084 HH+MM:SS.TH SERIAL: 00809F TIME: 12:14:42.66

JOBNAME: HSMVM0A SYSTEM NAME: VM0A ERRORID: SEQ=03567 CPU=0000 ASID=003C TIME=12:14:42.6

SEARCH ARGUMENT ABSTRACT

PIDS/5752SC1B4 RIDS/IEFW21SD#L RIDS/IEFDB400 AB/S005C PRCS/00000214 REGS/00E10 REGS/0C6DB RIDS/IEFDB402#R

SYMPTOM DESCRIPTION

---------

PIDS/5752SC1B4 PROGRAM ID: 5752SC1B4
RIDS/IEFW21SD#L LOAD MODULE NAME: IEFW21SD
RIDS/IEFDB400 CSECT NAME: IEFDB400
AB/S005C SYSTEM ABEND CODE: 005C
PRCS/00000214 ABEND REASON CODE: 00000214
REGS/00E10E REGISTER/PSW DIFFERENCE FOR ROC: 01E
REGS/0C6DB REGISTER/PSW DIFFERENCE FOR ROC: 02E
RIDS/IEFDB402#R RECOVERY ROUTINE CSECT NAME: IEFDB402

Chapter 8. Diagnosing an abend
Abend analysis

OTHER SERVICEABILITY INFORMATION

RECOVERY ROUTINE LABEL: IEFAB4ED
DATE ASSEMBLED: 05223
MODULE LEVEL: UA20441
SUBFUNCTION: DYNAMIC ALLOCATION

TIME OF ERROR INFORMATION

PSW: 071C2000 83AA2110 INSTRUCTION LENGTH: 02 INTERRUPT CODE: 000D
FAILING INSTRUCTION TEXT: BFFB148 0A0D98EC B08807FE

REGISTERS 0-7
GR: 00000000 0405C000 7F10C128 00000024 006EA338 006A3648 006A2D64 7F363028

• OPERLOG, SYSLOG, or job log or VERBEXIT MTRACE

Look for message IEA995I or other messages with an abend code in the message text. The message might also give a reason code. For example:

IEA995I SYMPTOM DUMP OUTPUT 694
USER COMPLETION CODE=4039 REASON CODE=00000000
TIME=05.07.41 SEQ=33565 CPU=0000 ASID=0247
PSW AT TIME OF ERROR 078D1000 99937C66 ILC 2 INTC 0D
ACTIVE LOAD MODULE ADDRESS=1987A4C0 OFFSET=000BD7A6
NAME=CEEPLPKA
DATA AT PSW 19937C60 - 00181610 0A0D58D0 D00498EC
AR/GR 0: 00C4BB3E/04000000 1: 00000000/04000FC7
2: 00000000/00AE1E08 3: 00000000/0002000D
4: 00000000/1992CA78 5: 00000000/00A1954
6: 00000000/00000000 7: 00000000/198758E0
8: 00000000/00AE1E08 9: 00000000/00A34C6
A: 00000000/00A1954 B: 00000000/19937890
C: 00000000/009EA58 D: 00000000/00A4608
E: 00000000/1992BBAE F: 00000000/00000000
END OF SYMPTOM DUMP

In message texts, an abend code can be called a system completion code or a SYS CODE. A message can show an abend code in the variable text without identifying it as an abend code; use the message explanation to understand the variable text.

• VERBEXIT SYMPTOM output from a dump: Format the dump completely, as described in step 1 on page 101 of “Gathering additional problem data for abends” on page 97. Look for AB/S0hhh and PRCS/hhhhhhh in the symptoms.

In the following VERBEXIT SYMPTOM output, the primary symptom string indicates a system abend code of 'X'03C' and a return code of 'X'2D000810'.

Primary Symptom String:

RIDS/NUCLEUS#L RIDS/IARYTASS PIDS/5752SIC1CR AB/S003C RIDS/IARRR#R
VALU/HCO099680 REGS/E0084 REGS/088FA PRCS/20000810 VALU/CNAGEMENT

<table>
<thead>
<tr>
<th>Symptom</th>
<th>Symptom data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIDS/NUCLEUS#L</td>
<td>NUCLEUS#L</td>
<td>Routine identifier</td>
</tr>
<tr>
<td>RIDS/IARYTASS</td>
<td>IARYTASS</td>
<td>Routine identifier</td>
</tr>
<tr>
<td>PIDS/5752SIC1CR</td>
<td>5752SIC1CR</td>
<td>Component identifier</td>
</tr>
<tr>
<td>AB/S003C</td>
<td>003C</td>
<td><strong>ABEND code - system</strong></td>
</tr>
<tr>
<td>RIDS/IARRR#R</td>
<td>IARRR#R</td>
<td>Routine identifier</td>
</tr>
<tr>
<td>VALU/HCO099680</td>
<td>C0099680</td>
<td>Error related hexadecimal value</td>
</tr>
<tr>
<td>REGS/E0084</td>
<td>E0084</td>
<td>Program register</td>
</tr>
<tr>
<td>REGS/088FA</td>
<td>088FA</td>
<td>Program register</td>
</tr>
<tr>
<td>PRCS/20000810</td>
<td>20000810</td>
<td>Return code</td>
</tr>
</tbody>
</table>
Abend analysis

The dump does not contain a secondary symptom string.

- Dump title

Look at the dump title; some titles contain the abend and reason codes. Use the DISPLAY DUMP;TITLE or DISPLAY DUMP;ERRDATA to display the dump title and any error information associated with captured dumps, or dumps written to pre-allocated or dynamically allocated data sets. In response to the DISPLAY command, message IEE853I or IEE854I are issued containing the requested information. Look in the IEE853I or IEE854I message replies for the abend code, reason codes and the registers. For example:

```
IEE853I 12.54.26 SYS1.DUMP TITLES 939
SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=002
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00000000M
DUMP00 TITLE=ABEND=S0C4,RC=0010,COMPON=SDSF-ESTAE,COMPID=5647-A01
,ISSUER=ISFSTAE,SDSF ABEND ROUTINE
DUMP TAKEN TIME=17.49.48 DATE=nn/nn/nnnn
DUMP01 TITLE=ABEND=S0C4,RC=0010,COMPON=SDSF-ESTAE,COMPID=5647-A01
,ISSUER=ISFSTAE,SDSF ABEND ROUTINE
DUMP TAKEN TIME=nn.nn.nn DATE=nn/nn/nnnn
```

For a dump that has been copied from the SYS1.DUMPxx data set or for an SVC dump you are viewing in IPCS, use the IPCS LIST TITLE subcommand to obtain the dump title.

You know you are done when you locate the abend and reason codes. You can then look up a description of the abend code using the product documentation or LookAt and follow the recommendations.

Related information:
- For LookAt and IBM product documentation, see www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/ and Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165.
- For information about the IPCS STATUS FAILDATA subcommand, see z/OS MVS IPCS Commands.
- For information about the logrec data set, see z/OS MVS Diagnosis: Tools and Service Aids.
- For information about abend codes, see the product documentation. For example:
  - z/OS MVS System Codes
  - z/OS UNIX System Services Messages and Codes
  - z/OS Communications Server: IP Messages Volume 1 (EZA)
  - z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)
  - z/OS Communications Server: IP Messages Volume 3 (EZY)
  - z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM).

Identifying the module and component

In some cases, the abend code indicates the source of the problem and how to repair it. However, there are cases when you cannot identify if the problem was caused by a z/OS, a vendor, or an installation problem. In this case, you must analyze the abend code to see which module or component was involved to conduct a more granular search for a known problem.
Abend analysis

Steps for identifying the module and component

**Before you begin:** You need to know how to use IPCS and also have access to the following:
- SVC dump, SYSMDUMP, and SADUMP
- Job log, system log, OPERLOG or application error log.
- Master trace

You should also be able to locate and use:

Perform the following steps to identify the module that caused the abend and the offset of the failing instruction into the module.

1. Do one of the following, depending on the problem data available.
   - For an **SVC dump**, obtain the component name, component identifier, and module name from the dump title.
   - **Obtain the PIDS/cccccccc and RIDS/cccccccc** symptoms from the search argument. PIDS is the program identifier, which is the four character product identifier and the five character component identifier. RIDS identifies the module.
     
     If the search argument in an SVC dump does not identify the program and module or if the problem involves multiple abends, analyze the dump for the failing task or service request. See [z/OS MVS Diagnosis: Tools and Service Aids](http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/) for information about analyzing an SVC dump.
   - Obtain the address in the right half of the program status word (PSW) in **STATUS FAILDATA dump output**. The zArchitecture PSW is 128 bits (16 bytes). The combination of bits 31 and 32 show the current addressing mode. These are the low order bit of the left half and the high order bit of the right half of the PSW. The meanings are as follows:
     - 00 - 24-bit mode
     - 01 - 31-bit mode
     - 10 - Invalid
     - 11 - 64-bit mode
     
     In some places the PSW is shown in a shorter 64-bit (8 bytes) form. This indicates that the addressing mode is 24-bit or 31-bit and the PSW is shown in an ESA/390 form. In that case bit 31, the low order bit in the first half, shows the addressing mode:
     - 0 - 24-bit mode
     - 1 - 31-bit mode
      
     Subtract the instruction length from the PSW address to obtain the address of the failing instruction. Do not subtract the instruction length in the following cases; the failing instruction is at the PSW address.
     - Program check interruptions for which the processing of the instruction identified by the old PSW is nullified. See [z/Architecture Principles of Operation](http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/) for the interruption action. Some examples are:
       - Page translation exception: interrupt code = 0011
       - Segment translation exception: interrupt code = 0010
       - Access register translation exception
         
         The following interrupt codes result in the operation being nullified:
         - LFX translation exception = 0026
Abend analysis

- LSX translation exception = 0027
- ASX-translation exception = 0021
- ASTE-validity exception = 002B
- ASTE-sequence exception = 002C
- LSTE sequence exception = 002E
- ASTE instance exception = 002F
- Region-first-translation exception
- Region-second-translation exception
- Region-third-translation exception

Obtain the PSW and registers for the error from the **STATUS CPU REGISTERS** subcommand.

In the following **STATUS CPU REGISTERS** output, the address can be found in the second half of the PSW. Note that this presentation uses both the first (bit 32) and last (bit 63) bits in the PSW shown combine to indicate addressing mode. Bit 64 will be on when the PSW represents code running in 64 bit mode.

CPU STATUS:
PSW=070C1000 83D00B72 (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
DISABLED FOR PER
ASID(X'0015') 03D00B72. DATASY02+03CA IN EXTENDED PRIVATE
ASCB21 at F9C000, JOB(DALESY01), for the home ASID
ASXB21 at 6FE038 for the home ASID. No block is dispatched
HOME ASID: 0015 PRIMARY ASID: 0015 SECONDARY ASID: 0015
GPR VALUES
0-3 00000000 03D017B0 00000000 03D01A12
4-7 03D00EC1 03D00CE8 006D4FF8 FD000000
8-11 03D025BF 83D007A8 03D015C0 03D017A7
12-15 03D01830 03D015C0 03D019EB 03D00DA9
IEA11015I The requested ALETs are zero.
CONTROL REGISTER VALUES
0-3 5EB1EE40 00C0407F 002B5040 00800015
4-7 00000015 01756540 FE000000 00C0407F
8-11 00000000 00000000 00000000 00000000
12-15 01F7C27F 00C0407F DF881755 7F704008
THE PRECEDING STATUS CPU INCLUDED THE REGS OPTION

Example for for 31-bit:
PSW: 040C0000 816B65A6 Instruction Length: 04 Interrupt Code: 0011
Failing instruction text: 58F0C030 50F0B222 5BF0B240
Translation Exception Address: 00000000_7F37B003

Example for 64-bit:
Time of Error Information:
PSW: 04046001 80000000 80000000 0178F356
Instruction length: 04 Interrupt code: 0004
Failing instruction text: 000A5023 00000032 0004172

STATUS CPU REGISTERS supplies the name of the failing module and its offset without having to do a separate WHERE subcommand.

2. Do one of the following:
   - If analyzing the dump interactively, use the instruction address in an IPCS **WHERE** subcommand to obtain the name of the load module and the offset of the address into the load module. If the module name is not proceeded with IECNV01, then IPCS has given the load module name. If you enter the **STATUS CPU REGISTERS** subcommand, a **WHERE** is performed automatically.

Use the AMBLIST service aid to list the CSECTs in the load module. Use the offset into the load module to identify the CSECT containing the failing
Abend analysis

instruction. Then subtract the starting address of the CSECT from the instruction address to obtain the offset into the CSECT.

For instructions on using the AMBLIST service aid, see AMBLIST in z/OS MVS Diagnosis: Tools and Service Aids.

- If the WHERE command does not present a module name, follow this procedure:
  - Using the IPCS Browse panel, enter the PSW instruction address and ASID obtained from the time of error information. Browse backwards looking for the CSECT name eye-catcher. IBM module names are generally followed by an assembly date and a product identifier or PTF level, such as HBB7720; most eye-catchers are at the beginning of a module, but some are at the end.

3. The module prefix identifies the component, subsystem, or product, if provided by IBM. See the module identification chart in z/OS MVS Diagnosis: Reference. For example, using the information in the following output from you can determine what component was involved in an error from the module prefix. The ADY module prefix indicates a DAE-type error.

<table>
<thead>
<tr>
<th>Module Prefix</th>
<th>Component Name</th>
<th>Product ID</th>
<th>Component ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF</td>
<td>TSO and TSO/E session manager</td>
<td>5665</td>
<td>28505</td>
</tr>
<tr>
<td>ADY</td>
<td>Dump analysis and elimination (DAE)</td>
<td>5752</td>
<td>SC143</td>
</tr>
</tbody>
</table>

4. Continue diagnosis as follows:
- For a z/OS component, continue with “Steps for searching the problem reporting databases.”
- For an IBM subsystem or product, continue diagnosis with the subsystem or product. See Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165 for a listing of components and products.
- For an installation-provided program, including an installation exit routine, continue diagnosis with that program, using the dump for the abend.

Related information:
- See “Searching problem reporting databases” on page 8 for more information.
- See z/OS MVS Diagnosis: Tools and Service Aids for dump analysis of a problem in an installation-provided program.
- See z/OS MVS IPCS Commands for information about IPCS subcommands.

Searching the problem reporting databases

Search arguments are used to search problem reporting databases. If the problem you are diagnosing was already reported and the symptoms are in the database, the search produces a match. Searching is an iterative process; you might need to gather additional data and continue your search.

Steps for searching the problem reporting databases

1. Search the problem reporting database to determine if the problem was previously reported. See “Searching for a known problem” on page 10.
   Use the free-format search argument abstract as a symptom string to determine if the problem already exists. For example ABEND05C RSN00000241.
   For example, the following search argument abstract would generate the free-format search argument: ABEND03C RSN20000810:
2. If the search provides no match, remove some symptoms to broaden the search. If the search provides too many symptoms, try adding some symptoms to limit the scope. Check to see which matches pertain to the operating system environment.

3. If the search finds that the problem was previously reported, request the problem fix. If not, continue with "Gathering additional problem data for abends." Use the problem data gathered there to create more symptoms; use these symptoms in later searches.

4. If you still cannot find the cause of the abend or if the problem is new, report the problem to IBM.

Provide the information in Chapter 16, “Problem diagnostic worksheet,” on page 173, such as:

- Any changes made to the system recently, preceding when the problem began occurring (for example, PTFs or new products installed or new hardware).
- Problem type: abend
- Search argument
- Dump, formatted by IPCS, online or printed
- Failing input request: macro, command, or statement
- SDWAVRA keys, lengths, and contents
- Offset of the failing instruction into the module or CSECT
- Accompanying messages: identifiers and texts
- Logrec report, if used
- All printed output and output data sets related to the problem
- Data on any related problems
- Module name and level
- Name and level of the operating system(s) with a list of program temporary fixes (PTF) applied at the time of the problem and all installation modifications, exits, and products with other than Class A service
- Other problem data developed while using the diagnosis book for the component, subsystem, or program

**Gathering additional problem data for abends**

Gathering additional data will increase your chances of finding a match in the problem reporting databases. Use the procedures outlined in this section to create more symptoms; use these symptoms in later searches.

**Steps for gathering additional data for abends**

It is important to gather the following information:

- The impact of the problem to system or sysplex
- The names of jobs, functions or programs that were running at the time of the error
- The existence of any new software maintenance or hardware changes
- The associated messages from job log, SYSLOG, or OPERLOG
- The related component traces that are active.
Abend analysis

This procedure is divided into the following sections:

- “Steps for gathering trace data for abends”
- “Steps for collecting additional messages and logrec for abends” on page 99
- “Steps for obtaining a dump for the error” on page 101

Before you begin any of these tasks: Complete the steps in Chapter 8, “Diagnosing an abend,” on page 85.

Steps for gathering trace data for abends

Use the following steps to gather additional data from system trace table:

1. Analyze the system trace table, which is formatted by the SYSTRACE CURRENT ERROR subcommand. A system trace provides a record of system events. Use it to create a picture of the processing occurring at the time of the error.

2. Starting at the end of the trace, back up to the entry for the abend being diagnosed:
   - *SVC D or *SVCE D in the IDENT CD/D columns
   - The abend code in the right 3 bytes in the UNIQUE-3 column
   - The reason code in the UNIQUE-1 column

   Example: In the following SYSTRACE output, the *SVC D indicates that an abend code has been loaded for processor 02. When examining system trace output, look for RCVY entries that represent entry into a recovery routine following an error or interruption.

<table>
<thead>
<tr>
<th>PR</th>
<th>ASID</th>
<th>TCB-ADDR</th>
<th>IDENT CD/D</th>
<th>PSW</th>
<th>ADDRESS...</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>000D</td>
<td>006F8E88</td>
<td>SSRV 12D</td>
<td>8120FDB8</td>
<td>006F8E88 000B0000 00000000 00000000</td>
</tr>
<tr>
<td>01</td>
<td>002E</td>
<td>006F8E88</td>
<td>SVC 30</td>
<td>070C3000</td>
<td>827FAF36 00000000 00000001 072CFBF4</td>
</tr>
<tr>
<td>01</td>
<td>002E</td>
<td>006F8E88</td>
<td>PC ... 0</td>
<td>81157326</td>
<td>00100</td>
</tr>
<tr>
<td>01</td>
<td>002E</td>
<td>006F8E88</td>
<td>PT ... 0</td>
<td>8118782C</td>
<td>002E</td>
</tr>
<tr>
<td>02</td>
<td>000D</td>
<td>006F8E88</td>
<td>DSP 070C0000</td>
<td>8101A9B0</td>
<td>00000000 0000001 072CFBF4</td>
</tr>
<tr>
<td>02</td>
<td>000D</td>
<td>006F8E88</td>
<td>SVC D 070C0000</td>
<td>8101A9B2</td>
<td>00D12090 0010035C 000295A8</td>
</tr>
<tr>
<td>02</td>
<td>000D</td>
<td>006F8E88</td>
<td>PC ... 0</td>
<td>811D7626</td>
<td>00506</td>
</tr>
<tr>
<td>01</td>
<td>002E</td>
<td>006F8E88</td>
<td>SVC ... 0</td>
<td>827FAF36</td>
<td>00000000 00000028 0000002C</td>
</tr>
<tr>
<td>01</td>
<td>002E</td>
<td>006F8E88</td>
<td>SVC 30</td>
<td>070C3000</td>
<td>827FAF64 00000000 00000028 072CFBF4</td>
</tr>
<tr>
<td>01</td>
<td>002E</td>
<td>006F8E88</td>
<td>SVC A 070C1000</td>
<td>80F8146A</td>
<td>00000000 FD000236 00F81468</td>
</tr>
</tbody>
</table>

   For system trace, when viewing program checks, look for the PGM entry just before a RCVY entry.

<table>
<thead>
<tr>
<th>PR</th>
<th>ASID</th>
<th>TCB-ADDR</th>
<th>IDENT CD/D</th>
<th>PSW</th>
<th>ADDRESS...</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>00C4</td>
<td>009FF540</td>
<td>PGM 004</td>
<td>07BD3400</td>
<td>A9308D12 00040004 00000000 00000000</td>
</tr>
<tr>
<td>06</td>
<td>00C4</td>
<td>009FF540</td>
<td>RCVY PROG</td>
<td>940C4000</td>
<td>00000000 0000000</td>
</tr>
</tbody>
</table>

3. In the entry, note the processor in the PR column, the address space identifier in the ASID column, and the task control block (TCB) address in the TCB-ADDR column. The ASID should be the same as the ASID identified in STATUS FAILDATA or STATUS CPU output.

4. Continue backing up, looking for the following entries:
   - The entry for the system service being processed when the abend occurred, if the abend occurred during system processing. This entry will have SVC, SSRV, or SVCE in the IDENT column and the same ASID as the abend entry.
   - Problem entries, which have an asterisk (*) before the identifier in the IDENT column.
   - Other entries for the same processor, PR column.
   - Other entries for the same address space, ASID column.
Abend analysis

- Repeated requests by a program for one or more system services. This pattern indicates a loop. An enabled loop has multiple types of entries before the pattern repeats. Continue diagnosis with the program requesting the system services.

You should now be able to determine the source of the abend or have more information to search the problem reporting database.

Related information
- For information about IPCS subcommands: SELECT, SUMMARY, VERBEXIT LOGDATA, and VERBEXIT MTRACE, see [z/OS MVS IPCS Commands](#).
- For the format and contents of the CDE, RB, RTM2WA, SDWA, VRAMAP (VRA keys), and TCB, see the version of [z/OS MVS Data Areas](#) that corresponds to the release you are running in your environment. See the z/OS library at [http://www.ibm.com/systems/z/os/zos/bkserv/](http://www.ibm.com/systems/z/os/zos/bkserv/)
- For the formats of system trace entries, see [z/OS MVS Diagnosis: Tools and Service Aids](#).
- For information about the SLIP command, see [z/OS MVS System Commands](#).
- See [EREP User’s Guide](#) for formatting logrec records.
- See [z/OS MVS System Messages, Vol 6 (GOS-IEA)](#) for message IEA995I.
- See [z/OS MVS System Messages, Vol 9 (IGF-IWM)](#) for the IOS messages.
- For the PGM parameter, see [z/OS MVS ICL Reference](#).
- For interactive TSO/E commands, see [z/OS TSO/E Command Reference](#).

Steps for collecting additional messages and logrec for abends
Use the following steps to gather additional messages and logrec:

1. Collect and analyze messages and logrec records about the problem. Use the ERRORID from the dump message and time stamps to select messages and software, symptom, and hardware logrec records related to the problem. Look in the following:
   - The job log
   - A TSO/E user’s ISPF transaction log or session manager log
   - The hardcopy log, also known as the system log (SYSLOG)
   - VERBEXIT MTRACE dump output, which shows the buffer for system messages
   - VERBEXIT LOGDATA dump output, which formats the logrec buffer
   - The logrec data set, formatted by EREP

2. Look for the following:
   - Symptom dump message IEA995I for a previous, related abend
   - Messages identifying a failing program with a nonzero return code
   - I/O error messages

3. Identify the program being processed when the abend occurred by obtaining the job name from the following:
   - SUMMARY output
   - VERBEXIT LOGDATA output
   - Messages in the job log
   - Messages in VERBEXIT MTRACE output
   - SELECT output
     For example, in the following SELECT output, the job name NVAST in address space 0073 contains an error.
Abend analysis

4. If a batch job was being processed, obtain the program name from the PGM parameter on the JCL EXEC statement.
   In the following output, the PGM parameter of JCL statement indicates that the program name is UNIONE.
   
   ```
   //BANK1 EXEC PGM=UNIONE,PARM='@PLANID=1,10S,SHR',
   // REGION=1024K,COND=(8,LE)
   //BANKLOG DD DSN=NULLFILE,DISP=SHR
   ```

5. If interactive work was being processed, use the command being processed to identify the program.

6. Analyze the problem data for multiple problems. Collect data for related problem ERRORIDs that occur in a similar time frame. You can find this data in the logrec data set. The time stamps are a few seconds before or after the time stamp for the abend being diagnosed. The data involves the following:
   - The same job step
   - The same task (TCB) or service request (SRB)
   - The same home address space (ASID)
   - The address spaces involved in cross-memory mode processing
   - The same processor (CPU), if the problem occurred while the system was disabled for input/output (I/O) and external interrupts (EXT), as indicated in STATUS CPU dump output.

Look for the following:

- In the output from IPCS SUMMARY TCBERROR, look at the task completion codes in job step program TCB CMP fields; a nonzero completion code indicates an abend. You are looking at the correct abend if it has an associated RTM2WA. If a related task abended seconds before the abend being diagnosed, check the task’s CDE, RTM2WA, and SDWA control blocks for the module name and other data about the abend. The output contains one RTM2WA for each abend being processed.

In the following SUMMARY TCBERROR output, the nonzero CMP field of the TCB indicates an error.

   ```
   TCB: 009F3EB8
   CMP...... 940C9000  PKF...... 80  LMP...... FF  DSP...... FF
   TSFLG.... 00  STAB..... 00FD200  NDS........ 00020000
   JSCB..... 00FF40C  BITS..... 00000000  DAR...... 00
   RTWA...... 00000000  FBT1..... 00
   Task non-dispatchability flags from TCBFLGS5:
   Secondary non-dispatchability indicator
   Task non-dispatchability flags from TCBNDS2:
   SVC Dump is executing for another task
   ```

- In VERBEXIT LOGDATA output or the logrec reports and in messages in all locations, look for previous abends and symptom records for earlier problems that did not cause abends. The previous abend or the earlier problem might have led to the abend being diagnosed.

- Look for the name of the program that called the abending module.
  - The address of the calling program can be in the second half of the PSW stored in the caller’s RB, which will precede the running RB, except for branch entries.
  - Determine the linkage conventions of individual save areas. The calling program’s address might be in a save area.
  - In register 14 of the top RB.
Abend analysis

- If a command or macro was being processed, obtain the name of the module issuing the command or macro. The name is in the NAME field of the CDE for a request block (RB) for the abending module’s TCB.

Check for problems in the calling program. The calling program might have caused the abend being diagnosed.

Investigate the following:
- Many abends relating to the same area of the system.
- Many TCBs with the same abend code.

You should now be able to identify the program causing the abend. If not, go to the next step.

Steps for obtaining a dump for the error

Use the following steps to obtain a dump and collect additional data using IPCS:

1. If a dump was not written for the abend, recreate the problem and obtain a dump by doing one of the following:
   - Set a **SLIP command** to obtain an SVC dump. For example:
     ```
     SLIP SET,C=0C9,JOBNAME=RMF,A=SVCD,END
     ```
     This **SLIP** trap will request a dump when an ABEND0C9 occurs in the RMF address space. For more information, see the topic on the **SLIP** command in **z/OS MVS System Commands**.
   - Insert a **SYSABEND**, **SYSUDUMP** or **SYSMDUMP** DD statement in the JCL for the abending job step to obtain an **ABEND** dump. For more information, see **z/OS MVS JCL Reference**.

2. Use IPCS to look at the dump. Use IPCS subcommands in the order indicated by the following list. If using IPCS interactively for an SVC dump, respond yes to the IPCS message that asks if summary data can be used by dump access.
   - **STATUS FAILDATA**
   - **STATUS SYSTEM**

   **Example**: In the following **STATUS** **SYSTEM** output, **AMDSADMP** indicates that this dump was scheduled. Also note the date and time the dump was taken.

   ```
   SYSTEM STATUS:
   Nucleus member name: IEANUC01
   I/O configuration data:
   IODF data set name: IODF.IODF12
   IODF configuration ID: TC4Syst
   EDT ID: 00
   Sysplex name: ENGTEST2
   TIME OF DAY CLOCK: BE5E67AF 7ED6370E 02/15/2006 14:33:32.124515 GMT
   TIME OF DAY CLOCK: BE5E24A1 5B96370E 02/15/2006 09:33:32.124515 local
   Program Producing Dump: SADUMP
   Program Requesting Dump: AMDSADMP
   c. **STATUS CPU REGISTERS DATA CONTENTION**
   d. **STATUS WORKSHEET**
   e. **SUMMARY FORMAT**
   f. **VERBEXIT LOGDATA**
   g. **VERBEXIT SUMDUMP**
   h. **SYSTRACE**
   i. **VERBEXIT MTRACE**

   **Example**: In the following **VERBEXIT** **MTRACE** output, message **IEF450I** indicates a system abend of X’522’ with a reason code of X’00’.

   ```
   0001 007A5F54 N 000000 AN03 93039 10:26:08.31 00000281
   IEF450I SLIP TRAP ID=X13E MATCHED
   0001 007A5F54 N 00000000 AN03 93039 10:26:08.34 00000281
   ```
### Abend analysis

IEA989I SLIP TRAP ID=X13E MATCHED
0001 007A5F54 N 0000000 ANO3 93039 10:26:08.43 00000281
IEA989I SLIP TRAP ID=X13E MATCHED
0001 007A5F54 N 0000000 ANO3 93039 10:26:08.49 00000281
IEA989I SLIP TRAP ID=X13E MATCHED
0001 007A5F54 N 4000000 ANO3 93039 10:26:09.21 TSU05807 00000091
IEF450I LASSEC2 AAIRACF AAIRACF - ABEND=S522 U0000
REASON=00000000
0001 007A7430 N 4000000 ANO3 93039 10:26:09.45 TSU06038 00000091
IEF450I LAMMLF AAIUSER AAIUSER - ABEND=S522 U0000
REASON=00000000
0001 007A7430 M 4000000 ANO3 93039 10:26:09.59 TSU05807 00000090
IEF377I LASSEC2 AAIRACF AAIRACF
0001 007A5F54 E 064 00000090
LASSEC2.SPFLOG1.LIST NOT CATLGD 2
0001 007E02C N 4000000 ANO3 93039 10:26:09.66 TSU05807 00000090
/HASP395 LASSEC2 ENDED
0001 007A79C0 N 0200000 ANO3 93039 10:26:10.06 TSU05807 00000081
/HASP250 LASSEC2 IS PURGED

j. Subcommands selected from the list below

k. VERBEXIT SYMPTOM

3. Before the VERBEXIT SYMPTOM subcommand, add other IPCS subcommands, depending on the problem indicated in the abend explanation or accompanying messages. Pick the subcommands from the following list:

#### Table 8. Summary of IPCS dump subcommands by problem

<table>
<thead>
<tr>
<th>Problem involves</th>
<th>IPCS dump command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation/unallocation of jobs</td>
<td>VERBEXIT ALCMWAIT</td>
</tr>
<tr>
<td>Asynchronous operations manager (AOM)</td>
<td>VERBEXIT AOMDATA 'TRCDUMP'</td>
</tr>
<tr>
<td>Auxiliary storage</td>
<td>ASMCHECK</td>
</tr>
<tr>
<td></td>
<td>VERBEXIT ASMDATA</td>
</tr>
<tr>
<td>Availability management</td>
<td>VERBEXIT AVMDATA</td>
</tr>
<tr>
<td>Callable service requests</td>
<td>CBFORMAT addr STRUCTURE(CSRCPOOL)</td>
</tr>
<tr>
<td></td>
<td>CBSTAT addr STRUCTURE(CSRCPOOL)</td>
</tr>
<tr>
<td>Communications</td>
<td>COMCHECK</td>
</tr>
<tr>
<td>Cross-system coupling facility (XCF)</td>
<td>COUPLE SUMMARY ALL</td>
</tr>
<tr>
<td></td>
<td>COUPLE DETAIL ALL</td>
</tr>
<tr>
<td></td>
<td>COUPLE EXCEPTION ALL</td>
</tr>
<tr>
<td>Data-in-virtual</td>
<td>DIVDATA SUMMARY ALL</td>
</tr>
<tr>
<td>Data lookaside facility of VLF</td>
<td>DLFDATA</td>
</tr>
<tr>
<td>Global resource serialization</td>
<td>VERBEXIT GRSTRACE</td>
</tr>
<tr>
<td>Input/output</td>
<td>IOSCHECK</td>
</tr>
<tr>
<td>JES2</td>
<td>VERBEXIT JES2</td>
</tr>
<tr>
<td>JES3</td>
<td>VERBEXIT JES3</td>
</tr>
<tr>
<td>Language Environment</td>
<td>VERBEXIT LEDATA</td>
</tr>
<tr>
<td>MVS message service (MMS)</td>
<td>VERBEXIT MMSDATA</td>
</tr>
<tr>
<td>z/OS UNIX System Services (OMVS)</td>
<td>OMVSDATA SUMMARY</td>
</tr>
<tr>
<td></td>
<td>OMVSDATA DETAIL</td>
</tr>
<tr>
<td></td>
<td>OMVSDATA EXCEPTION</td>
</tr>
</tbody>
</table>
Table 8. Summary of IPCS dump subcommands by problem (continued)

<table>
<thead>
<tr>
<th>Problem involves</th>
<th>IPCS dump command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real storage manager (RSM)</td>
<td>RSMDATA SUMMARY</td>
</tr>
<tr>
<td>System resources manager (SRM)</td>
<td>VERBEXIT SRMDATA</td>
</tr>
<tr>
<td>Storage Management Subsystem (SMS)</td>
<td>VERBEXIT SMSDATA</td>
</tr>
<tr>
<td>Time sharing option (TSO)</td>
<td>VERBEXIT TSODATA</td>
</tr>
<tr>
<td>Virtual storage manager (VSM)</td>
<td>VERBEXIT VSMDATA</td>
</tr>
<tr>
<td>Virtual lookaside facility (VLF)</td>
<td>VLFDATA</td>
</tr>
</tbody>
</table>

4. Use the RSMDATA SUMMARY output to get a summary of real storage usage in the system. Use the RSMDATA EXCEPTION report to determine where errors might have occurred. The following is an example of RSMDATA SUMMARY output:

**RSMDATA SUMMARY REPORT**

```
<table>
<thead>
<tr>
<th></th>
<th>Tot real</th>
<th>Prf real</th>
<th>Below Prf</th>
</tr>
</thead>
<tbody>
<tr>
<td>In configuration</td>
<td>131,072</td>
<td>98,234</td>
<td>4,096</td>
</tr>
<tr>
<td>Available for allocation</td>
<td>126,084</td>
<td>93,247</td>
<td>4,093</td>
</tr>
<tr>
<td>Allocated</td>
<td>54,127</td>
<td>53,253</td>
<td>184</td>
</tr>
<tr>
<td>Percent usage</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Common fixed frames</td>
<td>3,291</td>
<td>3,145</td>
<td>19</td>
</tr>
<tr>
<td>Percent of available</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total fixed frames</td>
<td>8,283</td>
<td>-</td>
<td>28</td>
</tr>
<tr>
<td>Percent of available</td>
<td>6</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
```

V=R Region:
- First frame number X'000006'
- Last frame number X'0004B'
- Size (in frames) 70

Total disabled reference (DREF) pages in real: 2,309

Number of shared data pages:
- Valid and fixed in real: 3
- Valid and pageable in real: 1,356
- On auxiliary storage: 0

Number of 64-bit common memory pages:
- Backed in real: 513
- Fixed in real: 144
- DREF in real: 256
- On auxiliary storage: 0

**Figure 8. RSMDATA SUMMARY report**

5. Examine the VRADATA output in the STATUS FAILDATA, VERBX LOGDATA or EREP report for an error for additional clues about the error. For some components, the data consists of a key, a length, and the contents.

**Example:** In the following Variable Recording Area from STATUS FAILDATA output, the VRA key is X'1A' and the length is X'94'.

**VARIABLE RECORDING AREA (SDWAVRA)**

```
+000   Key: 1A   Length: 94
```
Abend analysis

STATUS FAILDATA will not format an SDWA for a dump requested by SLIP. If SDWA data is not in the dump, obtain problem data from STATUS CPU REGISTERS or view the SDUMP 4K SQA buffer. (See Reading the SDUMPX 4K SQA buffer in z/OS MVS Diagnosis: Tools and Service Aids.)

You should now have extracted enough problem data to do a search for a known problem, identify the source of the problem, or report the new problem to IBM or the appropriate vendor.

Related information

For information about the SYSMDUMP DD, see
- z/OS MVS Diagnosis: Tools and Service Aids
- z/OS MVS IPCS Commands
- z/OS UNIX System Services Planning
- z/OS MVS JCL Reference
- z/OS MVS System Commands
Chapter 9. Diagnosing a system hang or wait state

Overview of a hang or wait

A system hang or wait can occur gradually as a resource contention problem or abruptly when a disabled wait state is loaded for a critical software-detected error. Externally, the following list of symptoms might be noticed:

- A disabled coded wait state is loaded
- A hang during IPL or system initialization
- The consoles can be locked
- There can be contention for system resources
- The system code can be looping.

When there is a system failure or outage, a stand-alone dump must be taken for problem diagnosis. OPERLOG, SYSLOG, and EREP reports from the time frame of the system outage are also important.

This section will only discuss system hangs and waits. When a job or subsystem is hung, see Chapter 10, “Diagnosing a job or subsystem hang,” on page 121.

**Symptoms of a wait or hang:** The system enters a wait or the entire system hangs. The terms hang and wait are used synonymously in these procedures. Some symptoms of a hang:

- No response occurs on the user’s or system operator’s console.
- No communication with the system through the console occur.
- No response from subsystems (TSO/E, CICS, IMS, DB2, and others) occur.
- The system does not issue or receive messages on the console.
- A series of messages that indicate waits followed by bursts of activity.
- A message indicating a wait appears on the system console.
- The program status word (PSW) contains X’070E0000 00000000’.
- The job entry subsystem does not respond to any commands. For example, in a JES2 system, enter a SDI1 command and JES2 does not respond.

There are two types of wait states: enabled and disabled.

**Enabled wait**

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

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

The most common causes of an enabled wait are that the system is waiting for:

- Work – the system has no active jobs to process or all active jobs are swapped out.
- Action – an operator reply or other action.
Hang and wait analysis

- Missing interrupts – the system is waiting for a critical device, which is busy, not ready, reserved by another system, or has a mount pending. If the system residence (SYSRES) or paging (PAGE) volumes have missing interrupts, the operator may not get a message.
- System resource – work is waiting for a resource, which can be a lock, queue, input/output (I/O) device, page, or device allocation.

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 wait state
You can restart the system.

- An attempt by the operating system to communicate with the operator. When the system cannot send a message to a console, the system can use a restartable wait state to contact the operator and obtain a response.
- A way to preempt processing. For a SLIP trap with an action of wait, the system will issue a message, then enter a restartable wait.
- A symptom of another problem.

non-restartable
You cannot restart the system. After capturing a stand-alone dump, you must reIPL the system.

Steps for diagnosing a system hang

Before you begin: You need to know how to use IPCS and you need access to the following types of information:
- Stand-alone dump
- EREP report of SYS1.LOGREC
- OPERLOG or SYSLOG
- The level of z/OS operating system. Use the IPCS CBFORMAT CVT command to find the level of the z/OS.

The following is an example of the CVT output:

```
CVT: 00FD4938

-0028 PROD1..... 5P7,0,6  PROD1..... HBB7709  VERID..... MGL..... 2064  RELNO..... 038
+0000 TCBP..... 00000218  OEF00..... 00F24EC  LINK..... 00FD48B4  AUSCB..... 00FD4F20  BUF..... 00000000  XAPG..... 00FDE310
+0018 OVLO0..... 00FF63DE  PCNVT..... 00FEDC04  PLT00..... 00FEB004  LL000..... 018E50F0  LLTTR...... 8146288  XTLER..... 00FED310
+0030 SYSAO..... 00EAD9BB  BTERM..... 00FEF8D0  DATE..... 0106114F  MSLT..... 00FD4F8  ZDAB..... 00DA0000  XPAR..... 00FF9740
+0048 OEF01..... 00FF250C  YSS..... 0000  VPSM..... 0000  EXIT..... 0A03  BRET..... 07F7  SVDCB.....
```

- The state of the system. Use IPCS STATUS CPU and note the PSW and mode of each CPU. For example:

  CPU(X'00') STATUS:
  PSW=07F60000 00000000 00000000 00000000
  No work wait
  CPU is in no work wait

Normally, a wait state code appears in the program status word (PSW) when the operating system enters a wait state. Use this code and the associated reason code to diagnose and fix the problem. Explanations for wait state codes are found in z/OS MVS System Codes.

The following steps will guide you through diagnosing a hang or wait:
Hang and wait analysis

1. “Collecting the problem description” on page 109
2. “Diagnosing a hang or wait during IPL” on page 110
3. “Diagnosing an enabled wait state” on page 112
4. “Diagnosing a coded disabled wait state” on page 114
5. “Diagnosing a system partitioned from a sysplex because of status update missing” on page 117
6. “Searching the problem reporting databases” on page 117
7. “Gathering additional data for hangs and waits” on page 119

Use the following flowchart to guide diagnosis of a system hang:
Hang and wait analysis

Figure 9. Flowchart for system hang analysis
Collecting the problem description

The problem descriptions found in ["Gathering diagnosis data" on page 5] and [Table 1 on page 7] indicate you have a hang, a disabled wait state, or an enabled wait state that needs diagnosis.

Steps for collecting the problem description

Perform the following steps to collect the problem description:

1. Ensure that the symptom descriptions above and in [Table 1 on page 7] identify the problem is a hang.
   
   If you see the system activity on the console is high and no jobs are being processed, the problem is a loop. Use the procedure in [Chapter 11, “Diagnosing a loop,” on page 129].

2. Describe what was happening on the system prior to the hang or wait and record this information in [Chapter 16, “Problem diagnostic worksheet,” on page 173]. This includes:
   
   - What is the status for the system on the Hardware Management Console (HMC)?
   - What jobs were started just prior to the hang?
   - What commands were entered and responses received?
   - What recovery procedures did you attempt?
   - What error messages were received?
   - Were there environmental changes? For example, was a new device installed or software maintenance applied.
   - Was the impact to a subsystem like DB2 or the entire system workload?

3. Did the hang during IPL or system initialization? If yes, go to [“Diagnosing a hang or wait during IPL” on page 110].

4. Determine the state of the system by entering the state IPCS STATUS CPU command. Note the PSW for each CP.

   a. If every CP is showing a no work wait, go to [“Diagnosing an enabled wait state” on page 112]. For example:
   
   CPU(X'00') STATUS:
   PSW=07060000 00000000 00000000 00000000
   No work wait
   CPU is in no work wait

   b. If any CP is showing a disabled coded wait state, go to [“Diagnosing a coded disabled wait state” on page 114]. For example:
   
   CPU(X'01') STATUS:
   PSW=000A0000 800200A2
   Disabled wait state code 00A2 SUPPLMNT INFO 80020
   Wait occurred because system monitor control information cannot be read or written.
   ASCB6 at 5FDE88 and TCB6G at 5FF500 for the home ASID
   HOME ASID: 0006 PRIMARY ASID: 0006 SECONDARY ASID: 0006

   c. If the CP is not showing a no work wait or a coded disabled wait, start diagnosis by checking for resource contention. Go to [“Diagnosing an enabled wait state” on page 112].
Hang and wait analysis

Diagnosing a hang or wait during IPL

If a hang or wait occurs during IPL or early on during system initialization, obtain a stand-alone dump, SYSLOG, and note the last message issued to the screen or log. The objectives for analyzing the output of a stand-alone dump are:

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

Steps for diagnosing a hang or wait during IPL

1. Enter the IPCS IPLDATA STATUS command to determine how far along in the IPL or nucleus initialization program (NIP) processing the system is. There will be an entry for each initialization routine. The last entry indicates the last initialization routine to execute. Use this module name in a search for a known problem. The following example indicates IEAIP99 ... Page frame table and cleanup as the last entry.

```plaintext
*** IPL Statistics ***
IEAIP10 00:00:00.000 ISNIRIM - Read SCPINFO
IEAIP20 00:00:01.688 Test Block storage to 2G
IEAIP31 00:00:00.018 Fast FIND service
IEAIP30 00:00:00.002 LOAD service
IEAIP46 00:00:00.164 Read SCHIBs into IPL workspace
IEAIP49 00:00:00.000 Process Load and Default parameters
IEAIP50 00:00:00.774 IPL parmlib - process LOADxx and NUCLSTxx
IEAIP51 00:00:00.019 System architecture
IEAIP43 00:00:00.032 Find and Open IDDF data set
IEAIP60 00:00:00.008 Read NCRs from IODF
IEAIP70 00:00:00.208 UIM environment - load CBD and IOS services
IEAIP71 00:00:00.120 Build DFT for each device
IEAIP8 00:00:00.007 Read EDT information from IODF
IEAIP40 00:00:00.093 Read MLTs from nucleus
IEAIP42 00:00:00.018 Read NMLs from nucleus (IEANynnn modules)
IEAIP41 00:00:01.388 Read PDS directory entries and EESD records
IEAIP05 00:00:01.056 Build and sort NUCMAP
IEAIP02 00:00:03.779 Load nucleus modules
IEAIP04 00:00:00.020 Allocate PFT and SQA/ESQA
IEAIP14 00:00:00.000 Build LSQA/ELSQA for Master
IEAIP06 00:00:00.000 IARM - RSM blocks, master SGT
IEAIP09 00:00:00.054 IAXMI - FFT, master RAB, etc.
IEAIP07 00:00:00.037 Update AMODE for nucleus resident SVCs
IEAIP03 00:00:00.027 Build UCBs, ULUT, etc.
IEAIP18 00:00:00.172 Copy and relocate EDT to ESQA
IEAIP99 00:00:00.465 Page frame table and cleanup

Total IPL Time: 00:00:10.162
```

Figure 10. IPL statistics example

The following NIP example indicates IEAVNPFF ... Loadwait/Restart as the last entry.

```plaintext
*** NIP Statistics ***
IEAVNI0 00:00:00.024 NIP Base
IEAVNI0M 00:00:00.077 Invoke NIP RIMs
IEAVNI6 00:00:03.358 Service Processor Interface
IEAVNPFF 00:00:00.023 Loadwait/Restart
```
Hang and wait analysis

2. If the report is complete, with the last entries indicating master scheduler initialization is complete with total times as in the following example:

FINSHMSI 00:00:00.001 Wait for attached CMDs

IEEMB860 00:05:17.024 Uncaptured time: 00:00:00.810

Total Time: 00:07:13.473

Enter the IPCS SELECT ALL command to verify which system address spaces are up and have completed initialization. For example:

<table>
<thead>
<tr>
<th>ASID</th>
<th>JOBNAME</th>
<th>ASCBADDR</th>
<th>SELECTION CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td><em>MASTER</em></td>
<td>00FD3400</td>
<td>ALL</td>
</tr>
<tr>
<td>0002</td>
<td>PCAUTH</td>
<td>00FBDE80</td>
<td>ALL</td>
</tr>
<tr>
<td>0003</td>
<td>RASP</td>
<td>00FBDD00</td>
<td>ALL</td>
</tr>
<tr>
<td>0004</td>
<td>TRACE</td>
<td>00FBDB80</td>
<td>ALL</td>
</tr>
<tr>
<td>0005</td>
<td>DUMPSRV</td>
<td>00FBDA00</td>
<td>ALL</td>
</tr>
<tr>
<td>0006</td>
<td>XCFAS</td>
<td>00FB1E80</td>
<td>ALL</td>
</tr>
<tr>
<td>0007</td>
<td>GRS</td>
<td>00FB1D00</td>
<td>ALL</td>
</tr>
<tr>
<td>0008</td>
<td>SMSPDSE</td>
<td>00F80400</td>
<td>ALL</td>
</tr>
<tr>
<td>0009</td>
<td>CONSOLE</td>
<td>00F80280</td>
<td>ALL</td>
</tr>
<tr>
<td>000A</td>
<td>WLM</td>
<td>00F4F300</td>
<td>ALL</td>
</tr>
<tr>
<td>000B</td>
<td>ANTMAIN</td>
<td>00F4F180</td>
<td>ALL</td>
</tr>
<tr>
<td>000C</td>
<td>ANTAS000</td>
<td>00F4F000</td>
<td>ALL</td>
</tr>
<tr>
<td>000D</td>
<td>OMVS</td>
<td>00F4DE80</td>
<td>ALL</td>
</tr>
<tr>
<td>000F</td>
<td>IEFSCHAS</td>
<td>00FC6E80</td>
<td>ALL</td>
</tr>
<tr>
<td>0010</td>
<td>JESXCF</td>
<td>00F8B500</td>
<td>ALL</td>
</tr>
<tr>
<td>0011</td>
<td>ALLOCAS</td>
<td>00F8B380</td>
<td>ALL</td>
</tr>
<tr>
<td>0012</td>
<td>IOSAS</td>
<td>00F97280</td>
<td>ALL</td>
</tr>
<tr>
<td>0013</td>
<td>IXGLOGR</td>
<td>00F97100</td>
<td>ALL</td>
</tr>
<tr>
<td>0014</td>
<td>JESZ</td>
<td>00FC0D80</td>
<td>ALL</td>
</tr>
</tbody>
</table>

3. Do a search using symptoms including the last initialization routine to run, the last message issued to the log or screen, and the last address space to initialize.

Table 9. Common wait states that occur during IPL

<table>
<thead>
<tr>
<th>Wait state code</th>
<th>Reason code</th>
<th>Explanation</th>
<th>Find information in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'064'</td>
<td>X'005'</td>
<td>Indicates an ABEND was issued during NIP. To diagnose using a SADUMP, enter the SYSTRACE ALL command, go to the bottom of the output and enter the FIND *SVC PREV command to locate the the ABEND issued.</td>
<td>Chapter 8, “Diagnosing an abend,” on page 85</td>
</tr>
</tbody>
</table>

**EXAMPLE: WAITX'064' RSNX'005'**

| 00 0001 00000000  | +SVCE       | 040C1000 814EE1E2 00000010 84000000 84878000 10000201 00000000 0001 0001 BCSA7BA2DAC255B 08000004 00400000 |

| X'064'          | X'009'      | Indicates a program check occurred during NIP. To diagnose using a SADUMP, format the LCCA by entering the CBFORMAT LCCAx command (where x is CP ID). | Chapter 8, “Diagnosing an abend,” on page 85 |
Hang and wait analysis

Table 9. Common wait states that occur during IPL (continued)

<table>
<thead>
<tr>
<th>Wait state code</th>
<th>Reason code</th>
<th>Explanation</th>
<th>Find information in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'040'</td>
<td>Not applicable</td>
<td>Indicates an ABEND was issued during NIP. Gather additional information by entering the IPCS STATUS CPU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>From this example, a search argument including symptoms WAIT040 ABEND878 RC10 IEAVNPB2 would be built to check for a known problem.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:

```plaintext
CPU (X'00') STATUS:
PSW=00020000 00000000 00000000 00000000 00878040
(Running in PRIMARY, key 0, AMODE 31, DAT OFF)
Disabled for PER I/O MCH
NIP RIM IEAVNPB2 has failed
ABEND=878 REASON=00000010

Register values
0-3 84000000 84878000 03A5EF90 0000E676
4-7 00FCC9B 00FCD9B 0393F42B 00FCC00
8-11 0187C51B 00001000 00000000 00000030
12-15 00000001 00000000 FE000424 00000010
ASCB1 at FCC00B, JOB(*MASTER*), for the home ASID
ASXBI at FCC370 and a local SRB for the home ASID
HOME ASID: 0001 PRIMARY ASID: 0001 SECONDARY ASID: 0001
```

Diagnosing an enabled wait state

When the IPCS STATUS CPU command does not show a no work wait or a coded disabled wait, start diagnosis by checking for resource contention using the following steps.

Steps for diagnosing an enabled wait state

1. **Verify that IPCS STATUS CPU report shows every CPU in a no work wait:**
   ```plaintext
   CPU(X'00') STATUS:
   PSW=07060000 00000000 00000000 00000000
   No work wait
   CPU is in no work wait
   ```

2. **Enter the IPCS ANALYZE EXCEPTION command to look for resource contention.**
   ```plaintext
   CONTENTION EXCEPTION REPORT
   JOBNAME=PMIMTAPE ASID=0065 TCB=007DD0F8
   ```
   ```plaintext
   JOBNAME=PMIMTAPE HOLDS THE FOLLOWING RESOURCE(S):
   RESOURCE #0003: There are 0025 units of work waiting for this resource
   ```
Hang and wait analysis

NAME=MAJOR=SYSIEFSD MINOR=Q4 SCOPE=SYSTEM

STATUS FOR THIS UNIT OF WORK:
This address space is on the SRM IN queue.
JOBNAME=PCICBDTS ASID=0266 TCB=008723A8

JOBNAME=PCICBDTS HOLDS THE FOLLOWING RESOURCE(S):

RESOURCE #0002: There are 0022 units of work waiting for this resource
NAME=MAJOR=SYSIEFSD MINOR=Q10 SCOPE=SYSTEM
STATUS FOR THIS UNIT OF WORK:
This address space is on the SRM IN queue.

********************************************************* END OF DATA *****

If resource contention exists, use the IPCS FIND command on the TCB or SSRB address that is identified as the holder of a resource in the analyze exception report to see if the TCB is waiting for any other resources. If found waiting, get the TCB or SSRB address of the holder of that resource and repeat the process until the bottom of the contention chain is reached.

3. Enter an IPCS SUMMARY FORMAT JOBNAME(xyz) for the holder of the resource in contention.

4. Use the IPCS FIND command to locate the TCB or SSRB that is identified as the holder of that resource. If found waiting, get the TCB/SSRB address of the holder of that resource and repeat the process, until the bottom of the contention chain is reached.

   a. If the holder of the resource is a TCB, go to “Examining the TCB status” on page 126.

   b. If the holder is an SSRB, either find the SSRB in the SUMMARY FORMAT srb address STR(SRB) command. Using the PSW address from the CPSW field, use the IPCS WHERE command or browse storage to find the module name that determines where the SRB was last running.

   For example:

5. Use the module name to search for a known problem.
Hang and wait analysis

6. If the search reveals no contention, use the IPCS SYSTRACE ALL command to examine the system trace table for the ASIDs that are executing.

7. Use the FIND WAIT command in the SYSTRACE report to check for any WAIT type system trace entries.

8. If no WAIT entries exist, there might be contention on CPU resources. Scroll through SYSTRACE noting the ASIDs associated with the entries. If all the entries are for a couple of ASIDs and they are mostly CLKC or EXT type entries, it might indicate a loop. Go to Chapter 11, “Diagnosing a loop,” on page 129.

9. If WAIT entries are found, then there is no contention on any CPU resource. Talk to the operator to get more specific information on what appears to be hung from an operations perspective. If a specific job or class of jobs are hung (like Batch or TSO users), then get the job name or the specific TSO userid and go to Chapter 10, “Diagnosing a job or subsystem hang,” on page 121.

Diagnosing a coded disabled wait state

When the IPCS STATUS CPU command shows a disabled coded wait state, use the following steps to start diagnosis.

Steps for diagnosing a coded disabled wait state

1. Obtain the disabled wait state code from the stand-alone dump, if obtained or the Hardware Management Console (HMC). The IPCS STATUS WORKSHEET report from a stand-alone dump often contains a corresponding wait state message. For example:
Hang and wait analysis

MVS Diagnostic Worksheet

Dump Title: WAIT 0A2 REASON 15A ZOS 1.7 INSTALAC SYSPLEX

CPU Model 2066 Version 00 Serial no. 0123B4 Address 00
Date: 05/22/2006 Time: 20:48:49.817141 Local

Wait State Message Issued at 20:43:48 on Day 142 of 2006:
IXC436W THIS SYSTEM HAS LOST TIME SYNCHRONIZATION WITH THE OTHER
SYSTEMS IN THE SYSPLEX AND
HAS BEEN PLACED INTO A NON-RESTARTABLE
WAIT STATE CODE: 0A2 REASON CODE: 15A

SYSTEM RELATED DATA
CVT SNAME (154) IEASYSFI VERID (-18)
CUCB (64) 00FD5140 PVT (164) 00FFB108 GDA (230) 0210A278
RTMCT (23C) 00F4FB20 ASMVT (2C0) 00FD75D8 RCEP (490) 01907F18
CSD Available CPU mask: 8000 Alive CPU mask: 80000000 00000000
Number of active CPUs: 00000001

PROCESSOR RELATED DATA

<table>
<thead>
<tr>
<th>NAME</th>
<th>OFFSET</th>
<th>CPU 00</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSW at time of dump</td>
<td>00020000</td>
<td>80000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0015A0A2</td>
</tr>
</tbody>
</table>

Figure 11. IPCS STATUS WORKSHEET report from a stand-alone dump

2. Find the wait state code using Z/OS MVS System Codes. If there is no recommended action:
   a. Look up the wait state code in the wait state code to module table in Z/OS MVS System Codes
   b. Use the module name to identify the component using the module identification table in Z/OS MVS Diagnosis: Reference

3. Perform a search using the wait state code, reason code, module name and component identifier to look for a known problem. If you cannot find a match, report the problem to IBM.
### Table 10. Common disabled wait states

<table>
<thead>
<tr>
<th>Wait state code</th>
<th>Reason code</th>
<th>Explanation</th>
<th>Go to information in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'01B'</td>
<td></td>
<td>This is a restartable wait state that is loaded when an active SLIP trap requests an action of WAIT. The conditions specified on the SLIP command are met. The system enters a wait state, as requested. Information about the environment when the slip trap matches is presented in message IEE844W. For example:</td>
<td>The topic on SLIP problem data in the SLIP work area in <a href="http://www.ibm.com/systems/z%E3%82%AA%E3%82%B9/mvs/diagnosis/tools_and_service_aids">z/OS MVS Diagnosis: Tools and Service Aids</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message IEE844W: SENDING SLIP TRAP 0001 MATCHED. ACTION=WAIT TYPE=PER. PER INFO:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X'040'</td>
<td>The system ended a task during nucleus initialization program (NIP) processing.</td>
</tr>
<tr>
<td></td>
<td>X'005'</td>
<td>Indicates that an ABEND was issued during NIP. To diagnose using a SADUMP, do a SYSTRACE ALL, max PF8 to the bottom of the output and do a ‘F *SVC PREV’ to locate the the ABEND issued.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXAMPLE: WAITX'064' RSNX'005'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00 0001 00000000 +SYSCE D 040C1000 814E1EE2 00800010 84000000 84878000 10000201 00000000 0001 BECSA78A2DAC2558 00800004 00400000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X'009'</td>
<td>A program check occurred during NIP. Accompanying message IEA304W further explains this wait state and entry code. If the message does not appear on the console, you can find the message in the wait state message area (WSMA). The WSMA is described in <a href="http://www.ibm.com/systems/z%E3%82%AA%E3%82%B9/mvs/diagnosis/tools_and_service_aids">z/OS MVS Data Areas</a> in the z/OS library at <a href="http://www.ibm.com/systems/z/os/zos/bkserv/">http://www.ibm.com/systems/z/os/zos/bkserv/</a>. To diagnose using a stand-alone dump, format the LCCA by entering the CBFormat LCCA: command (where x is the CPU ID).</td>
<td>Chapter 8, “Diagnosing an abend,” on page 85</td>
</tr>
<tr>
<td></td>
<td>X'004'</td>
<td>The operator entered the VARY XCF,sysname,OFFLINE command to remove the system from the sysplex.</td>
<td>Ask the operator or system programmer why the system was varied out of the sysplex before continuing with diagnosis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X'104'</td>
<td>I/O has been prevented as a result of a system being fenced.</td>
</tr>
<tr>
<td></td>
<td>X'10C'</td>
<td>Cross-system coupling facility (XCF) or cross-system extended services encountered an unrecoverable error and stopped the system. The system also issues this wait state in response to an operator request to stop the system. See <a href="http://www.ibm.com/systems/z%E3%82%AA%E3%82%B9/mvs/diagnosis/reference">z/OS MVS Diagnosis: Reference</a> for information on diagnosing sysplex problems.</td>
<td>“Diagnosing an enabled wait state” on page 112</td>
</tr>
</tbody>
</table>
Diagnosing a system partitioned from a sysplex because of status update missing

A system that is partitioned from the sysplex because of status update missing is really indicating a system hang or wait. A system in a sysplex indicates its health by updating a timestamp value on the SYSPLEX couple dataset every second. If the timestamp is not updated for the failure detection interval as defined in the COUPLExx parmlib member in use (85 seconds is the recommended value), the system will be partitioned from the sysplex. Use the following steps to guide your diagnosis:

**Steps for diagnosing a system partitioned because of status update missing**

When partitioned, the system is be put into a X'0A2' wait state with one of the following reason codes:
- X'104' if a System Failure Management (SFM) Policy is active and IO is fenced.
- X'10C' if the status update missing.

1. Verify the system was partitioned from the sysplex as indicated by message IXC101I, which is in the OPERLOG:
   ```
   IXC101I SYSPLEX PARTITIONING IN PROGRESS FOR SYS22 REQUESTED BY XCFAS. REASON: SFM STARTED DUE TO STATUS UPDATE MISSING
   ```
2. Verify the system is in a X'0A2' wait state by entering the IPCS STATUS CPU command:
   ```
   CPU(X'01') STATUS:
   PSW=000A0000 800200A2
   Disabled wait state code 00A2 SUPPLMNT INFO 8002
   Wait occurred because system monitor control information cannot be read or written.
   ASCB6 at F42700, JOB(XCFAS), for the home ASID
   ASXB6 at 5F3ED0 and TCBG at 5FF500 for the home ASID
   HOME ASID: 0006 PRIMARY ASID: 0006 SECONDARY ASID: 0006
   ```
3. Enter the IPCS COUPLE SYSPLEX EXCEPTION to identify the reason the system is being partitioned.
4. If the reason is that the system entered a coded disabled wait state prior to sysplex partitioning, go to “Diagnosing a coded disabled wait state” on page 114 otherwise, go to “Diagnosing an enabled wait state” on page 112.

**Searching the problem reporting databases**

Search arguments are used to search problem reporting databases. If the problem you are diagnosing was already reported and the symptoms are in the database, the search produces a match. Searching is an iterative process; you might need to use the procedures in “Gathering additional data for hangs and waits” on page 119 to gather additional data and continue your search.

**Steps for searching the problem reporting databases**

Use the following steps to search the problem reporting databases and determine if the problem was previously reported:
1. Develop a free-format search argument using the symptoms obtained from the analysis performed. The free-format search argument can include any of the following symptoms:
   - WAITxxx RSNyyyyyyyy (where xxx is the disabled wait state code and yyyyyyy is the associated reason code)
   - Module or CSECT name
   - Resource name that was found to be in contention
Hang and wait analysis

2. If an argument is not provided, use the primary symptom string in VERBEXIT SYMPTOM output, if available, or use the following symptoms:
   - Program identifier: PIDS/cccccccc
   - CSECT name or module name: RIDS/cccccccc
   - Wait state:
     - If a disabled wait, with a wait state code: WS/D0hhh
     - If an enabled wait: WS/E0000
   - If ANALYZE EXCEPTION output indicates a lockout: PCSS/LOCKOUT
   - Input request (call, command, macro, statement), if one is associated with the problem: PCSS/cccccccccc
   - Symptoms created from information in the STATUS CPU output

3. Select the problem type on the search tool panel, based on STATUS CPU output:

   Table 11. Selecting the problem type for STATUS CPU output

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>STATUS CPU Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled wait</td>
<td>DISABLED WAIT STATE CODE</td>
</tr>
<tr>
<td>Enabled wait</td>
<td>NO WORK WAIT</td>
</tr>
<tr>
<td>Enabled wait</td>
<td>DISABLED FOR ccc (not I/O or EXT)</td>
</tr>
<tr>
<td>Hang</td>
<td>None of the above</td>
</tr>
</tbody>
</table>

4. If the search finds that the problem was previously reported, request the problem fix.
   Searching is an iterative process. If the search finds no match, you can remove some symptoms or change the symptoms and search again. Continue searching for matches by adding, removing, and changing symptoms using the steps in “Gathering additional data for hangs and waits” on page 119.

5. If you still cannot find the cause of the hang or wait or if the problem is new, report the problem to IBM using the procedure in Chapter 15, “Reporting problems to IBM,” on page 169. Record the following problem data in Chapter 16, “Problem diagnostic worksheet,” on page 173:
   - Problem type: disabled wait, enabled wait, or hang
   - Search argument
   - Dump formatted by IPCS online
   - SMF records, if obtained
   - Accompanying messages: identifiers and texts
   - Hard-copy log, beginning 15 to 30 minutes before the problem, or master trace, if not wrapped between the problem and dump
   - Logrec records, beginning 15 to 30 minutes before the problem and edited using the SPOTCHK and TIMESEQ parameters
   - All output data sets related to the problem
   - Data on any related problems
   - Module name and level
   - Name and level of the operating system(s) with a list of program temporary fixes (PTF) applied at the time of the problem and all installation modifications, exits, and products with other than Class A service
Hang and wait analysis

- Other problem data developed while using the procedures in this document or other diagnosis books for the component, subsystem, or program

You know you are done when you find a match for your problem or report the problem.

Related information:
- See "Searching problem reporting databases" on page 8 for more information on developing search arguments.
- See z/OS MVS IPCS Commands for the IPCS subcommands:
  ANALYZE
  STATUS
  VERBEXIT SYMPTOM
  STATUS CPU

Gathering additional data for hangs and waits

Gathering additional data will increase your chances of finding a match in the problem reporting databases. Use the procedures outlined in this section to gather additional data and continue searching the problem reporting databases.

Steps for gathering messages and logrec for hangs

Use the following steps to collect and analyze messages and logrec records about the problem.

1. Use time stamps to select messages and software, symptom, and hardware logrec records related to the problem. Look in the following:
   - OPERLOG or SYSLOG
   - VERBEXIT MTRACE dump output, which shows the buffer for system messages
   - VERBEXIT LOGDATA dump output, which formats the logrec buffer
   - Logrec data set, formatted by EREP

2. Use the COPYCAPD command to check for any SVC dumps captured in dataspaces that did not have a chance to get written out to a dataset prior to the system hang. For example:

<table>
<thead>
<tr>
<th>Number</th>
<th>Time stamp</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>04/10/2006 15:41:18</td>
<td>END OF MEMORY RESOURCE MANAGER HANG DETECTED: TCB = 009BBE88, NAME = ................</td>
</tr>
<tr>
<td>2</td>
<td>04/10/2006 15:42:24</td>
<td>END OF MEMORY RESOURCE MANAGER HANG DETECTED: TCB = 009D64B0, NAME = ................</td>
</tr>
<tr>
<td>3</td>
<td>04/10/2006 15:44:43</td>
<td>IXC431I XCF STALLED GROUP</td>
</tr>
<tr>
<td>4</td>
<td>04/10/2006 15:46:35</td>
<td>JES2/XCF Env on current &amp; other sysplex members via IEADMCJ2</td>
</tr>
<tr>
<td>5</td>
<td>04/10/2006 15:51:35</td>
<td>END OF MEMORY RESOURCE MANAGER HANG DETECTED: TCB = 009A4998, NAME = ................</td>
</tr>
<tr>
<td>6</td>
<td>04/10/2006 15:52:43</td>
<td>END OF MEMORY RESOURCE MANAGER HANG DETECTED: TCB = 009A4758, NAME = ................</td>
</tr>
<tr>
<td>7</td>
<td>04/10/2006 15:53:51</td>
<td>END OF MEMORY RESOURCE MANAGER HANG DETECTED: TCB = 00985CF8, NAME = ................</td>
</tr>
<tr>
<td>8</td>
<td>04/10/2006 15:55:00</td>
<td>END OF MEMORY RESOURCE MANAGER HANG DETECTED: TCB = 009A4758, NAME = ................</td>
</tr>
</tbody>
</table>

   These SVC dumps can be extracted into dump data sets using the COPYCAPD command and then diagnosed individually. Go to Chapter 8, “Diagnosing an abend,” on page 85.

Related information:
- See z/OS MVS IPCS Commands for the IPCS subcommands.
- For formatting of logrec records, see Recording logrec error records in z/OS MVS Diagnosis: Tools and Service Aids.
- See z/OS MVS Diagnosis: Reference for logrec reports.
- For explanations of the messages, see:
  - z/OS MVS System Messages, Vol 1 (ABA-AOM)
Hang and wait analysis

- z/OS MVS System Messages, Vol 2 (ARC-ASA)
- z/OS MVS System Messages, Vol 3 (ASB-BPX)
- z/OS MVS System Messages, Vol 4 (CBD-DMO)
- z/OS MVS System Messages, Vol 5 (EDG-GFS)
- z/OS MVS System Messages, Vol 6 (GOS-IEA)
- z/OS MVS System Messages, Vol 7 (IEE-IEE)
- z/OS MVS System Messages, Vol 8 (IFE-JGD)
- z/OS MVS System Messages, Vol 9 (IGF-IWM)
- z/OS MVS System Messages, Vol 10 (IXC-IZP)
- z/OS MVS Dump Output Messages
- The message book for a subsystem or program

You know you are done when your search produces a match.
Chapter 10. Diagnosing a job or subsystem hang

Overview of a hang or wait

When a job or subsystem hang occurs, you might notice part of the system is not functioning or that a job is in the system for a long time without processing.

Symptom of a job hang:
• A job remains in the system for a long time and does not end.

Symptom of a subsystem hang:
• A subsystem does not respond to any commands. For example, in a JES2 system, enter a $DI1 command and JES2 does not respond.

Steps for diagnosing a job or subsystem hang

The following procedures will guide you through diagnosing a job or subsystem hang:
1. “Gathering additional data for a job or subsystem hang” on page 123
2. “Determining the status of a hung job or subsystem” on page 123
3. “Determining if a job is waiting for resources” on page 124
4. “Determining address space dispatchability” on page 124
5. “Examining the SRB status” on page 125
6. “Examining the TCB status” on page 126
7. “Examining why a job is not running” on page 128

Use the following flowchart to guide diagnosis of a job or subsystem hang:
Job or subsystem hang analysis

Figure 12. Flowchart for job or subsystem hang analysis
Gathering additional data for a job or subsystem hang

It is important to gather information about what the job or subsystem was doing at the time of the hang. This includes answering the following questions:

- What recovery procedures did you attempt? For example, was the MODIFY, CANCEL or FORCE command entered?
- What error messages were received at the time of the hang?
- What commands were entered and responses received?
- Were there environmental changes? For example, a new device installed or a software maintenance upgrade.

Record this information in Chapter 16, “Problem diagnostic worksheet,” on page 173.

Step for gathering additional data

Before you begin: You must know how to use IPCS and understand how to take an SVC dump. For complete information on SVC dump, see using the IEADMCxx parmlib member in the topic on SVC dump in z/OS MVS Diagnosis: Tools and Service Aids.

Request an SVC dump in one of the following ways:

- Use the DUMP command for the hung job and any other related ASIDs.
- If this is a subsystem or system address space, use IEADMCxx, the DUMP command parmlib member. IEADMCxx allows you to specify the collection of dump data without having to remember and identify all the systems, address spaces, and data spaces involved. IEADMCxx from SYS1.SAMPLIB defines the dump options for specific jobs, functions, subsystem, as documented in z/OS MVS Initialization and Tuning Reference.
- Specify the following:
  DUMP COMM=(dumptitle) Rxx,ASID=(1,xx),SDATA=(GRSQ,SQA,CSA,RGN,TRT,COUPLE,XESDATA,NUC),END

  Where xx is the ASID associated with the hung job. You can also use JOBNAME=(xyz)).
  If using a IEADMCxx parmlib member, enter:
  DUMP COMM=(title),PARMLIB=xx

  You must also determine the level of the z/OS operating system:
  - Use the IPCS CBFORMAT CVT command. For an example, see “Steps for diagnosing a system hang” on page 106.

Determining the status of a hung job or subsystem

Sometimes a loop can appear to be a hang. Use the following steps to identify if the job is running:

Steps for determining the status of a hung job or subsystem

1. In the SVC dump, look for entries in the system trace table by issuing IPCS SYSTRACE JOBNAME(name). If there are no entries, continue with “Steps for determining if a job is waiting for resources” on page 124.
2. If there are entries, summarize to determine if:
   - All the entries are in SRB mode or all the entries are for one or two TCB’s.
Job or subsystem hang analysis

- The PSW addresses are all within a certain range and repeating. (If the PSW addresses in CLKC and EXT type trace entries are all within a specific range, go to Chapter 11, “Diagnosing a loop,” on page 129.)

If both are true, use the IPCS WHERE command to identify which module that the PSW address is pointing to.

3. Use the module name and the symptom LOOP to build a search argument to check for a known problem. For details, see “Extracting problem symptoms and search arguments” on page 8.

Determining if a job is waiting for resources

Use the following steps to identify if the job or subsystem is waiting for resources:

**Steps for determining if a job is waiting for resources**

1. Enter the IPCS ANALYZE RESOURCE command.
2. Look in the report for the job name that is hung to determine if it is waiting for system resources. If it is waiting for resource, note the resource name and examine the status of the unit of work holding the resource. **For example:**

   JOBNAME=SDSF  ASID=003D  SSRB=02FC4900  --> unit of work holding resource
   
   RESOURCE #0002: There are 0018 units of work waiting for this resource
   
   NAME=LOCAL LOCK FOR ASID 003D
   DATA=INTERRUPTED AND NOW DISPATCHABLE

3. Do a FIND on the TCB or SSRB address of the holder to check if it is waiting on another resource. Repeat until the bottom of the contention chain is reached.
4. Choose from the following:
   a. **If the unit of work that is holding the resource is not dispatchable**, go to “Examining the SRB status” on page 125 or “Examining the TCB status” on page 126 to determine why.
   b. **If the unit of work that is holding the resource is dispatchable**, go to “Steps for examining why a job is not running” on page 128.
   c. **If the job is not holding or waiting for a resource**, go to “Steps for examining address space dispatchability.”

Determining address space dispatchability

If you are working with a complete SVC dump, as indicated by messages IEE911E or IEA611I, examine address space dispatchability using the following steps:

**Steps for examining address space dispatchability**

1. Enter IPCS SUMMARY FORMAT JOBNAME(xyz) to obtain a report for the hung job.
2. Locate the ASCB (F ASCB). **For example:**
Job or subsystem hang analysis

Examining the SRB status

Steps for examining the SRB status

1. Enter the IPCS CBFORMAT srb address STR(SRB) to format the SRB or SSRB and related control blocks that describe the environment of the system request block (SRB).

For example:

<table>
<thead>
<tr>
<th>Field</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCB:</td>
<td>00F2A2B0</td>
</tr>
<tr>
<td>DSP1 (+X'72')</td>
<td></td>
</tr>
<tr>
<td>Address space non-dispatchability bits:</td>
<td></td>
</tr>
<tr>
<td>- If X'80', an SVC dump is in progress, which does not indicate a problem.</td>
<td></td>
</tr>
<tr>
<td>- If X'40', an address space is failing. Check field MCC (ASCB+X'A8') for the memory termination completion code (MCC=8840D000). If so, this job is terminating with an ABEND40D. Extract the associated reason code from field ARC (ASCB+X'174').</td>
<td></td>
</tr>
<tr>
<td>- If X'10' or X'32', the address space is logically or physically swapped out. To determine the reason for the swap, issue the IPCS VERBX SRMDATA and FIND the job name (F jobname).</td>
<td></td>
</tr>
<tr>
<td>LOCK (+X'80')</td>
<td></td>
</tr>
<tr>
<td>Address space local lock word:</td>
<td></td>
</tr>
<tr>
<td>- If the word contains X'7FFFFFFF', a TCB or SSRB is suspended holding the local lock. To locate the PSW and registers for the lock holder, enter FIND IHSA. Use the IPCS WHERE command on the address from CPSW (current PSW) in the IHSA to identify the lock holder. Use this module name in a search.</td>
<td></td>
</tr>
<tr>
<td>- If the word contains X'0000004X', this identifies that the lock holder is currently executing on CP 4x at the time of the dump. Do an IPCS SYSTRACE CPU(4x), go to the bottom of the output and scroll back examining the trace entries to see what unit of work is executing. For information on interpreting system trace, see MVS Diagnosis: Tools and Service Aids.</td>
<td></td>
</tr>
<tr>
<td>- If the word contains X'4FFFFFFF' or X'FFFFFFFFFF' (the SRB or TCB ready to run id), then the unit of work is on the dispatching queue and waiting to get dispatched. Go to “Steps for examining why a job is not running” on page 128.</td>
<td></td>
</tr>
</tbody>
</table>
Job or subsystem hang analysis

2. Use the PSW address from the CPSW field in the IPCS WHERE command or browse storage to look for a module name to determine where the SRB was last executing.

3. Use the module name and hang to search for a known problem.

Examining the TCB status

Steps for examining the TCB status

1. Enter the IPCS SUMMARY FORMAT JOBNAME(xyz) to generate a report of address space related control blocks representing the job, which includes the ASCB, SRB’s, TCB’s, RB’s, and others.

2. Locate the TCB.
   a. If you have identified a TCB as holding or waiting for a resource, enter the FIND TCB: xxxxxxxx where xxxxxxxx is the virtual address of the TCB.
   b. If you have not identified a TCB, scroll to the bottom of the SUMMARY FORMAT report to locate the last TCB on the chain.

3. Examine the TCB non-dispatchability bits, ignoring the SDUMP non-dispatchability indicator:

   Task non-dispatchability flags from TCBFLGS4:
   a. If RB is in a wait --- important
   b. If SVC Dump is executing for another task ----> ignore, always on in an svcdump

4. If RB is in a WAIT or with no flags on:
   a. Do a FIND ACTIVE to locate the RB control blocks (these can be PRBs, SVRBs, IRBs or SIRBs).
b. Look at the last RB formatted. Verify that the wait or suspend count in the high order byte of the RBLINK is greater than zero. Extract the PSW address from the OPSW field. For example:

```
+0000 RSV..... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
+0018 SQE...... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
+0020 XB...... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
+0038 flagsz... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 0-000003 018FF2A0
+0020 GPR0-3... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
+0030 GPR4-7... 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8
+0040 GPR8-11.. 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0
+0060 RSV...... E2C8D9F0 F4D3D740
```

Left halves of all registers contain zeros

- D 00000008 00000F08 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- 4-7 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8 008FDC8
- 8-11 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0 008FC1B0
- +0060 RSV..... E2C8D9F0 F4D3D740

For example:

```
Command ===> ip w 25400f00

******************************************************************************* TOP OF DATA ******
ASID(X'0020') 25400F00. SHR04LP+01F8 IN EXTENDED PRIVATE
ASID(X'0020') 25400F00. AREA(Subpool251Key08)+0F00 IN EXTENDED PRIVATE
******************************************************************************* END OF DATA ******
```

- If PSW points into ISGGWAIT, the TCB is waiting on an ENQ resource. Check the storage pointed to from GPR1 saved in this SRVB. It will point to storage containing the ENQ parmlist. The ENQ parmlist +4 points to the major name and +8 points to the minor name associated with the resource. For a mapping of the parmlist, see SVC 38 in z/OS MVS [Diagnosis: Reference]

- If the PSW points to IEAVEWAT, a program call (PC) entered the WAIT as requested. To determine who the requester was, locate the current linkage stack entry for this TCB by entering F LSE: PREV. Enter the IPCS WHERE command on the PSW address from field PSWE to determine who performed the WAIT (PC 30D). Use this module name when doing a search for a known problem. For example:

```
  LSE: 7F0B8140
  GENERAL PURPOSE REGISTER VALUES
  00-01.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  02-01.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  04-05.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  06-07.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  08-09.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  10-11.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  12-13.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  14-15.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  ACCESS REGISTER VALUES
  00-03.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  04-07.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  08-11.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  12-15.... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  PKM...... 8040 SASN..... 0006 EAX...... 0000 PASN..... 0006
  PSW...... 07041000 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  TARG..... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  MSTA..... 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  TYPE...... 0D
  PC STATE ENTRY
  RFS...... 0D80 NES...... 0000

Chapter 10. Diagnosing a job or subsystem hang 127
Job or subsystem hang analysis

5. If the abnormal wait non-dispatchability flag is on, check for a subtask that is also abending. Do a FIND on TCB examining the CMP field in the TCB for one that is non-zero. Next, repeat step 3 examining the TCB non-dispatchability bits.

6. If any other TCB non-dispatchability flags are on, determine the name to the TCB non-dispatchability flag set by examining the value of TCBFLGS field, bytes 4 and 5, and field TCBNDSP. Interpret bit settings using the mapping of the TCB in the version of z/OS MVS Data Areas that corresponds to the release of z/OS you are running in your environment. See Use the TCB non-dispatchability bit name in a search for a known problem.

7. Build a symptom string including:
   • The word WAIT
   • The module name
   • The component ID identified
   • Any other symptoms. For example, the resource waited on (SYSZTIOT or CMSEQDQ lock) or the TCB non-dispatchability bit name (TCBSTP).

Examining why a job is not running

When a job is not running, but the TCBs or SRBs are dispatchable, determine what is preventing the job from being dispatched. This often occurs when higher priority work is monopolizing the CPs. Use the following steps to examine the activity in the system trace.

Steps for examining why a job is not running

1. Enter the IPCS SYSTRACE ALL command.
2. Enter FIND WAIT in the SYSTRACE report and identify WAIT type trace entries.
   a. If you find WAIT entries, there are intervals when the dispatcher did not find any dispatchable work to dispatch. Review the previous sections of Chapter 10, “Diagnosing a job or subsystem hang,” on page 121 to ensure nothing overlooked.
   b. If you do not find any WAIT entries, examine which ASIDs are running by scrolling through the IPCS SYSTRACE ALL report.
3. In a IPCS VERBX SRMDATA report, compare the service classes and periods of the jobs executing in SYSTRACE to those that are hung.
   For example:
   ```
   JOB ZFS
   ASID 01F5
   OUCB 02700780 IN QUEUE
   +11 (NSW) NONSWAPPABLE
   +11 (PVL) PRIVILEGED PROGRAM
   (ASCBSRSME) RAX ADDRESS IS 020848C8
   SERVICE CLASS = SYSSTC
   WORKLOAD = SYSTEM
   INTERNAL CLASS= $SRMG000D
   PERIOD = 01
   ```
   If these are of higher priority, examine where they are running by using the IPCS WHERE on several of the PSW addresses from the SYSTRACE report.
4. Search for a known problem using the module, CSECT, or both.
Chapter 11. Diagnosing a loop

Overview of a loop

A loop is a repetitive set of instructions being performed by a job or unit of work. A job or function that is looping can appear to be hung or can use a high amount of CP resource and lock out other work from getting service.

The three types of loops are:

**Disabled loop**

A *disabled loop* is repetitive execution, usually in system level code, with the IO and EXT type interrupts prevented with a PSW mask of X’x4’ in the high order byte of the PSW. A disabled loop is bound to one CP in the system. If in a multi-processor environment and resources are held, a spin loop is detected. If on a uniprocessor, a disabled loop will result in a system outage.

**Enabled loop**

An *enabled loop* occurs under a unit of work (TCB or SRB) that is executing on behalf of a job or function. It is executing with a PSW that is enabled for I/O and external interrupts with a mask of X’x7’ in the the high order byte. A unit of work that is looping enabled, is interrupted periodically for IO, EXT or CLKC type interrupts, which are traced in the system trace table.

**Spin loop**

A *spin loop* is a timed disabled loop in system code controlled by the installation with specifications in the EXSPATxx (excessive spin condition actions) parmlib member. The system can spin or loop disabled waiting for a resource, such as a lock, to be released by another CP in a multi-processing environment. See z/OS MVS Initialization and Tuning Reference for more information on the EXSPATxx parmlib member.

Symptoms of a loop:

**Disabled loop symptoms**

Disabled loops are easier to identify than enabled loops. Symptoms include:

- System CP usage increases for unexplained reasons.
- There is no communication with the system through the master and alternate consoles.
- Console communications are locked out. To check for communication with the console, enter DISPLAY T command and the system will not respond.

**Enabled loop symptoms**

Enabled loops allow some or all interrupts. The loops are usually caused by an error in an application program. All or most of the loop is in code running in problem state, but the loop can include system code if any instructions in the loop request system services. An enabled loop can run on more than one central processor. The loop will uselessly consume resources and might take over all system operation.
Additional symptoms include:
- A bottleneck, indicating that the system slows down periodically, thus creating a performance problem.
- A job stays in the system for a long time without changing status or ending.
- Low priority work slows down or stops running (a result of a higher priority enabled loop).
- System CP usage increases for unexplained reasons or CP usage of an address space is much higher than normal.

Spin loop symptoms

A spin loop occurs when one processor in a multiprocessor environment is unable to communicate with another processor or requires a resource currently held by another processor. The processor that has attempted communication is the detecting or spinning processor. The processor that has failed to respond is the failing processor.

The detecting processor continuously attempts its communication with the failing processor until either:
- It is successful.
- A specified time interval has passed.

When the communication is not successful within this interval, an excessive spin loop time out exists. The detecting processor then initiates recovery processing for the condition.

MVS processing for excessive spin-loop conditions can provide recovery without any operator prompts or actions required. The following recovery actions can be defaulted to or specified in the EXSPATxx parmlib member:

**SPIN** Continue spinning for another interval to allow the event to complete

**ABEND** End the current unit of work on the failing processor but allow the recovery routines to retry

**TERM** End the current unit of work on the failing processor and do not allow the recovery routines to retry

**ACR** Invoke alternate CP recovery (ACR) to take the failing processor offline.

- The system chooses the appropriate action without requiring any decision or action. If an action taken in response to an occurrence of an excessive spin loop does not resolve the condition, the system takes the next action when the next excessive spin loop time out occurs. The default order in which the system takes the actions is SPIN, ABEND, TERM, and ACR.
- An installation can change the order of the actions, except the first one, that the system takes.
- For hardware-related errors that formerly caused message IEA490A, the system immediately initiates ACR processing without working through the sequence of actions and without requiring any intervention.
- There is a default spin loop time-out interval. You can change this interval through the combination of a parameter in EXSPATxx parmlib member and entering the SET command.
To avoid unnecessary recovery actions, system functions that can validly exceed the interval are exempt from excessive spin-loop processing, so that they will not cause any recovery actions. If they exceed the time out interval, these system functions do cause an excessive spin loop record to be written to the logrec data set.

The installation can still control excessive spin loop recovery through operator actions. See EXSPATxx (excessive spin condition actions) in z/OS MVS Initialization and Tuning Reference.

Steps for diagnosing a loop

The following steps guide you through diagnosing a loop:

1. “Gathering additional data for a loop” on page 132
2. “Analyzing the dump to determine the type of loop” on page 133
3. “Diagnosing a disabled loop” on page 134
4. “Diagnosing an enabled loop” on page 135
5. “Diagnosing an excessive spin (spin loop)” on page 138
6. “Analyzing a logrec error record” on page 140
7. “Searching the problem reporting databases” on page 141

Use the following flowchart to guide diagnosis of a loop:
Gathering additional data for a loop

By gathering the correct data you can determine what recovery actions are necessary.

Steps for gathering loop data

Gather the following types of data:
1. The description of the external symptoms, including any software or hardware changes.
2. What recovery actions were attempted? Was a MODIFY, CANCEL or FORCE command entered?
3. Request an SVC dump using the system DUMP command for the job or jobs involved in the loop. If this is a subsystem or system address space, define the dump options using the applicable dump parmlib member, IEADMCxx documented in z/OS MVS Initialization and Tuning Reference. Otherwise, specify the following:

   DUMP COMM=(dumptitle) Rxx,ASID=(1,xx),SDATA=(GRSQ,SQA,CSA,RGN,TRT,COUPLE,XESDATA,NUC),END

Where xx is the ASID associated with the looping job. You can also specify it with JOBNAME=(xyz). If an SVC dump is not possible or the system is hung, request a stand-alone dump.

4. If the DUMP command cannot be entered because the system is hung, or does not complete, request a stand-alone dump of the system. See the topic on stand-alone dump in z/OS MVS Diagnosis: Tools and Service Aids.

5. If this is an enabled loop and there is a SYSMDUMP DD coded in the JCL, you can enter a CANCEL command for the looping job with the DUMP option.

Related information

- See z/OS MVS Diagnosis: Tools and Service Aids for information about stand-alone dumps, SVC, SYSMDUMP, and stand-alone dumps.
- See z/OS MVS JCL Reference for the SYSMDUMP DD statement.
- See z/OS MVS System Commands for information about the DUMP and CANCEL commands.

Analyzing the dump to determine the type of loop

You can determine what type of loop you have by analyzing the dump.

**Step for analyzing the dump for loop type**

**Before you begin:** You need access to IPCS.

**Analyze any dump for the type of loop.**

Format the dump with an IPCS STATUS CPU subcommand. Under the heading CPU(X'nn') STATUS, look for the following:

   DISABLED FOR cccccc
   System processing was disabled for one or more types of interrupts for the module running at the time of the dump. The system can be disabled for program event recording (PER), I/O, external interrupts (EXT), and machine checks (MCH).

The type is:

- **Disabled loop:** If the system was disabled for I/O or EXT or both.
- **Enabled loop:** If the system was not disabled for I/O or EXT.

In this example, the statement DISABLED FOR PER indicates an ENABLED loop.

**Example: STATUS CPU Subcommand Output**

CPU STATUS:
Warnings regarding STRUCTURE(ASCB) at ASID(X'0001') 00FD5F00:
   Located via STRUCTURE(ASVT) at ASID(X'0001') 00F336D0
   Storage not in dump

   PSW=070C2000 8AC2CB2A (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
Loop analysis

DISABLED FOR PER
ASID(X'0001') 0AC2CB2A. AREA(PRIVATE)+02CB2A IN EXTENDED PRIVATE ASCB76 at F52080, JOB(CATALOG), for the home ASID

Related information:

See z/OS MVS IPCS Commands for the STATUS subcommand.

Diagnosing a disabled loop

A disabled loop is not visible in the system trace output because disabled routines do not take interrupts. Normally, a disabled loop results in a spin loop in a multiprocessor environment. When analyzing a stand-alone dump for a disabled loop, use the stored status data to determine the module involved in the loop. Disabled loops often result in a system outage if they persist. Therefore, you usually be working with a stand-alone dump.

Steps for diagnosing a disabled loop

1. In an IPCS session, enter the IPCS STATUS CPU command to examine status of each CP. For example:

   CPU(X'02') STATUS:
   PSW=04042000 80000000 00000000 011E8592
   (Running in PRIMARY, key 0, AMODE 31, DAT ON)
   Disabled for PER I/O EXT
   NCPU ASID(X'0001') 011E8592, IAEUCB1,IEAVELX+073A IN READ ONLY NUCLEUS
   ASCB1 at FD3400, JOB(MASTER)+, for the home ASID
   ASKB1 at FD3598 and a local SRB for the home ASID
   HOME ASID: 0001 PRIMARY ASID: 0001 SECONDARY ASID: 0001
   CLTE: 0176820
   HLDING LCKS: CPU
   CURRENT FRR STACK IS: NORMAL

   CPU(X'04') STATUS:
   PSW=04042000 80000000 00000000 017D5126
   (Running in PRIMARY, key 0, AMODE 31, DAT ON)
   Disabled for PER I/O EXT
   NCPU ASID(X'0001') 017D5126, IAEUCB1,IXARC+034E IN READ ONLY NUCLEUS
   ASCB1 at FD3400, JOB(MASTER)+, for the home ASID
   ASKB1 at FD3598 for the home ASID. No block is dispatched
   HOME ASID: 0001 PRIMARY ASID: 0003 SECONDARY ASID: 0367
   CLTE: 01B19900
   HLDING LCKS: CPU RSM RSMCM RSMST RSMAD
   CURRENT FRR STACK IS: NORMAL

In the preceding example, the PSW for CP 2 indicates that it is executing disabled in CSECT IEAVELX, the lock manager. The status for CP 4 indicates several locks are held, therefore, this is a normal spin in IEAVELX. The loop that is causing the problem is in IAXRC.

2. Use the name of the CSECT executing on CP 4, IARXC and the symptom LOOP to check for a known problem.

3. From the module prefix, identify the component, subsystem, or product, if provided by IBM.

   Use the module name to query the SMP/E zone for a module entry with that module name. If the search does not find a match, the module is not an IBM module. If the search indicates a match, use the FMID to positively identify the product.

4. Continue diagnosis as follows:
Loop analysis

a. If all the addresses are in components of z/OS, continue with “Searching the problem reporting databases” on page 141.

b. If all the addresses are in an IBM subsystem or product, continue diagnosis with the diagnosis publication for the subsystem or product. See Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165 for the correct publication.

c. If all the addresses are in components of z/OS and in an IBM subsystem or product, continue with “Searching the problem reporting databases” on page 141 and with the diagnosis book for the subsystem or product. See Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165 for the correct publication.

d. If any of the addresses are in an installation-provided program, including an installation exit routine, continue diagnosis with that program, using the dump.

If some addresses are in the program or routine and some in system modules, the loop is probably in the program or routine and includes one or more requests for system services.

Diagnosing an enabled loop

Enabled loops are often quite large and can include several distinct operations, such as I/O, SVCs, and module linkages. Because the loop is enabled, it is interrupted, preempted, and resumed many times. This makes the loop pattern difficult to recognize. The steps below can help make that identification easier.

For an enabled loop you should have either an SVC dump or a stand-alone dump.

Steps for diagnosing an enabled loop

Verify the system is in an enabled loop by examining the activity in the system trace table.

1. Enter the IPCS SYSTRACE ALL TIME(LOCAL) command. The following is an example of the system trace table output:

```
```

The columns represent:
- The PR, ASID, and TCB-ADDR columns tell you who is executing
- The IDENT CD/D columns tell you the type of event

Figure 14. System trace table entry
**Loop analysis**

- The PSW column tells you where
- The UNIQUE columns tell you how
- The PASD tells you which ASID is being addressed
- The TIMESTAMP tells you when.

2. To diagnose an enabled loop, look for patterns of repetitive entries in system trace. Typically, the bounds of an enabled loop are identified by the PSW addresses in EXT, CLKC, and IO events. In the preceding example, the bounds of the loop appear to go from PSW address 151AFBA6 to 151AFBD0.

3. Enter the IPCS WHERE command on these PSW addresses to determine which module or CSECT the loop is in or browse storage at that location. For example, under IPCS Option 1:

   Command ==> 
   
   CURRENT DEFAULTS: 
   Source ==> DSNAME('H44IPCS.PMR59356.B019.DUMP') 
   Address space ==> ASID(X'0001') 
   
   OVERRIDE DEFAULTS: 
   Source ==> DSNAME('H44IPCS.PMR59356.B019.DUMP') 
   Address space ==> ASID(X'301') 
   
   Password ==> 
   POINTER: 
   Address ==> 151AFBA6 
   Remark ==> ASID(X'0301') 

   Browse backward (PF 7) looking for an module or CSECT identifier.

   Command ==> 
   151AEAA0 1606B1F8 00000000 00000000 00000000 | .....8........... |
   151AEB00 00000000 04000005 00000000 00000000 | ................ |
   151AEEC0:151AEECF. LENGTH(X'10')--All bytes contain X'00' 
   151AEE00 00000000 00000000 00000000 00000000 | ................ |
   151AEE00 00000000 00000000 03002100 30004001 | ................ |
   151AEEF0.:151AEF0F. LENGTH(X'20')--All bytes contain X'00' 
   151AEF10 00000000 00000000 00000000 00000000 | ................ |
   151AEF20.:151AEFFF. LENGTH(X'10')--All bytes contain X'00' 
   151AFE00 09ECDDB4 A7C50046 151B0688 C3D8D4D6 | ....xE.....hCQMO |
   151AFE01 09C5C4C3 40404040 4040F0F3 61F0F261 | REDC 03/02/ |
   151AFE02 0F6F6F1F 48F4F7A0 40404040 40404040 | 0619.47 |
   151AFE03 40404040 40404040 40404040 F2F2F040 | 05305 |
   151AFE04 D72F2F9 F6F0F040 F5F6F9F7 60C9F0F3 | PK20600 5697-103 |
   151AFE05 40404040 40404040 40404040 F2F2F040 | 05305 |
   151AFE06 40404040 40404040 40404040 F2F2F040 | 05305 |
   151AFE07 40404040 40404040 40404040 F2F2F040 | 05305 |
   151AFE08 40404040 40404040 40404040 F2F2F040 | 05305 |

   Browse backward (PF 7) looking for an module or CSECT identifier.

4. Obtain the PSW addresses from the system trace entries involved in the loop. For ANALYZE output and SUMMARY KEYFIELD CURRENT output, use the TCB address.

   If using a PSW address, ignore the leftmost bit of the leftmost digit. The leftmost bit of the leftmost digit denotes addressing mode and is not part of the address.

5. Do one of the following actions, for each address in the loop:
   - If analyzing the dump interactively, use the address in a WHERE subcommand to obtain the name of the load module.
   - If analyzing printed output, find the address:
Loop analysis

- In dump output from the LIST or VERBEXIT SUMDUMP subcommand. Look for the CSECT name eye-catcher. IBM module names are generally followed by an assembly date and a product identifier or PTF level, such as HBB7720 or UY01234; most eye-catchers are at the beginning of a module, but some are at the end.
- In a module listed for the LPAMAP or VERBEXIT NUCMAP subcommand. LPAMAP will list load modules. Use AMBLIST to obtain the offsets of CSECTS within those load modules. NUCMAP lists CSECTs with offsets, but can only be used for modules within the nucleus.

In the following example, **STATUS CPU** output and **WHERE** subcommand, the PSW identifies the address as X'13206AA'.

```plaintext
CPU STATUS:

PSW=040C2000 813206AA (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
   DISABLED FOR PER I/O EXT
   ASID(X'0006') 013206AA. IEANUC01.IXLM2SP+07AA IN READ ONLY NUCLEUS
   ASCB1 at FD1780, JOB(=MASTER=), for the home ASID
   ASXB1 at FD1A30 for the home ASID. No block is dispatched
   HOME ASID: 0001 PRIMARY ASID: 0006 SECONDARY ASID: 0006

Using the WHERE subcommand, the load module name is IXLM2SP plus an offset of 7AA.
   ASID(X'0006') 013206AA. IEANUC01.IXLM2SP+07AA IN READ ONLY NUCLEUS
```

6. From the module prefix, identify the component, subsystem, or product, if provided by IBM.

Use the module name to query the SMP/E zone for a module entry with that module name. If the search does not find a match, the module is not an IBM module. If the search indicates a match, use the FMID to positively identify the product.

7. Continue diagnosis as follows:

   - If all the addresses are in components of z/OS, continue with “Searching the problem reporting databases” on page 141.
   - If all the addresses are in an IBM subsystem or product, continue diagnosis with the diagnosis book for the subsystem or product.
   - If all the addresses are in components of z/OS and in an IBM subsystem or product, continue with “Searching the problem reporting databases” on page 141 and with the diagnosis book for the subsystem or product.
   - If any of the addresses are in an installation-provided program, including an installation exit routine, continue diagnosis with that program, using the dump.

   If some addresses are in the program or routine and some in system modules, the loop is probably in the program or routine and includes one or more requests for system services.

**Related information:**

- See the topics on IBM component, subsystem, product program identifier or module prefix in z/OS MVS Diagnosis: Reference
- See z/OS MVS IPCS Commands for the IPCS subcommands.
- See z/OS MVS Diagnosis: Tools and Service Aids
- See z/OS MVS Diagnosis: Reference for the SVC summary.
- For the format and contents of the EED and TCB, see z/OS MVS Data Areas in the z/OS library at [http://www.ibm.com/systems/z/os/zos/bkserv/](http://www.ibm.com/systems/z/os/zos/bkserv/)
Diagnosing an excessive spin (spin loop)

To avoid unnecessary recovery actions, system functions that can validly exceed the interval are exempt from excessive spin-loop processing, so that they will not cause any recovery actions. If they exceed the time-out interval, these system functions do cause an excessive spin loop record to be written to the logrec data set.

You can examine the in-storage logrec buffer for entries that recovery routines have made, but which were not written to the logrec data set because of a system problem. Very often it is these records that are the key to the problem solution. See the topic on obtaining information from the logrec in z/OS MVS Diagnosis: Tools and Service Aids.

When the system writes a dump, the dump includes the records in the logrec buffer in storage; the buffer records have been either written to the logrec data set or are queued to be written to the logrec data set. When you begin to diagnose a dump for a system problem, you can use IPCS to view the system records in the logrec recording control buffer.

Before you begin You need to have IPCS installed and the following information:
- An EREP report of SYS1.LOGREC
- Any SVC dumps, stand-alone dumps showing the ABEND071, or SYSMDUMP dump
- SYSLOG and OPERLOG. Look for spin loop messages like IEE331A or IEE178I. Note that IEE331A is rare; IEE178I is more likely to occur. For example:

```
00 IEE331A PROCESSOR (0) IS IN AN EXCESSIVE DISABLED SPIN LOOP WAITING FOR CPU IN STOPPED STATE
REPLY U OR SPIN TO CONTINUE SPIN
REPLY ABEND TO TERMINATE WORK ON PROCESSOR (1) WITH RETRY,
REPLY TERM TO TERMINATE WORK ON PROCESSOR (1) WITHOUT RETRY,
OR STOP PROCESSOR (1) AND REPLY ACR
(AFTER STOPPING THE PROCESSOR, DO NOT START IT)
```

```
IEE178I AUTOMATIC RECOVERY IS IN PROGRESS NO OPERATOR ACTION IS REQUIRED
PROCESSOR (00) DETECTED AN EXCESSIVE DISABLED SPIN LOOP WAITING FOR LOCK RELEASE FROM PROCESSOR (03).
AUTOMATIC RECOVERY ACTION IS SPIN
```

Steps for diagnosing an excessive spin

Use the following steps:

1. In an EREP report of SYS1.LOGREC or an IPCS VERBX LOGDATA report in a dump, enter FIND S0071 (X'071'). You might find multiple records for S0071 with different reason codes.

2. The following example is for an ABEND071 RSN10, which is the first record written to logrec for a spin condition:

Example: VERBEXIT LOGDATA Output

```
SEARCH ARGUMENT ABSTRACT

PIDS/575251CM RIDS/IEANUC01#L RIDS/IEAVTEXS AB/S0071 PRCS/00000010 REGS/0E68E REGS/0C7AA RIDS/IEAVTEXS#R
SYMPTOM DESCRIPTION
------- -----------
```
Look for a SOFTWARE RECORD for abend X’071’ with the following reason codes:

- Reason code X’10’: Recovery action of SPIN was taken, and a logrec entry was written to provide information about the CSECT/module that was detected as looping. No attempt was made to abend the unit of work.
- Reason code X’20’: Recovery action of ABEND was taken, targeting the looping program with a retryable ABEND071.
- Reason code X’30’: A recovery action of TERM was taken, targeting the looping program with a non-retryable ABEND071.

In a logrec record for an abend X’071’, look in the SEARCH ARGUMENT ABSTRACT for the symptom RIDS/IEAVTEXS. If found, look under the heading VARIABLE RECORDING AREA (SDWAVRA) for the following problem data:

- EX SPIN RECORD in the EBCDIC text
- An array of up to 16 pointers: addresses of the functional recovery routines (FRR) on the interrupted FRR stack
- A binary number: index into the FRR stack of the current FRR
- Array of processors causing the spin from SVTSPCP
- Spin loop timeout interval
- Spin loop recovery actions
- Control registers

In the following output, the SDWAVRA indicates an excessive spin record and contains an array of pointers to the FRR stack.

**Example: VERBEXIT LOGDATA Output — Variable Recording Area**

```
VARIABLE RECORDING AREA (SDWAVRA)
+000  KEY: 39  LENGTH: 0E
+002  C5E740E2  D7C9D540  D9C5C3D6  D9C4  | EX SPIN RECORD |
+010  KEY: 37  LENGTH: 04
+012  C6D9D9E2  | FRRS |
+016  KEY: 38  LENGTH: 40
+018  811F43FB  811D74ED  81319571  81086D05  | A..8A...A.N.A._.|
+028  811D74ED  00000000  00000000  00000000  00000000  | A.............|
+038  00000000  00000000  00000000  00000000  | ................|
+048  00000000  00000000  00000000  00000000  | ................|
```

a. Enter the IPCS WHERE command on the PSW address at the time of error from the logrec entries found for an S0071. This should identify the CSECT/module that is excessively looping.

b. Search for a known problem using the symptoms: ABEND071 and the module or CSECT name identified. If IPCS WHERE fails to identify a CSECT/module name, browse storage preceding the psw address looking for a module eye catcher using IPCS Option 1.

3. Identify the modules containing each instruction in the loop, as follows: For automatic spin loop recovery, if the dump is an SVC dump: Use a STATUS
Loop analysis

WORKSHEET subcommand to obtain the dump title. Obtain the component name, component identifier, or module name from the title.

In the following example, STATUS WORKSHEET output, the dump title is for the XES component.

MVS Diagnostic Worksheet
Dump Title: COMPON=IXL,COMPID=5752SCIXL,ISSUER=IXLM1REC,MODULE=IXLM2SP
,ABEND=S0071,REASON=00000030

CPU Model 9021 Version A6 Serial no. 300359 Address 03
Date: 03/30/93 Time: 10:32:38 Local

Original dump dataset: SYS1.DUMP32

4. From the module prefix, identify the component, subsystem, or product, if provided by IBM.

Use the module name to query the SMP/E zone for a module entry with that module name. If the search does not find a match, the module is not an IBM module. If the search indicates a match, use the FMID to positively identify the product.

5. Continue diagnosis as follows:
   - If all the addresses are in components of z/OS, continue with “Searching the problem reporting databases” on page 141.
   - If all the addresses are in an IBM subsystem or product, continue diagnosis with the diagnosis book for the subsystem or product.
   - If all the addresses are in components of z/OS and in an IBM subsystem or product, continue with “Searching the problem reporting databases” on page 141 and with the diagnosis book for the subsystem or product.
   - If any of the addresses are in an installation-provided program, including an installation exit routine, continue diagnosis with that program, using the dump.

If some addresses are in the program or routine and some in system modules, the loop is probably in the program or routine and includes one or more requests for system services.

Related information:
   - See z/OS MVS IPCS Commands for the IPCS subcommands.
   - See the IBM component, subsystem, product program identifier or module prefix in z/OS MVS Diagnosis: Reference.
   - See z/OS MVS Diagnosis: Tools and Service Aids for information about analyzing an SVC dump for a problem in an installation-provided program.

Analyzing a logrec error record

Analyze a logrec error record for a disabled loop. The excessive-spin logrec error record can identify the module running on the processor causing the spin condition.

Steps for analyzing a logrec error record

Before you begin: You need access to the logrec error record.

Use the following steps to identify the module running on the processor causing the spin condition, as follows:
1. Locate the 16 FRR addresses from the stack that was current when the target processor was restarted. These addresses appear after the identification text EX SPIN RECORD at the start of the VRA.

2. Identify the current FRR on the stack from the INDEX=x value that follows the sixteen addresses. The value of x can be 0 through 16.
   If x is 0, the stack contains no current FRRs. Otherwise x is an index indicating which of the 16 addresses points to the current FRR. For example, if x is 2, the second address points to the current FRR.

3. Use a storage map to identify the component that owns the FRR at this address.

   For example: In the following output example, the current FRR is 81319571, which is the third FRR in the stack. This is the current FRR because INDEX=03.

   VERBEXIT LOGDATA output — FRR Stack
   +012  C6D909E2
   +016  KEY: 38  LENGTH: 40
   +018  811F43F8  811D74ED  81319571  81086005  A..A.A..N.A..A.
   +028  811D74ED  00000000  00000000  00000000  A.............
   +038  00000000  00000000  00000000  00000000  ...............
   +048  00000000  00000000  00000000  00000000  ...............
   +058  KEY: 37  LENGTH: 06
   +05A  C9D5C4C5  E77E
   +060  KEY: 38  LENGTH: 01
   +062  03

---

Searching the problem reporting databases

Search arguments are used to search problem reporting databases. If the problem you are diagnosing was already reported and the symptoms are in the database, the search produces a match. Searching is an iterative process; you might need to gather additional data and continue your search.

Steps for searching the problem reporting databases

Use the following steps to search the problem reporting databases and determine if the problem was previously reported:

1. Create a search argument using the symptoms applicable for the type of loop being diagnosed:
   a. Disabled loop search argument - LOOP module/CSECT name
   b. Enabled loop search argument - LOOP module/CSECT name
   c. Spinloop search argument - ABEND071 module/CSECT name.
   Tip: Use free-format search arguments. For more information, see “Searching problem reporting databases” on page 8.

2. If the search finds no match, remove some symptoms or add some symptoms. Search again. Continue searching for matches by adding and removing symptoms.
   If the search finds that the problem was previously reported, request the fix.

3. If the search does not produce a match, continue with the next step. Use problem data from the preceding steps to create more symptoms; use these symptoms in later searches.
4. If you still cannot find the cause of the loop or if the problem is new, report the problem to IBM using the procedure in Chapter 15, “Reporting problems to IBM,” on page 169. Provide the following problem data as listed in Chapter 16, “Problem diagnostic worksheet,” on page 173.

- Problem type: type of loop
- Search argument
- Dump, formatted by IPCS, online or printed
- Range of loop
- SMF records, if obtained
- Accompanying messages: identifiers and texts
- Hard-copy log, beginning 15 to 30 minutes before the problem, or master trace, if not wrapped between the problem and dump
- All printed output and output data sets related to the problem
- Data on any related problems
- Module name and level
- Name and level of the operating system(s) with a list of program temporary fixes (PTF) applied at the time of the problem and all installation modifications, exits, and products with other than Class A service
- Other problem data developed while using the procedures in this document or other diagnosis books for the component, subsystem, or program

Related information
- See “Searching problem reporting databases” on page 8 for more information.
- See z/OS MVS IPCS Commands for the IPCS subcommands.
Chapter 12. Diagnosing an output problem

Overview of analyzing output problems

Most output problems occur during installation and testing of new functions or applications.

This chapter explains what information you will need to properly diagnose an output problem.

Symptoms of output problems: Your output is incorrect, incomplete, or missing, but messages indicate successful processing.

- **Incorrect output**: The processing produced all the expected output, but some output is incorrect. For example:
  - Some values in a report are wrong
  - The text in a message is incorrect
  - The return or reason code is not valid
  - The records in are not producing expected records out.

- **Incomplete output**: The processing did not produce all the expected output. For example, a column is missing from a report.

- **Missing output**: Some or all of the expected output is missing. For example, a report is missing.

Steps for diagnosing output problems

**Before you begin**: You need to have IPCS installed and have access to the following information:

- Job log
- Input and output data sets for the program
- The JCL for the program
- System log
- Logrec data set

Use the following flowchart to guide you through diagnosing an output problem:
The following steps will guide you through diagnosing output problems:

1. “Collecting problem data for an output problem” on page 145
2. “Analyzing data set allocation for an output problem” on page 145
3. “Analyzing the inputs and outputs” on page 145
4. “Analyzing installation exits for an output problem” on page 146
5. “Identifying the program or component” on page 147
6. “Searching the problem reporting databases for an output problem” on page 148
7. “Gathering additional data for output problems” on page 149
   - “Messages and logrec for output problems” on page 149
Collecting problem data for an output problem

To properly diagnose an output problem you need to collect as much information as possible.

Step for collecting problem data

- If the output is a data set, collect the following:
  - All input data sets for the program
  - Input macros, commands, and statements that are used to request output from the program
  - All output data sets produced by the program
  - The job log
- If the output is a message, return code, or reason code, collect the incorrect or incomplete message or code.

Analyzing data set allocation for an output problem

For missing data set output, analyze data set allocation.

Steps for analyzing data set allocation

1. Look for messages in the job log indicating that all data sets used by the program were properly allocated and unallocated. Look for:
   - The data set that should have contained the output
   - The data set that contained the input used by the program to create the output
   - Indications that the output was sent to another data set or different system
2. If problems are found, correct the JCL.
3. If the problem is not found, continue diagnosis with "Analyzing the inputs and outputs."

Related information:

- See z/OS MVS JCL Reference for JCL coding.
- For explanations of the messages, see:
  - z/OS MVS System Messages, Vol 1 (ABA-AOM)
  - z/OS MVS System Messages, Vol 2 (ARC-ASA)
  - z/OS MVS System Messages, Vol 3 (ASB-BPX)
  - z/OS MVS System Messages, Vol 4 (CBD-DMO)
  - z/OS MVS System Messages, Vol 5 (EDG-GFS)
  - z/OS MVS System Messages, Vol 6 (GOS-IEA)
  - z/OS MVS System Messages, Vol 7 (IEB-IEA)
  - z/OS MVS System Messages, Vol 8 (IEF-IGD)
  - z/OS MVS System Messages, Vol 9 (IGF-IWM)
  - z/OS MVS System Messages, Vol 10 (IXC-IZP)

Analyzing the inputs and outputs

For incorrect or incomplete data set output, analyze the inputs and outputs.
Output problem analysis

Steps for analyzing the inputs and outputs

Perform the following steps:

1. Compare the input and the output. For example, if a device model number is wrong in an IOS report, compare it to the model number specified on the in the IODF.

2. Compare the output received to examples of the output shown in the user’s guide for the request.

3. Check the call, command, macro, or statement used to request the output. Make sure that all fields contain desired values.

4. Match your findings to the following:
   - **For missing data set output**, check the macro, command, or statement used to request the output. Make sure that the missing output was supposed to be received.
   - **For an incorrect or incomplete message**, match the message received to the message in the book that explains it.
   - **For incorrect return and reason codes**, match the codes to the expected codes. Make sure that the code received and the code in the book are both hexadecimal or both decimal.

Related information:

- For the message text or a return or reason code in a message, see:
  - z/OS MVS System Messages, Vol 1 (ABA-AOM)
  - z/OS MVS System Messages, Vol 2 (ARC-ASA)
  - z/OS MVS System Messages, Vol 3 (ASB-BPX)
  - z/OS MVS System Messages, Vol 4 (CBD-DMO)
  - z/OS MVS System Messages, Vol 5 (EDG-GFS)
  - z/OS MVS System Messages, Vol 6 (GOS-IEA)
  - z/OS MVS System Messages, Vol 7 (IEB-IEE)
  - z/OS MVS System Messages, Vol 8 (IEF-IGD)
  - z/OS MVS System Messages, Vol 9 (IGF-IWM)
  - z/OS MVS System Messages, Vol 10 (IXC-IZP)
  - z/OS MVS Dump Output Messages

- The message book for a subsystem or program.

- For a return or reason code with an abend code, see z/OS MVS System Codes

- For a return or reason code for a macro, see:
  - z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN
  - z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG
  - z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU
  - z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO
  - z/OS MVS Programming: Authorized Assembler Services Guide
  - z/OS MVS Programming: Assembler Services Reference ABE-HSP
  - z/OS MVS Programming: Assembler Services Reference JAR-XCT
  - z/OS MVS Programming: Assembler Services Guide

Analyzing installation exits for an output problem

Analyze all installation exit routines used in obtaining the output.

Steps for analyzing installation exits

1. Check for problems in the logic of each exit routine. Many installation exits are invoked using the dynamic exits facility, which allows an updated exit to be refreshed into the system.
2. If no logic problems are found, remove the options that cause each exit routine to be invoked.
3. Rerun the program.
4. If this action stops the problem, the problem is in the exit routine that was eliminated. Continue diagnosis with that routine.

Related information:
See z/OS MVS Installation Exits for more information on coding installation exit routines.

Identifying the program or component

Identifying the component that is involved with the output enables you to determine the source of the problem.

Steps for identifying the program or component

Identify the program or component involved with the output from one of the following steps:

1. For output from a batch job, obtain the program name from the PGM parameter on the JCL EXEC statement. In the following example the program name is obtained from the JCL EXEC statement:

   In the following example, the name of a program involved with the output is IKJEFT01. It can be found on the PGM parameter of the EXEC statement:

   ```
   /*
   //************************************************************************
   // Batch TSO job (PGM=IKJEFT01)
   //************************************************************************
   //IKJEFT01 EXEC   PGM=IKJEFT01
   //************************************************************************
   //*/
   ```

2. Identify the program or component involved with the output from one of the following:

   • For output from interactive work, use the command being processed to identify the program.
   • For an error message, use the message prefix to identify the program or component or look for the component listed in the message explanation. Look at the preface of any of the z/OS MVS system messages books to find the element or component that the message is associated with and the book where you can find the explanation of the message:

     For example, if you have an incorrect message with message number AHL002A, you can tell from the preface of z/OS MVS System Messages, Vol 1 (ABA-AOM) that the message was issued by GTF, and the explanation can be found in the same book:

```
<table>
<thead>
<tr>
<th>Prefix</th>
<th>Component</th>
<th>Book Title</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHL</td>
<td>Generalized trace facility (GTF)</td>
<td>z/OS MVS System Messages, Vol 1</td>
<td>SA22-7631</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ABA-AOM)</td>
<td>SA22-7590</td>
</tr>
<tr>
<td></td>
<td></td>
<td>z/OS MVS Dump Output Messages</td>
<td></td>
</tr>
</tbody>
</table>
Output problem analysis

3. For a return or reason code accompanying an abend, see the component listed in the explanation of the code. The following example identifies the component issuing an abend:

00D

**Explanation:** An error occurred during processing of a CTRACE or CTRACEWR macro. Register 15 contains xxnnnmmx where nnnn is a reason code that further describes the error.

**Reason Code (hex)**

<table>
<thead>
<tr>
<th>Reason Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>For the CTRACE macro, the parameter list version number is incorrect.</td>
</tr>
<tr>
<td>0002</td>
<td>For the CTRACE macro, the component name either does not begin with an alphabetic or national character, or it contains one or more characters that are not alphanumeric or national characters.</td>
</tr>
</tbody>
</table>

Source: Component trace

4. For a component of z/OS, continue with "Searching the problem reporting databases for an output problem."

5. For an IBM subsystem or product, continue diagnosis with the subsystem or product.

6. For an installation-provided program, including an installation exit routine, continue diagnosis with that program.

**Related information:**

- See [z/OS MVS JCL Reference](#) for the EXEC statement.
- For message prefixes for IBM components, subsystems, and products, see the following books:
  - [z/OS MVS System Messages, Vol 1 (ABA-AOM)](#)
  - [z/OS MVS System Messages, Vol 2 (ARC-ASA)](#)
  - [z/OS MVS System Messages, Vol 3 (ASB-BPX)](#)
  - [z/OS MVS System Messages, Vol 4 (CBD-DMO)](#)
  - [z/OS MVS System Messages, Vol 5 (EDG-GFS)](#)
  - [z/OS MVS System Messages, Vol 6 (GOS-JEA)](#)
  - [z/OS MVS System Messages, Vol 7 (IEF-IEE)](#)
  - [z/OS MVS System Messages, Vol 8 (IEF-IGD)](#)
  - [z/OS MVS System Messages, Vol 9 (IGF-IWM)](#)
  - [z/OS MVS System Messages, Vol 10 (IXC-IZP)](#)
- See [z/OS MVS System Codes](#) for explanations of the abend codes.
- See [z/OS MVS Diagnosis: Reference](#) to relate an IBM component, subsystem or product to a program identifier or module prefix.

---

**Searching the problem reporting databases for an output problem**

Use free-format search arguments to search the problem reporting databases. If the problem you are diagnosing was already reported and the symptoms are in the database, the search produces a match. Searching is an iterative process; you might need to gather additional data and continue your search.
Step for searching the problem reporting database

Use the following steps to search a problem reporting database to determine if this is a known problem.

1. Use a free-format search argument developed from the symptoms. For more information, see “Searching problem reporting databases” on page 8.

2. If the search finds no match, remove some symptoms or change the symptoms. Search again. Continue searching for matches by adding, removing, and changing symptoms.

3. If the search finds that the problem was previously reported, request the problem fix. If not, continue with the next step. Use problem data from following steps to create more symptoms; use these symptoms in later searches.

Related information:

See “Searching problem reporting databases” on page 8 for more information on developing a search argument.

Gathering additional data for output problems

Gathering additional data will increase your chances of finding a match in the problem reporting databases. Use the procedures outlined in this section to gather additional data and continue searching the problem reporting databases.

Steps for gathering additional information for output problems

Use the following steps to collect additional information about:

- “Messages and logrec for output problems”
- “Determine path for output problems”
- “Teleprocessing for output problems” on page 150

Messages and logrec for output problems

1. 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.

2. Look in the following:
   - Job log
   - TSO/E user’s ISPF transaction log or session manager log
   - System log (SYSLOG) for the console with master authority or the alternate console
   - Logrec data set, formatted by EREP.

3. Check for:
   - I/O errors that could affect the output
   - Operator interactions that could affect the output
   - Problems with the access method or function involved: For example, VSAM, BTAM, JES, or WTO.

Determine path for output problems

1. Analyze the path the data should take.

2. For the request being processed, determine the correct path for the data from input to output.

3. Determine each program and component involved.
Output problem analysis

4. Check for messages about problems in these programs and components.
5. If an installation-provided exit routine receives control during the processing, check the routine.
6. Look at the environment. Specifically, look for recent hardware and software changes to the system and to any applications. A change in one program can affect others; for example, a change to an application that updates a database affects all other users of the database.
7. If needed, recreate the problem, using a SLIP trap and traces to obtain the data needed to isolate the problem.

Teleprocessing for output problems

1. Analyze the path the data should take.
2. Determine how the data flows through the programs and components that process it and through the systems and hardware. Use this knowledge to recreate the problem, using traces to checkpoint the data at certain spots along the path.
3. Track the data from a point where it was correct to a point where the data stopped or became incorrect.

Related information:

- For formatting of logrec records, see [Recording logrec error records](#) in [z/OS MVS Diagnosis: Tools and Service Aids](#).
- See [z/OS MVS Diagnosis: Tools and Service Aids](#) for the logrec records.
- For explanations of the messages, see:
  - [z/OS MVS System Messages, Vol 1 (ABA-AOM)](#)
  - [z/OS MVS System Messages, Vol 2 (ARC-ASA)](#)
  - [z/OS MVS System Messages, Vol 3 (ASB-BPX)](#)
  - [z/OS MVS System Messages, Vol 4 (CBD-DMO)](#)
  - [z/OS MVS System Messages, Vol 5 (EDG-GFS)](#)
  - [z/OS MVS System Messages, Vol 6 (GOS-IEA)](#)
  - [z/OS MVS System Messages, Vol 7 (IEB-IEE)](#)
  - [z/OS MVS System Messages, Vol 8 (IEF-IGD)](#)
  - [z/OS MVS System Messages, Vol 9 (IGF-IWM)](#)
  - [z/OS MVS System Messages, Vol 10 (IXC-IZP)](#)
  - [z/OS MVS Dump Output Messages](#)
  - The message book for a subsystem or program
- See [z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures](#) and [z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT](#) to diagnose VTAM® problems.
- See the SLIP chapter in [z/OS MVS System Commands](#) for information on setting a SLIP trap.
- See [z/OS MVS Diagnosis: Tools and Service Aids](#) for requesting dumps and traces.
- See [z/OS MVS Installation Exits](#) for exit routines.

Reporting output problems to IBM

If you have completed the procedures listed in this chapter, cannot find a match in the problem reporting databases, and you believe the failure was caused by a defect in IBM code, call the IBM Support Center.

Provide the following problem data:

- Problem type: INCORROUT, that is, incorrect, incomplete, or missing output in a data set, message, return code, or reason code.
Output problem analysis

- Search argument
- All input associated with the problem, including all data sets, commands, macros, and statements
- All output associated with the problem, including data sets, reports, and records
- JCL for all data sets involved
- Source code for any exit routine involved
- Accompanying messages: identifiers and texts
- Hard copy log, beginning 30 to 60 minutes before the problem, or master trace, if not wrapped between the problem and dump
- Name and level of the operating system(s) with a list of program temporary fixes (PTF) applied at the time of the problem and all installation modifications, exits, and products with other than Class A service

Related information:

- See Chapter 15, “Reporting problems to IBM,” on page 169 for more information on reporting a problem.
- For formatting of logrec records, see the topic on recording logrec error records in z/OS MVS Diagnosis: Tools and Service Aids.
Output problem analysis
Chapter 13. Diagnosing a performance problem

Overview of a performance problem

Most performance problems appear as unacceptable response times or resource usage. For example, system processing is slow because a program is using an excessive amount of system resources.

Symptoms of a performance problem:
- Jobs take more time to execute than normal
- Poor or erratic response time exists
- Service level objectives are being exceeded
- Users complain about slow response time
- Unexpected changes occur in response times or resource utilizations
- Other indicators show stress
- Monitor III Workflow/Exceptions occur.
- System resource indicators occur (for example, paging rates, DASD response)
- Expected throughput on the system is not being attained
- CP utilization for a job or address space is higher than normal.

Steps for diagnosing a performance problem

Before you begin: You need access to following types of information:
- System messages
- SVC dump
- Job log
- TSO/E user’s ISPF transaction log or session manager log
- System log
- Logrec data set
- z/OS Resource Measurement Facility (RMF), SMF, or other system monitoring programs, such as IBM OMEGAMON z/OS Management Console.

RMF provides comprehensive guidance for diagnosing performance problems. Diagnosing a problem: the first steps in z/OS RMF Performance Management Guide offers a practical, task-oriented approach to analyzing performance issues using RMF.

Use the following flowchart to guide you through diagnosing a performance problem:
Performance problem analysis

Use the following steps to guide you through diagnosing a performance problem:

1. "Collecting data using commands" on page 156
2. "Checking for resource contention or loop" on page 157
3. "Searching the problem reporting database" on page 158
4. "Gathering additional data for performance problems" on page 159
5. "Analyzing a dump for performance problems" on page 160
6. "Reporting performance problems to IBM" on page 161

---

Collecting data using commands

Collecting information with commands will give you a better understanding of the source of the problem. In a JES2 system, use JES2 commands to determine why JES2 is not able to schedule work. This topic is divided into two sections:

- "Steps for collecting data using DISPLAY" on page 156
- "Steps for using JES2 commands to collect data" on page 156

---

Steps for collecting data using DISPLAY

Use the following DISPLAY commands in the order listed to find the source of your problem:
1. Determine if the system is waiting for an operator action, for example, mounting of a volume. Other jobs might have to wait until the action is completed. In this case, the operator should perform the action. The following command displays outstanding messages requiring operator action:

```
DISPLAY R,LIST
```

In the following example of **DISPLAY R,LIST** output, there is one message, IEF434D, requiring immediate operator action. The system waits until a valid reply is entered. The operator should enter a valid reply to this message.

```
IEE112I 13.39.37 PENDING REQUESTS FRAME LAST F E SYS=SY1
RM=0 IM=0 CEM=2 EM=0 RU=0 IR=0 AMRF
ID:R/K T JOB ID MESSAGE TEXT
 3 C *ILR005E PLPA PAGE DATA SET FULL, OVERFLOWING TO
    COMMON DATA SET
 5 R *IEF434D CRITJOB - INVALID REPLY. REPLY 'HOLD'
    OR 'OHOLD'
```

2. Look for the generalized trace facility (GTF) in the started tasks. GTF may slow performance. A job step name of STARTING indicates that the system has not yet successfully completed initiation of the first step. If unsuccessful initiation of the job continues, diagnose this problem. The following command displays detailed information for active jobs and started tasks.

```
DISPLAY A,LIST
```

In the following example of **DISPLAY A,LIST** output, all of the steps have been successfully initiated. However, the performance problem could be due to GTF being active:

```
IEE114I 14.51.49 93.181 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 GTF IEFPROC NSW S
```

3. Look for the loss of a hardware component indicated by a message on the hard copy log. This loss may be causing jobs to wait. In this case, correct the hardware problem. The following command, **DISPLAY M** displays the hardware configuration.

```
DISPLAY M
```

4. Look for the name and status of current SLIP traps.
   a. Enter the **DISPLAY SLIP** command for a summary of SLIPs running on the system:

```
DISPLAY SLIP
DISPLAY SLIP=xxxx
```

In the following output of SLIPs Displayed using the **DISPLAY SLIP** command the name and status of the current SLIP traps are shown:

```
IEE735I 13.42.00 SLIP DISPLAY FRAME LAST F E SYS=SY1
ID STATE ID STATE ID STATE ID STATE ID STATE
X013 ENABLED X028 ENABLED X067 ENABLED X0F3 ENABLED X13E ENABLED
X222 ENABLED X322 ENABLED X33E ENABLED X622 ENABLED X804 ENABLED
X806 ENABLED X80A ENABLED X9FB ENABLED X837 ENABLED XD37 ENABLED
XE37 ENABLED
```

b. Pick SLIP IDs that are enabled and enter the **DISPLAY SLIP=xxxx** command for each one to check for enabled PER traps. In this case, disable the traps.
Performance problem analysis

In the output from DISPLAY SLIP above, all the SLIPs appeared enabled. The DISPLAY SLIP=X013 command yields the following enabled PER trap with an action of STRACE, which can slow performance:

IEE7351 09.14.03 SLIP DISPLAY    FRAME LAST    F    E    SYS=SY1
ID=X013,PER-SB,ENABLED(ACTIVE),ACTION=STRACE,SET BY TSO KLOGAN

Related information:
- See z/OS MVS System Commands for the SLIP and DISPLAY commands.
- See z/OS MVS Diagnosis: Tools and Service Aids for information about GTF.

Steps for using JES2 commands to collect data

Use the following procedures to collect problem data with JES2 commands:

1. Use the job entry subsystem display commands to find the status of jobs, queues, printer setups, requirements of SYSOUT data sets, and other problem data.
   
   $D J1-9999
   
   $D J1-9999 output displays the status of jobs. If the display shows that a range of jobs has been held, use the JES2 $A J command to release specific jobs. Or, use the JES2 $A A command to release all jobs in the system.
   
   In the following $D J1-99 output, all of the displayed jobs are being held:
   
   $D J1-99
   
   JOB00005 $HASP608 IEBGENER AWAITING HARDCOPY PRIO HELD 15 ANY
   JOB00006 $HASP608 XEQN1 AWAITING XMITTER POK PRIO HELD 9 ANY
   JOB00007 $HASP608 IEBGENER AWAITING HARDCOPY PRIO HELD 15 ANY
   JOB00008 $HASP608 IEBGENER AWAITING EXECUTION A PRIO HELD 9 ANY

2. Use the following command to display the status of tasks started under JES2:
   
   $D $S1-9999
   
   The following $D $S1-9999 output shows tasks started under JES2:
   
   $D $S1-99
   
   STC0001 $HASP608 SYSLOG EXECUTING $ PRIO 15 IBM2
   STC00002 $HASP608 $MASCOMM AWAITING HARDCOPY PRIO 15 ANY
   STC00003 $HASP608 INIT EXECUTING $ PRIO 15 IBM2
   INITASID=0015
   STC00004 $HASP608 IRRDPTAB ON PRT1 PRIO 1 IBM2

3. Use the following command to display the status of time-sharing users:
   
   $D T1-9999

4. Use the following command to display the status of data set groups queued for output and the status of JES2-controlled local printers.
   
   $D F
   $D U,PRTS

   If these displays show that no printers are set up with the needed forms, use the JES2 $T PRTnnnn command to change a printer’s setup to the needs of the output forms queue.
   
   In the following $D F and $D U,PRTS output, there is only one item queued for the printers. All of the printers, however, are either drained or halted, which means that none of the printers are started. No printing can occur.
   
   $D F
   
   $HASPa621 OUT R=LOCAL F=STD C=**** T=**** W= (NONE)
   $HASPa621 PRMODE=LINE CLASS A=3
   
   $D U, PRTS
5. Use the following command to display the number of queued jobs.

```
$D Q
```

If the problem is that jobs are not running because they are held, you can use the JES2 $A command with appropriate parameters to release the jobs.

Related information:

See [z/OS JES2 Commands](#) for the JES2 commands.

### Checking for resource contention or loop

You can analyze output from RMF, SMF or another system monitoring program to look for resource contention and loops.

### Steps for checking resource contention

**Before you begin:**

Perform the following steps to check for resource contention:

1. 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. The following lists some of the potential problems to look for:
   - 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

2. Identify the program, job or function that is involved in the performance problem. If using more resources than normal, gather RMF, trace output, or both from the time frame of when the slowdown occurred.

   For a problem caused by resource contention, use dump output from "Analyzing a dump for performance problems" on page 160 or the following two choices:
   - Use the ANALYZE output to identify the problem causing the contention:
     - In the following output, resource #0002, which is device 687, shows an intercept condition and is not running. There is one unit of work waiting for this device.

```
CONTESTION EXCEPTION REPORT
JOBNAME=IOS. ASID=0001 UCBAPE=00FC72C0
JOBNAME=IOS. HOLDS THE FOLLOWING RESOURCE(S):
RESOURCE #0002: There are 0001 units of work waiting for this resource
   NAME=I/O Device 687 (TAPE ) VOLSER=....... 
```
Performance problem analysis

DATA=(IOS) Active I/O with ASSIGN held.
   (IOS) Device not ready.
   (IOS) Intercept condition.

STATUS FOR THIS UNIT OF WORK:
   IRAI0102I This address space is on the SRM IN queue.

• For a problem caused by a batch program and identified through JES2
  commands, obtain the program name from the PGM parameter on the JCL
  EXEC statement.

In the following example for obtaining the program name, the name of
the program involved with the output is IKJEFT01. It can be found on the PGM
parameter of the EXEC statement:

    /*
    /*====================================================
    /* Batch TSO job (PGM=IKJEFT01)
    /*====================================================
    //IKJEFT01 EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
    /*

3. Continue diagnosis depending on what program caused the problem.
   • For a component of z/OS, continue with “Searching the problem reporting
database.”
   • For an IBM subsystem or product, continue diagnosis with the subsystem or
     product.
   • For an installation-provided program, including an installation exit routine,
     continue diagnosis with that program.

Related information:
   • See z/OS MVS System Management Facilities (SMF) for the SMF reports.
   • See z/OS RMF User's Guide for information.
   • See z/OS MVS IPCS Commands for the ANALYZE subcommand.
   • See z/OS MVS JCL Reference for the EXEC statement.
   • See z/OS MVS Diagnosis: Reference to find the IBM component, subsystem, or
     product for a program identifier or module prefix.

Searching the problem reporting database

Search arguments are used to search problem reporting databases. If the problem
you are diagnosing was already reported and the symptoms are in the database,
the search produces a match. Searching is an iterative process; you might need to
gather additional data and continue your search.

Steps for searching the problem reporting databases

1. Search a problem reporting database to determine if the problem was
   previously reported.
   Include any of the following free-format search arguments that apply to the
   performance problem, and the word PERFORMANCE. For example:
   • System address space name
   • Function name
   • Product involved
   • Module or CSECT name
   • Resource names
   • Message ID
Performance problem analysis

- Symptoms created from information in STATUS CPU output
  For more information, see “Searching problem reporting databases” on page 8.

2. If the search finds no match, remove some symptoms or add some symptoms. Search again. Continue searching for matches by adding or removing symptoms.

3. If the search finds that the problem was previously reported, request the problem fix. If not, continue with “Gathering additional data for performance problems.” Use the problem data gathered there to create more symptoms; use these symptoms in later searches.

Related information:
- See “Searching problem reporting databases” on page 8.
- See z/OS MVS IPCS Commands for the STATUS and VERBEXIT SYMPTOM subcommands.

Gathering additional data for performance problems

Gathering additional data will increase your chances of finding a match in the problem reporting databases. Use the procedures outlined in this section to gather additional data and continue searching the problem reporting databases.

Steps for gathering additional information for performance problems

1. Collect and analyze messages and logrec records about the problem. Use time stamps to select messages and software, symptom, and hardware records for logrec. Look at any messages or records that occurred before and during the time the problem was first reported. Look in the following:
   - The job log
   - A TSO/E user’s ISPF transaction log or session manager log
   - The system log (SYSLOG) for the console with master authority or the alternate console
   - VERBEXIT MTRACE dump output, which shows the buffer for system messages
   - VERBEXIT LOGDATA dump output, which formats the logrec buffer
   - Logrec data set, formatted by EREP

2. Use IPCS to look at the dump.

Related information:
- See z/OS MVS IPCS Commands for the VERBEXIT LOGDATA and VERBEXIT MTRACE subcommands.
- For formatting of logrec records, see Recording logrec error records in z/OS MVS Diagnosis: Tools and Service Aids
- For explanations of the messages, see:
  - z/OS MVS System Messages Vol 1 (ABA-AOM)
  - z/OS MVS System Messages Vol 2 (ARC-ASA)
  - z/OS MVS System Messages Vol 3 (ASB-BPX)
  - z/OS MVS System Messages Vol 4 (CBD-DMO)
  - z/OS MVS System Messages Vol 5 (EDG-GFS)
  - z/OS MVS System Messages Vol 6 (GOS-IEX)
  - z/OS MVS System Messages Vol 7 (IEB-IEE)
  - z/OS MVS System Messages Vol 8 (IEF-IGD)
Performance problem analysis

- [z/OS MVS System Messages, Vol 9 (IGF-IWM)]
- [z/OS MVS System Messages, Vol 10 (IXC-IZP)]
- See the message book for a subsystem or program.
- See [z/OS MVS Dump Output Messages](#) for dump output messages.

Analyzing a dump for performance problems

When all else fails, analyzing a dump might yield some additional information about performance problems. A dump captures a moment in time during the system execution and potentially a small amount of SYSTRACE data. This is typically not enough data to diagnose performance problems, but you might find a loop or a resource contention for the jobs that are performing poorly.

Steps for collecting and analyzing a dump for performance problems

**Before you begin:** Request a dump. If you did not obtain a dump, recreate the problem and request the dump using the following steps:

1. Enter a DUMP command to request an SVC dump and reply with SDATA options to dump global resource serialization control blocks and the nucleus.
2. Specify the address spaces that are experiencing performance reduction. For example, if the problem appears to be in JES2 or JES3, specify the JES address space in the reply.
   
   The following example shows a request for an SVC dump with a sample of the parameters you might use:

   ```
   DUMP COMM=(text)
   REPLY id,ASID=1,SDATA=(GRSQ,NUC,CSA,SQA,TRT),END
   ```

   **Tip:** Be sure to give your dumps meaningful names.

3. Use IPCS to format the dump.
4. If your dump contains the JES address space and your problem appears to be in JES2 or JES3, format the dump using IPCS as follows:
   - For a JES2 system, select JES2 in the Component Analysis panel. Then select JES2 Control Blocks from the JES2 Component Data Analysis panel. From the JES2 Control Block List panel, select the control blocks you wish to format.
   - For a JES3 system, select JES3 in the Component Analysis panel. Then select JES3 Control Block Information from the IPCS JES3 - Primary Options panel. You can also use JMF to analyze performance problems for JES3.

Related information:

- See [z/OS MVS Diagnosis: Tools and Service Aids](#) for requesting an SVC dump.
- See [z/OS MVS System Commands](#) for the DUMP operator command.
- See [z/OS JES2 Diagnosis](#) for information on diagnosing JES2 problems and using IPCS for JES2 diagnosis.
- See [z/OS JES3 Diagnosis](#) for information on diagnosing JES3 problems, using IPCS for JES3 diagnosis, and JMF.
Reporting performance problems to IBM

If you have completed the procedures listed in this chapter, cannot find a match for your performance problem in the problem reporting databases, and believe the problem is a defect in IBM code, call the IBM Support Center.

Provide the following problem data:
- Problem type: performance
- Search argument
- Dump, formatted by IPCS, online or printed
- System responses to DISPLAY and JES commands
- Parmlib members analyzed
- SMF records, if obtained in steps from the hang or wait procedure
- Hard copy log, beginning 30 to 60 minutes before the problem, or master trace, if not wrapped between the problem and dump
- Logrec records, beginning 30 to 60 minutes before the problem
- All printed output and output data sets related to the problem
- Name and level of the operating system(s) with a list of program temporary fixes (PTF) applied at the time of the problem and all installation modifications, exits, and products with other than Class A service

Related information:
- See Chapter 15, “Reporting problems to IBM,” on page 169 for more information.
- For formatting of logrec records, see Recording logrec error records in z/OS MVS Diagnosis: Tools and Service Aids.
Performance problem analysis
Part 4. Diagnosis reference material

Before calling IBM, it is important to gather the correct information. The following topics can help you find specific diagnosis information for z/OS base elements and features and help you have the correct information available to discuss with the IBM support specialist. This section covers the following topics:

- Chapter 14, “Diagnosis information for z/OS base elements and features,” on page 165
- Chapter 15, “Reporting problems to IBM,” on page 169
- Chapter 16, “Problem diagnostic worksheet,” on page 173
Chapter 14. Diagnosis information for z/OS base elements and features

You can find specific diagnosis information for z/OS base elements and features in the following publications and information centers:

Table 13. Where to find diagnosis information for z/OS base elements and features

<table>
<thead>
<tr>
<th>Product</th>
<th>Title</th>
</tr>
</thead>
</table>
| IBM information centers and product libraries | • IBM product libraries
• z/OS basic skills information center at [http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp](http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp)
  – IBM WebSphere® MQ information center
  – IBM TotalStorage® DS6000™ information center
  – IBM TotalStorage DS8000® information center
  – CICS Transaction Gateway for z/OS. |
| APPC | • [z/OS MVS Programming: Writing Transaction Programs for APPC/MVS](http://www.ibm.com/support/docview.wss?uid=swg27002317) |
| BDT | • [z/OS BDT Diagnosis Reference](http://www.ibm.com/support/docview.wss?uid=swg27000851) |
| Binder (Program Management) | • See the topic on binder serviceability aids in [z/OS MVS Program Management: User's Guide and Reference](http://www.ibm.com/support/docview.wss?uid=swg27000851) |
| C/C++ | • See the topic on diagnosing problems in [z/OS XL C/C++ User’s Guide](http://www.ibm.com/support/docview.wss?uid=swg27001880) |
| CICS | • [CICS Problem Determination Guide](http://www.ibm.com/support/docview.wss?uid=swg27000828)
• [CICSPlex SM Problem Determination](http://www.ibm.com/support/docview.wss?uid=swg27000828)
• [CICSVR V3R1 Messages and Problem Determination](http://www.ibm.com/support/docview.wss?uid=swg27000828)
• [CICS Debugging Tools Interfaces Reference](http://www.ibm.com/support/docview.wss?uid=swg27000828)
• [CICS Diagnosis Reference](http://www.ibm.com/support/docview.wss?uid=swg27000828) |
| Common Information Model | • See the troubleshooting topic in [z/OS Common Information Model User’s Guide](http://www.ibm.com/support/docview.wss?uid=swg27000828) |
Table 13. Where to find diagnosis information for z/OS base elements and features (continued)

<table>
<thead>
<tr>
<th>Product</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communications Server</td>
<td>• z/OS Communications Server: IP Diagnosis Guide</td>
</tr>
<tr>
<td></td>
<td>• z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures</td>
</tr>
<tr>
<td></td>
<td>• z/OS Communications Server: SNA Diagnosis Vol 2, FFS7 Dumps and the VIT</td>
</tr>
<tr>
<td></td>
<td>• z/OS Communications Server Troubleshooting Information Center</td>
</tr>
<tr>
<td></td>
<td>• The topic on Troubleshooting EE problems in z/OS Communications Server: SNA Network Implementation Guide</td>
</tr>
<tr>
<td></td>
<td>• The topic on Problem determination in z/OS Communications Server: SNA Customization</td>
</tr>
<tr>
<td></td>
<td>• The topic on Diagnosing CSM problems in z/OS Communications Server: CSM Guide</td>
</tr>
<tr>
<td></td>
<td>• The topic on Diagnosing problems in z/OS Communications Server: IPv6 Network and Application Design Guide</td>
</tr>
<tr>
<td></td>
<td>• The topic on Diagnosing problems in z/OS Communications Server: IP Configuration Guide</td>
</tr>
<tr>
<td></td>
<td>• The topic on Diagnosing problems in z/OS Communications Server: IP System Administrator’s Commands</td>
</tr>
<tr>
<td>Cryptographic Services</td>
<td>• See the topic on Diagnosis reference information in z/OS Cryptographic Services ICSF System Programmer’s Guide</td>
</tr>
<tr>
<td>Bulk Data Transfer</td>
<td>• z/OS BDT Diagnosis Reference</td>
</tr>
<tr>
<td>Debug Tool for z/OS</td>
<td>• Debug Tool Users Guide</td>
</tr>
<tr>
<td>Debug Tool for z/OS</td>
<td>• Debug Tool Reference and Messages</td>
</tr>
<tr>
<td>Debug Tool for z/OS</td>
<td>• Debug Tool Summary of Commands and Built-in Functions</td>
</tr>
<tr>
<td>DFSMS</td>
<td>• z/OS DFSMSdfp Diagnosis</td>
</tr>
<tr>
<td></td>
<td>• z/OS DFSMSshm Diagnosis</td>
</tr>
<tr>
<td></td>
<td>• z/OS DFMSrmm Diagnosis Guide</td>
</tr>
<tr>
<td>DFSORT™</td>
<td>• z/OS DFSORT Messages, Codes and Diagnosis Guide</td>
</tr>
<tr>
<td>Distributed File Service (SMB)</td>
<td>• See the topics on</td>
</tr>
<tr>
<td></td>
<td>- client does not communicate</td>
</tr>
<tr>
<td></td>
<td>- tuning and debugging guidelines</td>
</tr>
<tr>
<td>Distributed File Service (zFS)</td>
<td>• See the topic on performance and debugging in z/OS Distributed File Service zSeries File System Administration</td>
</tr>
<tr>
<td>Global Resource Serialization</td>
<td>• See the topic on diagnosing global resource serialization in z/OS MVS Planning: Global Resource Serialization</td>
</tr>
<tr>
<td>Graphical Data Display Manager (GDDM®)</td>
<td>• GDDM Diagnosis</td>
</tr>
</tbody>
</table>
Table 13. Where to find diagnosis information for z/OS base elements and features (continued)

<table>
<thead>
<tr>
<th>Product</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>• IBM System z9® Enterprise Class</td>
<td></td>
</tr>
<tr>
<td>• IBM System z9 Business Class</td>
<td></td>
</tr>
<tr>
<td>Hardware Configuration Definition (HCD)</td>
<td>• See the topic on problem determination for HCD in <a href="http://www.ibm.com/servers/resourcelink">z/OS HCD User’s Guide</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS and z/VM HCD Messages</a></td>
</tr>
<tr>
<td>HCM</td>
<td>• See the topic on problem determination in <a href="http://www.ibm.com/servers/resourcelink">z/OS and z/VM HCM User’s Guide</a></td>
</tr>
<tr>
<td>High Leveler Assembler (HLASM)</td>
<td>• See the topic on Programming and diagnostic aids in <a href="http://www.ibm.com/servers/resourcelink">HLASM General Information</a></td>
</tr>
<tr>
<td></td>
<td>• See the topic on Diagnosing assembly errors in <a href="http://www.ibm.com/servers/resourcelink">HLASM Programmer’s Guide</a></td>
</tr>
<tr>
<td></td>
<td>• See the topic on Maintaining High Level Assembler Toolkit Feature on VM in <a href="http://www.ibm.com/servers/resourcelink">HLASM Toolkit Feature Installation and Customization Guide</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">HLASM Installation and Customization Guide</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">HLASM Toolkit Feature Interactive Debug Facility User’s Guide</a></td>
</tr>
<tr>
<td>IMS™</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">IMS Version 8: Diagnosis Guide and Reference</a></td>
</tr>
<tr>
<td>Infoprint</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS Infoprint Server Messages and Diagnosis</a></td>
</tr>
<tr>
<td>Interactive Problem Control System (IPCS)</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS IPCS Commands</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS IPCS Customization</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS IPCS User’s Guide</a></td>
</tr>
<tr>
<td>ISPF</td>
<td>• See the topic on Diagnostic Tools and Information in <a href="http://www.ibm.com/servers/resourcelink">z/OS ISPF Messages and Codes</a></td>
</tr>
<tr>
<td>JES2</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS JES2 Diagnosis</a></td>
</tr>
<tr>
<td>JES3</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS JES3 Diagnosis</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS JES3 Diagnosis Reference</a></td>
</tr>
<tr>
<td>Language Environment</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS Language Environment Debugging Guide</a></td>
</tr>
<tr>
<td>MVS</td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS Diagnosis: Reference</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS Diagnosis: Tools and Service Aids</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS Programming: Resource Recovery</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS IPCS Commands</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS IPCS Customization</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="http://www.ibm.com/servers/resourcelink">z/OS MVS IPCS User’s Guide</a></td>
</tr>
<tr>
<td>Product</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>Print Service Facility</td>
<td>PSF for z/OS: Diagnosis</td>
</tr>
<tr>
<td>Program Management (Binder)</td>
<td>Program Management: User’s Guide and Reference</td>
</tr>
<tr>
<td>RMF</td>
<td>See the topic in <a href="http://www.ibm.com/servers/resourcelink">z/OS RMF Performance Management Guide</a></td>
</tr>
<tr>
<td>Security Server LDAP</td>
<td>See the topic on debugging the LDAP server in <a href="http://www.ibm.com/servers/resourcelink">z/OS Integrated Security Services LDAP Server Administration and Use</a></td>
</tr>
<tr>
<td>Security Server RACF</td>
<td>z/OS Security Server RACF Diagnosis Guide</td>
</tr>
<tr>
<td>System Display and Search Facility (SDSF)</td>
<td>z/OS SDSF Operation and Customization</td>
</tr>
<tr>
<td>SMP/E</td>
<td>SMP/E Messages, Codes, and Diagnosis, SMP/E User’s Guide</td>
</tr>
<tr>
<td>TSO/E</td>
<td>z/OS TSO/E System Diagnosis: Data Areas</td>
</tr>
<tr>
<td>WebSphere Application Server for z/OS</td>
<td>WebSphere Application Server for z/OS V5.0: Diagnosis</td>
</tr>
</tbody>
</table>
Chapter 15. Reporting problems to IBM

Before you begin: Be familiar with the information in this document; know how to collect the data that your software specialist needs to solve your problem.

This chapter covers the following topics:
- “Software support service checklist”
- “Automatic problem reporting” on page 171
- “Invoking IPCS as a background job” on page 171

Software support service checklist

In order to understand and resolve your software support service request in the most expedient way, it is important that you gather information about the problem and have it on hand when discussing the situation with the software specialist. The following information is required:

- Definition of the problem
  It is very important that you are as specific as possible in explaining a problem or question to our software specialists. Our specialists want to be sure that they provide you with exactly the right solution so, the better they understand your specific problem scenario, the better they are able to resolve it. To assist you with problem identification, see the Problem Resolution Worksheet (Appendix A).

- Background information
  If possible, obtain all data about a problem soon after the problem occurs. Otherwise, updates to the system can cause discrepancies in the data. Ask yourself the following questions:
  - What levels of software were you running when the problem occurred? Please include all relevant products, for example: operating system as well as related products.
  - Has the problem happened before, or is this an isolated problem?
  - What steps led to the failure?
  - Can the problem be recreated? If so, what steps are required?
  - Have any changes been made to the system? (workload, hardware, netware or software)
  - Were any messages or other diagnostic information produced? If yes, what were they?
  - It is helpful to have the message number(s) of any messages received when you place the call for support or to document in the ETR.
  - Define your technical problem statements in specific terms and provide the version and release level of the product(s) in question.

- Relevant diagnosis information
  It is often necessary that our software support specialists analyze specific diagnostic information, such as storage dumps and traces, in order to resolve your problem. Gathering this information is often the most critical step in resolving your problem. Product specific diagnostic documentation can be very helpful in identifying what information is typically required to resolve problems. You should keep all problem data until the problem is resolved or until the data
is successfully transmitted to IBM. The following are examples of problem data that the support center might ask you to provide:

- Any changes made to the system recently, preceding when the problem began occurring (for example, PTFs or new products installed or new hardware).
- Problem type (for example: abend, hang, loop)
- Search arguments
- Dump data, see “Invoking IPCS as a background job” on page 171
- Failing input request: macro, command, or statement
- SDWAVRA keys, lengths, and contents
- Offset of the failing instruction into the module or CSECT
- Accompanying messages: identifiers and texts
- Logrec report, if used
- All printed output and output data sets related to the problem
- Data on any related problems
- Module name and level
- Name and level of the operating system(s) with a list of program temporary fixes (PTF) applied at the time of the problem and all installation modifications, exits, and products with other than Class A service
- Other problem data developed while using the diagnosis book for the component, subsystem, or program

- Severity level

  You need to assign a severity level to the problem when you report it, so you need to understand the business impact of the problem you are reporting. A description of the severity levels is in the following table.

### Table 14. Severity levels and examples

<table>
<thead>
<tr>
<th>Severity</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Critical Impact/System Down: Business critical software component is inoperable or critical interface has failed. This indicates you are unable to use the program resulting in a critical impact on operations. This condition requires an immediate solution.</td>
</tr>
<tr>
<td>2</td>
<td>Significant impact: A software component is severely restricted in its use, causing significant business impact. This indicates the program is usable but is severely limited.</td>
</tr>
<tr>
<td>3</td>
<td>Moderate impact; A noncritical software component is malfunctioning, causing moderate business impact. This indicates the program is usable with less significant features.</td>
</tr>
<tr>
<td>4</td>
<td>Minimal impact; A noncritical software component is malfunctioning, causing minimal impact, or a nontechnical request is made.</td>
</tr>
</tbody>
</table>

- Mention the following items that apply to your situation:
  - If you are under business deadline pressure
  - When you are available (for example, when you will be able to work with IBM Software Support)
  - Where you can be reached
  - A knowledgeable alternate contact with whom IBM can speak
  - Other open problems (PMRs/Incidents) with IBM regarding this service request
  - If you are participating in an early support program (ESP)
  - If you have researched this situation prior to calling IBM and have detailed information or documentation to provide for the problem.
Automatic problem reporting

Parts of the system automatically report the need for service to IBM; for example, the central processor complex (CPC) reports problems directly to IBM. If the system contains a Hardware Management Console (HMC), you should be aware that problems in the Sysplex Timer® and in direct access storage devices (DASD) might be automatically reported, even though the problems are recorded by MVS:

- MVS captures information about the problems and creates the following logrec records:
  - ETR record: For problems in the Sysplex Timer
  - DASD-SIM record: For problems in DASD
- For a unique Sysplex Timer or DASD error, HMC creates a problem record (PMR) in RETAIN to notify IBM that service is needed.

Invoking IPCS as a background job

Before calling IBM, format the dump using the IPCS subcommands recommended in the appropriate procedure. Some of the IPCS subcommands take time to run for a large dump and transferring the dump to IBM is often time consuming. You can start an IPCS session before calling IBM. Then, during the call, the output can be browsed as needed. See “Step for invoking IPCS as a background job.”

Sometimes you might discover it is easier to create a second dump with a subset of the original dump. Send the second dump to IBM for initial diagnosis. For example, sometimes only the address spaces of the suspected problem job are necessary to diagnose system hangs. Of course, this is not always the case, so if you do send a subset of the dump to IBM, do not delete the original dump.

Step for invoking IPCS as a background job

Use the IPCS COPYDUMP subcommand to reduce the size of a very large dump, such as a stand-alone dump. An initial review of ASIDs 1-10 and others that are known to be involved in the problem, greatly reduces the size and transfer time of the dataset to be sent to the support center for initial diagnosis.

Use COPYDUMP to extract the problem address spaces that you want to analyze. COPYDUMP always includes address spaces 1 through 4 in the new dump data set, as well as any data spaces associated with the address spaces. Use the LISTDUMP subcommand to see the address spaces available in a dump data set.

The IPCS COPYDUMP subcommand can copy a single unformatted dump from one data set to another. Use the following example JCL to guide your creation of a batch job that invokes IPCS as a background job. This job opens the dump and extracts the desired ASIDs, using the IPCS COPYDUMP command, and saves the result in another data set.
The options on the example IPCS command do the following:

**DEFER**
Defers the use of a dump directory. In this example **COPYDUMP** had no need for the dump directory.

**NOPARM**
Indicates not to use an IPCSPIXnn parmlib member. This eliminates the allocation of the problem and data set directories named by IPCSR00 on the production system.

For a complete list of IPCS options, see [z/OS MVS IPCS Commands](https://www.ibm.com/support/knowledgecenter/SSEKQY_2.1.0/com.ibm.zos.v2r11.iseries.s which is not visible in the image).
Chapter 16. Problem diagnostic worksheet

Use this worksheet when calling IBM Technical Support to help you resolve your problem.

Table 15. What is the impact of your problem?

<table>
<thead>
<tr>
<th>Impact</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>__ 1 Critical business impact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__ 2 Significant business impact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__ 3 Some business impact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__ 4 Minimal business impact</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is there a system outage? If yes, how many systems are affected:

Is the problem repetitive? Can you recreate the problem?

   Number of occurrences:

Details:

Table 16. How is your system configured?

<table>
<thead>
<tr>
<th>System environment and level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP model and serial number:</td>
</tr>
<tr>
<td>z/OS level:</td>
</tr>
</tbody>
</table>

How many systems are involved with the problem?

   LPAR
   VM
   NATIVE
   Sysplex

What other hardware devices are involved?

Table 17. What are the external symptoms

<table>
<thead>
<tr>
<th>External symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded system wait state</td>
</tr>
<tr>
<td>System hung or partitioned from sysplex</td>
</tr>
<tr>
<td>Loop or high system overhead</td>
</tr>
<tr>
<td>Loop or high CP usage by job</td>
</tr>
<tr>
<td>Job/subsystem/application/function failure</td>
</tr>
<tr>
<td>Job/subsystem/application/function hang</td>
</tr>
<tr>
<td>Output incorrect or missing</td>
</tr>
<tr>
<td>Performance or slowdown</td>
</tr>
<tr>
<td>Error message issued</td>
</tr>
</tbody>
</table>

Details:

Table 18. What symptom information did you collect?

<table>
<thead>
<tr>
<th>Symptoms extracted from diagnostic information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump title:</td>
</tr>
</tbody>
</table>
### Table 18. What symptom information did you collect? (continued)

**Symptoms extracted from diagnostic information**

<table>
<thead>
<tr>
<th>ABEND code(s):</th>
<th>Wait state code:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID(s):</td>
<td>Module name(s) and rmid:</td>
</tr>
<tr>
<td>Component ID(s):</td>
<td>Other:</td>
</tr>
</tbody>
</table>

### Table 19. Which type of documentation did you obtain?

**Documentation obtained**

<table>
<thead>
<tr>
<th>Dump produced:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSABEND, SYSUDUMP, CEEDump (formatted dump)</td>
</tr>
<tr>
<td>SLIP, Console, SVC, SYSMDUMP, TDUMP (unformatted dump)</td>
</tr>
<tr>
<td>SADUMP</td>
</tr>
<tr>
<td>Joblog, SYSLOG, OPERLOG or other:</td>
</tr>
<tr>
<td>EREP report of SYS1.LOGREC:</td>
</tr>
<tr>
<td>GTF data set:</td>
</tr>
<tr>
<td>CTRACE data set:</td>
</tr>
<tr>
<td>Other:</td>
</tr>
</tbody>
</table>

### Table 20. What recovery actions did you attempt?

**Recovery actions**

<table>
<thead>
<tr>
<th>Program terminated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job canceled</td>
</tr>
<tr>
<td>Job restarted</td>
</tr>
<tr>
<td>Job forced</td>
</tr>
<tr>
<td>Device taken offline</td>
</tr>
<tr>
<td>Restart key on HMC selected</td>
</tr>
<tr>
<td>System partitioned or re-ipled</td>
</tr>
<tr>
<td>Sysplex restarted</td>
</tr>
</tbody>
</table>

Other:
Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted
mobility or limited vision, to use software products successfully. The major
accessibility features in z/OS enable users to:

• Use assistive technologies such as screen readers and screen magnifier software
• Operate specific or equivalent features using only the keyboard
• Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user
interfaces found in z/OS. Consult the assistive technology documentation for
specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to z/OS TSO/E
about accessing TSO/E and ISPF interfaces. These guides describe how to use
TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF
keys). Each guide includes the default settings for the PF keys and explains how to
modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library
Server versions of z/OS books in the Internet library at:

http://www.ibm.com/systems/z/os/zos/bkserv/
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

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo
106-8711

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS™, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Programming Interfaces Information

This book documents information that is NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at [http://www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

These terms are trademarks or registered trademarks of Ricoh Co., Ltd., in the United States, other countries, or both:

- AFP
- Infoprint

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation 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.
## Index

### A
- **abend**
  - finding the module 93
  - flowchart 88
  - hardware-detected 85
  - message text 86
  - obtaining the reason code 89
  - overview 85
  - searching databases 96
  - software-detected 85
  - symptoms 85
- **abend code**
  - obtaining 89
  - steps for obtaining 89
  - VERBEXIT MTRACE 92
  - where to find 89
- **abend example**
  - determining the dump type 101
  - dump title 93
  - finding the abend code 90, 101
  - finding the job name 99
  - finding the program name 100
  - free-format search argument 97
  - identifying the module prefix 96
  - obtaining the dump title 93
  - RSMDATA SUMMARY report 103
  - searching for multiple problems 100
  - SELECT output 100
- **abend task**
  - abend code, obtaining steps 89
  - component, identifying steps 94
  - data, collecting steps 101
  - data, gathering steps 98
  - databases, searching steps 96
  - logrec, collecting steps 99
  - messages, collecting steps 99, 101
  - module, identifying steps 94
  - trace data, gathering steps 98
  - abnormal wait 128
  - accessibility 175
- **address**
  - where to find 94
- **address space**
  - status 124
- **AIRH**
  - messages 73
- **AIRH messages** 73
- **AIRSHREP.sh**
  - install script 39
- **AMATERSE** 28
- **APAR**
  - finding 12
  - Assist On-Site 28
  - automatic problem reporting 171
  - automatic spin loop 139

### B
- **best practices**
  - stand-alone dump 25
- **bpptomtext** 16
  - example 16

### C
- **CEEDUMP**
  - incorrect JVM path 39
  - check
    - common storage 49
    - frames and slots usage 61
    - Logrec dump rate 55
    - message arrival rate 66
  - coded wait state
  - guidance for diagnosis 87
  - collecting data
    - performance problem 154
    - using JES2 commands 156
  - command
    - no system response 129
  - commands
    - pfa.display 44
    - Predictive Failure Analysis (PFA) 44
  - common storage
    - output report 52
  - communication
    - console
      - locked out 129
  - component owner
    - identifying 6
  - compressing
    - dump data sets 28
  - console
    - disabled loop
      - locked out 129
      - locked out 129
  - control interval (CI)
    - splits 27
COPYDUMP
  syntax 28
  using 27
CP
example 14
creating PFA directories
  example 40
creep
  definition 50
CSA + SQA
  output 52
CSECT 134

D
DAE 10
data set allocation 145
data set output
diagnosing 145
database
for problem reporting 12
databases
  abend symptoms 96
  problem reporting 8
diagnosing
job or subsystem hang
  steps 121
diagnosis
best practices 25
diagnosis 105
hang 105
  information for elements 165
  loop 129
  output problem 143
  overview xi
  performance problem 153
  starting xi
  steps 4
  synopsis xi
  system hang 105
  wait 105
diagnosis data
gathering 5
disability 175
DISABLED FOR PER 133
disabled loop
diagnosing 134
disabled wait 106
disabled wait state 114
DISPLAY
  STATUS 45
DISPLAY command
elements 45
documentation
where to find 22
dummy wait 105
dump
for loop 133
dump directory
  defining 26
dump title
  obtain the title 93
dumps
  compressing 28
dynamic exits 146

E
enabled loop
diagnosing 135
enabled wait 105
encrypting dump data 28
example
  ANALYZE RESOURCE 124
  ASCB control block 125
  bpxmtext 16
  CBFORMAT srb 125
  creating PFA directories 40
  finding the abend address 95
  finding the abend code 92
  free-format search argument 96
  hang
    abend during NIP 111
    CBFORMAT CVT 106
    IEA944W 116
    STATUS CPU 117
    identifying control 127
    identifying the module 96
    IEE611I 6
    IEA794I 5
    IEA911E 5
    IEA995I 86
    IPCS STATUS WORKSHEET 14
    ISMF ABEND PANEL 86
    ISPF abend panel 86
    Java path 42
    loop
      VERBEXIT LOGDATA output 141
      VERBEXIT LOGDATA Output 138
      VRA 139
      LSE 127
    message indicating an abend 85
    mount point 41
    mount points 40
    MVS Diagnostic Worksheet 140
    program check interruptions 94
    program name 100
    RB 127
    RSMDATA SUMMARY report 103
    SELECT output 100
    spin loop messages 138
    STATUS CPU REGISTERS 95
    subsystem hang 121
    SUMMARY FORMAT 126
    SUMMARY FORMAT JOBNAME 124
    SUMMARY TCBERROR 100
    SVC dump request 86
    SVC dumps 5
    symbolic link (symlink) 40
    SYSTEM status 101
    SYSTEM TRACE TABLE 135
    TSO/E abend message 87
    VERBEXIT MTRACE output 101
    VERBEXIT SYMPTOM 92
    VERBX SRMDATA report 128
    VRA 103
    zFS attributes 41
examples
  DISPLAY 45
exception handling
  Predictive Failure Analysis (PFA) 37
excessive spin
diagnosing 138
excessive spin loop  
  steps for diagnosing  138  
time out  130  
EXSPATxx  130  
external symptoms  5  
extracting  
  search argument  8  

F  
fast tape  26  
Flash  12  
  URL  23  

flowchart  
  abend analysis  88  
  job or subsystem hang  123  
  loop  132  
FMID (function management identifier)  
  for operating system  14  
format  
  for symptoms  9  
frames and slots usage  
  prediction report  63  
free-format search  
  building  9  

H  
hang  
  dispatchable TCB or SRB  128  
during IPL  
  steps  110  
  job not running  128  
  logrec, gathering  
  steps  119  
  messages, gathering  
  steps  119  
  overview  105  
  problem description, collecting  
  steps  109  
  problem reporting, searching  
  steps  117  
  symptom  105  
  system trace activity  128  
  wait states during IPL  111  

hang example  
  ABEND issued  112  
  ANALYZE EXCEPTION  112  
  CBFORMAT  113  
  COPYCAPID  119  
  COUPLE SYSPLEX EXCEPTION  117  
  CP disabled coded wait  109  
  CP no work wait  109  
  IPL statistics  110  
  IPLDATA STATUS  110  
  MVS diagnostic worksheet  115  
  NIP statistics  111  
  SELECT ALL  111  
  STATUS CPU  106, 112  
  wait  111, 112  
  wait states during IPL  111  

hardware-detected  
  abend  85  
Hot Topics  23  
how PFA works  36  

I  
IBMLink  12  
  example  12  
  identifying  
    system problems  5  
IEE844W  116  
  incomplete output  145  
  information  
    where to find more  22  
  input  
    analyzing  145  
  install script  
    AIRSHREP.sh  39  
  installation exits  
    analyzing  146  
  installing  
    PFA  37  
    shared file system  39  
  installing PFA  39  
  introduction  
    problem determination  3  
IPCS  
  COPYDUMP  27  
    performance  27  
IPCS STATUS WORKSHEET  
  example  14  
IPCS subcommand  
  by problem  102  
IPL  
  hang  110  
  information  13  
ISGWIWAIT  127  
ISMF  
  ABEND PANEL  86  
ISPF  
  abend panel  86  

J  
Java  
  level  37, 41  
Java 5.0  
  example  41  
Java 6.0  
  example  41  
Java path  
  example  42  
  updating  41  
JCL  
  for dumps  29  
JES2  
  collecting data  156  
job  
  dispatchability  124  
  hang  
    overview  121  
    symptom  121  
  name  
    finding  124  
  job or subsystem hang  
    gathering data  123  
    step  123  
  status  
    determine  123  
    waiting for resources  124
Predictive Failure Analysis (PFA) (continued)
example
typical check 36
exception handling 36, 37
installing 37
messages 73
overview 35
pfa,display 44
prerequisites 37
problem
categories 6
problem data
collecting 3
report to IBM 171
problem determination
introduction to 3
problem diagnostic worksheet 173
problem indicators
overview 6
table 7
problem reporting
additional data for output 149
automatic 171
databases 8
output problems 148
to IBM 171
problem reporting database
search 12
problem resolution
overview 3
problem source identification
definition 3
introduction to 3
problem type
indicators 7
program check 111
program check interruptions
example 94
program name 100
PSI
definition 3
PTF
finding 12
PUTDOC 28

R
real storage manager
usage 103
reason codes
determining meaning 16
z/OS UNIX 16
remote access 28
remote checks
PFA_COMMON_STORAGE_USAGE 49
PFA_FRAMES_AND_SLOTS_USAGE 61
PFA_LOGREC_ARRIVAL_RATE 55
PFA_MESSAGE_ARRIVAL_RATE 66
reporting example
finding the system level 14
how many CPUs 14
resource contention
performance 157
RETAIN symptom 9
RMF
analyze output 157

Index 185
SYMptom (continued)
  RETAIN format  9
  spin loop  130
  wait  105
SYMptoms
  abend  85
  search argument  8
  selecting  9
SYSPLEX
  system partitioned  117
SYSTME LEVEL
  analyzing for abend  98
SYSTRACE
  wait entries  128

T
  task
    abend overview  85
  tasks
    hang, diagnosing steps  106
    resource contention, checking steps  157
  TB status
    examining  126
  tech notes
    URL  23
  techdocs
    URL  23
  Technical help database  12
  technotes
    URL  23
  Technotes  12
  testing
    dump operations  29
  tip
    query trace  6
    routing information  5
  tools
    IBM Health Checker for z/OS  3
  trace
    query  6
    trace data
      gathering for abends  98
  TRSMAIN  28
  TSO/E abend message
    example  87

U
  uniprocessor  14

V
  VERBEXIT MTRACE output  101
VERBEXIT SYMPTOM
  example 92
  subcommands 102

W
  wait
    coded 105
    disabled 106
    non-restartable 106
    restartable 106
    enabled 105
    causes 105
    definition 105
    dummy 105
    no work wait 105
    missing interrupt 106
    non-restartable 106
    restartable 106
    symptom 105
    wait state
      coded
      diagnosing 114
    worksheet
      reporting problems 173

Z
  z/OS Basic Skills information center  xii
  z/OS UNIX
    reason codes  16
Readers’ Comments — We’d Like to Hear from You

z/OS
Problem Management
Version 1 Release 11
Publication No. G325-2564-06

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via e-mail to: mhvrdfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address