MVS Interactive Problem Control System (IPCS) Commands
Note

Before using this information and the product it supports, read the information in “Notices” on page F-1.

This edition applies to version 1, release 13, modification 0 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SA22-7594-12.

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

Figures ............................................................ vii

Tables ................................................................. ix

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

How to send your comments to IBM ......................... xiii
If you have a technical problem ............................. xiii

Summary of changes ............................................. xv
Changes made in z/OS Version 1 Release 13, as 
updated September 2012 ....................................... xv
Changes made in z/OS Version 1 Release 13 ..........  xv
Changes made in z/OS Version 1 Release 11 ..........  xvi

Chapter 1. Introduction ........................................... 1-1
IPCS processing sources facilities, and modes ........ 1-1
Starting IPCS ....................................................... 1-2
Starting IPCS with customized access .................. 1-2
Starting IPCS without customized access .............. 1-2
Directing IPCS output .......................................... 1-2
Attention processing in IPCS ............................... 1-3
Attention processing for IPCS subcommands 
and CLISTS .................................................... 1-3
Attention processing for IPCS REXX Execs. .......... 1-4
Messages and user completion codes ..................... 1-4
Using IPCS parameters ......................................... 1-5
Syntax conventions ............................................. 1-6

Chapter 2. Literal values ......................................... 2-1
Types of literal values........................................ 2-1
Positive integers .............................................. 2-2
Signed integers .................................................. 2-2
General values .................................................. 2-3
Symbols ............................................................ 2-7

Chapter 3. Data description parameter ...................... 3-1
Parts of the data description parameter .............. 3-1
Address, LENGTH, and POSITIONS parameters .... 3-2
Address processing parameters ........................... 3-7
Attribute parameters .......................................... 3-13
Array parameters .............................................. 3-17
Remark parameters ............................................ 3-18

Chapter 4. TSO/E commands ..................................... 4-1
Entering TSO/E commands .................................. 4-1
Task directory of TSO/E commands for IPCS ......... 4-1
ALTLIB command — identify libraries of CLISTS 
and REXX EXECs ............................................ 4-1
BLS9 command — session of TSO commands .......... 4-2
BLS9CALL command — call a program ................ 4-3
IPCS command — start an IPCS session .............. 4-5
IPCSDDIR command — initialize a user or sysplex 
dump directory ............................................. 4-6
SYSDSCAN command — display titles in dump 
data sets .................................................. 4-7

Chapter 5. IPCS subcommands ................................ 5-1
Entering subcommands ....................................... 5-1
Abbreviating subcommands and parameter 
operands ...................................................... 5-1
Overriding defaults .......................................... 5-1
Online help ..................................................... 5-2
Standard subcommand return codes ................... 5-2
Task directory for subcommands ......................... 5-2
  Analyze a dump ........................................... 5-2
  View dump storage ....................................... 5-3
  View trace information ................................. 5-3
  Check system components and key system areas .... 5-4
  Retrieve information in variables .................... 5-5
  Maintain the user dump directory or sysplex 
dump directory ........................................... 5-6
  Perform utility functions ............................... 5-6
  Debug a dump exit program ............................ 5-7
ADDDUMP subcommand — add a source 
description to a dump directory ....................... 5-7
ALTER subcommand — change a name in the IPCS 
inventory .................................................. 5-9
ANALYZE subcommand — perform contention 
analysis .................................................. 5-9
APPCDATA subcommand — analyze APPC/MVS 
component data ......................................... 5-18
ARCHCHECK subcommand — format access register 
data .................................................. 5-22
ASCBEXIT subcommand — run an ASCB exit 
routine .................................................. 5-25
ASCHDATA subcommand — analyze APPC/MVS 
transaction scheduler data ............................. 5-27
ASMCHECK subcommand — analyze auxiliary 
storage manager data ................................. 5-28
CBFORMAT subcommand — format a control 
block ...................................................... 5-29
CBSTAT subcommand — obtain control block 
status ..................................................... 5-35
CLOSE subcommand — release resources in use 
by IPCS .................................................. 5-38
COMCHECK subcommand — analyze 
communications task data ........................... 5-40
COMPARE subcommand — compare dump data ....... 5-43
COPYCAPD subcommand — copy captured 
dump data .............................................. 5-46
COPYDDIR subcommand — copy source 
description from dump directory .................. 5-49
COPYDUMP subcommand — copy dump data ........ 5-52

© Copyright IBM Corp. 1988, 2012
COPYTRC subcommand — copy trace entries or records 5-58
COUPLE subcommand — analyze cross-system coupling data 5-62
CTRACE subcommand — format component trace entries 5-66
DOCPU subcommand — obtain stand-alone dump data for multiple processors 5-77
DIVDATA subcommand — analyze data-in-virtual data 5-79
DLFDATA subcommand — format data lookaside facility data 5-82
DROPDUMP subcommand — delete source description data 5-84
DROPMAP subcommand — delete storage map records 5-87
DROPSYM subcommand — delete symbols 5-88
END subcommand — end an IPCS session 5-90
EPTRACE subcommand — using 72-byte save areas 5-91
EQUATE subcommand — create a symbol 5-92
EVALDUMP subcommand — format dump attributes 5-97
EVALMAP subcommand — format a storage map entry 5-100
EVALPROF subcommand — format PROFILE subcommand options 5-104
EVALSYM subcommand — format the definition of a symbol 5-105
EVALUATE subcommand — retrieve dump data for a variable 5-109
FIND subcommand — locate data in a dump 5-114
FINDMOD subcommand — locate a module name 5-118
FINDSWA subcommand — locate a scheduler work area (SWA) block 5-120
FINDUCB subcommand — locate a UCB 5-120
GO subcommand — resume IPCS trap processing 5-121
GRSDATA subcommand — format Global Resource Serialization data 5-122
GTFTRACE subcommand — format GTF trace records 5-126
HELP subcommand — get information about subcommands 5-133
INTEGER subcommand — format or list a number 5-133
IOSCHECK subcommand — format I/O supervisor data 5-135
IPCSDATA subcommand — request a report about IPCS activity 5-141
IPLDATA subcommand — request IPL reports 5-149
ISPEXEC subcommand — request an ISPF dialog service 5-150
LIST subcommand — display storage 5-151
LISTDUMP subcommand — list dumps in dump directory 5-153
LISTEDT subcommand — format the eligible device table (EDT) 5-160
LISTMAP subcommand — list storage map entries 5-163
LISTSYM subcommand — list symbol table entries 5-165
LISTTOD subcommand — list TOD clock image 5-168
LISTUCB subcommand — list UCBs 5-170
LITERAL subcommand — assign a value to a literal 5-172
LOGGER subcommand — format system logger address space data 5-174
LPAMAP subcommand — list link pack area entry points 5-174
MERGE and MERGEEND subcommands — merge multiple traces 5-176
NAME subcommand — translate an STOKEN 5-179
NAMETOKN subcommand — display the token from a name/token pair 5-181
NOTE subcommand — generate a message 5-184
OMVSDATA subcommand — format z/OS UNIX data 5-187
OPCODE subcommand — retrieve operation code 5-189
OPEN subcommand — prepare resources for use by IPCS 5-190
PATCH subcommand 5-193
Adding or replacing a patch 5-195
Deleting patches 5-195
Listing patches 5-196
PROFILE subcommand — set preferred line and page size defaults 5-197
RENUM subcommand — renumber symbol table entries 5-201
RSMDATA subcommand — analyze real storage manager data 5-202
RUNARRAY subcommand — process an array of control blocks 5-202
RUNCHAIN subcommand — process a chain of control blocks 5-216
RUNCPOOL subcommand — process a CPOOL 5-223
Examples 5-225
SCAN subcommand — validate system data areas 5-226
SELECT subcommand — generate address space storage map entries 5-229
SETDEF subcommand — set defaults 5-232
SMFDATA subcommand — obtain system management facilities records 5-241
SSIDATA subcommand — display subsystem information 5-242
STACK subcommand — create a symbol in the stack 5-242
STATUS subcommand — describe system status 5-243
STRDATA subcommand — format coupling facility structure data 5-253
SUMMARY subcommand — summarize control block fields 5-263
SYMDEF subcommand — display an entry in the system symbol table 5-276
SYSTRACE subcommand — format system trace entries .................................................. 5-277
TCBEXIT subcommand — run a TCB exit routine ..................................................... 5-284
TRAPLIST subcommand — list the status of IPCS traps ........................................... 5-286
TRAPOFF subcommand — deactivate IPCS traps ....................................................... 5-288
TRAPON subcommand — activate IPCS traps ............................................................. 5-290
TSO subcommand — run a TSO/E command ............................................................... 5-293
VERBEIXT subcommand — run an installation-supplied or an IBM-supplied verb exit routine ........................................................... 5-296
VERBEIXT ALCWAIT subcommand — list jobs waiting for devices .......................... 5-300
VERBEIXT ASMDATA subcommand — format auxiliary storage manager data ........... 5-300
VERBEIXT AVMDATA subcommand — format availability manager data ................... 5-301
VERBEIXT BLSAIPST subcommand — format system initialization data ..................... 5-301
VERBEIXT CBDATA subcommand — format component broker data ......................... 5-301
VERBEIXT DAEDATA subcommand — format dump analysis and elimination data ......... 5-302
VERBEIXT GRSTRAICE subcommand — format Global Resource Serialization data ....... 5-304
VERBEIXT IEAVTSFS subcommand — format SVC dump measurements and statistics report .................................................................... 5-306
VERBEIXT IEFENFVX subcommand — list ENF listeners ........................................... 5-312
VERBEIXT IEFIVAWT subcommand — list pending XCF work for tape allocation ........ 5-312
VERBEIXT IEFIVIGD subcommand — list global tape device information .................... 5-312
VERBEIXT JESXCF subcommand — format data for JES XCF component .................... 5-313
VERBEIXT LEDATA subcommand — format Language Environment data ................. 5-314
VERBEIXT LOGDATA subcommand — format logrec buffer records ......................... 5-317
VERBEIXT MMSDATA subcommand — format MVS message service data .................. 5-320
VERBEIXT MTRACE subcommand — format master trace entries .............................. 5-320
VERBEIXT NUCMAP subcommand — map modules in the nucleus ............................... 5-322
VERBEIXT SADMPMSG subcommand — format stand-alone dump message log .......... 5-326
VERBEIXT SRMDATA subcommand — format System Resource Manager data ............. 5-326
VERBEIXT SUMDUMP subcommand — format SVC summary dump data ..................... 5-326
VERBEIXT SYMTPOM subcommand — format symptom string ..................................... 5-328
VERBEIXT VSMDATA subcommand — format virtual storage management data .......... 5-329
VLFDATA subcommand — format virtual lookaside facility data ............................... 5-332
WHERE subcommand — identify an area at a given address ...................................... 5-334
WLMDATA subcommand — analyze workload manager data .................................... 5-343
XESDATA subcommand — format cross system extended services data ....................... 5-345

Chapter 6. IPCS dialog controls. ........................................... 6-1
Using dialog controls ........................................................................................................ 6-1
Commands, PF keys, and codes for panels ................................................................. 6-2
Selection and data entry panels ......................................................................................... 6-3
Pointer and storage panels ................................................................................................. 6-3
Dump display reporter panels ............................................................................................. 6-4
IPCS inventory panel .......................................................................................................... 6-5
Storage panel ....................................................................................................................... 6-6
IPCS dialog primary commands .................................................................................. 6-7
ASCII primary command — display characters as ASCII ........................................... 6-7
CANCEL primary command — end the BROWSE option .............................................. 6-7
CBFORMAT primary command — format a control block ........................................... 6-7
DOWN primary command — scroll data forward ......................................................... 6-8
EBCDIC primary command — display characters as EBCDIC ...................................... 6-9
END primary command — end a subcommand or panel .............................................. 6-9
EQUATE primary command — create a user-defined symbol ....................................... 6-9
EXCLUDE primary command — exclude lines from display ......................................... 6-10
FIND primary command — search for a specified value ................................................ 6-11
IPCS primary command — invoke an IPCS subcommand, CLIST, or REXX exec ........ 6-15
LEFT primary command — scroll data left ..................................................................... 6-17
LOCATE primary command — scroll the display to show specific data ........................ 6-18
MORE primary command — scroll data ........................................................................ 6-20
OPCODE primary command — display operation code ............................................... 6-20
RENUM primary command — renumber symbol entries .............................................. 6-21
REPORT primary command — process IPCS output streams ....................................... 6-21
RESET primary command — remove pending commands ........................................... 6-25
RETURN primary command — display the IPCS primary option menu ........................ 6-25
RFIND primary command — repeat the FIND command ............................................ 6-25
RIGHT primary command — scroll data right ............................................................... 6-26
SELECT primary command — select a pointer to display storage ............................... 6-27
SORT primary command — sort an IPCS report ......................................................... 6-27
STACK primary command — create an IPCS-defined symbol ..................................... 6-28
UP primary command — scroll data backward .............................................................. 6-29
WHERE primary command — identify an area at a given address ............................... 6-30
IPCS dialog line commands ............................................................................................. 6-32
D line command — delete screen output ......................................................................... 6-32
E line command — edit a pointer ...................................................................................... 6-33
Chapter 7. IPCS CLISTs and REXX EXECs .............. 7-1
Task Directory for IPCS CLISTs and REXX EXECs .... 7-1
  Analyze a dump ........................................ 7-1
  Customize an IPCS session ......................... 7-1
  Print dump analysis reports ...................... 7-2
  Print storage data ..................................... 7-2
  Sample CLISTs and REXX EXECs .................... 7-2
BLSCBSAA CLIST — print a stand-alone dump screening report .... 7-3
BLSCBSAP CLIST — print a stand-alone dump detailed report .... 7-3
BLSCBSVA CLIST — print an SVC dump screening report .... 7-4
BLSCBSVB CLIST — obtain an SVC dump screening report .... 7-5
BLSCBSVP CLIST — print an SVC dump detailed report .... 7-5
BLSCBSYA CLIST — print a SYSMDUMP dump screening report .... 7-6
BLSCBSYB CLIST — obtain a SYSMDUMP dump screening report .... 7-6
BLSCBSYP CLIST — print a SYSMDUMP dump detailed report .... 7-7
BLSCDDIR CLIST — create a dump directory .......... 7-8
BLSCDROP CLIST — issue IPCS DROPDUMP for uncataloged DSNAME entries .... 7-9
BLSCPRNT CLIST — print a dump .................... 7-11
BLSCPSQA CLIST — print global system queue areas .......... 7-12
BLSCSCAN CLIST — obtain a stand-alone dump screening report .... 7-12
BLSXWHERE REXX EXEC — find all modules with the same entry point name .... 7-14

Chapter 8. IPCS batch mode .................. 8-1
JCL needed to run IPCS in batch mode .................. 8-1
IPCS cataloged procedure ......................... 8-2
  Running CLISTs with BLSJIPCS ................. 8-2

IPCS symbols .............................................. A-1
  Defining symbols ...................................... A-1
  Creating symbols ...................................... A-1
  IPCS symbol definitions .............................. A-1

IPCS special symbols for system control blocks .............. B-1

Control blocks and data areas scanned, mapped, and formatted .... C-1

Print dump to IPCS conversion summary ................... D-1

Accessibility .............................................. E-1
  Using assistive technologies ...................... E-1
  Keyboard navigation of the user interface ....... E-1
  z/OS information ..................................... E-1

Notices ................................................. F-1
  Programming interface information ................. F-3
  Policy for unsupported hardware .................. F-3
  Trademarks ............................................ F-3

Index .............................................. X-1
Figures

5-1. Syntax of the EPTRACE IPCS subcommand ............................ 5-91
5-2. LISTUCB command report for device 0410 5-172
5-3. PROFILE-Defined Defaults ........................................ 5-197
6-1. Using FIND on the Dump Display Reporter Panel .................. 6-14
6-2. Result of Using FIND .............................................. 6-15
6-3. Result of Using PF5/RFIND ....................................... 6-15
6-4. Using E on the Pointer Panel .................................... 6-34
6-5. Pointer Editing Panel ................................................ 6-34
6-6. Result of Using Edit .............................................. 6-35
Tables

1-1. Destination of IPCS Output . . . . . . . 1-3
5-1. Standard subcommand return codes  5-2
5-2. Structures recognized by EPTRACE  5-92
5-3. GRS resource status values . . . . . 5-126
5-4. EXCLUDE parameter naming conventions 5-198
6-1. Selection and Data Entry Panels - Commands and PF Keys . . . . . . . . . . . .6-3

6-2. Pointer and Storage Panels - Commands and PF Keys . . . . . . . . . . . . 6-3
6-3. Dump display reporter panel - commands and PF keys . . . . . . . . . . . . 6-4
B-1. IPCS special symbols . . . . . . . . . . . . B-1
D-1. AMDPRDMP - IPCS conversion summary  D-1
About this document

This document supports z/OS (5694-A01).

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

This document contains reference information about using IPCS. It presents, in alphabetic order:

- TSO/E commands for IPCS
- IPCS subcommands
- IPCS primary commands
- IPCS line commands
- IPCS CLISTs and REXX execs

It also gives examples of output generated by subcommands.

Who should use this document

This document is for anyone who analyzes unformatted dumps and traces on an MVS system.

Where to find more information

Where necessary, this document references information in other documents, 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.

The following table lists the titles and order numbers for documents related to other products.

<table>
<thead>
<tr>
<th>Short title used in this document</th>
<th>Title</th>
<th>Order number</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS/MVS Diagnosis Guide and Reference</td>
<td>IMS/MVS Version 2 Diagnosis Guide and Reference</td>
<td>LY27-9526</td>
</tr>
</tbody>
</table>

Information updates on the web

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

This document 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/zos/basics/index.jsp
How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:
1. Send an email to mhvrdfs@us.ibm.com
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRDFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   U.S.A.
4. Fax the comments to us as follows:
   - From the United States and Canada: 1+845+432-9405
   - From all other countries: Your international access code +1+845+432-9405

Include the following information:
- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number:
  - z/OS V1R13.0 MVS IPCS Commands
  - SA22-7594-13
- The topic and page number related to your comment
- The text of your comment.

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 submit.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:
- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM support portal at [http://www.ibm.com/systems/z/support/](http://www.ibm.com/systems/z/support/)
Summary of changes

This document contains terminology, maintenance, and editorial changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Changes made in z/OS Version 1 Release 13, as updated September 2012

This document contains information previously presented in z/OS MVS IPCS Commands, SA22-7594-12, which supports z/OS Version 1 Release 13.

New information:

- "RSMDATA subcommand — analyze real storage manager data" on page 5-202 is updated for storage-class memory (SCM)
- "RSMDATA subcommand — analyze real storage manager data" on page 5-202 is updated for 1 MB pageable large pages
- "RSMDATA subcommand — analyze real storage manager data" on page 5-202 is updated for 2 GB fixed large pages

Changed information:

- "RSMDATA subcommand — analyze real storage manager data" on page 5-202 is updated with a shortened RSMDATA summary report added to the IPCS RSMDATA Summary command
- "SYSTRACE subcommand — format system trace entries" on page 5-277 is updated with trace table snapshots for ITCH
- "Attribute parameters" on page 3-13 of data description parameter is updated with optional keyword EP() for INSTRUCTION
- "IPLDATA subcommand — request IPL reports" on page 5-149 is updated with the Result of IPL Data Information

Changes made in z/OS Version 1 Release 13

This document contains information previously presented in z/OS MVS IPCS Commands, SA22-7594-11, which supports z/OS Version 1 Release 12.

New information:

- The CBFORMAT subcommand contains new options for specifying STRUCTURES. See "CBFORMAT subcommand — format a control block" on page 5-29.
- A new IPCS subcommand, DOCPU, gives you the ability to obtain stand-alone dump information from multiple processors using one command. See "DOCPU subcommand — obtain stand-alone dump data for multiple processors" on page 5-77.
- The RSMDATA subcommand contains a new option for specifying high virtual common. See "RSMDATA subcommand — analyze real storage manager data" on page 5-202.
• The SYSTRACE subcommand contains new options for specifying
  CPUMASK(cpu-hexadecimal-mask) and CPUTYPE(ZAAP | ZIIP | STANDARD). See
  “SYSTRACE subcommand — format system trace entries” on page 5-277.

• A new VERBEXIT subcommand for ENF codes: IEFENFVX. See “VERBEXIT
  IEFENFVX subcommand — list ENF listeners” on page 5-312.

**Changed information:**

• In the STATUS subcommand, CPU parameter, a table of CPU start and end times appears.

**Deleted information:**

• The SMFDATA subcommand erroneously told readers that the output logstream must be connected to the current coupling facility structure. That information was eliminated from this documentation.

---

**Changes made in z/OS Version 1 Release 11**

The document contains information previously presented in z/OS MVS IPCS Commands, SA22-7594-09, which supports z/OS Version 1 Release 10.

**New information:**

• A new line about SNAPTRC is added to the STATUS WORKSHEET report.
• CPUD, SRBPT, SVT, and SVTX symbols have been added to IPCS Symbol definitions in “IPCS symbols” on page A-1.
• NOENQ parameter is supported in BLSCDDIR CLIST to suppresses ENQ processing that is intended to block other instances of IPCS from using the directory being prepared for use by IPCSDDIR. See “BLSCDDIR CLIST — create a dump directory” on page 7-8.
• The XP code is created to export dump description to RECFM = VB data set in IPCS Inventory Panel. See “IPCS inventory panel” on page 6-5.

**Changed information:**

• CBFORMAT subcommand is updated with clarifications of FORMAT and MODEL parameters. See “CBFORMAT subcommand — format a control block” on page 5-29.
• CBFORMAT subcommand is updated with clarification of the role played by the STRUCTURE(cbname) option and some special options. See “Address, LENGTH, and POSITIONS parameters” on page 3-2.
• The "Readers' Comments - We'd Like to Hear from You" section at the back of this publication has been replaced with a new section “How to send your comments to IBM” on page xiii. The hardcopy mail-in form has been replaced with a page that provides information appropriate for submitting readers comments to IBM.

**Deleted information:**

• The IPCS problem management subcommands are removed from z/OS. As a result, the appendix "Problem Management Subcommands" is deleted from the book.
• Appendix B. IPCS Special Symbols for System Control Blocks, Appendix C. Control Blocks and Data Areas Scanned, Mapped, and Formatted, and Appendix D. Print Dump to IPCS Conversion Summary are removed. You can use the IPCSDATA command to check what symbols, control blocks, and data areas are
supported by this service level of IPCS with the products installed locally. See
“IPCSDATA subcommand — request a report about IPCS activity” on page
5-141.

- The SETDEF subcommand no longer supports the PROBLEM and NOPROBLEM
  keywords. The documentation for the two keywords is deleted accordingly.
Chapter 1. Introduction

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

IPCS processing sources facilities, and modes

- **Sources for IPCS processing**
  IPCS processes the contents of the following sources:

<table>
<thead>
<tr>
<th>Source</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVC dump data set</td>
<td>Dump written to a data set on DASD or tape</td>
</tr>
<tr>
<td>SYSMDUMP dump data set</td>
<td>ABEND dump written to data sets defined by SYSMDUMP DD statements</td>
</tr>
<tr>
<td>Stand-alone dump</td>
<td>Dump written by the stand-alone service aid</td>
</tr>
<tr>
<td>Trace data set</td>
<td>Data set created by the generalized tracing facility (GTF) or by component trace</td>
</tr>
<tr>
<td>Active system storage</td>
<td>The following in central storage:</td>
</tr>
<tr>
<td></td>
<td>- Storage for the address space in which IPCS is currently running</td>
</tr>
<tr>
<td></td>
<td>- Private storage</td>
</tr>
<tr>
<td></td>
<td>- Any common storage accessible by an unauthorized problem-state program</td>
</tr>
<tr>
<td>Data sets</td>
<td>Virtual storage access method (VSAM) data sets and other data sets for browsing</td>
</tr>
<tr>
<td></td>
<td>Note: For information about how to reference VSAM objects, see “Address processing parameters” on page 3-7</td>
</tr>
</tbody>
</table>

- **IPCS processing facilities**
  IPCS can browse and analyze the records in any of these data sets using general purpose facilities. Special purpose facilities are also included to process two groups of these data sets:
  - The dump data sets and active system storage — for these sources, you can:
    - Browse virtual and other system storage, and control information placed in dumps by the dump-writing program.
    - Request various types of dump data reports.
    - Selectively format trace records found in the dump.
    - Run your own special purpose analysis and reporting CLISTs, REXX execs, Interactive System Productivity Facility (ISPF) dialogs, and exit routines.
  - Trace data sets — IPCS provides specialized processing to facilitate formatting trace data sets. See the z/OS MVS IPCS User’s Guide for further information.

- **IPCS processing modes**
  Using IPCS, you can process dumps in:
  - Full screen mode during an interactive TSO/E session, a session during which line mode messages are shown immediately when written, where interactive ISPF services are also available.
Starting IPCS

The procedure you follow to start IPCS depends on the specific customization, if any, that you or your installation have provided. The z/OS MVS IPCS User's Guide contains a more detailed description of procedures for starting IPCS, and z/OS MVS IPCS Customization explains how to customize access to IPCS.

Starting IPCS with customized access

There should be an option on an ISPF selection panel for starting the IPCS dialog. To start the IPCS dialog, select the appropriate option.

Starting IPCS without customized access

If access to IPCS has not been customized, you can use the following procedure:

1. Logon to TSO/E.
2. (Optional) — Unless you want to use IPCS in line mode, you can skip this step. To start IPCS in line mode, do the following:
   a. Add SYS1.SBLSCLI0 to the SYSPROC concatenation:
      ```
      ALTIB ACTIVATE APPLICATION(CLIST) DA('SYS1.SBLSCLI0')
      ```
   b. Enter the IPCS command:
      ```
      IPCS
      ```
      At this point, you can enter IPCS commands in line mode. You do not need to proceed to the next step unless you want to start the IPCS dialog from IPCS line mode.
3. Start the ISPF dialog:
   ```
   ISPF
   ```
4. Choose the TSO/E commands option from the ISPF menu.
5. Start the IPCS dialog by entering the following at the prompt:
   ```
   EX 'SYS1.SBLSCLI0(BLSCLIBD)'
   ```

Directing IPCS output

Depending on which message routing parameters are in effect (PRINT, NOPRINT, PDS, NOPDS, TERMINAL, NOTERMINAL) and depending in which mode (full-screen, line, batch) you are using IPCS, the output can be directed to different mediums. Note that certain non-report type messages are always routed to the terminal or the SYSTSPRT data set.

The following table provides a summary of the output destination possibilities.
Table 1-1. Destination of IPCS Output

<table>
<thead>
<tr>
<th>Message routing parameters</th>
<th>Using IPCS in line or full-screen mode, the output is directed to:</th>
<th>Using IPCS in batch mode, the output is directed to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT, PDS, and TERMINAL</td>
<td>IPCSPRNT data set, IPCSPDS data set, and Terminal</td>
<td>IPCSPRNT, IPCSPDS, and SYSTSPRT data sets</td>
</tr>
<tr>
<td>PRINT, PDS, and NOTERMINAL</td>
<td>IPCSPRNT and IPCSPDS data sets</td>
<td>IPCSPRNT and IPCSPDS data sets</td>
</tr>
<tr>
<td>PRINT, NOPDS, and TERMINAL</td>
<td>IPCSPRNT data set and Terminal</td>
<td>IPCSPRNT and SYSTSPRT data sets</td>
</tr>
<tr>
<td>PRINT, NOPDS, and NOTERMINAL</td>
<td>IPCSPRNT data set</td>
<td>IPCSPRNT data set</td>
</tr>
<tr>
<td>NOPRINT, PDS, and TERMINAL</td>
<td>IPCSPDS data set and Terminal</td>
<td>IPCSPDS and SYSTSPRT data sets</td>
</tr>
<tr>
<td>NOPRINT, PDS, and NOTERMINAL</td>
<td>IPCSPDS data set</td>
<td>IPCSPDS data set</td>
</tr>
<tr>
<td>NOPRINT, NOPDS, and TERMINAL</td>
<td>Terminal</td>
<td>SYSTSPRT data set</td>
</tr>
<tr>
<td>NOPRINT, NOPDS, and NOTERMINAL</td>
<td>Terminal</td>
<td>SYSTSPRT data set</td>
</tr>
</tbody>
</table>

Note: Unless a different ddname is used on the OPEN subcommand, IPCS associates PRINT with the IPCSPRNT data set.

Attention processing in IPCS

To cancel any IPCS processing, use the attention interrupt key. When you press the attention interrupt key during an IPCS session, IPCS indicates that you have suspended mainline IPCS processing and have initiated an attention interrupt by displaying a message.

Attention processing for IPCS subcommands and CLISTS

- For subcommands and CLISTs running in IPCS line mode, IPCS displays the following message:
  IPCS*
- For subcommands and CLISTs running in the IPCS dialog, IPCS displays the following message:
  Processing suspended--Enter a null line, TIME, END, or ABEND

You can do the following in response to either attention message:

- Resume processing by entering a null line after the attention interrupt. If you are using session manager support at your terminal, press the ERASE EOF key and then press Enter to enter a null line.
- Enter the TSO/E TIME command. The command runs without ending the interrupted processing and the attention interrupt remains in effect.
- Enter the TSO/E ABEND command. The IPCS session abnormally ends with an IPCS user code of X'072' (decimal 114). The abend produces a dump if you have a SYSABEND, SYSUDUMP, or SYSDUMP data set allocated to your session.
- Enter the TSO/E END command. IPCS ends the interrupted processing.
Attention processing

- Perform other processing by entering any other TSO/E command or an IPCS subcommand or CLIST. This causes IPCS to end the interrupted processing and run the new command, subcommand, or CLIST.

If you interrupt and end a subcommand that modifies the problem directory or the data set directory, the modification to the directory might be incomplete.

The ATTN statement of CLIST processing is not supported under IPCS. The scheduling of the attention interrupt causes the IPCS attention exit to be bypassed and control reverts to the terminal monitor program (TMP) level.

Attention processing for IPCS REXX Execs

For REXX execs running in IPCS line mode, the system displays message IRX0920I:
ENTER HI TO END, A NULL LINE TO CONTINUE, OR AN IMMEDIATE COMMAND

You can do the following in response to this message:
- Enter the HI command to end the exec. If the system was processing an IPCS subcommand from the exec at the time of the interrupt, the system allows the subcommand to run to completion before ending the exec.
- Enter a null line after the attention interrupt to resume processing.
- Enter an immediate command. If the system was processing an IPCS subcommand from the exec at the time of the interrupt, the system allows the subcommand to run to completion before processing the immediate command. See z/OS TSO/E REXX Reference for information about immediate commands.

For REXX execs running in the IPCS dialog, IPCS displays the following message:
Enter HI to end, a null line, TIME, or an immediate command

You can do the following in response to this message:
- Enter the HI command to end the exec. If the system was processing an IPCS subcommand from the exec at the time of the interrupt, the system also ends the subcommand.
- Enter a null line after the attention interrupt to resume processing.
- Enter the TSO/E TIME command. The command runs without ending the interrupted processing and the attention interrupt remains in effect.
- Enter an immediate command. If the system was processing an IPCS subcommand from the exec at the time of the interrupt, the system allows the subcommand to run to completion before processing the immediate command. See z/OS TSO/E REXX Reference for information about immediate commands.

Messages and user completion codes

Messages that appear during an IPCS session can come from many sources. The following table identifies the three major types of messages that appear during an IPCS session and the books in which you will find explanations for those messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPCS</td>
<td>z/OS MVS Dump Output Messages</td>
</tr>
<tr>
<td>TSO/E</td>
<td>z/OS TSO/E Messages</td>
</tr>
</tbody>
</table>
User completion codes indicate a problem with IPCS processing. See the IPCS topic in [z/OS MVS Diagnosis: Reference](https://www.ibm.com/support/knowledgecenter) for explanations of the codes.

### Using IPCS parameters

A typical IPCS function invocation is divided into two parts: the **operation**, or command or subcommand name, followed by the **operand**, which consists of parameters. The operation can be a TSO/E command, IPCS subcommand, IPCS primary command, or IPCS line command.

The parameters that are used with the TSO/E commands, IPCS subcommands, and IPCS primary commands are of two types: **positional** and **keyword**.

**Positional parameters**

Positional parameters follow the command name in a certain order. In the command descriptions within this book, the positional parameters are shown in lowercase characters. In the following example, **iosvirba** is a positional parameter on the FINDMOD subcommand:

```
FINDMOD iosvirba
```

**Keyword parameters**

Keyword parameters are specific names or symbols that have a particular meaning to IPCS. You can include these parameters in any order following the positional parameters. In the command descriptions, the keywords are shown in uppercase characters and any variables associated with them are shown in lowercase characters. However, the keywords may be entered in either uppercase or lowercase:

```
TERMINAL | NOTERMINAL
FILE(ddname)
```

Long keywords such as TERMINAL and NOTERMINAL might make syntax easier to read, but it might be a burden to type long keywords. IPCS primary commands, IPCS subcommands and TSO/E commands that are supplied with IPCS provide two ways to allow abbreviating long keywords:

- Some keywords that you tend to use often support explicit, short aliases. For example, you can type C for CHARACTER.
- All keywords support unambiguous truncations. For example, you can enter LEN for LENGTH, because this truncated form is currently unambiguous on all the subcommands that support the LENGTH keyword.

If you are composing a command procedure that you hope will remain useful for a long time, do not truncate keywords in it. As IPCS responds to new demands, new keywords are introduced that might make the previous acceptable truncations
IPCS parameters

ambiguous. Use truncations only when you type commands manually, or when you compose command procedures for short-term use.

Many parameters are unique to an IPCS subcommand. However, two different sets of parameters are used by many subcommands:
- Parameters in the Chapter 3, “Data description parameter,” on page 3-1
- Parameters defined through “SETDEF subcommand — set defaults” on page 5-232

Syntax conventions

For IPCS subcommands, IPCS primary commands, IPCS line commands, and TSO/E commands, the syntax in this book uses the following conventions.

Note: The defaults for the SETDEF-defined parameters are not shown in each subcommand syntax diagram because they are individually set by each IPCS user. Unless a special situation is noted for a particular subcommand, see “SETDEF subcommand — set defaults” on page 5-232 for an explanation of each SETDEF-defined parameter.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Syntax example</th>
<th>Sample entry example</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPERCASE</td>
<td>Uppercase indicates the item must be entered using the characters shown. Enter the item in either uppercase or lowercase.</td>
<td>SUMMARY KEYFIELD</td>
<td>summary keyfield</td>
</tr>
<tr>
<td>lowercase</td>
<td>Lowercase indicates a variable item. Substitute your own value for the item.</td>
<td>LENGTH(length)</td>
<td>length(24)</td>
</tr>
<tr>
<td>' '</td>
<td>Apostrophes indicate a parameter string. Enter the apostrophes as shown.</td>
<td>VERBX VSMDATA 'parameter,parameter'</td>
<td>verbx vsmdata 'error,global'</td>
</tr>
<tr>
<td>( )</td>
<td>Parentheses must be entered as shown.</td>
<td>FLAG(severity)</td>
<td>flag(info)</td>
</tr>
<tr>
<td>{ }</td>
<td>Single braces represent group-related items that are alternatives. You must enter exactly one of the items.</td>
<td>{ COMCHECK</td>
<td>COMK }</td>
</tr>
<tr>
<td>[ ]</td>
<td>Single brackets represent single or group-related items that are optional. Enter one or none of the items.</td>
<td>GTFTRACE [DEBUG]</td>
<td>gtftrace</td>
</tr>
<tr>
<td>{ } { }</td>
<td>Stacked braces represent group-related items that are alternatives. You must enter exactly one of the items.</td>
<td>{ ASCBEXIT } { pgmname } { ASCBX } { * }</td>
<td>ascbx *</td>
</tr>
<tr>
<td>[ ] [ ]</td>
<td>Stacked brackets represent group-related items that are optional. Enter one or none of the items.</td>
<td>[ TERMINAL ] [ NOTERMINAL ]</td>
<td>terminal</td>
</tr>
<tr>
<td>___</td>
<td>Underscore indicates a default option. If you select an underscoped alternative, you need not specify it when you enter the command.</td>
<td>SCAN [ SUMMARY ] [ NOSUMMARY ]</td>
<td>scan</td>
</tr>
</tbody>
</table>
### Syntax conventions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Syntax example</th>
<th>Sample entry example</th>
</tr>
</thead>
<tbody>
<tr>
<td>`</td>
<td>`</td>
<td>Or-sign indicates a mutually-exclusive choice. When used with brackets, enter one or none of the items. When used with braces, you must enter one of the items.</td>
<td>RDCM[(ALL</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Ellipsis indicates that the preceding item or group of items can be repeated one or more times.</td>
<td>SUB((subname[^.subname]...))</td>
<td>sub((sub1. func2.svc3))</td>
</tr>
</tbody>
</table>
Chapter 2. Literal values

Four types of literal values can be used with IPCS subcommands and primary commands.

### Types of literal values

- **Positive integers**: See “Positive integers” on page 2-2.

<table>
<thead>
<tr>
<th>To describe</th>
<th>Specify</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive binary numbers</td>
<td>B’bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb’</td>
</tr>
<tr>
<td>Positive decimal numbers</td>
<td>nnnnnnnnn</td>
</tr>
<tr>
<td>Positive hexadecimal numbers</td>
<td>X’xxxxxxxx’ or X’xxxxxxxx_xxxxxxxx’</td>
</tr>
</tbody>
</table>

  An underscore (_) might be used between hexadecimal digits to improve legibility for values greater than 32 bits.

- **Signed integers**: See “Signed integers” on page 2-2.

<table>
<thead>
<tr>
<th>To describe</th>
<th>Specify</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed binary numbers</td>
<td>B’[+</td>
</tr>
<tr>
<td>Signed decimal numbers</td>
<td>[+</td>
</tr>
<tr>
<td>Signed hexadecimal numbers</td>
<td>X’[+</td>
</tr>
</tbody>
</table>

  An underscore (_) might be used between hexadecimal digits to improve legibility for values greater than 32 bits.

- **General values**: See “General values” on page 2-3.

<table>
<thead>
<tr>
<th>To describe</th>
<th>Specify</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fullword pointers</td>
<td>A’xxxxxxxx’ or A’(Ln)xxxxxxxx_xxxxxxxx’</td>
<td>none</td>
</tr>
<tr>
<td>EBCDIC character strings</td>
<td>C’c...’</td>
<td>none</td>
</tr>
<tr>
<td>Signed binary fullwords</td>
<td>F’[+</td>
<td>-]nnnnnnnnnn’ or F’(Ln)[+</td>
</tr>
<tr>
<td>Signed binary halfwords</td>
<td>H’[+</td>
<td>-]nnnnn’ or H’(Ln)[+</td>
</tr>
<tr>
<td>Picture strings</td>
<td>P’p...’</td>
<td>none</td>
</tr>
<tr>
<td>ASCII character strings</td>
<td>Q’q...’</td>
<td>none</td>
</tr>
<tr>
<td>Any string of characters</td>
<td>‘...’ or “...”</td>
<td>valid only for the FIND primary command</td>
</tr>
<tr>
<td>ASCII text strings</td>
<td>S’s...’</td>
<td>none</td>
</tr>
<tr>
<td>EBCDIC text strings</td>
<td>T’t...’</td>
<td>none</td>
</tr>
<tr>
<td>Uppercase or lowercase letters or numbers</td>
<td>blank,</td>
<td>sign, or comma before and after the value</td>
</tr>
<tr>
<td>Hexadecimal strings</td>
<td>X’xx...’</td>
<td>none</td>
</tr>
</tbody>
</table>
### Literal Values

<table>
<thead>
<tr>
<th>To describe</th>
<th>Specify</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previously entered search value</td>
<td>valid only for the FIND primary command</td>
<td></td>
</tr>
</tbody>
</table>

- Symbols: See “Symbols” on page 2-7.

### Positive integers

Whenever an IPCS subcommand requires a number between 0 and 2\(^{31}\), that number can be entered in any of the following ways:

- **nnnnnnnnnn**
  - This notation describes a decimal number. The value, nnnnnnnnnn, is a positive 1- to 10-digit decimal number.

  **Note:** The maximum value that can be entered using decimal notation is 2147483647 (2\(^{31}\)-1), one less than the maximum positive integer that IPCS can process (for example, as a data length or a page size). In order to designate the maximum value to IPCS, hexadecimal or binary notation must be used.

- **X'xxxxxxxx' or X'xxxxxxxx_xxxxxxxx'**
  - This notation describes a hexadecimal number. The value, xxxxxxxx, is a positive 1- to 8-digit hexadecimal number, preceded by X. Hexadecimal digits A through F can be entered using either uppercase or lowercase letters.

  An underscore (_) might be used between hexadecimal digits to improve legibility for values greater than 32 bits.

- **B'bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb'**
  - This notation describes a binary number. The value, bbbbbbbbbbbbbbbbbbbbbbbbbbb, is a positive 1- to 31-digit binary number preceded by B.

### Signed integers

When an IPCS subcommand requires a number between -2\(^{31}\) and 2\(^{31}\)-1, you can specify the number using any of the following notations:

- **[+]nnnnnnnnnn**
  - This notation describes a decimal number. The value, nnnnnnnnnn, is a 1- to 10-digit decimal number preceded by an optional plus (the default) or minus sign.

- **F'[+]nnnnnnnnnn'**
  - This notation describes a 1- to 10-digit decimal number preceded by an F and an optional plus (the default) or minus sign.

- **X'[+]xxxxxxx'**
  - This notation describes a hexadecimal number. The value, xxxxxxxx, is a 1- to 8-digit hexadecimal number preceded by X and an optional plus (the default) or minus sign. Hexadecimal digits A through F can be entered in either uppercase or lowercase.

- **B'[+]bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb'**
B'−bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb'
This notation describes a binary number. The value, bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb, is a 1- to 31-digit binary number preceded by B and an optional plus (the default) or minus sign.

General values

When an IPCS subcommand accepts a literal value that can describe a string and a number, that value can be expressed as follows:
• Preceded by a letter indicating the type of literal and an apostrophe. The letter can be entered in uppercase or lowercase.
• Followed by an apostrophe.

When the primary commands in the IPCS dialog accept a literal value that can describe a string and a number, that value can be expressed in the same manner as described for the IPCS subcommands and as follows:
• Preceded or succeeded by a letter indicating the type of literal value. The letter can be entered in uppercase or lowercase.
• The literal value can be delimited by either quotation marks or by apostrophes. If the delimiter character is used as part of the value, then each delimiter that is represented in the value must be doubled. For example, if you want to find the EBCDIC character string dump's, enter:
  FIND C'dump''s' or FIND C"dump's"

IPCS accepts 64-bit addresses and signed binary values. The explicit length notation is indicated by an expression within parentheses beginning with the letter "..L" in upper or lower case and followed by a length expressed in decimal. Standard TSO/E separator characters may be used between parts of the expression. The total length of the expression may not exceed 256 characters. See type codes A, F, and H for examples.

IPCS supports the following types of values:

A'[(Ln)]xxxxxxxx_xxxxxxxx'
This notation describes a fullword pointer. The value, xxxxxxx_xxxxxxxx, is a 1- to 16-digit hexadecimal expression. IPCS provides leading zeros if you enter fewer than 16 digits.

The length may be explicitly specified as 1-8 bytes or will default to 4 bytes.

Examples:
A' (L8) F4'
A' (L8) 00000000_000000F4'

C'c....'
This notation describes an EBCDIC character string containing one to 256 characters. The value, c..., is subjected to editing as follows:
• Data entered manually from a terminal may be translated by the TSO/E Terminal I/O Controller.
• IPCS translates each pair of adjacent apostrophes into a single apostrophe.
• The FIND primary command accepts either 'ABC'C or C'ABC' as the same search value.

Note: Lowercase letters are not translated to uppercase when the search argument is formed.
**Literal Values**

*Example:*

find C'aBc'

*Result:* IPCS finds the first occurrence of aBc.

\[ F'[(Ln)][+]|-(nnnnnnnnnn) \]

This notation describes a signed binary fullword. The value, [+|-]nnnnnnnnnnn, is a 1- to 10-decimal digit number preceded by an optional plus (the default) or minus sign. IPCS provides leading zeros if you enter fewer than ten digits.

*Example:*

\[ F'(L8) 124' \]

\[ H'[(Ln)][+]|-(nnnnn) \]

This notation describes a signed binary halfword. The value, [+|-]nnnnn, is a 1- to 5-decimal digit number, preceded by an optional plus (the default) or minus sign. IPCS provides leading zeros if you enter fewer than 5 digits.

*Example:*

\[ H'(L8) 75' \]

\[ P'p...' \]

This notation describes a picture string containing one to 256 characters. With picture strings you can enter the type of string to be found instead of the exact characters to be found. Each character “p” can be any of the following:

- Blank
- Alphabetic character
- Decimal digit

or it can be a symbol used to represent a class of characters, as follows:

**Symbol**

<table>
<thead>
<tr>
<th>Description of Class</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any character</td>
<td>=</td>
</tr>
<tr>
<td>Alphabetic characters</td>
<td>@</td>
</tr>
<tr>
<td>Numeric characters</td>
<td>#</td>
</tr>
<tr>
<td>Special characters</td>
<td>$</td>
</tr>
<tr>
<td>Non-blank characters</td>
<td>~</td>
</tr>
<tr>
<td>Invalid characters</td>
<td>.</td>
</tr>
<tr>
<td>Non-numeric characters</td>
<td>-</td>
</tr>
<tr>
<td>Lowercase alphabets</td>
<td>&lt;</td>
</tr>
<tr>
<td>Uppercase alphabets</td>
<td>&gt;</td>
</tr>
</tbody>
</table>

Use of picture strings results in either an equal or an unequal condition.

**Note:** Picture strings can be used only in a search argument or in a comparison. They cannot be used to specify:

- A PAD value on a COMPARE subcommand
- A MASK value on a COMPARE, EVALUATE, or FIND subcommand or on a FIND primary command
- A symbolic literal on a LITERAL subcommand

*Example 1:*

find p'aBc'

*Result:* IPCS finds the first occurrence of string aBc.

*Example 2:*
FIND P'¬>'

*Result:* IPCS finds the first occurrence of a string consisting of a non-blank character followed by an uppercase letter.

Q'Q...'  
This notation describes an ASCII character string containing one to 256 characters. The value, Q..., is subjected to editing as follows:

- Data entered manually from a terminal may be translated by the TSO/E Terminal I/O Controller.
- IPCS translates each pair of adjacent apostrophes into a single apostrophe.
- The FIND primary command accepts either ‘ABC’Q or Q’ABC’ as the same search value.
- The characters entered are interpreted as ISO-8 ASCII characters and are limited to those characters for which corresponding EBCDIC graphics are supported.

*Note:* Lowercase letters are not translated to uppercase when the search argument is formed.

*Example:*  
find Q'aBc'  
*Result:* IPCS finds the first occurrence of aBc.

quoted-string  
When the FIND primary command is used from the storage panel of IPCS browse, the character translation currently being employed determines how a quoted string is interpreted:

- If characters are being shown in EBCDIC, the quoted string is interpreted as a text string T't...'.
- If characters are being shown in ASCII, the quoted string is interpreted as an ASCII text string S'...'.

S'S...'
This notation describes ASCII text strings containing one to 256 characters. ASCII text strings are phrases without regard to case. Either uppercase or lowercase is processed.

Use of ASCII text strings results in either an equal or unequal condition.

*Note:* ASCII text strings may only be used in a search argument or a comparison. They CANNOT be used to specify:

- A pad value on a COMPARE subcommand.
- A MASK value on a COMPARE, EVALUATE, or FIND subcommand or an a FIND primary command.
- A symbolic literal on a LITERAL subcommand.

*Example:*  
find s'ABC'
*Result:* IPCS finds the first occurrence of any of the following possibilities:
abc  
Abc  
ABc  
ABC  
aBC  
abC
T't...' 
This notation describes text strings containing one to 256 characters. Text strings are phrases without regard to case. Either uppercase or lowercase is processed.

Use of text strings results in either an equal or an unequal condition.

Note: Text strings can be used only in a search argument or in a comparison. They cannot be used to specify:

• A PAD value on a COMPARE subcommand
• A MASK value on a COMPARE, EVALUATE, or FIND subcommand or on a FIND primary command
• A symbolic literal on a LITERAL subcommand

Example:
find t'ABC'

Result: IPCS finds the first occurrence of any one of the following possibilities:
abc
Abc
ABc
ABC
aBc
abC
aBC
AbC

word 
When the FIND primary command is used from the storage panel of IPCS browse, the character translation currently being employed determines how a word is interpreted:
• If characters are being shown in EBCDIC, the quoted string is interpreted as a text string T't...'.
• If characters are being shown in ASCII, the quoted string is interpreted as an ASCII text string S'...'.

You determine whether characters are shown in EBCDIC or ASCII by using the EBCDIC and ASCII primary commands.

X'xx...' 
This notation describes a hexadecimal string containing one to 256 characters. The value, xx..., must contain two hexadecimal digits for each byte described. For legibility, you can place one or more TSO/E separator characters between groups of hexadecimal digits, such as:

- Blanks (X'40')
- Commas (X'6B')
- Tabs (X'05')

Each group divided in this manner must describe one or more complete bytes.

* This notation (the asterisk), which is accepted only by the FIND primary command in the IPCS dialog, specifies the repetition of the same search value that was used on the preceding FIND primary command.
Symbols

When an IPCS subcommand accepts a literal value, the value can be entered as a symbol. The definition of the symbol and the data associated with the symbol are contained in the dump directory. You can use symbolic literals so that IPCS can manage many dumps and traces without having to allocate and open the dump and trace data sets frequently.

Defining a Symbol

Define a symbol using a LITERAL subcommand. For example:

```
literal a c'ABCDE'
```

If the EVALUATE subcommand requests a storage key for a symbolic literal, IPCS returns the FF value used when the storage key is not available.

**Note:** IBM does not recommend using a symbolic literal as the basis for indirect addressing. IPCS will accept such an indirect address and try to resolve it to the appropriate dumped central storage, but may not be able to resolve it depending on the dump and the local and global defaults in effect.

If you define a symbol based on a literal symbol, the resulting definition is an independent copy of the literal data. For instance:

```
literal a c'X'
remove b a
literal a c'Y'
```

This sequence leaves symbol A associated with C'Y' and symbol B associated with C'X', rather than C'Y'. This sequence is consistent with the following EQUATE subcommands, which leave symbol F with the same definition as symbol ASVT and symbol G with the same definition as symbol CVT:

```
equate f cvt
equate g f
equate f asvt
```

Referring to a Symbolic Literal

An IPCS command or subcommand refers to the name of the address space containing the literal as LITERAL and refers to the literal by its symbol. For example:

```
literal(a)
```

Location of a Symbol

IPCS treats each literal value as residing in the first 1 through 256 bytes of an address space that it shares with no other literals. Because an address space contains \(2^{31}\) bytes, most or all bytes in the address space for a symbolic literal are not available. The following sequence of subcommands associates symbol Y with an address space in which no bytes are available:

```
literal x c'Q'
equate y x position(10) length(10) character
```

Only the first byte of the address space was populated by the LITERAL subcommand. The EQUATE subcommand tries to define symbol Y with 10 bytes of storage that are not available.
Literal Values
Chapter 3. Data description parameter

You describe storage in a dump by using the data description (data-descr) parameter.

Parts of the data description parameter

The parts of the data-descr parameter are:
• An address (required when data-descr is explicitly specified on a subcommand)
  Types of addresses are:
  – Symbolic address
  – Relative address
  – Literal address
  – General-purpose register
  – Floating-point register
  – Indirect address
• Address processing parameters (optional)

<table>
<thead>
<tr>
<th>To describe an address in:</th>
<th>Specify the parameter:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute storage</td>
<td>ABSOLUTE</td>
</tr>
<tr>
<td>Virtual storage</td>
<td>ASID(asid) [CPU(cpu)</td>
</tr>
<tr>
<td>A data space</td>
<td>ASID(asid) DSPNAME(dspname) [SUMDUMP]</td>
</tr>
<tr>
<td>Physical block number</td>
<td>BLOCK(block-number)</td>
</tr>
<tr>
<td>Component data</td>
<td>COMPDATA(component-id)</td>
</tr>
<tr>
<td>Supplementary dump data</td>
<td>DOMAIN(domain-id)</td>
</tr>
<tr>
<td>The header record</td>
<td>HEADER</td>
</tr>
<tr>
<td>Relative byte address group number</td>
<td>RBA</td>
</tr>
<tr>
<td>Central storage</td>
<td>REAL [CPU(cpu)]</td>
</tr>
<tr>
<td>One of the CPU status records</td>
<td>STATUS [CPU(cpu)]</td>
</tr>
<tr>
<td>The physical block</td>
<td>TTR(ttr)</td>
</tr>
<tr>
<td>A dump source</td>
<td>ACTIVE, MAIN, STORAGE, DSNAMES(dname), DATASET(dname), FILE(ddname), DDNAME(ddname), or PATH(hfspath)</td>
</tr>
</tbody>
</table>

• An attribute parameter (optional)

<table>
<thead>
<tr>
<th>To describe an address in:</th>
<th>Specify the parameter:</th>
</tr>
</thead>
<tbody>
<tr>
<td>An area</td>
<td>AREA</td>
</tr>
<tr>
<td>A bit string</td>
<td>BIT or HEXADECIMAL</td>
</tr>
<tr>
<td>A character string</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>A signed binary number</td>
<td>SIGNED</td>
</tr>
<tr>
<td>An unsigned binary number</td>
<td>UNSIGNED</td>
</tr>
<tr>
<td>A pointer</td>
<td>POINTER</td>
</tr>
<tr>
<td>A module</td>
<td>MODULE</td>
</tr>
</tbody>
</table>
Data Description Parameter

<table>
<thead>
<tr>
<th>To describe an address in:</th>
<th>Specify the parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A control block</td>
<td>STRUCTURE</td>
</tr>
<tr>
<td>A floating point number</td>
<td>FLOAT</td>
</tr>
<tr>
<td>An instruction stream</td>
<td>INSTRUCTION</td>
</tr>
<tr>
<td>A packed decimal number</td>
<td>PACKED</td>
</tr>
<tr>
<td>A zoned decimal number</td>
<td>ZONED</td>
</tr>
</tbody>
</table>

- Array parameters (optional)

<table>
<thead>
<tr>
<th>To provide</th>
<th>Specify the parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>An array</td>
<td>ENTRIES(xx [:yy])</td>
</tr>
<tr>
<td>An array of dimension</td>
<td>DIMENSION(nnn) [ENTRY(xx)]</td>
</tr>
<tr>
<td>A single entity</td>
<td>SCALAR</td>
</tr>
</tbody>
</table>

- A remark parameter (optional)

<table>
<thead>
<tr>
<th>To provide</th>
<th>Specify the parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A comment about an address</td>
<td>REMARK('text')</td>
</tr>
<tr>
<td>No comment</td>
<td>NOREMARK</td>
</tr>
</tbody>
</table>

Address, LENGTH, and POSITIONS parameters

An address, which is required, and LENGTH and POSITIONS parameters, which are optional, specify the three properties of the data:

- An address is the logical origin of the data, the address passed between programs to indicate where it is and thus, the location at which the data is said to reside.

  Depending on the subcommand's syntax, address can be a positional or keyword parameter.

  An example of specifying address as a positional parameter is:
  ```
lst 54.% length(9) asid(22)
  ```

  An example of address as a keyword parameter is:
  ```
find address(54.%) length(9) asid(22)
  ```

  Address may be expressed as a single address, an address expression, or a range of addresses.

  **Note:** The DROPMAP, LISTMAP, and SCAN subcommands are exceptions to the rule that an address is required in a data description. These subcommands accept address processing parameters without an address and interpret that to mean all addresses contained within an address space.

- LENGTH is the number of bytes spanned by the data (or a single entry in an array); its size in IPCS terms.

- POSITIONS is the signed offset between the logical origin of the data and its physical origin.

  Where the offset is negative (as it is with system CVTs, RBs, TCBs, and UCBs), the data is said to have a prefix.
Data Description Parameter

If the address is a **positional parameter**, the syntax is as follows:

```
address[:address] [LENGTH(length)] [POSITIONS(position[:position])]
```

If the address is a **keyword parameter**, the syntax is as follows:

```
{ ADDRESS(address[:address]) }  
{ RANGE(address[:address]) }  
[ LENGTH(length) ]  
[ POSITIONS(position [:position ] ) ]
```

**address [: address]**

ADDRESS(address : address)

RANGE(address : address)

**address expression**

Specify the address as:

- A single address
- A range of addresses
- An address expression

A **single address** is a symbolic address, relative address, literal address, general-purpose register, floating-point register, or indirect address.

*Example:*

```
list +73
```

*Result:* LIST displays a relative address, X’73’ bytes beyond X, the current address.

A **range of addresses** is any pair of addresses, address expressions, and registers (general-purpose and floating-point), separated with a colon. A range of addresses includes both end-points of the range. If you specify a range of addresses and LENGTH, the length of the range overrides the LENGTH value.

*Example:*

```
scan range(7819b.:8019b.) asid(6)
```

*Result:* SCAN processes only the storage map entries for ASID 6 that originate between X’7819B’ and X’8019B’ inclusive.

An **address expression** is an address followed by any number of expression values. An address expression has the format:

```
address[[%?]...]±value[[%?]...]±value[[%?]...]
```

**address**

A symbolic address, relative address, literal address, indirect address, or general-purpose register. You cannot use floating-point registers (and it is not advisable to use general-purpose registers) in an address expression.

For any symbol that has a positive or negative origin point, be sure to use the +0 displacement for indirect addressing.

**value**

An address modifier that is either:

- A 1- to 19-digit decimal number followed by the letter N. The N may be in uppercase or lowercase.
- A 1- to 16-digit hexadecimal number that is not followed by a period.

Underscores may be used between pairs of hexadecimal digits to improve legibility.
Data Description Parameter

Value must be preceded by a plus (+) or a minus (-) sign and cannot be the first value in an address expression. You can use address modifiers with general-purpose registers but you cannot use address modifiers with floating-point registers.

Types of addresses
An address can be any one of the following types:

- Symbolic address
  A symbolic address is a symbol consisting of at least one and no more than 31 characters. The first character must be a letter or the following characters:
  $ (X'5B')
  # (X'7B')
  @ (X'7C')
  The same characters plus the decimal digits, 0 through 9, may be used for any of the remaining characters.

Notes:
1. A symbolic address provides a complete description of a block of storage to IPCS:
   - Address, LENGTH, and POSITIONS parameters
   - Address processing parameters
   - An attribute parameter
   - Array parameters
   - A remark parameter
2. A symbolic address can be defined and used by the same IPCS subcommand if the following conditions are met:
   - The symbolic address conforms to IPCS naming conventions. See "IPCS symbols" on page A-1 for a list of the IPCS naming conventions supported by the MVS/System Product. The diagnostic guides for other products that you have installed may supplement this list.
   - IPCS is able to associate the symbolic address with the type of AREA, MODULE, or STRUCTURE required by IPCS naming conventions. This will occur if, for example, you enter
     list ascb1
     or you enter
     list ascb1 structure(ascb)
   - It will not occur if you enter
     list ascb1 structure

Example:
list x

Result: LIST displays X, the current address.

- Relative address
  A relative address value can designate a maximum of 16 hexadecimal digits. Underscores (_) can be used as separators when the value is entered.
Data Description Parameter

- **Literal address**
  Before OS/390® Release 10, a literal address is a maximum of eight hexadecimal digits. If the initial digit is a letter A through F, the literal address **must end with a period**. Otherwise, the period can be omitted. The maximum address is ‘7FFFFFFF’.

  The following list explains valid literal address ranges.
  - If the address is absolute, real, or virtual, the address can range from 0 through $2^{64} - 1$.
  - If the address is in the status record, the address can range from 0 through 4095.
  - If the address is in the dump header record, the address can range from 0 through 4159.

  Example:
  ```
  WHERE fe2b8.
  ```
  Result: WHERE identifies the area in storage in which the address resides. Underscores (_) can be used as separators when the value is entered. IPCS accepts literal addresses beginning with a decimal digit without regard to the presence of a trailing period.

- **General-purpose register**
  A general-purpose register is designated as a decimal integer followed by an R. The decimal integer can range from 0 through 15.

  With OS/390 Release 10 and higher, 64-bits of general-purpose registers are recorded as part of an unformatted dump. When dumps are produced on OS/390 Release 10 on processors lacking support for the z/Architecture® instruction set and 64-bit registers, the fullword values actually available are prefixed with 32 bits of binary zeros.

  Example:
  ```
  list 0r:15r terminal
  ```
  Result: LIST displays the contents of all 16 general-purpose registers as they were at the time of the dump to the terminal.

- **Floating-point register**
  A floating-point register is designated as a decimal integer followed by a D for double precision. The decimal integer can be 0 through 15.

  Example:
  ```
  list 0d:6d
  ```
  Result: LIST displays the seven floating-point double precision registers in hexadecimal.

  **Notes:**
  1. Single precision floating point register notation, a decimal integer followed by an E, is accepted but interpreted as a reference to the corresponding double precision floating point register.
  2. Two or three decimal digit values ending in D or E are going to be interpreted as precise instances of floating point registers, so it is very important that you end an address with a period if you want it to be literal.

- **Indirect address**
  An indirect address is a symbolic, relative, or literal address, or a general-purpose register followed by a maximum of 255 percent signs (%)
or question marks (?). The address may include up to a maximum of 255
exclamation points (!) to indicate a 64-bit address value.

Each percent sign, question mark, or exclamation point indicates one level
of indirect addressing. Indirect addressing is a method of addressing in
which one area of dump data is used as the address of other dump data.
The address preceding the percent sign, question mark, or exclamation
point is used to locate a pointer in the dump as follows:

• If the address preceding the percent sign, question mark, or exclamation
point is a symbolic address that describes a pointer, the contents of the
pointer are retrieved from the dump.

• If the address preceding the percent sign, question mark, or exclamation
point is not a symbolic address that describes a pointer, IPCS verifies
that the addressed storage is acceptable for indirect addressing:
  – If the addressed storage begins on a fullword or doubleword
    boundary, IPCS accepts the fullword or doubleword pointer.
  – If not, IPCS checks the data type of the address storage. if the
    addressed storage has a data type of POINTER, IPCS accepts the
    pointer, even though it does not begin on a fullword or doubleword
    boundary.

Once IPCS accepts a pointer, it retrieves the contents of that pointer from
the dump. The pointer is interpreted to form an address as follows:

• If the address is followed by a percent sign, the pointer is interpreted as
  a 24-bit address. If a fullword pointer was retrieved from the dump,
  nonzero bits in the first byte are set to zeros to form the address.

• If the address is followed by a question mark, the pointer is interpreted
  as a 31-bit address. If a fullword pointer was retrieved from the dump,
  the initial bit is set to zero to form the address.

• If the address is followed by an exclamation point, the pointer is
  interpreted as a 64-bit address.

It is not recommended that you use registers in indirect addresses. For
compatibility with TSO/E TEST, general-purpose registers will be accepted
in an address expression, but the resolution of the expression by IPCS will
generally prove unsatisfactory. You cannot use floating-point registers in an
address expression.

LENGTH(length)
The length of the area beginning at the specified address. The length can be
specified in decimal (nnn), hexadecimal (X’xxx’), or binary (B’bbb’) notation.

The following list explains valid address length ranges.

• If the address is absolute, real, or virtual, the length can range from 1
  through 2**41.

• If the address is in the status record, the length can range from 1 through
  4096.

• If the address is in the dump header record, the length can range from 1
  through 4160.

If you specify the LENGTH parameter and a range of addresses, the length of
the range overrides the LENGTH value. If the length exceeds the upper limit
for an addressing mode, the length is adjusted to include the last valid address
for that addressing mode.

If you omit the LENGTH parameter, the subcommand uses the default length.
Data Description Parameter

**Note:** When STRUCTURE(cbname) attribute parameter is specified, IPCS can supply a preferred length that overrides the default length. See "Attribute parameters" on page 3-13 for more details.

**Example:**
equate abc a72f4. length(80) area

**Result:** EQUATE creates a symbol table entry for symbol ABC associating it with an 80-byte area beginning at X'A72F4'.

**POSITIONS(position[:position])**
The offset of the initial and, optionally, the final byte of the area. The offsets can be specified in signed decimal (+ | -nnn or F'+ | -Jnn'), signed hexadecimal (X'+ | -Jxxx'), signed binary (B'+ | -Jbb').

**Example 1:**
list 400. position(30) length(10) structure

**Result:** LIST displays locations X'41E' (decimal 1054) through X'427'. IPCS uses offset caption +0000001E for the line of storage displayed.

**Example 2:**
list asvt positions(512:519)

**Result:** LIST displays the cross section of the ASVT containing fields ASVTASVT and ASVTMAXU. The ending position is an alternate means to designate the length of the storage.

**Example 3:**
list +5 position(0) length(5)

**Result:** LIST performs the following steps:
1. The definition of the current symbol, X, is retrieved.
2. The POSITION(0) specification in conjunction with the explicit offset specification, +5, causes 5 to be added to the address of X before 0 is stored as a new offset.
3. The LENGTH(5) specification causes the updated definition of X to be stored with a length of 5 bytes.
4. The 5 bytes of storage are displayed.

This combination of explicit offset and the POSITION parameter can be used to move down (or up) within storage, in increments.

**Address processing parameters**

Address processing parameters are optional. They describe an address space within which the data to be processed resides.

**Note:** Address processing parameters DSNAME, FILE, BLOCK, and RBA are the only address processing parameters you can use when referencing VSAM data sets.
Data Description Parameter

[ ABSOLUTE
[ ASID(asid) [CPU(cpu)|NOCPU] [SUMDUMP] ]
[ ASID(asid) DSPNAME(dspname) [SUMDUMP] ]
[ BLOCK(block-number) ]
[ COMPDATA(component-id) ]
[ DOMAIN(domain-id) [CPU(cpu)] ]
[ HEADER ]
[ RBA {{[0]rba-group]} ]
[ REAL "[CPU(cpu)] ]
[ STATUS [CPU(cpu)] ]
[ TTR(ttr) ]
[ ACTIVE | MAIN | STORAGE ]
[ DSNNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(hfspath) ]

ABSOLUTE
The storage at the address or address range is in absolute storage in a system
dump or ACTIVE storage (that is, the LPAR in which IPCS is running).
Access to the absolute storage in ACTIVE requires read authority to facility
class BLSACTV SYSTEM. Without such authority, all ABSOLUTE storage are
treated as inaccessible.

ASID(asid)
The storage at the address or address range is in an address space or a data
space. IPCS accesses the storage differently, depending on the type of
information source:
• For dumps, IPCS accesses address spaces using a valid ASID.
• For ACTIVE storage, IPCS accesses storage from the system where it is
  executing on demand as an enabled application. Access to sensitive storage
  is restricted by proper authority to facility classes BLSACTV.ADDRSPAC or
  BLSACTV.SYSTEM. See “Facility class authority” on page 3-11 for more
discussions about the facility class authority.
• For stand-alone dumps, IPCS simulates dynamic address translation or
  central storage prefixing, depending on the parameter you specify. (See the
descriptions for the CPU and NOCPU parameters.)

The ASID can range from 1 through 65,535. You can specify the ASID in
decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’).

Example:
equate abc a72f4. asid(1) length(80) area

Result: EQUATE creates a symbol table entry for symbol ABC, associating it
with an 80-byte area beginning at X’A72F4’. ASID(1) indicates that this address
is in virtual storage and IPCS simulates dynamic address translation.

BLOCK(block-number)
The storage at the address or address range is in physical block number
“block-number” as follows:
BLOCK(0) is the first physical block.
BLOCK(1) is the second physical block.
BLOCK(2) is the third physical block.
BLOCK(3) is the fourth physical block.
...
The block number can range from 0 through 2^{24}-1. You can specify the block
number in decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’).
Data Description Parameter

For VSAM data sets, BLOCK(0) is the first control interval, BLOCK(1) is the second, and so on.

**COMPDATA(component-id)**

The storage supplied as part of a dump to facilitate analysis of a specific component. Use the LISTDUMP subcommand to find the COMPDATA records available in a dump. For example, the stand-alone dump program can produce the following COMPDATA records:

**AMDSAMSG**

Requests display of messages displayed at the operator's console during the dumping process.

**AMDSA001 - AMDSA005**

Request display of self-dump information from stand-alone dump when it detects errors in its own processing.

**AMDSA009**

Request display of internal control blocks used by stand-alone dump during its processing.

See z/OS MVS Diagnosis: Tools and Service Aids for more information about stand-alone dump COMPDATA records.

Records written by a stand-alone dump use `component-ids` that begin with the same prefix characters as that component's module names ("AMDSA"). This is true for all IBM-supplied components.

**CPU(cpu)**

The storage within the CPU address that provides the context for the ASID, DOMAIN, REAL, or STATUS parameter. The CPU parameter applies only to stand-alone dumps.

- For the ASID and REAL parameters, this is the processor whose prefix register is used when IPCS simulates prefixing.
- For the STATUS parameter, this is the processor whose registers were saved by a store-status operation during the dumping of the operating system.

The CPU address can range from 0 to 63 and may be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...'). If you specify this parameter and omit ASID, REAL, and STATUS, the subcommand uses the default ASID. See “Facility class authority” on page 3-11 for more details.

**DOMAIN(identifier)**

The dump storage that supplements the storage pages that record system status. The valid `domain-ids` are:

**DOMAIN(VECTOR)**

The vector registers recorded by stand-alone dump. Stand-alone dumps might contain vector registers that are for each processor in the configuration. If you do not use the CPU parameter to specify the address of the CPU containing vector records you want, IPCS uses a default CPU address.

**DOMAIN(SDUMPBUFFER)**

The diagnostic data in the SDUMP buffer. The requester of a system-initiated dump puts the data in the SDUMP buffer.

**DOMAIN(SUMDUMP)**

The highly volatile diagnostic data that is useful for problem determination.
Data Description Parameter

DSPNAME (dspname)
The data space dspname that is associated with the specified ASID. If the dump is not a stand-alone dump, and the DSPNAME and SUMDUMP parameters are specified or are the default, IPCS accesses only that data space information which was collected in DOMAIN(SUMDUMP) records.

As of z/OS V1R9, IPCS users can access data spaces via ACTIVE storage:
• Without special authority, the data spaces that are visible to an authorized application can be accessed.
• Authority to the facility classes BLSACTV.ADDRSPAC and BLSACT.SYSTEM can provide access to the data spaces that are not directly accessible by an authorized application.

HEADER
The storage at the address or address range is in the header record for a system dump or ACTIVE storage. When you use this parameter, the subcommand accesses data in the header record from offset 0. That is, the subcommand processes data in the header record at the address you specify.

NOCPU
The storage at the address or address range is in virtual storage in a system stand-alone dump. IPCS is to simulate dynamic address translation and use the results to directly access absolute storage without the use of prefix registers.

If you specify the NOCPU parameter and omit ASID, the subcommand uses the default ASID.

RBA([0]rba-group]
The storage at the address or address range is in relative byte address group number "rba-group." Each relative byte address group consists of up to 2^31 bytes from a data set as follows:
   RBA(0) contains the first 2^31 bytes.
   RBA(1) contains the second 2^31 bytes.
   RBA(2) contains the third 2^31 bytes.
   RBA(3) contains the fourth 2^31 bytes.
   ...

Note: IPCS interprets RBA(0) (or just RBA) as the first 2^64 bytes of a data set.

The group number can range from 0 through 2^{31}-1. If the group number is omitted, it defaults to 0. You can specify the group number in decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’).

For VSAM data sets, IPCS masks the boundaries between control intervals, allowing them to be referenced as part of a single address space.

REAL
The storage at the address or address range is in central storage in a system stand-alone dump. IPCS is to simulate prefixing for the specified or current default CPU.

If you specify the REAL parameter and omit the CPU parameter, the subcommand uses the default CPU.

STATUS
The storage at the address or address range is in one of the CPU status records in a system stand-alone dump. Stand-alone dumps contain a CPU status record for each CPU that was active on the system at the time of the dump. The CPU status record for a particular CPU contains an image of a 4096-byte prefixed save area (PSA) just after a STORE STATUS operation was performed from the
CPU to the PSA. The status information stored by the STORE STATUS operation includes the current PSW and the general registers. IPCS supports access to each CPU’s status as a 4096-byte CPU status address space.

When you use STATUS, the parameter accesses data in the status records from offset eight. That is, the parameter processes data in the status record eight bytes beyond the address you specify. See the AMDDATA mapping macro for more information.

If you specify this parameter and omit CPU, the subcommand uses the default CPU.

*Example:*

```
list 100 status cpu(0) length(8)
```

*Result:* LIST displays the PSW that is placed in the store status record at X’100’ of a stand-alone dump.

**SUMDUMP**

The dump storage containing the DOMAIN(SUMDUMP) records, provided that the dump is not a stand-alone dump. For dumps other than stand-alone dumps, the SUMDUMP parameter can be specified or may be the default.

*Note:* The SUMDUMP parameter does not apply to stand-alone dumps.

**TTR(ttr)**

The storage at the address or address range is in the physical block that has the relative track and record address of “ttr”. The value of ttr can range from 0 through $2^{24}$-1. You can specify the ttr in decimal, hexadecimal (X’xxx’), or binary (B’bbb’).

**ACTIVE or MAIN or STORAGE**

**DSNAME(dsname) or DATASET(dsname)**

**FILE(ddname) or DDNAME(ddname)**

Specifies the source that contains the address space or address range. If one of these parameters is not specified, IPCS uses your current source.

*Note:* Do not use these parameters for:

- Volatile common or private storage
- Prefixed storage

**ACTIVE or MAIN or STORAGE** specifies that the address or address range is in the central storage in which IPCS is currently running.

A header record similar to those used for system dumps is supplied by IPCS to enable common dump analysis programs to function.

Storage is accessed incrementally on demand, and IPCS generally remains enabled. As a result, ACTIVE storage might be subject to frequent changes, which can prevent the analysis programs from producing useful results.

ABSOLUTE, ASID, DSPNAME, and HEADER keywords are supported. Access to sensitive storage areas, such as ABSOLUTE, is limited using facility classes. When the user does not have the authority, the access attempts are handled as though the storage in question was not included in a dump.

With no special authority, IPCS can access the following storage:

- The common and private storage in its own ASID visible to a key 8 application
- The data spaces owned by its own ASID and visible to a key 8 application

Before z/OS V1R9 no data space access was supported.
With read authority to facility class BLSACTV.ADDRSPAC, IPCS can look at the following storage (in addition to those storage areas it can access with no special authority):

- The common and private storage visible to a key 0 application
- All data spaces owned by its own ASID

Before z/OS V1R9 no data space access was supported.

With read authority to facility class BLSACTV.SYSTEM, IPCS can look at the following storage (in addition to those storage areas it can access with no special authority):

- The ABSOLUTE storage
- Other ASIDs
- The data spaces owned by other ASIDs

BLSACTV.SYSTEM support was added in z/OS V1R9.

**Note:** IPCS artificially attributes CADS ownerships to ASID(1) as it also does for the page frame table space, ASID(1) DSPNAME(IARPFT). Consistent with this perspective BLSACTV.SYSTEM authority is required to access these data spaces.

**DSNAME** or **DATASET** specifies that the address or address range is in the cataloged data set *dsname*.

For VSAM data sets, you can:

- Access the data portion of the cluster by:
  - Specifying the cluster data set name for *dsname*
  - Specifying the optional data portion data set name for *dsname*
  - Specifying *dsname* in pseudo-PDS notation, providing a member name of “data”, as in
    ```
    DSNAMEx(vsam.cluster.dsname(data))
    ```

- Access the index portion of the cluster by:
  - Specifying the optional index portion data set name for *dsname*
  - Specifying *dsname* in pseudo-PDS notation, providing a member name of “index”, as in
    ```
    DSNAMEx(vsam.cluster.dsname(index))
    ```

**FILE** or **DDNAME** specifies that the address or address range is in the data set *ddname*.

**z/OS UNIX files** can be referenced with this notation.

- Those **z/OS UNIX files** whose size is a multiple of 4160 bytes will be treated as **z/OS unformatted dumps**.
- **No trace formatting support** is provided for **GTF** or component traces that have been copied into **z/OS UNIX files**.
- **RBA access** is supported for any **z/OS UNIX file**.

For VSAM data sets, allocate the data or index portions of the VSAM cluster to use the **FILE** parameter in pseudo-PDS notation. For example, specify FILE(vsam.cluster.dsname(data)) or FILE(vsam.cluster.dsname(index)).

Specifying the name of the required portion with the **DSNAME** parameter instead avoids allocating the portions.

**PATH(hfspath)**

Specifies a valid path name.
Data Description Parameter

- You can reference path names directly. There is no need to associate a path with a ddname before asking IPCS to process a z/OS UNIX file path.
- Fully qualified path names are limited to 44 characters. Enclosing apostrophes or quotation marks are not counted toward the limit.
- You can use partially-qualified path names. IPCS will determine the fully-qualified names.

You can enter PATH as follows:

`PATH('/pathname')`

`PATH("/pathname used in IPCS dialog")`

You can always enter path names within apostrophes. Quotation marks can be used as an alternative to apostrophes when supplying a source name to the defaults or browse options of the IPCS dialog. The rules for entering a path name within quotation marks are standard:

- If the path name contains an apostrophe and you used that punctuation to delimit the name, two adjacent apostrophes need to be entered.
- If the path name contains a quotation mark and you used that punctuation to delimit the name, two adjacent quotation marks need to be entered.

Always use quoted string notation when your path name contains blanks, commas, horizontal tabulation characters (EBCDIC X'05'), apostrophes (single quotation marks), or quotation marks.

`PATH((x/y/z))`

Quoted string notation is not always required. You can enter most path names without enclosing them with punctuations.

`PATH(partially-qualified-name)`

IPCS accepts existing z/OS UNIX file paths that can be qualified when they are entered, as if the fully-qualified name had been entered.

**Note:** Path names are case-sensitive. Path names “/ABC”, “Abc”, and “abc” refer to three different paths.

### Attribute parameters

Attribute parameters are optional. They designate the type of data and thus, the way IPCS should format the storage in which the data resides. If you omit all attribute parameters, the default is AREA.

```
[ AREA[(name)] ]
[ BIT | B | HEXADECIMAL | X ]
[ CHARACTER | C ]
[ FLOAT ]
[ INSTRUCTION EP(x'addr') ]
[ MODULE[(name)] ]
[ PACKED ]
[ POINTER | PTR ]
[ SIGNED | F ]
[ STRUCTURE[(cbname)] ]
[ UNSIGNED ]
[ ZONED ]
```

**AREA[(name)]**

The storage indicated by the address or in the range is an area of storage (a subpool, a buffer, and so on.) that is not a module or control block.
If you display or print the area, each line contains four or eight words, depending on line width, in hexadecimal format followed by their character equivalent. This parameter is frequently used when creating a symbol table entry for the storage at the address or in the address range.

If you specify a name, IPCS automatically creates a storage map entry for it. The name can be a maximum of 31 alphanumeric characters and the first character must be alphabetic.

Example:
equate abc a72f4. asid(1) length(80) area

Result: EQUATE creates a symbol table entry for symbol ABC associating it with an 80-byte area beginning at X'A72F4'. ASID(1) indicates that this address is in virtual storage.

**BIT or HEXADECIMAL**
The storage indicated by the address or in the address range is bit string data. If you display or print the data, it is shown in hexadecimal format. B or X is the abbreviation.

**CHARACTER**
The storage indicated by the address or in the address range is character string data. If you display or print the data, it is shown in character format. C is the abbreviation.

Example:
list abc+80n length(20) c

Result: LIST displays a 20-byte field following a symbolic address in character format.

**FLOAT**
The storage indicated by the address or in the address range is a floating point number or numbers. If you display or print the data, it is shown as a string of hexadecimal digits.

If you specify LENGTH, it must be 4, 8 or 16. If you specify any other value, the subcommand changes the attribute to AREA. If you omit the length parameter, the subcommand uses the length associated with the symbol, if you used one, or the default length. If this length is not 4, 8 or 16, the subcommand changes the length to the nearest shorter length, if possible, or to 4 otherwise.

**INSTRUCTION EP(x'addr')**
The storage indicated by the address or in the address range is an instruction stream. If you display or print the data, the output format depends on HLASM services that provide formatting support. Only contiguous streams of instructions guarantee correct formatting.

Example: Use the LIST subcommand to display the instruction stream at the given address EP9:
list ep9 instr

Result:

EP9
LIST 02CC48. ASID(X'01AA') LENGTH(X'69') INSTRUCTION
0002CC48 | 90EC 000C | STM R14,R12,X'C'(R13)
0002CC4C | 18CF | LR R12,R15
0002CC4E | 41F0 0000 | LA R15,X'0',
0002CC52 | 5800 C19C | L R0,X'19C'(,R12)
0002CC56 | 5890 1000 | L R9,X'0'(,R1)
0002CC5A | 58F0 93E4 | L R15,X'3E4'(,R9)
Data Description Parameter

You can use optional keyword EP() to provide IPCS with an address of the beginning of a module. In this case an additional column with relative address of every instruction from the beginning of the module will be displayed.

Example: Use the LIST subcommand to display the instruction stream at the given address 000007AE and the address of the beginning of the module 000007B8:

```
list 000007AE length(x'64') instr EP(000007B8)
```

Result:

```
LIST 07AE. ASID(X'0005') LENGTH(X'64') INSTRUCTION
000007AE | ___ | 1850 | SR R5,0
000007B0 | ___ | 55F0 07E4 | CL R15,X'7E4'
000007B4 | ___ | 4740 078C | BC X'4',X'78C'
000007B8 | 0000 | 41F0 0030 | LA R15,X'30'
000007BC | 0004 | 5F00 07E0 | AL R15,X'7E0'
000007D0 | 0008 | ACFB 04A3 | STNSM X'4A3',X'FB'
000007D4 | 000C | 5F00 F000 | L R15,X'0'(,R15)
000007D8 | 0010 | 0CEF | BASSM R14,R15
000007DA | 0012 | AD04 04A3 | STOSM X'4A3',X'04'
000007E0 | 0016 | 18E0 | LR R14,0
000007F0 | 0018 | 0B0E | BSM 0,R14
000007FC | 001A LENGTH(X'40')==>Displayed as AREA 000007D2 LENGTH(X'60')==>All bytes contain X'00'
```

**MODULE[(name)]**

The storage indicated by the address or in the address range is a module. If you display or print the data, each line contains four or eight words, depending on line width, in hexadecimal format followed by their character format. This parameter is frequently used when creating a symbol table entry for the storage indicated by the address or in the address range.

If you omit the name, the storage is given the attribute of MODULE to distinguish it from AREA and STRUCTURE.

If you specify a name, IPCS automatically creates a storage map entry for it. The name can be a maximum of 31 alphanumeric characters and the first character must be alphabetic.

**PACKED**

The storage indicated by the address or in the address range is a signed packed decimal number or numbers. If you display or print the data, it is shown as a string of hexadecimal digits.

If you specify LENGTH, it must be 1 through 16. If you specify any other value, the subcommand changes the attribute to AREA. If you omit the length parameter, the subcommand uses the length associated with the symbol, if you used one, or the default length. If this length is greater than 16, the subcommand changes the length to 16.
Data Description Parameter

**POINTER**
The storage indicated by the address or in the address range is a pointer or pointers. If you display or print the data, it is shown in hexadecimal format.

If you specify LENGTH, it can range from 1 through 4. If you specify any other length, the subcommand changes the attribute to AREA.

If you omit the length, the subcommand uses the length associated with the symbol, if you used one, or the default length. If this length exceeds four, the subcommand uses a length of four. PTR is the abbreviation.

**SIGNED or F**
The storage indicated by the address or in the address range is a signed binary number or numbers. If you display or print the data, it is shown as a signed number or numbers translated to decimal.

If you specify LENGTH, the length must be two or four. If you specify any other value, the subcommand changes the attribute to AREA.

If you omit the length parameter, the subcommand uses the length associated with the symbol, if you used one, or the default length. If this length is not two or four, the subcommand changes lengths of one or three to two and changes lengths greater than four to four. F is the alias.

**STRUCTURE[(cbname)]**
The storage indicated by the address or in the address range is a control block. If you display or print the data, each line contains four or eight words, depending on line width, in hexadecimal format followed by their character format. This parameter is frequently used when creating a symbol table entry for the storage indicated by the address or in the address range.

If you omit cbname, the storage is given the attribute STRUCTURE to distinguish it from AREA and MODULE.

If you specify a cbname, IPCS automatically creates a storage map entry for it to assess whether the instance of STRUCTURE(cbname) that you have identified is a usable one, and to supply a preferred length that overrides the default length. (An explicit LENGTH or range supplied in the data description can, in turn, override the preferred length.) The following sources of this information are consulted, selecting the first one listed.

1. A scan exit routine is used.
2. A model is used if one is available and the model has described a control block identifier and a control block length.
3. A table of z/OS data area lengths is consulted regarding a default data length.
4. The default data length established by SETDEF is used.

The cbname can be a maximum of 31 characters and the first character must be alphabetic.

The CBFORMAT subcommand requires specification of the STRUCTURE parameter, except with its own MODEL and FORMAT parameters. The CBSTAT subcommand always requires the STRUCTURE(cbname) parameter. The parameter may be omitted for either, however, if the referenced symbol already exists in the symbol table and if the referenced symbol contains the attribute STRUCTURE(cbname).

The CBSTAT subcommand can use another value, STORESTATUS, in place of cbname. See “CBSTAT subcommand — obtain control block status” on page 5-35 for a description and an example.
Data Description Parameter

Example 1:

cbstat 7fa030. structure(tcb)

Result: CBSTAT displays the status for the TCB control block at the given address.

Example 2:
equate mytcb 522c0. structure(tcb)

Result: EQUATE explicitly verifies that the storage at X’522C0’ is a TCB and makes a symbol table entry for MYTCB and a storage map entry for location X’522C0’. In verifying the TCB, IPCS checks various pointers in the TCB to other control blocks, such as RBs, CDEs, and so on. In the process, these control blocks may also be validated and entered in the storage map but not in the symbol table.

UNSIGNED

The storage indicated by the address or in the address range is an unsigned binary number or numbers. If you display or print the data, it is shown as an unsigned number or numbers translated to decimal.

If you specify LENGTH, it can range from one through four. If you specify any other length, the subcommand changes the attribute to AREA.

If you omit the length, the subcommand uses the length associated with the symbol, if you used one, or the default length. If this length exceeds four, the subcommand uses a length of four.

ZONED

The storage indicated by the address or in the address range is signed zoned decimal number or numbers. If you display or print the data, it is shown as a string of hexadecimal digits.

If you specify LENGTH, the length must be 1 through 31. If you specify any other value, the subcommand changes the attribute to AREA. If you omit the length parameter, the subcommand uses the length associated with the symbol, if you used one, or the default length. If this length is greater than 31, the subcommand changes the length to 31.

Array parameters

Array parameters are optional. They indicate whether the data consists of a single item (SCALAR) or consists of adjacent, similar items (ENTRIES).

ENTRY(\texttt{xx[:yy]}) or \texttt{ENTRIES(\texttt{xx[:yy]})}

The storage indicated by the address or in the address range is an array. You specify the number of elements in the array with the values \texttt{xx} and \texttt{yy}. The value \texttt{xx} must be less than or equal to \texttt{yy}. These values can range from -2^{31} to 2^{31}-1 and can be specified using signed decimal (\texttt{[+|-]nnn}), hexadecimal (X’[+|-]xxx’, or binary (B’[+|-]bbb’). (Plus is the default.) The size of the total array is the length of storage in the specified address range or specified with the LENGTH parameter, multiplied by the number of array elements.

The number of elements in the array can range from -2^{63} to 2^{63}-1. The difference between the lower and the upper values can be no more than 15 decimal digits.
If you specify an array whose size exceeds the upper limit for the addressing mode, the subcommand changes the array to a scalar and adjusts its length to include the last valid address for that addressing mode. If you specify ENTRY or ENTRIES and SCALAR, the subcommand uses the SCALAR parameter and ignores ENTRY or ENTRIES.

Example:
```plaintext
list 7FFFD018. length(4) entries(6:10)
```

Result: Assuming that you have located a segment table at X'7FFFD000', LIST displays five segment table entries beginning at X'7FFFD018' (each segment table entry is four bytes). The total length of the five entries is 20 bytes.

**DIMENSION(nnn) or MULTIPLE(nnn)**

The storage indicated by the address or in the address range is an array of dimension nnn. The number nnn can be a maximum of $2^{31}$ and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...'). Each array element occupies the length of storage in the specified address range or the length specified with the LENGTH parameter. The total size of the array is the size of an element, multiplied by nnn. The dimension may be no longer than 15 decimal digits.

If you specify an array whose size exceeds the upper limit for the addressing mode, the subcommand changes the array to a scalar and adjusts its length to include the last valid address for that addressing mode.

Example:
```plaintext
equate sgt001 5d7c00. absolute length(4) dimension(256)
```

Result: Assuming that the master segment table is located at X'5D7C00' in absolute storage with a length of 4 and a dimension of 256, EQUATE defines the master segment in the symbol table with these attributes.

**SCALAR**

The storage indicated by the address or in the address range is a single entity with non-repeating fields. If you omit all array and scalar parameters, the default is SCALAR.

If you specify SCALAR and either ENTRY or ENTRIES, the subcommand uses the SCALAR parameter and ignores ENTRY or ENTRIES.

Example:
```plaintext
list a72f4. asid(1) length(x'50') area scalar
```

Result: LIST displays the storage as a single entity of non-repeating fields, beginning at the absolute address X'A72F4' for a length of 80 (X'50') bytes.

### Remark parameters

Remark parameters are optional. They associate a description with the data consisting of up to 512 characters of text.

```plaintext
[ REMARK('text') | NOREMARK ]
```

**REMARK(text)**

A textual description of the storage indicated by the address or in the address range. The description must be entered within apostrophes, and any apostrophes which appear within the description must be paired. The text can be a maximum of 512 characters. The remark is stored in the symbol table.

Example:
Data Description Parameter

equate abc a72f4. asid(1) length(80) area scalar +
   remark('input params from EXEC statement')

Result: IPCS creates a symbol table entry for the symbol ABC. EQUATE
associates the entry with an 80-byte area beginning at the absolute address,
X'A72F4'. The ASID(1) indicates that this address is in virtual storage and IPCS
simulates dynamic address translation for ASID(1); AREA indicates that the
symbol is neither a module nor a control block; SCALAR indicates that the
symbol is a single block of storage, not an array; REMARK is your description
of the 80-byte area.

NOREMARK
No textual description is to be associated with the storage.
This parameter may be used when equating a new symbol to one previously
defined. It will prevent IPCS from copying the remark text stored with the
existing symbol.

Example:
equate abc+73 asid(1) length(80) area scalar +
   noremark

Result: Assuming symbol, ABC, already exists in the symbol table, EQUATE
overlays the new address and attributes for ABC but does not delete the
existing remark.
Data Description Parameter
Chapter 4. TSO/E commands

This chapter describes the TSO/E commands that perform IPCS functions. It also describes those TSO/E commands that have special considerations when they are entered from an IPCS session.

Entering TSO/E commands

The following TSO/E commands can be processed at any time during a TSO/E session. Except for the IPCS command, which starts an IPCS session, you can also run TSO/E commands during an IPCS session.

To run a TSO/E command whose name does match an IPCS subcommand, use the IPCS subcommand named TSO (see “TSO subcommand — run a TSO/E command” on page 5-293). To run a TSO/E command whose name does not match an IPCS subcommand, type the command and press ENTER.

Task directory of TSO/E commands for IPCS

The following table identifies the TSO/E commands by the tasks they perform:

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin an IPCS session</td>
<td>“IPCS command — start an IPCS session” on page 4-3</td>
</tr>
<tr>
<td>Identify libraries of CLISTs and REXX EXECs</td>
<td>“ALTLIB command — identify libraries of CLISTs and REXX EXECs”</td>
</tr>
<tr>
<td>Initialize a dump directory</td>
<td>“IPCSDDIR command — initialize a user or sysplex dump directory” on page 4-6</td>
</tr>
<tr>
<td>View dump titles</td>
<td>“SYSDSCAN command — display titles in dump data sets” on page 4-7</td>
</tr>
<tr>
<td>Pass control to TSO command processors</td>
<td>“BLS9 command — session of TSO commands” on page 4-2</td>
</tr>
<tr>
<td>Pass control for a System/370 interface</td>
<td>“BLS9CALL command — call a program” on page 4-3</td>
</tr>
</tbody>
</table>

ALTLIB command — identify libraries of CLISTs and REXX EXECs

Use the ALTLIB command to identify libraries of CLISTs or REXX EXECs.

The function, operands, and syntax of the ALTLIB command are the same as those documented in z/OS TSO/E Command Reference. However, the following special considerations apply for using ALTLIB in an IPCS session.

Using ALTLIB in the IPCS Dialog

When you activate the IPCS dialog for an ISPF logical screen, the system creates an ALTLIB environment for IPCS that will be used whenever you ask IPCS to process a CLIST or REXX EXEC. This ALTLIB environment is separate from the following ALTLIB environments:
- The ALTLIB environment maintained by ISPF
- The ALTLIB environment maintained by another IPCS dialog logical screen
- The ALTLIB environment used in IPCS line mode
TSO/E command ALTLIB

To display or update the ALTLIB environment for the IPCS dialog logical screen, use the following command with appropriate operands:

IPCS ALTLIB

You can also enter ALTLIB without the IPCS prefix from option 4 of the IPCS dialog. You cannot use the QUIET option of the ALTLIB command. The QUIET option requires ISPF services, which are not made available to TSO/E commands by IPCS.

Changes that you make to the ALTLIB environment for that logical screen will remain in effect until the next ALTLIB command is entered or until you exit the IPCS dialog.

Note: The following command is a request to display or update the separate ALTLIB environment maintained by ISPF, not the ALTLIB environment maintained by the IPCS dialog:

TSO ALTLIB

Using ALTLIB in IPCS line mode or batch mode

When you use IPCS in line mode or batch mode, IPCS continues to use the same ALTLIB environment in effect when it received control. ALTLIB commands entered before the use of IPCS remain in effect. ALTLIB commands entered during the IPCS session will display or update this environment. This ALTLIB environment is not affected by ending IPCS.

BLS9 command — session of TSO commands

Use the BLS9 command to pass control to a succession of unauthorized TSO command processors. A “temporary steplib” can be specified for the duration of the BLS9 command session.

Authorized TSO commands are supported through linkage that ignores any TASKLIB data sets in effect for unauthorized commands.

Related subcommand

END

Syntax

BLS9

[ TASKLIB(dsname ...) ]

[ TEST | NOTEST ]

Operands

TASKLIB(dsname ...)

TASKLIB(dsname) specifies a list of load module libraries to be searched for unauthorized command processors invoked during the BLS9 session and for any modules the unauthorized command processors invoke using system-aided linkages.

TEST

NOTEST

TEST specifies that any ABEND that occurs during a BLS9 session is to be permitted to continue so that the TSO TEST command can be used.

Note: TSO TEST and TSO TMP will describe the situation as “BLS9 ENDED DUE TO ERROR+” whether the ABEND occurred in BLS9 command processing or in the processing of a command invoked by the BLS9 command.
NOTEST specifies that the BLS9 command is to intercept and briefly diagnose any ABEND that occurs during a BLS9 session, allowing a SYSABEND, SYSDUMP, or SYSUDUMP data set to be produced to document the error but blocking the use of TSO TEST.

**BLS9CALL command — call a program**

Use the BLS9CALL command to pass control to a processing program that expects the interface established by the IBM® System/370 standard linkage conventions. Such processing programs include assemblers, compilers, and data set utilities among others.

**Related commands**
- ATTCHMVS REXX host command environment
- CALL command of the z/OS TSO/E element
- CALLMVS REXX host command environment
- JCL EXEC PGM=program

**Syntax**

```
BLS9CALL program [ parm ]
[ HEADING(heading) | TITLE(title) | NOHEADING | NOTITLE ]
[ LIBRARY(library ...) | NOLIBRARY ]
[ MEMBER(member) ]
[ PAGE(page) ]
[ STATUS | NOSTATUS ]
[ SYSLIB(syslib) ]
[ SYSLIN(syslin) ]
[ SYSLMOD(syslmod) ]
[ SYSPRINT(sysprint) ]
[ SYSPUNCH(syspunch) ]
[ SYSTEM(systerm) ]
[ SYSUT1(sysut1) ]
[ SYSUT2(sysut2) ]
[ SYSUT3(sysut3) ]
[ SYSUT4(sysut4) ]
```

**Operands**

**program**

program specifies the 1-8 character name of the command processor to be given control. The program can reside in a library specified on the BLS9CALL command, the job pack area, the logon procedure steplib, the link pack area, or the system link library.

**parm**

Specifies a character string to be passed to the processing program. Enclose the character string with apostrophes. If not specified, the default is a null string.

**HEADING(heading)**

**TITLE(title)**

**NOHEADING**

**NOTITLE**

Specifies the heading or title to be passed to the processing program. Enclose the heading or the title in apostrophes.

**LIBRARY(library ...)**

**NOLIBRARY**

Specifies the libraries to be searched before the logon procedure steplib and
TSO/E command BLS9CALL

the system link library when searching for an unauthorized program and any modules it invokes using system-aided linkages.

Note: These libraries are not searched when an authorized program is invoked.

MEMBER(member)

Specifies a member of the SYSLMOD library. The member is typically an argument passed as a parameter to a linkage editor.

PAGE(page)

Specifies a page number to be passed to the processing program.

STATUS

NOSTATUS

Specifies whether the completion status of the processing program is to be displayed if the program terminates without an abend. (On abend, the status always is displayed.)

SYSLIB(syslib)

Specifies the file name to be passed to the processing program and used instead of SYSLIB.

SYSLIN(syslin)

Specifies the file name to be passed to the processing program and used instead of SYSLIN.

SYSLMOD(syslmod)

Specifies the file name to be passed to the processing program and used instead of SYSLMOD.

SYSPRINT(sysprint)

Specifies the file name to be passed to the processing program and used instead of SYSPRINT.

SYSPUNCH(syspunch)

Specifies the file name to be passed to the processing program and used instead of SYSPUNCH.

SYSTEM(systerm)

Specifies the file name to be passed to the processing program and used instead of SYSTEM.

SYSUT1(sysut1)

Specifies the file name to be passed to the processing program and used instead of SYSUT1.

SYSUT2(sysut2)

Specifies the file name to be passed to the processing program and used instead of SYSUT2.

SYSUT3(sysut3)

Specifies the file name to be passed to the processing program and used instead of SYSUT3.

SYSUT4(sysut4)

Specifies the file name to be passed to the processing program and used instead of SYSUT4.
IPCS command — start an IPCS session

Use the IPCS command to start an IPCS session. IPCS is a TSO/E command that initializes the IPCS environment. Once the IPCS command is processed, you may use the IPCS subcommands. Before running the IPCS command, you must allocate a dump directory.

Related subcommands
END
SETDEF

Syntax

IPCS

[ PARM(nn|00) | NOPARM ]
[ TASKLIB(dsname) | NOTASKLIB ]

Operands

PARM(nn|00)
NOPARM

PARM(nn) specifies the member of parmlib that IPCS uses as its initialization parameters for this session. The first six characters of the member name are “IPCSPR” and nn is the 2-digit decimal number that is appended to it. When specifying the number, a leading zero is optional.

The IPCSPRnn member specifies parameters for problem management and data set management facilities. See z/OS MVS Initialization and Tuning Reference for the syntax of the IPCSPRnn parmlib member.

NOPARM specifies that no IPCSPRnn member of parmlib should be accessed for this IPCS session. If NOPARM is specified, IPCS facilities for problem analysis may be used during the session, but those for problem management and data set management may not be used.

The default is PARM(00), which causes IPCSPR00 to be used.

TASKLIB(dsname)
NOTASKLIB

TASKLIB(dsname) specifies a list of load module libraries to be searched for analysis programs. The libraries must be cataloged and will be searched in the order entered.

NOTASKLIB specifies that only the standard load module libraries should be searched for analysis programs during the IPCS session.

For example, request that IPCS search the load libraries IPCSU1.DEBUG.LOAD and IPCSU1.DIAGNOS.LOAD, enter:

ipcs tasklib('ipcsu1.debug.load' 'ipcsu1.diagnos.load')

IPCSU1.DEBUG.LOAD will be searched for programs before data set IPCSU1.DIAGNOS.LOAD.

You may enter each data set name using one of the following notations:

- Enter a fully-qualified data set name within apostrophes. For example, to specify data set IPCSU1.DEBUG.LOAD, enter:

  ipcs tasklib('ipcsu1.debug.load')

- A data set name beginning with your TSO/E prefix qualifier and ending with the qualifier “LOAD” may be designated by entering the qualifiers...
between them. If your TSO/E prefix is IPCSU1 and you want to specify
data set IPCSU1.DEBUG.LOAD, enter:

```
ipcs tasklib(debug)
```
The data set name entered is edited in three ways:

- Lowercase letters are changed to uppercase.
- The TSO/E prefix qualifier is added before the entered name.
- The final qualifier “LOAD” is appended to the name.

- A data set name beginning with your TSO/E prefix qualifier and ending
with the qualifier “LOAD” may also be designated by entering the
qualifiers including the final qualifier. For example, if your TSO/E prefix
is IPCSU1, the following command specifies data set
IPCSU1.DEBUG.LOAD:

```
ipcs tasklib(debug.load)
```
The following command specifies data set IPCSU1.LOAD:

```
ipcs tasklib(load)
```
The data set name entered is edited in two ways:

- Lowercase letters are changed to uppercase.
- The TSO/E prefix qualifier is added before the name.

---

**IPCSDDIR command — initialize a user or sysplex dump directory**

Use the IPCSDDIR command to:

- Initialize a user dump directory or a sysplex dump directory
- Reset a directory to contain only initialization records

To initialize the directory, the IPCSDDIR command writes two records to it: one
with a key of binary zeros (0) and the other with a key of binary ones (1). Once the
directory is initialized, you do not need to reinitialize it.

Initialization of the directory is required before IPCS subcommands can use it.

**Syntax**

```
IPCSDDIR dsname

[ REUSE  | NOREUSE ]
[ CONFIRM | NOCONFIRM ]
[ ENQ     | NOENQ ]
```

**Operands**

- **dsname**
  - The name of the data set for the dump directory.

- **REUSE**
  - **NOREUSE**
  - REUSE requests that the system delete all records from the data set and
    write the initialization records to the data set. The directory must have the
    VSAM REUSE attribute to use this option.
    
  - NOREUSE requests that the system write the initialization records to the
    data set. When using IPCSDDIR NOREUSE, the data set should contain no
    records; if the initialization records are already present, the command will
    fail.
TSO/E command IPCSDDIR

**NOCONFIRM**
CONFIRM causes the IPCS user to be prompted before IPCS runs an IPCSDDIR REUSE command.

NOCONFIRM authorizes immediate processing of an IPCSDDIR REUSE command.

**ENQ**
NOENQ
ENQ causes IPCSDDIR to serialize access to the dump directory during its initialization. This is the default and the recommended option.

NOENQ suppresses ENQ processing that is intended to block other instances of IPCS from using the directory being prepared for use by IPCSDDIR. IPCS itself uses this option when it has already established the needed serialization. Manual use of this option is not recommended.

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>04</td>
<td>Attention, command completed with a condition that might be of interest to the user.</td>
</tr>
<tr>
<td>08</td>
<td>Error, command encountered an error condition that might be of interest to the user.</td>
</tr>
<tr>
<td>12</td>
<td>Severe, an error condition or user request forced early end to the command processing.</td>
</tr>
<tr>
<td>16</td>
<td>Ending, an error condition from a called service routine forced an early end to the processing.</td>
</tr>
</tbody>
</table>

**SYSDSCAN command — display titles in dump data sets**

Use the SYSDSCAN command to display the titles of the dumps in dump data sets. The date and time when each dump was produced is included in the display.

**Syntax**

SYSDSCAN  [ xx [:yy] | 00:09 ]

**Operands**

**xx[:yy]**
Specifies one or a range of SYS1.DUMPnn data sets. xx and yy can be any positive decimal numbers from 00 through 99. A leading zero is optional and xx must be less than or equal to yy.

If you omit this operand, the default range is 00:09.

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>other</td>
<td>Either a nonzero return code from IKJPARS or a nonzero return code from dynamic allocation.</td>
</tr>
</tbody>
</table>

**Compatibility for Different Versions of MVS:**

1. Of all MVS/SP Version 4 IPCS facilities, only the SYSDSCAN command correctly processes MVS/SP Version 3 or Version 2 dumps.

3. Of all MVS/SP Version 3 IPCS facilities, only the SYSDSCAN command correctly processes MVS/SP Version 2 dumps. However, it does not correctly process MVS/SP Version 4 dumps.


Chapter 5. IPCS subcommands

This topic presents a task directory for and descriptions of the individual IPCS subcommands. The subcommands used to manage problems and data sets are described in Appendix A.

Entering subcommands

Enter a subcommand as directed by the syntax diagrams. See “Syntax conventions” on page 1-6 for more information.

- Entering subcommands in IPCS line mode
  Enter a subcommand at the IPCS prompt. For example:
  
  IPCS
  ANALYZE CONTENTION

- Entering subcommands from an IPCS batch job
  After the batch job has established an IPCS session, you can enter subcommands as you do from IPCS line mode. The following example shows how to enter a subcommand from the JCL or TSO/E job stream:
  
  //SYSTSIN DD *
  IPCS
  ANALYZE CONTENTION
  */

- Entering subcommands from the IPCS dialog
  There are two ways to enter subcommands from the IPCS dialog:
  - Choose option 4 (COMMAND) and enter the subcommand on the command line:
    
    ===> ANALYZE CONTENTION
  - Use the IPCS primary command to prefix the subcommand invocation from any command or option line of the IPCS dialog. For example:
    
    COMMAND ===> IPCS ANALYZE CONTENTION

Abbreviating subcommands and parameter operands

You can enter subcommands and parameter operands spelled exactly as they are shown or you can use an acceptable abbreviation (also referred to as an alias). When abbreviating enter only the significant characters; that is, you must type as much of the parameter as is necessary to distinguish it from the other parameters. Most minimal abbreviations are indicated.

Overriding defaults

Some subcommands allow you to override the SETDEF-defined defaults for the processing of that single subcommand. Once the subcommand completes processing, the original defaults are in effect.

The syntax diagram will indicate what, if any, SETDEF-defined parameters are allowed for that subcommand. For an explanation of those parameters, see “SETDEF subcommand — set defaults” on page 5-232.
Online help

During an IPCS line mode or dialog session, you can use the HELP subcommand to obtain information about any IPCS subcommand. This information includes the function, syntax, and operands of a subcommand. For example, to get the syntax and operands of the ANALYZE subcommand, enter:

HELP ANALYZE

Standard subcommand return codes

Most IPCS subcommands use the following return codes:

Table 5-1. Standard subcommand return codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>04</td>
<td>Attention, subcommand completed with a condition that may be of interest to you.</td>
</tr>
<tr>
<td>08</td>
<td>Error, subcommand encountered an error condition that may be of interest to you.</td>
</tr>
<tr>
<td>12</td>
<td>Severe error, an error condition or user request forced an early end to the subcommand processing.</td>
</tr>
<tr>
<td>16</td>
<td>Ending error, an error condition from a called service routine forced an early ending of subcommand processing.</td>
</tr>
</tbody>
</table>

Additional return codes or special reasons for using the defined return codes are presented with the description of each subcommand.

Task directory for subcommands

The following tables organize the IPCS subcommands by the tasks they perform. These tasks are grouped into the following eight areas:

- “Analyze a dump”
- “View dump storage” on page 5-3
- “View trace information” on page 5-3
- “Check system components and key system areas” on page 5-4
- “Retrieve information in variables” on page 5-5
- “Maintain the user dump directory or sysplex dump directory” on page 5-6
- “Perform utility functions” on page 5-6
- “Debug a dump exit program” on page 5-7

Analyze a dump

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check resource contention</td>
<td>“ANALYZE subcommand — perform contention analysis” on page 5-9</td>
</tr>
<tr>
<td>Display access register data</td>
<td>“ARCHECK subcommand — format access register data” on page 5-22</td>
</tr>
<tr>
<td>Display ASCB-related data areas</td>
<td>“ASCBEXIT subcommand — run an ASCB exit routine” on page 5-25</td>
</tr>
<tr>
<td>Display z/OS UNIX System Services (z/OS UNIX) address spaces and tasks</td>
<td>“OMVSMDATA subcommand — format z/OS UNIX data” on page 5-18</td>
</tr>
</tbody>
</table>
### When you want to

<table>
<thead>
<tr>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format selected control blocks</td>
</tr>
<tr>
<td>Check the status of a control block or unit of work</td>
</tr>
<tr>
<td>Search for a module by name</td>
</tr>
<tr>
<td>Search for a UCB</td>
</tr>
<tr>
<td>Display a map of the link pack area</td>
</tr>
<tr>
<td>Translate an STOKEN</td>
</tr>
<tr>
<td>Display the token from a name/token pair</td>
</tr>
<tr>
<td>Repair data residing in a dump or manage the list of patches in effect for a dump.</td>
</tr>
<tr>
<td>Identify address spaces satisfying specified selection criteria.</td>
</tr>
<tr>
<td>Display system status at the time of the dump</td>
</tr>
<tr>
<td>Display formatted control blocks</td>
</tr>
<tr>
<td>Display TCB-related data areas</td>
</tr>
<tr>
<td>Identify area(s) containing a given address</td>
</tr>
</tbody>
</table>

### View dump storage

<table>
<thead>
<tr>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locate data in a dump</td>
</tr>
<tr>
<td>Display storage</td>
</tr>
<tr>
<td>Display the eligible device table (EDT)</td>
</tr>
<tr>
<td>Display one or more UCBs</td>
</tr>
<tr>
<td>Search through a chain of control blocks</td>
</tr>
</tbody>
</table>

### View trace information

<table>
<thead>
<tr>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display component trace data</td>
</tr>
<tr>
<td>Display data-in-virtual trace data</td>
</tr>
<tr>
<td>Display program control flow</td>
</tr>
<tr>
<td>When you want to</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Display GTF trace data</td>
</tr>
<tr>
<td>Merge several trace data reports</td>
</tr>
<tr>
<td>Display trace data in the master</td>
</tr>
<tr>
<td>trace table</td>
</tr>
<tr>
<td>Display system trace entries</td>
</tr>
</tbody>
</table>

### Check system components and key system areas

<table>
<thead>
<tr>
<th>To obtain a diagnostic report for</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Program-to-Program</td>
<td>“APPCDATA subcommand — analyze APPC/MVS component data” on page 5-18</td>
</tr>
<tr>
<td>Communication (APPC) component</td>
<td></td>
</tr>
<tr>
<td>APPC/MVS transaction scheduler</td>
<td>“ASCHDATA subcommand — analyze APPC/MVS transaction scheduler data” on page 5-27</td>
</tr>
<tr>
<td>Auxiliary storage manager (ASM)</td>
<td>“ASMCHECK subcommand — analyze auxiliary storage manager data” on page 5-28 “VERBEXIT ASMDATA subcommand — format auxiliary storage manager data” on page 5-300</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Availability management component</td>
<td>“VERBEXIT AVMDATA subcommand — format availability manager data” on page 5-301</td>
</tr>
<tr>
<td>Communications task component</td>
<td>“COMCHECK subcommand — analyze communications task data” on page 5-40</td>
</tr>
<tr>
<td>Cross-system coupling facility (XCF)</td>
<td>“COUPLE subcommand — analyze cross-system coupling data” on page 5-62</td>
</tr>
<tr>
<td>Cross system extended services (XES)</td>
<td>“XESDATA subcommand — format cross system extended services data” on page 5-345</td>
</tr>
<tr>
<td>Data-in-virtual component</td>
<td>“DIVDATA subcommand — analyze data-in-virtual data” on page 5-79</td>
</tr>
<tr>
<td>Data lookaside facility (DLF)</td>
<td>“DLFDATA subcommand — format data lookaside facility data” on page 5-82</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Dump analysis and elimination (DAE)</td>
<td>“VERBEXIT DAEDATA subcommand — format dump analysis and elimination data” on page 5-302</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Global resource serialization</td>
<td>“VERBEXIT GRSTRACE subcommand — format Global Resource Serialization data” on page 5-304</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Information Management System (IMS™)</td>
<td>See IMS/ESA Utilities Reference</td>
</tr>
<tr>
<td>product</td>
<td></td>
</tr>
<tr>
<td>IMS resource lock manager (IRLM)</td>
<td>See IMS/ESA Utilities Reference or</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Input/output supervisor (IOS)</td>
<td>“IOSCHECK subcommand — format I/O supervisor data” on page 5-135</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Job entry subsystem 2 (JES2)</td>
<td>See z/OS JES2 Diagnosis</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>Job entry subsystem 3 (JES3)</td>
<td>See z/OS JES3 Diagnosis</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>LOGREC buffer records</td>
<td>“VERBEXIT LOGDATA subcommand — format logrec buffer records” on page 5-317</td>
</tr>
<tr>
<td>component</td>
<td></td>
</tr>
<tr>
<td>MVS message service (MMS)</td>
<td>“VERBEXIT MMSDATA subcommand — format MVS message service data” on page 5-320</td>
</tr>
<tr>
<td>To obtain a diagnostic report for</td>
<td>Use</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Modules in the nucleus</td>
<td>&quot;VERBEXIT NUCMAP subcommand — map modules in the nucleus&quot; on page 5-322</td>
</tr>
<tr>
<td>Real storage manager (RSM) component</td>
<td>&quot;RSMDATA subcommand — analyze real storage manager data&quot; on page 5-202</td>
</tr>
<tr>
<td>Stand-alone dump message log</td>
<td>&quot;VERBEXIT SADMPMSG subcommand — format stand-alone dump message log&quot; on page 5-326</td>
</tr>
<tr>
<td>Storage management subsystem (SMS) component</td>
<td>See z/OS DFSMSdfp Diagnosis</td>
</tr>
<tr>
<td>System logger component</td>
<td>&quot;LOGGER subcommand — format system logger address space data&quot; on page 5-174</td>
</tr>
<tr>
<td>System resource manager (SRM) component</td>
<td>&quot;VERBEXIT SRMDATA subcommand — format System Resource Manager data&quot; on page 5-326</td>
</tr>
<tr>
<td>System symbol table (which is different from the IPCS symbol table - it contains system symbols for general system use)</td>
<td>&quot;SYMDEF subcommand — display an entry in the system symbol table&quot; on page 5-276</td>
</tr>
<tr>
<td>Subsystem Interface (SSI) component</td>
<td>&quot;SSIDATA subcommand — display subsystem information&quot; on page 5-242</td>
</tr>
<tr>
<td>Structures of the coupling facility</td>
<td>&quot;STRDATA subcommand — format coupling facility structure data&quot; on page 5-253</td>
</tr>
<tr>
<td>SVC summary dump data</td>
<td>&quot;VERBEXIT SUMDUMP subcommand — format SVC summary dump data&quot; on page 5-327</td>
</tr>
<tr>
<td>Symptom string</td>
<td>&quot;VERBEXIT SYMPTOM subcommand — format symptom string&quot; on page 5-328</td>
</tr>
<tr>
<td>Time Sharing Option Extensions (TSO/E) product</td>
<td>See TSO/E V2 Diagnosis: Guide and Index</td>
</tr>
<tr>
<td>Virtual lookaside facility (VLF) component</td>
<td>&quot;VLFDATA subcommand — format virtual lookaside facility data&quot; on page 5-332</td>
</tr>
<tr>
<td>Virtual storage manager (VSM) component</td>
<td>&quot;VERBEXIT VSMDATA subcommand — format virtual storage management data&quot; on page 5-329</td>
</tr>
<tr>
<td>Virtual Telecommunication Access Method (VTAM®) product</td>
<td>See VTAM Diagnosis</td>
</tr>
<tr>
<td>Workload manager (WLM)</td>
<td>&quot;WLMDATA subcommand — analyze workload manager data&quot; on page 5-343</td>
</tr>
</tbody>
</table>

### Retrieve information in variables

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format IPCS default values</td>
<td>&quot;EVALDEF subcommand — format defaults&quot; on page 5-94</td>
</tr>
<tr>
<td>Format a dump data set name or information regarding a dump data set</td>
<td>&quot;EVALDUMP subcommand — format dump attributes&quot; on page 5-97</td>
</tr>
<tr>
<td>Format information regarding an entry in the storage map for a dump data set</td>
<td>&quot;EVALMAP subcommand — format a storage map entry&quot; on page 5-100</td>
</tr>
<tr>
<td>Format information regarding an entry in the symbol table for a dump data set</td>
<td>&quot;EVALSYM subcommand — format the definition of a symbol&quot; on page 5-105</td>
</tr>
<tr>
<td>Format dump storage or protection keys</td>
<td>&quot;EVALUATE subcommand — retrieve dump data for a variable&quot; on page 5-109</td>
</tr>
</tbody>
</table>
## Maintain the user dump directory or sysplex dump directory

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a source description to a dump directory</td>
<td>“ADDDUMP subcommand — add a source description to a dump directory” on page 5-7</td>
</tr>
<tr>
<td>Delete records in a source description from a dump directory</td>
<td>“DROPDUMP subcommand — delete source description data” on page 5-84</td>
</tr>
<tr>
<td>Delete records of control blocks that have been located in a dump</td>
<td>“DROPMAP subcommand — delete storage map records” on page 5-87</td>
</tr>
<tr>
<td>Delete IPCS symbols from the IPCS symbol table</td>
<td>“DROPSYM subcommand — delete symbols” on page 5-88</td>
</tr>
<tr>
<td>Create an IPCS symbol with a user-defined name</td>
<td>“EQUATE subcommand — create a symbol” on page 5-92</td>
</tr>
<tr>
<td>List dumps represented in a dump directory</td>
<td>“LISTDUMP subcommand — list dumps in dump directory” on page 5-153</td>
</tr>
<tr>
<td>List storage map entries</td>
<td>“LISTMAP subcommand — list storage map entries” on page 5-163</td>
</tr>
<tr>
<td>List attributes of symbols in the IPCS symbol table</td>
<td>“LISTSYM subcommand — list symbol table entries” on page 5-165</td>
</tr>
<tr>
<td>Assign a value to an IPCS symbol in the symbol table</td>
<td>“LITERAL subcommand — assign a value to a literal” on page 5-172</td>
</tr>
<tr>
<td>Renumber all stack symbols in the IPCS symbol table</td>
<td>“RENUM subcommand — renumber symbol table entries” on page 5-201</td>
</tr>
<tr>
<td>Validate control blocks</td>
<td>“SCAN subcommand — validate system data areas” on page 5-226</td>
</tr>
<tr>
<td>Create storage map entries for address spaces satisfying specified selection criteria</td>
<td>“SELECT subcommand — generate address space storage map entries” on page 5-229</td>
</tr>
<tr>
<td>Add an IPCS symbol (Znnnnn) to the IPCS pointer stack</td>
<td>“STACK subcommand — create a symbol in the stack” on page 5-242</td>
</tr>
</tbody>
</table>

## Perform utility functions

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the dsname of a directory entry</td>
<td>“ALTER subcommand — change a name in the IPCS inventory” on page 5-9</td>
</tr>
<tr>
<td>End the use of resources by IPCS</td>
<td>“CLOSE subcommand — release resources in use by IPCS” on page 5-38</td>
</tr>
<tr>
<td>Perform logical data comparisons</td>
<td>“COMPARE subcommand — compare dump data” on page 5-43</td>
</tr>
<tr>
<td>Copy records describing a dump data set from one dump directory to another</td>
<td>“COPYDDIR subcommand — copy source description from dump directory” on page 5-49</td>
</tr>
<tr>
<td>Copy dump data from one data set to another</td>
<td>“COPYDUMP subcommand — copy dump data” on page 5-52</td>
</tr>
<tr>
<td>Copy trace data to a data set from one or more dump or trace data sets</td>
<td>“COPYTRC subcommand — copy trace entries or records” on page 5-58</td>
</tr>
<tr>
<td>End an IPCS session</td>
<td>“END subcommand — end an IPCS session” on page 5-90</td>
</tr>
<tr>
<td>Obtain descriptive information about the IPCS command and its subcommands</td>
<td>“HELP subcommand — get information about subcommands” on page 5-133</td>
</tr>
</tbody>
</table>
When you want to | Use the
---|---
Format an integer using decimal digits, hexadecimal digits, or four EBCDIC characters | "INTEGER subcommand — format or list a number" on page 5-133
Request ISPF dialog services | "ISPEXEC subcommand — request an ISPF dialog service" on page 5-150
Produce messages and control spacing and pagination | "NOTE subcommand — generate a message" on page 5-184
Prepare resources for use by IPCS | "OPEN subcommand — prepare resources for use by IPCS" on page 5-190
Control session output format | "PROFILE subcommand — set preferred line and page size defaults" on page 5-197
Set, change, and display IPCS session defaults | "SETDEF subcommand — set defaults" on page 5-222
Transfer system management facility (SMF) records to a preallocated SMF (VSAM) data set or log stream | "SMFDATA subcommand — obtain system management facilities records" on page 5-241
Invoke a non-IPCS TSO/E command or subcommand function | "TSO subcommand — run a TSO/E command" on page 5-293

**Debug a dump exit program**

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resume trap processing from a STOP trap</td>
<td>&quot;GO subcommand — resume IPCS trap processing&quot; on page 5-121</td>
</tr>
<tr>
<td>Display the status of currently active traps</td>
<td>&quot;TRAPLIST subcommand — list the status of IPCS traps&quot; on page 5-286</td>
</tr>
<tr>
<td>Selectively disable traps</td>
<td>&quot;TRAPOFF subcommand — deactivate IPCS traps&quot; on page 5-288</td>
</tr>
<tr>
<td>Selectively enable traps</td>
<td>&quot;TRAPON subcommand — activate IPCS traps&quot; on page 5-290</td>
</tr>
</tbody>
</table>

**ADDDUMP subcommand — add a source description to a dump directory**

Use the ADDDUMP subcommand to add a source description to a dump directory. The description is for an unformatted source that IPCS can format, for example, an SVC dump, a stand-alone dump, an SYSDUMP dump, a trace data set, a data set, or active storage. The directory is allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

If the source is a dump, IPCS does not initialize it, a process that takes time. If IPCS can access the dump and it is an unformatted dump from an z/OS MVS system or an MVS/ESA SP 5.2 or 5.2.2 system, IPCS accesses it to define symbols for the dump and place them in the symbol table in the record; for information about the symbol table, see z/OS MVS IPCS User’s Guide. IPCS defines the following symbols, as appropriate; for information about these symbols, see "IPCS symbols" on page A-1:

- DUMPINGPROGRAM
- DUMPORIGINALDSNAME
- DUMPREQUESTOR
ADD DUMP subcommand

DUMPTIMESTAMP
DUMPTOD
ERRORID
INCIDENTTOKEN
PRIMARYSYMPTOMS
REMTEDUMP
SECONDARYSYMPTOMS
SLIPTRAP
TITLE

Related subcommands
    DROPDUMP

Syntax

ADDDUMP

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE|MAIN|STORAGE ]
[ DSNAME(dsname)|DATASET(dsname) ]
[ FILE(ddname)|DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]

[ TEST | NOTEST ]

Parameters

ACTIVE or MAIN or STORAGE
DSNAME(dsname) or DATASET(dsname)
FILE(ddname) or DDNAME(ddname)
PATH(path-name)

Specifies the source storage or data set to be represented by the source
description. One of these parameters is required.

ACTIVE, MAIN, or STORAGE specifies central storage.

DSNAME or DATASET specifies a cataloged data set.

FILE or DDNAME specifies the ddname of a data set.

PATH specifies the path of a file or directory on a z/OS UNIX file.

Return codes

See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the ADDDUMP subcommand.

Example

Add a dump to your user dump directory.

- Action

    adddump dsname('sys1.dump.d930428.t110113.system1.s00001')

- Result

    IPCS creates in your user dump directory a source description for the dump
    with the data set name of sys1.dump.d930428.t110113.system1.s00001. IPCS
    accesses the dump but does not initialize it.
ALTER subcommand — change a name in the IPCS inventory

Use the ALTER subcommand to change the name of a dump or trace data set in an IPCS dump directory.

Syntax

```
ALTER
{DSNAME(dsname) | DATASET(dsname) | FILE(ddname) | DDNAME(ddname)}
NEWNAME({ DSNAME(dsname) | DATASET(dsname) | FILE(ddname) | DDNAME(ddname) })
```

-------- SETDEF-Defined Parameters -------------------------

Note: You must specify one of the following SETDEF parameters on the ALTER subcommand.

```
{ DSNAME(dsname) | DATASET(dsname) }
{ FILE(ddname) | DDNAME(ddname) }
[ PATH(path-name) ]
```

Note: You can override the following SETDEF parameters. See

```
“SETDEF subcommand — set defaults” on page 5-232
```

Parameters

DSNAME

Designates the current dsname or ddname of the dump or trace.

NEWNAME

Designates the new dsname or ddname of the dump or trace. The ALTER subcommand does not actually change the name of any data sets, only the association between dump directory data and a name.

For consistency with the TSO (and IDCAMS) ALTER command, you can use NEWNM as an abbreviation of the NEWNAME keyword.

The ALTER subcommand requires that the dump whose description is to be affected be explicitly specified.

ANALYZE subcommand — perform contention analysis

Use the ANALYZE subcommand to gather contention information from component analysis exits and format the data to show where contention exists in the dump. ANALYZE obtains contention information for I/O, ENQs, suspend locks, allocatable devices, real frames, global resource serialization latches, and other resources.

ANALYZE produces different diagnostic reports depending on the report type parameter or parameters. Specify one or more of these parameters to select the information you want to see. If you do not specify a report type parameter, you receive an EXCEPTION report.

- EXCEPTION displays contention information when a unit of work holds at least one resource for which contention exists and that unit of work is not waiting for another resource.
ANALYZE subcommand

When applicable, ANALYZE displays a resource lockout report following the EXCEPTION report when a unit of work holds a resource and is waiting for another resource that cannot be obtained until the first resource is freed. See 5-15 for an example of an EXCEPTION report and 5-16 for an example of a lockout analysis report.

- **RESOURCE** displays contention information organized by resource name. See the allocation/unallocation component in z/OS MVS Diagnosis: Reference for an example of a RESOURCE report.
- **ASID** displays contention information organized by ASID. Parts of this report are also produced by the STATUS CPU CONTENTION subcommand. See 5-12 for an example of an ASID report.
- **ALL** displays all contention information.

Obtaining contention information

IPCS gathers contention information once for each dump. ANALYZE invokes each ANALYZE exit routine specified by parmlib members embedded in the BLSCECT parmlib member. When contention information has not been previously gathered, IPCS issues this message:

```
BLS01000I Contention data initialization is in progress
```

The amount of time required to gather contention information depends on the size of the dump, how many address spaces it contains, the number of I/O devices, and the amount of contention in the dump. IPCS recommends that you run the ANALYZE subcommand in the background as part of a preliminary screening report. (See z/OS MVS IPCS User’s Guide for information about running IPCS subcommands in the background.)

In the event that no contention information is detected, IPCS issues:

```
BLS01002I No resource contention detected. Undetected contention is possible.
```

But if contention information is present, IPCS stores this data in the dump directory. When the contention information in the dump directory is inconsistent with the current exit routine list, this message is issued:

```
BLS01004I ANALYZE exit list in PARMILB member BLSCECT has changed. Correct BLSCECT member or issue DROPDUMP RECORDS TRANSLATION.
```

If the BLSCECT parmlib member is correct, enter:

```
COMMAND ===> DROPDUMP RECORDS(TRANSLATION)
```

This command deletes all contention information from the dump directory and lets you reenter the ANALYZE subcommand to gather the contention data again.

To perform its processing, the ANALYZE subcommand uses the contention queue element (CQE) create service to obtain contention data. The CQE service is IBM-supplied and can be used when writing your own dump exit. See z/OS MVS IPCS Customization for information about these services and for information about writing ANALYZE exits.

Syntax
Report type parameters

Use these parameters to select the type of report. If you omit a report type parameter, the default is EXCEPTION.

**EXCEPTION**

Specifies that contention information is to be reported only for units of work that have been determined to be “exceptions”. A unit of work is considered an “exception” when all of the following conditions apply:
- The unit of work holds at least one resource for which contention exists
- The unit of work is not waiting for another resource

The EXCEPTION report, which is organized by ASID, identifies the units of work that appear to be preventing work from being accomplished in the system. A second section of the EXCEPTION report can be produced (when applicable) indicating resource lockouts. The lockout analysis report lists all units of work that are involved in a circular chain of resource ownership.

**RESOURCE**

Specifies that the contention analysis report is to be organized by resource name. All resources are listed regardless of whether they are involved in contention.

**ASID**

Specifies that the contention analysis report is to be organized by ASID. The report uses the ASID number, the control block type and address, the CPU address and the system name (SYSNAME) to identify a unit of work that holds or is waiting for a resource. All units of work are listed regardless of whether they are involved in contention.

**ALL**

Specifies that all contention-related information found for this dump is to
BE REPORTED. NONCONTENTION INFORMATION, SUCH AS ALL ACTIVE I/O AND ALL
HOLDERS OF LOCAL AND CMS LOCKS, IS ALSO INCLUDED.

THE ALL PARAMETER INCLUDES EXCEPTION, RESOURCE AND ASID. THESE
OTHER PARAMETERS CAN BE SPECIFIED WITH ALL, BUT DO NOT CHANGE THE CONTENTS
OF THE GENERATED OUTPUT.

XREF OR NOXREF
XREF SPECIFIES THAT ADDITIONAL CROSS REFERENCING INFORMATION ABOUT RESOURCES
HELD AND RESOURCES WAITED FOR ARE TO BE DISPLAYED.

NOXREF SPECIFIES THAT THIS ADDITIONAL INFORMATION IS TO BE SUPPRESSED, AND
IS THE DEFAULT.

RETURN CODES
SEE "STANDARD SUBCOMMAND RETURN CODES" ON PAGE 5-2 FOR A DESCRIPTION OF THE
RETURN CODES PRODUCED BY THE ANALYZE SUBCOMMAND.

EXAMPLE 1
PRODUCE AN ASID CONTENTION REPORT.

- ACTION
  COMMAND ===> ANALYZE ASID XREF

- RESULT
  THE FOLLOWING REPORT IS PRODUCED.

1 CONTENTION REPORT BY UNIT OF WORK
2 JOBNAME=S1202   ASID=000E   TCB=009FA950
   JOBNAME=S1202   HOLDS THE FOLLOWING RESOURCE(S):

   RESOURCE #0004:
   NAME=Device Group 0015
   DATA=(ALC) ASSOCIATED WITH 3330,DASD,SYSDA,SYSSQ,3330B,SYSALLDA

   RESOURCE #0004 IS WAIVED ON BY:
   JOBNAME=S1203   ASID=000F   TCB=009FA950

3 JOBNAME=S1203   ASID=000F   TCB=009FA950
   JOBNAME=S1203   IS WAITING FOR RESOURCE(S):

   RESOURCE #0004:
   NAME=Device Group 0015
   DATA=(ALC) ASSOCIATED WITH 3330,DASD,SYSDA,SYSSQ,3330B,SYSALLDA

   RESOURCE #0004 IS HELD BY:
   JOBNAME=S1202   ASID=000E   TCB=009FA950

4 JOBNAME=S1301   ASID=0011   TCB=009FA950
   JOBNAME=S1301   HOLDS THE FOLLOWING RESOURCE(S):

   RESOURCE #0003:
   NAME=Device Group 0014
   DATA=(ALC) ASSOCIATED WITH 3330,DASD,SYSDA,SYSSQ,3330A,SYSALLDA

   RESOURCE #0003 IS WAIVED ON BY:
   JOBNAME=S1302   ASID=0012   TCB=009FA950

1 Names the contention report type, ASID. The report is
organized by ASID.
Identifies the unit of work by jobname, and lists the resource(s) it holds. If it holds more than one resource, they are displayed in the order in which they were encountered. When XREF is specified the report shows for each held resource:
- Other units of work that share the resource.
- Units of work that are waiting for the resource.

Resources that the job is waiting for are listed. XREF was specified, so the report identifies the unit of work that currently owns the resource.

Lists other units of work experiencing contention.

Example 2
Produce a RESOURCE contention report.

- Action
  COMMAND ===> analyze resource
- Result
  The following report is produced.

CONTENTION REPORT BY RESOURCE NAME

RESOURCE #0003:
  NAME=Device group 001B
  RESOURCE #0003 IS HELD BY:
    JOBNAME=S1400      ASID=0013      TCB=009FA490
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR

RESOURCE #0003 IS REQUIRED BY:
    JOBNAME=S1402      ASID=0014      TCB=009FA490
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR
    JOBNAME=S1403      ASID=0015      TCB=009FA490
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR

RESOURCE #0004:
  NAME=LOCAL LOCK FOR ASID 001A
  RESOURCE #0002 IS HELD BY:
    JOBNAME=DATJINT     ASID=001A      TCB=009FE240
    DATA=INTERRUPTED AND NOW DISPATCHABLE

Chapter 5. IPCS subcommands 5-13
**ANALYZE subcommand**

1. Names the contention report type, RESOURCE. The report is organized by resource name.
2. Identifies a resource experiencing contention. The report shows:
   - NAME - The name of the resource
   - Information about each job that either owns or is waiting to obtain the named resource:
     - JOBNAME - The job name
     - ASID - The associated home ASID
     - TCB - The address of the task control block (TCB) for the task that owns or is waiting to obtain the resource
     - DATA - Additional information that describes the named resource.
3. Identifies a resource experiencing contention. Because the resource shown in the example report is associated with a lock, the report shows:
   - NAME - The name of the resource
   - Information about each job that either owns or is waiting to obtain the named resource:
     - JOBNAME - The job name
     - ASID - The associated home ASID
     - TCB - The address of the task control block (TCB) for the task that owns or is waiting to obtain the resource
     - DATA - Additional information that describes the named resource.
4. Identifies a resource experiencing contention. Because the resource shown in the example report is associated with a latch, the report shows:
   - NAME - The latch set name
   - ASID - The identifier of the primary address space at the time the latch set was created
   - LATCH# - The number of the latch that has contention. This is followed by a ":" and the LATCHID String. Up to 255 characters of the LATCHID string will be displayed. If more than 255 characters exist, "T" will be appended after the 255th byte.
   - Information about each job that either owns or is waiting to obtain the latch:
     - JOBNAME - The job name
     - ASID - The associated home ASID
ANALYZE subcommand

- TCB - The TCB address of the requester, if the requester is a
task; the value '00000000', if the requester is an SRB
- DATA - Indicates whether the job requested exclusive or shared
access to the resource
- RETADDR - The contents of general purpose register (GPR) 14
at the time the requester called the Latch_Obtain service
- REQID - The requester ID (an 8-byte field that identifies the
latch requester) that was specified by the RequestId value on
the Latch_Obtain service.
- REQUEST - Time when the latch obtain was requested.
- GRANT - Time that the latch was granted ownership. This time
is in only provided for owners.

Additional information about latch resource values; for example,
Latch#, REQID, and LATCHID, that are provided by the latch set
creator can be found in the z/OS MVS Diagnosis: Reference

When XREF is specified:
- For each job that holds one or more resources, the report lists other
resources that are held. These other resource names are truncated to fit on
a single line. The full resource names are available in other sections of the
report.
- For each job that is waiting on one or more resources, the report gives the
name of the resources.

Resources that the job is waiting for are listed. XREF was specified, so the
report identifies the unit of work that currently owns the resource.

Example 3
Produce an EXCEPTION contention report.
- Action
  COMMAND ===> analyze exception
- Result
  The following report is produced.
ANALYZE subcommand

1 CONTENTION EXCEPTION REPORT
2 JOBNAME=S1202  ASID=000E  TCB=009FA950
   JOBNAME=S1202  HOLDS THE FOLLOWING RESOURCE(S):
3 RESOURCE #0004: There are 0001 units of work waiting for this resource
   NAME=Device Group 0015
   DATA=(ALC) ASSOCIATED WITH 3330,DASD,SYSDA,SYSSQ,3330B,SYSSALLDA
4 STATUS FOR THIS UNIT OF WORK:
   IRA10102I This address space is on the SRM WAIT queue.
   IRA10104I The reason for swap-out is long wait (3).
5 JOBNAME=S1301  ASID=0011  TCB=009FA950
   JOBNAME=S1301  HOLDS THE FOLLOWING RESOURCE(S):
   RESOURCE #0003: There are 0001 units of work waiting for this resource
   NAME=Device Group 0014
   DATA=(ALC) ASSOCIATED WITH 3330,DASD,SYSDA,SYSSQ,3330A,SYSSALLDA
   STATUS FOR THIS UNIT OF WORK:
   IRA10102I This address space is on the SRM WAIT queue.
   IRA10104I The reason for swap-out is long wait (3).
6 JOBNAME=MEGA  ASID=0014  TCB=009C0E88
   JOBNAME=MEGA  HOLDS THE FOLLOWING RESOURCE(S):
   RESOURCE #0006: There are 0002 units of work waiting for this resource
   NAME=Db3.XMITDATA.LATCH.SET  ASID=001D  Latch#=1
   DATA=EXCLUSIVE  RETADDR=82C63F6E  REQID=00AC41A000000000
   STATUS FOR THIS UNIT OF WORK:
   IRA10102I This address space is on the SRM IN queue.
7 BLS01005I No resource lockouts were detected for this dump

1 Names the contention report type, EXCEPTION.
2 Identifies the unit of work, by jobname, that holds a resource
   for which contention exists.
3 Lists the resources held by this unit of work. If more than
   one resource is held, the resources are displayed in the order
   in which they were encountered.
4 Indicates the status of this unit of work.
5 and 6 Identify other units of work that hold resources for which
   contention exists.
7 Indicates that no lockouts were detected. Therefore, a lockout
   analysis report will not appear at the end of this
   EXCEPTION report.

Example 4
Produce a lockout analysis report.
   Action
   COMMAND ===> analyze exception
   Result
   The following report is produced.
Indicates that this is a lockout analysis report, which is organized by ASID. A lockout occurs when a unit of work holds a resource and is waiting for another resource that cannot be obtained until the first resource is freed. The lockout report follows the EXCEPTION report with the lockout heading repeated for each unique set of resources involved.

Identifies IPCSJOB as the unit of work that holds a resource for which contention exists. The resources that are held and are waited for are displayed.

List the two other units of work and the resources that are held and are waited for. These resources caused IPCSJOB to become part of a lockout condition.
APPCDATA subcommand

APPCDATA subcommand — analyze APPC/MVS component data

Use the APPCDATA subcommand to generate reports about the Advanced Program-to-Program Communication (APPC) component of MVS. This subcommand provides information about the following topics:

- Status of the APPC component
- Configuration of local logical units (LU)
- Local transaction programs (TPs) and their conversations
- Allocate queues and their associated APPC/MVS server address spaces.
- TP FMH-5 attach requests
- APPC component trace status

See the APPC/MVS component chapter in z/OS MVS Diagnosis: Reference for examples of APPCDATA output.

Syntax

APPCDATA

-------- Report Type Parameters ---------------------------
[ ALL ]
[ CONFIGURATION ]
[ CONVERSATIONS[(asid|ALL)] ]
[ SERVERDATA ]
[ CTRACE ]
[ FMH5MANAGER ]
[ STATUS ]

-------- Data Selection Parameters ------------------------
[ DETAIL ]
[ EXCEPTION ]
[ SUMMARY ]

-------- SETDEF-Defined Parameters ------------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232

[ ACTIVE|MAIN|STORAGE ]
[ DNAME(dsname)|DATASET(dsname) ]
[ FILE(ddname)|DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters
Report type parameters
APPCDATA subcommand

Use these parameters to select the type of report. You can specify as many reports as you want. If you omit these parameters, the default is ALL.

**ALL**
Requests that information for all the APPCDATA options be presented.

**CONFIGURATION**
Requests information about the configuration of local LUs in terms of their connections to partner LUs.

The configuration summary report displays the following information:
- Local LU name and its status
- Number of partner LUs with which the local LU had sessions
- Number of partner/mode pairs for which sessions were established.
- VTAM generic resource name or *NONE*
- Local LU resource manager name and token
- Number of units of recovery (URs)
- Total expressions of interest

The configuration detail report includes information from the summary report and adds the following information for each partner LU:
- Partner LU name
- Number of modes that defined session characteristics
- Logon name for each mode.
- URIDs and expressions of interest for each UR
- Diagnostic information

**CONVERSATIONS[asid|ALL]**
Requests information for each local TP and its conversations for either a particular address space, specified as an address space identifier (ASID), or all address spaces. For this parameter, *asid* is a 1- to 4-character hexadecimal value. If no ASID is specified, information for all address spaces is displayed.

The conversations summary report displays for each address space the following information:
- A scheduler name
- Local TP name or *UNKNOWN*
- TP_ID
- Local LU name through which the session was established
- Work unit ID
- Number of conversations in which the TP was engaged

The conversations detail report includes information from the summary report and adds the following information for each conversation:
- Conversation identifier
- Conversation correlator
- Partner TP name or *UNKNOWN*
- Attach user identifier
- Conversation type
- Sync level
- Unit of recovery identifier (URID)
- Logical unit of work identifier (LUWID)
- Resource manager name
- LU name of the partner TP
- Logon mode
- Current state
- Time of day (TOD)
**APPDATA subcommand**

**SERVERDATA**
Requests information about allocate queues and their associated APPC/MVS server address spaces.

The SERVERDATA summary report displays the following information about allocate queues and APPC/MVS server address spaces.

- For each allocate queue:
  - Name of the TP whose allocate requests are being queued
  - Name of the LU at which the server resides
  - Userid that was specified on the allocate request
  - Profile of the security group to which the userid belongs
  - Name of the LU at which the client TP resides
  - Number of servers for the allocate queue
  - Number of allocate requests (elements) on the allocate queue
  - Total number of allocate requests that have been added to the allocate queue (includes allocate requests that have been received from the allocate queue)
  - Number of pending calls to the Receive_Allocate service
  - Keep time (the amount of time APPC/MVS is to maintain the allocate queue in the absence of servers)
  - Time at which the allocate queue was created
  - Time at which an allocate request was last received from the allocate queue
  - Time at which a server last called the Unregister_For_Allocates service to unregister from the allocate queue.

- For each APPC/MVS server:
  - Address space identifier (ASID) of the server address space
  - An indication of whether the server has an outstanding call to the Get_Event service
  - Number of events on the server's event queue
  - Number of allocate queues for which the server is currently registered.

The SERVERDATA detail report includes information from the summary report and adds the following information:

- For each APPC/MVS server for a given allocate queue:
  - Address space identifier (ASID) of the server address space
  - Time at which the server registered for each allocate queue
  - Time at which the server last issued the Receive_Allocate service
  - Time at which a Receive_Allocate request was last returned to the server
  - Total number of allocate requests returned to the server.

- For each pending Receive_Allocate request for a given allocate queue:
  - The address space identifier (ASID) of the server with the pending Receive_Allocate request.

- For each inbound allocate request for a given allocate queue:
  - Conversation identifier
  - Access method conversation identifier
  - Conversation type (basic or mapped)
  - Conversation correlator
APPCDATA subcommand

- Logon mode
- Partner LU name
- Sync level (“none” or “confirm”)
- Userid
- Security profile
- Time at which the system placed the request on the allocate queue
- Address of the access method control block (ACB) for the LU at which the APPC/MVS server resides.

- For each server event for a given server:
  - Event (“min” or “max”)
  - Event object (the allocate queue token of the allocate queue to which the event pertains)
  - Event qualifier.

- For each allocate queue for a given server:
  - Allocate queue token
  - Minimum and maximum one-time event threshold
  - Minimum and maximum continuous event threshold.

CTRACE
Displays the status of component trace for APPC, trace options, and other trace-related information.

The CTRACE summary report displays the following information:
- Trace status
- Most recently specified trace options
- Userids, ASIDs, and job names used as filters

The CTRACE detail report includes information from the summary report and adds the following details:
- Console identifiers of the operator who most recently started or stopped the trace
- Message-routing command and response token (CART)
- Information about the trace table

FMH5MANAGER
Requests information about the transaction program FMH-5 attach requests that are either waiting to be processed or are currently being processed.

The summary report displays the number of TP FMH-5 attach requests that are waiting to be processed and the number of requests currently being processed.

The detail report lists, for both types of requests, the LU names and the total number of requests they received. For each LU name, the requests are broken down into the number of requests originating from a specific partner LU name. If the request was being processed and dump data is available, the report displays the data.

STATUS
Displays a message about the overall status of the APPC component at the time of the dump.

Note: The reports generated by the APPCDATA subcommand contain information for IBM diagnostic use. The IBM Support Center might ask for this information for use in problem determination.

Data Selection Parameters
APPCDATA subcommand

Data selection parameters limit the scope of the data in the report. If no data selection parameter is selected, the default is to present a summary report for all topics listed below.

**DETAIL**
Requests detailed information for each of the selected topics.

**EXCEPTION**
Requests a list of exceptional or unusual conditions for each topic. The list of exceptions contains information for IBM diagnostic use.

**SUMMARY**
Requests summary information for each of the requested topics.

**Return codes**
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the APPCDATA subcommand.

ARCHECK subcommand — format access register data

Use the ARCHECK subcommand to format access register data associated with system control blocks, the active processors described by a stand-alone dump, or the processors described by an SVC dump.

**Syntax**

```
ARCHECK { data-descr } { CPU(nn)STATUS } { HEADER }
[ AR(nn| ALL) ]
[ ALET(aletvalue) ]
[ TRANSLATE | ANALYZE ]
```

**Parameters**

**data-descr**
Specifies the data description parameter, which supplies the location of the control block or access list you want. The data description parameter consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (see note below)
- Array parameters (optional)
- A remark parameter (optional)
Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

Note: The STRUCTURE(cbname) attribute parameter is required; all other attribute parameters are optional. Use one of the following values for cbname:

- ACCESSLIST
- RB
- SSRB
- TCB

When you specify STRUCTURE(ACCESSLIST), the ALET parameter is required to associate access registers with the access list.

**CPU(nn) STATUS**

CPU(nn) STATUS is for stand-alone dumps and requests formatting of the access registers in the STORE STATUS record associated with the specified CPU. The display shows the access register information at the time of the error.

**HEADER**

HEADER is for SVC dumps and produces the same output as CPU(nn) STATUS.

**AR(nn | ALL)**

Requests processing of either a specific access register or all non-zero access registers and is the default.

nn is a decimal number ranging from 0 to 15. If you do not supply a number, ALL is the default. When you specify AR(ALL), the contents of the access registers appears first, followed by more detailed information. The nature of the rest of the information you will see depends on whether you specify TRANSLATE or ANALYZE.

**ALET(alet value)**

Specify an 8-character hexadecimal ALET value instead of one of the saved access registers, to process a specific access list entry and control the use of the PASN or work unit access list. ALET is required with STRUCTURE(ACCESSLIST).

**TRANSLATE**

TRANSLATE identifies the target address space or data space for an ALET or access register and is the default. TRANSLATE works for stand-alone dumps only.

**ANALYZE**

ANALYZE formats the access list entry (ALE) and the address space second table entry control blocks. The ARCHECK service uses these control blocks to achieve access register addressability.

**Return codes**

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the ARCHECK subcommand.

**Example 1**

Display the contents of access register 5 for the RB at AD8BE0, in address space number 12 (X'000C').

- **Action**

  `COMMAND ===> archeck address(00ad8be0) asid(X'000C') structure(rb) ar(5)`

- **Result**
The display identifies the requested access register and the address space or data space associated with it.

Example 2
Get detailed information from a stand-alone dump about all the access registers associated with a central processor.
- Action
  COMMAND ===> archeck cpu(00) status ar(all) analyze
- Result
  IPCS produces the following report for the specified central processor:

1. ACCESS REGISTER VALUES
   0-3 00000000 00000000 00000000 00000000
   4-7 FFFFFFFF 00100007 0101000B FFFFFFFF
   8-11 00000000 00100006 00000000 00000000
   12-15 00000000 00000000 00000000 00000000

2. ALET TRANSLATION
   --------------------------------------------------

3. AR 04 VALUE: FFFFFFFF
   IEA11016I There are non-zero reserved bits in the ALET.

   AR 04 Not translatable
   --------------------------------------------------

4. AR 05 VALUE: 00010007
   IEA11013I The WORKUNIT access list is being used for translation.
   ALE: 7FFFD970
   +0000 OPTB1... 00   SN..... 01   EAX..... 0001
   +0008 ASTE.... 00026140   ASTSN... 00000001
   ASTE REAL ADDRESS: 00026140
   +0000 ATO.... 00C0F080   AX..... 0001   ATL..... 0030
   +0008 STD..... 80412000   LTD..... 00CA9F00

5. AR 05 addresses ASID (X'0004')

   1. Shows the contents of the access registers.
   2. Shows how the ALETS are translated and listed in numeric order with information about the translation results (described in items through 3).
   3. Shows the output message indicating an untranslatable ALET. An ALET is typically not translatable when errors are detected or dump data is insufficient for translation.
   4. Shows the translation results for a translatable ALET. Related information might include the access list entry used for translation processing and, if the ALET is addressing an address space, the address space second table entry (ASTE) control block.
   5. For translatable ALETS, a message indicates which space is accessible using the related access register.

Example 3
Obtain information about a particular access register using an access list you supply.
- Action
ARCHECK subcommand

Command:

```plaintext
====> archeck address(7fffd900) asid(12) str(accesslist)
alet(x'00010006') analyze
```

- Result

**ALET TRANSLATION**

-------------------------------

1. **ALET VALUE:** 00010006

2. IEA11013I The WORKUNIT access list is being used for translation.

3. **ALE:** 7FFFD960
   
   ```plaintext
   +0000 OPTB1... 01 SN...... 01 EAX..... 0001
   +0008 ASTE.... 00D26000 ASTSN... 00000001
   ```

4. **ASTE REAL ADDRESS:** 00D26000
   
   ```plaintext
   +0000 ATO.... 00C0F0B0 AX...... 0001 ATL..... 0030
   +0008 STD..... 0080B07F LTD..... 80412000 PALD.... 00CA9F00
   +0014 SQN..... 00000001 PROG.... 00F37E00
   ```

5. **ALET addresses ASID(X'0002')**

   Identifies the ALET value used for translation.

   The message that identifies the specified access list (address 7fffd900 in the command) as the WORKUNIT access list.

   The formatted ALE and ASTE control blocks used for translation.

   **Note:** The ASTE only appears of the ALET addresses an address space.

   Identifies the space that the translated ALET addresses.

ASCBEXIT subcommand — run an ASCB exit routine

Use the ASCBEXIT subcommand to run an installation-provided ASCB exit routine.

**Syntax**

---

Chapter 5. IPCS subcommands 5-25
ASCBEXIT subcommand

{ASCBEXIT } { pgmname | * }
{ASCBX   }

asid

[ AMASK(mask) ]

-------- SETDEF-Defined Parameters --------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

pgmname or *

pgmname specifies the name of an installation-provided exit routine that
must reside in a library available to IPCS, such as a step library, job library,
or link library. For information about writing ASCB exit routines, see [z/OS]
MVS IPCS Customization.

* specifies that the list of installation-provided ASCB exit routines
(identified in the BLSCUSER parmlib member) receives control.

Note: The z/OS MVS system does not supply any ASCB exit routines.

asid

Specifies the address space identifier (ASID) to be passed to the exit
routine. The ASID can range from 1 through 65535. You can specify the
ASID in decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’).

AMASK(mask)

Specifies an integer mask that ASCBEXIT is to AND to the dump addresses
passed by the exit to the storage access and format service routines. Only
X’7FFFFFFF’, X’00FFFFFF’, or the corresponding decimal or binary values
will be accepted.

Return codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 12   | Severe error; an error condition or user request forced early end to the
      | subcommand processing. |
| 16   | Ending error; an error condition from a called service routine forced an
      | early end to the subcommand processing. |
| other| An exit-generated return code |

Example

Invoke an installation-provided ASCB exit.

– Action
ASCBEXIT subcommand

COMMAND ====> ascbexit chekascb 7

– Result
This command runs the installation-provided routine, CHEKASCB, and passes it ASID 7. Note that CHEKASCB must be identified in the BLSCUSER parmlib member.

ASCHDATA subcommand — analyze APPC/MVS transaction scheduler data

Use the ASCHDATA subcommand to generate reports about the APPC/MVS transaction scheduler. This subcommand provides the following information:

• Status of the scheduler
• Subsystem name
• Default scheduler class
• Generic initiators, if any
• Summary information for each class

See the APPC/MVS component in z/OS MVS Diagnosis: Reference for examples of ASCHDATA output.

Note: The reports generated by ASCHDATA contain information for IBM diagnostic use. The IBM Support Center might ask you to provide this information for use in problem determination.

Syntax

ASCHDATA

-------- Report Type Parameters  ------------------------------

[ CLASS[(classname|ALL)] ]

-------- Data Selection Parameters  ---------------------------

[ DETAIL ]
[ EXCEPTION ]
[ SUMMARY ]

-------- SETDEF-Defined Parameters  -----------------------------
Note: You can override the following SETDEF parameters. See

“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE|MAIN|STORAGE ]
[ DSNNAME(dsname)|DATASET(dsname) ]
[ FILE(ddname)|DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

Report type parameters
Use these parameters to select the type of report. Specify only one; ASCHDATA produces the report type for each topic specified by a data selection parameter.
If you omit a report type parameter, the default is ALL.
ASCHDATA subcommand

CLASS[(classname|ALL)]
Requests APPC transaction scheduler information for either a particular scheduler class or all scheduler classes. For this parameter, classname is a valid 1- to 8-character scheduler class name. If no class name is specified, information for all scheduler classes is displayed.

The class summary report displays the following information for each scheduler class:
- Class name and status of each class, including:
  - Maximum and minimum number of initiators
  - Expected response time
  - Message limit
- Total number of jobs waiting for processing
- Total number of active initiators
- Total number of active waiting multi-trans initiators
- Total number of idle initiators

The class detail report includes information from the summary report and adds the following information:
- For each job waiting to run, the job identifier, local LU name, partner LU name, TP name, FMH5 userid, and time the job started waiting to run.
- For each active initiator, the address space identifier (ASID), TP start time, TP name, current job identifier, local LU name, partner LU name, and FMH5 userid.
- For each active waiting multi-trans initiators, the ASID and TP name.
- For each idle initiator, the ASID.

Data selection parameters
Data selection parameters limit the scope of the data in the report. If no data selection parameter is selected, the default is to present a summary report for all topics listed below.

DETAIL
Requests detailed information for each of the selected topics.

EXCEPTION
Requests a list of exceptional or unusual conditions for each topic. The list of exceptions contains information for IBM diagnostic use.

SUMMARY
Requests summary information for each of the requested topics.

Return codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the ASCHDATA subcommand.

ASMCHECK subcommand — analyze auxiliary storage manager data
Use the ASMCHECK subcommand to analyze and validate data associated with the auxiliary storage manager (ASM) to produce a report.

See the ASM component in z/OS MVS Diagnosis: Reference for an example of the ASMCHECK report and more information about diagnosing ASM problems.

Syntax
CBFORMAT subcommand — format a control block

Use the CBFORMAT subcommand to format and display a control block or data area that is defined in the exit data table. CBFORMAT can also be used to test and run user-written formatting routines and control block models. “Control blocks and data areas scanned, mapped, and formatted” on page C-11 lists the control blocks and data areas that CBFORMAT formats.

The maximum size of the control block or data area is 64 kilobytes.

After successful processing, CBFORMAT sets X, the current address, to the starting address of the data area being formatted. If a data area has no IPCS formatting support, IPCS issues message BLS17004I, which identifies the requested control block or data area name specified with the STRUCTURE parameter.

You can use the CBFORMAT subcommand to format literal data as if it was a valid instance of a control block. IBM does not recommend this use unless:

- The control block involved remains valid when removed from its original setting.
- You recognize that it is inappropriate, for example, to ask the service to format a symbolic literal as a task control block (TCB) and then to use the formatted TCB for diagnosis.

Syntax

```
{ASMCHECK | ASMK }
```

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See

“SETDEF subcommand — set defaults” on page 5-232

```
[ACTIVE|MAIN|STORAGE ]
[DSNAME(dsname) | DATASET(dsname) ]
[FILE(ddname) | DDNAME(ddname) ]
[PATH(path-name) ]
[FLAG(severity) ]

[PRINT | NOPRINT ]

[Terminal | NOTERMINAL ]

[TEST | NOTEST ]
```

Return codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the ASMCHECK subcommand.
CBFORMAT Subcommand

```{ CBFORMAT | CBF }

  data-descr

  [ EXIT | NOEXIT ]
  [ FORMAT(name [level]) ]
  [ MODEL(name) ]
  [ VIEW(fieldlist) ]
```

------ SETDEF-Defined Parameters --------------------------

Note: You can override the following SETDEF parameters. See

“SETDEF subcommand — set defaults” on page 5-232

  [ FLAG(severity) ]
  [ PRINT | NOPRINT ]
  [ TERMINAL | NOTERMINAL ]
  [ TEST | NOTEST ]

Parameters

data-descr

Specifies the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (see note below)
- Array parameters (optional)
- A remark parameter (optional)

See Chapter 3, “Data description parameter,” on page 3-1 for the use and syntax of the data description parameter.

Note: The STRUCTURE(cbname) attribute parameter is required, except with FORMAT and MODEL. All other attribute parameters are optional.

EXIT or NOEXIT

EXIT processes all formatting exits defined in the exit data table for a given control block, after the control block has formatted successfully.

NOEXIT requests no formatting exits, and is the default.

FORMAT(name[level])

FORMAT identifies the user-written formatter program to be used to format the data. See z/OS MVS IPCS Customization for details about formatting exits.

The level option can be one of the following:

HBB3310

It causes a BLSRESSY to be passed in ABITS(31) format.

HBB7703

It causes a BLSRESSY to be passed in ABITS(64) format.

If level is omitted, the default is HBB7703 for compatibility with CBFORMAT.
CBFORMAT Subcommand

Note: FORMAT is intended for use during program development of new formatter support. It does not require use of the STRUCTURE(cbname) attribute parameter.

MODEL(name)
MODEL(name) identifies the user-written control block model to be used to format the data. [z/OS MVS IPCS Customization] describes how to use formatting models.

Notes:
1. MODEL is intended for use during control block model development of new formatter support. It does not require use of the STRUCTURE(cbname) attribute parameter.
2. MODEL does not influence how IPCS resolves the data description. If a MODEL is used in resolution, it is the one that would have been used to support formatting STRUCTURE(cbname) except for this override.
3. When MODEL(name) supplies a control block length, the length is compared with the default length generated by the data-descr, and the longer of the two lengths is used during formatting.

VIEW(fieldlist)
VIEW sets the view control field of the format parameter. Values for fieldlist can be any combination of the following options:

hex value
A 4-digit hexadecimal value that displays a particular field you have defined in your model.

ALL
Displays all the control block fields.

DEFINED
Displays only the defined control block fields and is the default.

FLAGS
Displays significant bits in the flag bytes with explanations.

KEYFIELDS
Displays the key fields of defined control blocks.

LINK
Displays the control block linkage field and uses it to display attached blocks.

If VIEW is not specified, CBFORMAT uses a default of VIEW(DEFINED).

Return codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>04</td>
<td>Attention, subcommand completed with a condition that may be of interest to you.</td>
</tr>
<tr>
<td>08</td>
<td>Error, subcommand encountered an error condition that may be of interest to you.</td>
</tr>
<tr>
<td>12</td>
<td>Severe error, an error condition or user request forced early end to the subcommand processing.</td>
</tr>
<tr>
<td>16</td>
<td>Ending error, CBFORMAT did not recognize the control block type specified with the STRUCTURE parameter.</td>
</tr>
</tbody>
</table>
CBFORMAT Subcommand

Example 1
Format the CVT.
- Action
  COMMAND ===> cbformat cvt structure(cvt)
- Result
  This example formats the CVT. (No display is shown here because of the
  control block's size.) Note that the STRUCTURE parameter can be omitted
  from this example because IPCS always defines the CVT as a symbol and
  has STRUCTURE as part of its definition. If a symbol is defined in the IPCS
  symbol table and if that symbol has the STRUCTURE attribute assigned, the
  STRUCTURE parameter does not need to be specified.

Example 2
Format the CSD.
- Action
  COMMAND ===> cbformat f632d0. structure(csd)
- Result
  CBFORMAT generates the following formatted control block with offsets.

```
CSD: 00F632D0
+0000 CSD...... CSD  CPUJS.... 8000  CHAD..... 0000
+0008 CPUAL.... 8000  CPUOL.... 0001  SCFL1.... 00
+000C SCFL2.... 00  SCFL3.... 00  SCFL4.... 00
+0010 AXPAL.... 0000  AXPOL.... 0000  MF1CP.... 0000
+0016 ACR...... 00  FLAGS.... 80  MAFF..... 00000000 00000000
+0020 00000000 00000000 00000000 00000000 00000000 00000000
+0038 00000000 00000000 00000000 00000000 00000000 00000000
+0050 00000000 00000000 00000000 00000000 00000000 00000000
+0068 RV044.... 0000  DDRCT.... 0000  GDCC..... 00000001
+0070 GDINT.... 00000001  GDTOD.... 00000001  TCNT..... 00000000
+007C UCNT..... 00000000  MASK.... 80004000 20001000 08000400 02000100
+0090 00800040 00200010 00080004 00020001
+00A0 IOSID.... 00  IOML..... 02  CPUVF.... 0000
+00A8 CMT...... 019C5708
```

Example 3
Format a captured unit control block (UCB).
- Action
  COMMAND ===> cbformat 006f8028. structure(ucb)
- Result
  CBFORMAT generates the following formatted UCB with offsets. The Actual
  UCB Common Segment Address field is useful when you input a captured
  UCB address and want to learn the UCB's actual address. In this example,
  the captured UCB provides a view of the actual UCB at address 01D0E028.
Example 4

Format a base UCB of a parallel access volume.

- Action
  
  COMMAND ===> cbformat 00F0B808. structure(ucb)

- Result
  
  CBFORMAT generates the following formatted base UCB with offsets. After the formatted base UCB, the report provides information about each alias UCB associated with the base UCB. The information includes the alias UCB's device number, address, and whether it is available for I/O requests. In this example, the alias UCB with device number 01BC at address 01D42448 is not available for I/O requests.
CBFORMAT Subcommand

UCBPRFIX: 00F0B800
-0008 LOCK..... 00000000 IOQ..... 00FC1800

UCBOB: 00F0B808
+0000 JBNR.... 00 FL5..... 88 ID...... FF
+0003 STAT..... 84 CHAN..... 01B0 FL1..... 40
+0007 FLB..... 00 NXUCB..... 00000000 WGT..... 08
+000D NAME..... 1B0 TBYT1.... 30 TBYT2.... 30
+0012 DVCLS..... 20 UNTYP..... 0E FLC..... 00
+0015 EXTP..... F0B7E0 VTOC..... 00001000 VOLI..... 3381B0
+0022 STAB..... 04 DMCT..... 00 SQC..... 00
+0025 FL4..... 00 USER..... 0000 BASE..... 00F0B608
+002C NEXP..... 01D41F88

UCBCMXT: 00F0B7E0
+0000 ETI..... 00 STI..... 00 FL6..... 09
+0003 ATI..... 40 SNSCT..... 20 FLP1..... 22
+0006 SLSI..... 00 FL7..... 00 TEXT..... 01D54D38
+000C CHMRM..... 00 SATI..... 00 ASID..... 0000
+0011 WTOID..... 000000 DDF..... 00FCA728 CLEX..... 00F0B780
+001C OCTOF..... 0000

UCBXPX: 01D54D38
+0000 RSTEM..... 00 MIHKY..... 04 MIHTI..... 00
+0003 HOTIO..... 40 IQF..... 00000000 IOQL..... 00000000
+000C SLSA..... 001 SCLHS..... 0029 PMC1..... 289C
+0012 MBL..... 0049 LPM..... 00 LPUM..... 00
+0017 PIM..... 00 CHPID..... 60700000 00000000
+0020 LEVEL..... 01 IOSR..... 00 IOTKY..... 00
+0023 MIHFG..... 00 LVMSK..... 00000001

Actual UCB Common segment address 00F0B808

Device is dynamic

Base UCB of a parallel access volume

Base UCB has usable alias UCB 01B4 at address 01D42188

Base UCB has usable alias UCB 01B8 at address 01D422E8

Base UCB has unusable alias UCB 01BC at address 01D42448

Example 5
Format an alias UCB of a parallel access volume.
- Action
  COMMAND ===> cbformat 01d422e8. structure(ucb)
- Result
  CBFORMAT generates the following formatted alias UCB with offsets. After
the formatted alias UCB, the report states whether the alias UCB is available
for I/O requests and provides information about the base UCB.
CBSTAT subcommand — obtain control block status

Use the CBSTAT subcommand to analyze a specific control block. IBM provides exit routines that process ASCBs and TCBs; the exit routines are specified by parmlib members embedded in the BLSCECT parmlib member. CBSTAT generates a report for ASCBs that encompasses address space level information. Similarly, CBSTAT generates a report for TCBs that contains task level information about control blocks other than the TCB.

You can also use CBSTAT to get information about resource initialization modules (RIMs) that fail during IPL/NIP processing. Specify the STRUCTURE attribute parameter, but instead of a control block name, specify STORESTATUS. CBSTAT returns the name of the failing RIM(s) with corresponding abend and reason codes. (See the example on 5-37.)

IPCS may issue the accompanying messages when:

- No CBSTAT report is generated.
  
  BLS01040I No errors were detected by the CBSTAT exits

- CBSTAT does not analyze a requested control block.
  
  BLS01041I The CBSTAT exits defined in BLSCECT do not process  
  STRUCTURE(yyyyyyyy)
CBSTAT Subcommand

where yyyyyyyy is the name of the specified control block that CBSTAT does not analyze, such as the ASXB.

- The CBSTAT subcommand syntax check fails. This may occur when the address for the requested control block is not in virtual storage or when the STRUCTURE parameter is omitted.

BLS01043I CBSTAT requires the specification of a STRUCTURE in virtual storage

- The specified address cannot be accessed.

BLS18100I adr-space adr NOT AVAILABLE

- The control block identified in the STRUCTURE parameter fails the validity check.

BLS18058I Errors detected in STRUCTURE(name) at ASID(n) address

To perform its processing, the CBSTAT subcommand uses the CBSTAT service. This service is IBM-supplied and can be used when writing your own dump exit. See z/OS MVS IPCS Customization for information about these services and for information about writing CBSTAT exits.

Syntax

```
CBSTAT data-descr
```

Parameters

data-descr

Specifies the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (see note below)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

Note: The STRUCTURE(cbname) attribute parameter is required; all other attribute parameters are optional. The following values are valid for cbname:

- ASCB
- CSRCPOOL
- SSRB
- STORESTATUS
- TCB

Return codes
CBSTAT Subcommand

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>04</td>
<td>Attention, subcommand completed with a condition that may interest you.</td>
</tr>
<tr>
<td>08</td>
<td>Error, subcommand encountered an error condition that may interest you.</td>
</tr>
<tr>
<td>12</td>
<td>Severe error, no CBSTAT exits exist for the requested control block type or user request forced early end to the subcommand processing.</td>
</tr>
<tr>
<td>16</td>
<td>Ending error, the identified control block failed the validity check.</td>
</tr>
</tbody>
</table>

Example 1
Analyze the ASCB in the master scheduler address space.

- Action
  COMMAND ===> cbstat ascb1 structure(ascb)

- Result
  CBSTAT generates the following output for the master scheduler address space, after accessing and validity checking the ASCB. IPCS invokes the CBSTAT exits for ASCBs. Note that if the symbol, ascb1, is defined in the IPCS symbol table and if that symbol has the STRUCTURE attribute defined, the STRUCTURE parameter can be omitted from the example.

Example 2
Analyze a TCB at a specified address.

- Action
  COMMAND ===> cbstat 7fa030. structure(tcb)

- Result
  CBSTAT generates the following output for the specified TCB. IPCS invokes the CBSTAT exits for TCBs.

Example 3
Analyze an ASCB at a specified address.

- Action
  COMMAND ===> cbstat f62180. structure(ascb)

- Result
  CBSTAT generates the following output for the ASCB.

Example 4
View data about failing NIP RIMs.

- Action
CBSTAT Subcommand

COMMAND ===> cbstat structure(storestatus)

- Result
  
  CBSTAT generates the following output for the ASCB.

STATUS FOR STRUCTURE(STORESTATUS) AT 00FD7100 NOCPU ASID(X'0001')
IEA410011 NIP RIM IEAVNP11 has failed
IEA410021 ABEND=BC4 REASON=04

CLOSE subcommand — release resources in use by IPCS

Use the CLOSE subcommand to end the use of a source or data set by IPCS. CLOSE can end the use of the following:

- Dump data sets
- Trace data sets
- User dump directory
- Sysplex dump directory (for users with access authority)
- Central storage
- Print and table of contents (TOC) data sets

Note: When you end an IPCS session, IPCS automatically closes these data sets, except the sysplex dump directory.

See z/OS MVS IPCS User's Guide for information about closing the print and TOC data sets.

Syntax

CLOSE { ALL } { ACTIVE|MAIN|STORAGE } { DSNAME(dslist)|DATASET(dslist) } { FILE(ddlist | IPCSDDIR)|DDNAME(ddlist) } { PATH(path-name ...) } [ CONDITIONALLY | UNCONDITIONALLY ] [ PRINT ]

-------- SETDEF-Defined Parameter -------------------------
Note: You can override the following SETDEF parameter.
See “SETDEF subcommand — set defaults” on page 5-232

Parameters

ALL
ACTIVE or MAIN or STORAGE
DSNAME(dslist) or DATASET(dslist)
FILE(ddlist | IPCSDDIR) or DDNAME(ddlist)
PATH(pathname)

Specifies one or more source or print data sets to be closed. If you specify ALL and other source parameters, IPCS processes CLOSE ALL and ignores the other source parameters. If you omit these parameters, IPCS closes your current source data set.

ALL directs IPCS to close all data sets it is using.

ACTIVE, MAIN, or STORAGE directs IPCS to release resources that were needed to access the central storage that was specified as the source.
CLOSE Subcommand

DSNAME or DATASET specifies one or more names of cataloged data sets that IPCS is to close. The CLOSE subcommand closes the data sets in the order in which they are specified.

FILE or DDNAME specifies one or more ddnames of data sets that IPCS is to close. The CLOSE subcommand closes the data sets in the order in which they are specified.

When specifying more than one data set name or ddname, separate the names with a comma or a blank.

PATH specifies one or more paths of a file or directory on a z/OS UNIX file.

CLOSE FILE(IPCSDIR) indicates that you want to close your current dump directory. You have to specify its ddname; specifying a range for ddlist does not include your dump directory.

Default Values: You can change your current dump directory by closing it and opening another. This substitution has no effect on the local or global default values. IPCS establishes the local and global defaults when a session starts, using defaults from the dump directory available when the session started.

If you update your local or global defaults, IPCS records the updated defaults in your current dump directory. Depending on when you make the update, the updated dump directory will be the original directory used when the session started or the substitution dump directory.

CONDITIONALLY or UNCONDITIONALLY
Determine how IPCS should handle a data set that is already closed when the CLOSE subcommand is processed.

For CONDITIONALLY, IPCS does not issue messages about the data set being closed.

For UNCONDITIONALLY, IPCS issues messages about the data set being closed. UNCONDITIONALLY is the default.

PRINT
PRINT directs IPCS to close the print data set and the table of contents (TOC) data set, if it is open. In the process of doing a CLOSE PRINT, the default message routing parameter is set to NOPRINT so that subsequent subcommands do not attempt to write to a closed data set.

Support of dump directory substitution
- IPCS supports substitution when the change of dump directories is made while you are not using the IPCS dialog.
- IPCS supports substitution while you are using the IPCS dialog when the dialog activity is not using the original dump directory.
- IPCS does not allow substitution while you are using the IPCS dialog when the dialog activity is using the original dump directory. The reason is that unpredictable errors can potentially damage the new directory, because IPCS has data from the original directory and the data is not necessarily present in the new directory.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the CLOSE subcommand.

Example
CLOSE Subcommand

Close the TOC data set.

- Action
  COMMAND ===> close print
- Result
  Both the TOC and print data sets are closed. Note that when you end an
  IPCS session, IPCS closes both of these data sets automatically.

COMCHECK subcommand — analyze communications task data

Use the COMCHECK subcommand to generate reports about the attributes and
status of the communications task (COMMTASK) at the time of a dump. You can
request information for the following:
- MCS consoles
- Extended MCS consoles
- System console
- Subsystem console
- SMCS console
- Device independent display operator console support (DIDOCS) resident display
  control modules (RDCM)
- DIDOCS pageable display control modules (TDCM)
- Message queues and console management

You can select information for one or all MCS consoles and RDCM, TDCM, and
UCME control blocks. You can request the addresses of control blocks or
formatting of the blocks.

See z/OS MVS Diagnosis: Reference for examples of COMCHECK reports and more
information about diagnosing problems with communications task.

Syntax
Parameters

Report type parameters

Use these parameters to select the type of report. Specify only one. If you omit a report type parameter, the default is MCSINFO.

MCSINFO

Requests summary communications task information for console activity. MCSINFO analyzes the control blocks used to queue messages and manage consoles. MCSINFO produces the following statistics:
- The number of queued messages in the system at the time of the dump.
- The WTO limit (MLIM) in the dumped system.
- The number of messages that are queued to each console.
- Pending WTOR messages.

MCSINFO is the default when COMCHECK is specified without any other parameters.

DATABLKS[(LIST | address)]

Requests summary information that the IBM Support Center might request for problem determination.

ID(iiiiiiii)

Requests summary information for a console. Specify the console's 4-byte ID assigned by the system.

LISTNAMES(keyname)

Requests a list of extended MCS console names defined to a 1- to 8-character keyname.
**COMCHECK Subcommand**

**NAME(nnnnnnnn)**
Requests summary information for a console. Specify the console’s 2- to 8-character symbolic name.

**NAMELIST**
Requests a list of all console names defined in a sysplex.

**RDCM[(ALL | LIST | address)]**
Requests summary control block information for RDCMs.
- **ALL**
  - Gives the status of all active and defined RDCMs.
- **LIST**
  - Lists the address of each RDCM in the dump.
- **address**
  - Gives the status of one RDCM at the specified address.

**SBC**
Requests information about the delayed issue queue and additional information that the IBM Support Center might request for problem determination. It formats the supplemental branch entry console control block (SBC).

**SYSCONS**
Requests information about the status of the system console, including:
- The console name and ID
- The console’s attributes
- The console’s availability
- Message suppression for the console

**SYSPLEX[(CNTRLMEM | SYSMEM)]**
Requests summary information for the members of the sysplex. SYSPLEX with no delimiter prints the current number of sysplex members, the maximum number of members allowed in this sysplex, and additional information the IBM Support Center might request for problem determination.

- **CNTRLMEM**
  - Requests information for each sysplex control member that the IBM Support Center might request for problem determination.
- **SYSMEM**
  - Requests the names of the systems defined to the sysplex and additional information the IBM Support Center might request for problem determination.

**TDCM[(ALL | LIST | address)]**
Requests summary control block information for TDCMs.
- **ALL**
  - Gives the status of all active and defined TDCMs.
- **LIST**
  - Lists the status of each TDCM in the dump.
- **address**
  - Gives the status of one TDCM at the specified address.

**UCM**
Requests summary control block information for the unit control module (UCM) base, prefix, and extension.

**UCME[(ALL | LIST | address)]**
Requests the status of an MCS, SMCS, or subsystem console at the time of the dump. It formats the unit control module individual device entries (UCMEs).
### COMCHECK Subcommand

**ALL**
Gives the status of all active and defined MCS, SMCS, and subsystem consoles. It formats all console information.

**LIST**
Lists the address of each UCME in the dump.

**address**
Gives the status of one MCS, SMCS, or subsystem console. It formats the UCME at the specified address.

**UPDATES((ALL | LIST | address))**
Requests summary information that IBM might request for problem determination.

**Return codes**
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the COMCHECK subcommand.

**Example**
Find the status of an MCS console at the time of a dump.

- **First Action**
  Obtain a list of UCME addresses by entering on the command line:
  
  ```
  COMMAND ===> COMCHECK UCME
  ```

- **Result**
  COMCHECK produces a list of UCME addresses, similar to the following example:

  ```
  ADDRESS OF ALL ACTIVE UCMES ON SY1
  CONSOLE NAME ADDRESS TYPE
  -------------- ------- ----
  MCSY13E0 00FD64C0 MCS
  MCSY13D1 00FD6510 MCS
  MCSY13D2 00FD6560 MCS
  SUBSYS2 00FD6FBEC SUBSYSTEM
  ```

- **Second Action**
  To look at the UCME at address 00FD64C0, enter on the command line:
  
  ```
  COMMAND ===> COMCHECK UCME(00FD64C0)
  ```

- **Result**
  COMCHECK produces a report for the MCS console represented by that UCME. [z/OS MVS Diagnosis: Reference](#) shows a sample UCME report and explains the contents of the fields.

### COMPARE subcommand — compare dump data

Use the COMPARE subcommand to compare two data items. COMPARE makes the results of the comparison known to a CLIST or REXX exec by a return code and, optionally, makes the results known to you by a message. Each data item can be specified as a value or as the address of a data item.

- **Numeric comparison**
  Numeric comparison is performed if the PAD parameter is specified and both items to be compared have POINTER, SIGNED, or UNSIGNED data types.
COMPARE subcommand

- Numeric comparison between two unsigned (POINTER or UNSIGNED data types) items is accomplished by providing leading zero bytes to pad both items to a fullword (32-bit) precision and comparing the unsigned results.
- Numeric comparison between two SIGNED items is accomplished by propagating the sign bit to pad both items to a fullword (31-bit) precision and comparing the signed results.
- Numeric comparison between a SIGNED item and one that is unsigned is reduced to the following cases:
  - If the SIGNED value is negative, that number is less than any unsigned value.
  - Otherwise, a positive SIGNED value may be treated as unsigned, and the comparison completed as though unsigned numeric comparison had been requested.

• String comparison

String comparison is performed whenever numeric comparison is inappropriate. Comparison of strings whose lengths differ may be performed in two ways:
- The longer string may be truncated to the length of the shorter before comparison (TRUNCATE parameter).
- The shorter string may be padded to the length of the longer before comparison (PAD parameter). The character used for padding may be explicitly specified. If it is not, an EBCDIC blank (X'40') is used for data described as CHARACTER data or data described using a general value of types C or T. If the data was described using a general value associated with ISO-8 ASCII CHARACTER data (types Q or S), padding is performed using an ISO-8 ASCII blank (X'20'). Padding with a null character (X'00') is used for all other types of data.

• Syntax

```plaintext
{COMPARE | COMP}
  [ [ data-descr | ADDRESS(X) | (VALUE(value)) ]
    [ WITH [[(data-descr) | (ADDRESS(X)) | (VALUE(value))] ]
    [ LIST | NOLIST ]
    [ MASK(mask) | NOMASK ]
    [ PAD[(value)] | TRUNCATE ]
```

------- SETDEF-Defined Parameters ��数------------------------

Note: You can override the following SETDEF parameters. See

“SETDEF subcommand — set defaults” on page 5-232

  [ FLAG(severity) ]
  [ PRINT | NOPRINT ]
  [ TERMINAL | NOTERMINAL ]
  [ TEST | NOTEST ]

• Parameters

data-descr
ADDRESS(X)
COMPARE subcommand

**VALUE(value)**

Specifies the first operand for the comparison. The length of the comparison is determined by the length of the data described by this parameter or by the mask, if you specify one. The maximum length is $2^{31}$ bytes or, if you use a mask, 256 bytes.

The `data-descr` specifies the data description parameter, which designates dump data as the first operand for the comparison. The data description parameter consists of five parts:
- An address (required when `data-descr` is explicitly specified on the subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

[Chapter 3, “Data description parameter,” on page 3-1](#) explains the use and syntax of the data description parameter.

**VALUE(value)** designates a literal value as the first operand. Value may be specified as a:
- Positive integer
- Signed integer
- General value

See [Chapter 2, “Literal values,” on page 2-1](#) for more information, the syntax and examples.

If you specify **VALUE**, you cannot specify data description parameters with it. They will be ignored and processing will continue. IPCS issues this message:

```
BLS18032I Operand n uses both the value parameter and data description parameters. The data description parameters are ignored.
```

where n is either 1 or 2, to indicate which operand is in error.

If you omit this parameter, the default is `ADDRESS(X)`, the most recently accessed address.

**WITH [(data-descr) | (ADDRESS(X)) | (VALUE(value))]**

Specifies the second operand for the comparison.

**Note:** The rules for specifying the **VALUE** parameter on this operand are the same as those for specifying **VALUE** on the first operand.

**LIST or NOLIST**

LIST directs the subcommand to display the results of the comparison at your terminal.

NOLIST suppresses the display of the results of the comparison at your terminal.

**MASK(mask) or NOMASK**

**MASK(mask)** defines a value that is logically ANDed with both compare operands before performing the comparison. The mask must be the same size as the data items being compared. The mask value must be a general value. See [Chapter 2, “Literal values,” on page 2-1](#) for information about specifying a general value.

**NOMASK** suppresses masking.
COMPARE subcommand

**PAD[(value)] or TRUNCATE**

PAD authorizes numeric comparison and comparison of operands of differing lengths by padding the shorter compare operand before comparison.

PAD(value) specifies a 1-byte value to be used to pad data before comparison. Either a character (C'c') or a hexadecimal (X'xx') value may be specified.

TRUNCATE specifies that a string comparison be performed and that comparison of operands of differing length be performed by truncating the longer compare operand to the length of the shorter before comparison.

- **Return codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>The operands are equal.</td>
</tr>
<tr>
<td>04</td>
<td>The first operand is low.</td>
</tr>
<tr>
<td>08</td>
<td>The first operand is high.</td>
</tr>
<tr>
<td>12</td>
<td>The comparison is incomplete.</td>
</tr>
</tbody>
</table>

- **Example**

In the BLSCCOMP CLIST, instructions find the address space vector table (ASVT) from field CVTASVT in the communications vector table (CVT). A COMPARE subcommand compares the ASVT identifier field, ASVTASVT, with the character string 'ASVT'. If the comparison returns a nonzero completion code, the CVTASVT field that points to the ASVT might be damaged. The COMPARE subcommand is:

```
COMPARE ADDRESS(&ASVT+200) CHARACTER LENGTH(4)/* ASVTASVT */+
    WITH(VALUE(C'ASVT')) /* Expected, normal value */+
```

See the BLSCCOMP member in the IBM-supplied SYS1.SBLSCLI0 library for the complete listing.

COPYCAPD subcommand — copy captured dump data

Use the COPYCAPD subcommand to generate a report showing all captured dumps present in a standalone dump and then copy the captured dump data to an output data set. The generated report contains the following information:

- An ordinal number arbitrarily associated with the captured dump.
- The time when the dump capture process was started.
- The dump title.
- If present, the name of the dump data set to which part of the captured dump was written.

**Syntax**
COPYCAPD subcommand

COPYCAPD
{ captured-dump-number }
{ OUTDSNAME(dsname)|OUTDATASET(dsname)|ODS(dsname) }
{ OFILE(ddname)|OUTDDNAME(ddname) }
[ SPACE(nnnn[,mmmm]) ]

------ SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232
[ ACTIVE|MAIN|STORAGE ]
[ DSNAME(dlist)|DATASET(dsname) ]
[ FILE(ddname)|DDNAME(ddname) ]
[ PATH(hfsPath) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

captured-dump-number
Selects the captured dump to be copied. If this is omitted, COPYCAPD only produces a report describing captured dumps.

OUTDSNAME(dsname) or OUTDATASET(dsname) or ODS(dsname)
OFILE(ddname) or OUTDDNAME(ddname)
Specifies the output data set into which the dump is to be copied. An output data set must be specified.

OUTDSNAME, OUTDATASET, or ODS specifies the name of the output data set. After copying, IPCS closes and deallocates the data set.

If the designated data set exists, it must be cataloged. It is dynamically allocated and used by COPYCAPD. If the data set resides on a volume that is not mounted as RESIDENT or RESERVED, MVS MOUNT authorization is required.

If the designated data set does not exist, the system allocates a new data set with the specified name and the defaults RECFM=FBS, LRECL=4160, and system-determined BLKSIZE are used.

OFILE or OUTDDNAME specifies the ddname of the output data set. This data set must be allocated by JCL or the TSO/E ALLOCATE command before COPYCAPD is entered.

After copying, COPYCAPD closes the data set, but does not directly deallocate it. You may use the JCL option FREE=CLOSE to release the data set at the earliest possible moment.

SPACE(nnnn[,mmmm])
Specifies the primary space allocation, nnnn, and the secondary space allocation, mmmm, if a new data set is created. Space is allocated in units of 4160-byte dump records. Excess space is released at the completion of COPYCAPD processing.
COPYCAPD subcommand

If you omit this parameter, both the primary allocation and the secondary allocation defaults are 1500 records. If only the primary allocation is specified, the secondary allocation defaults to the primary allocation.

Return codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>End of file reached. The input data set has been closed and a dump has been copied to the output data set.</td>
</tr>
<tr>
<td>16</td>
<td>Subcommand processing ended after detection of a problem in the IPCS processing environment.</td>
</tr>
<tr>
<td>20</td>
<td>Subcommand processing ended as a result of an attention interruption you generated. The input data set has been closed. The output data set has been loaded with part of a dump.</td>
</tr>
</tbody>
</table>

Example 1

Request a report only. Normally, an IPCS user will first request a report to determine the available dump titles and time stamps. Once that information has been evaluated, the user can request another COPYCAPD subcommand to select a specific dump.

- Action
  
  COMMAND ===> COPYCAPD

- Result

Example 2

Request to report and copy the captured dump.

- Action
  
  COMMAND ===> COPYCAPD 1 SPACE(5000) OUTDSN(my.captured.dump)

- Result
COPYDDIR subcommand — copy source description from dump directory

Use the COPYDDIR subcommand to copy one or more source descriptions. A description is a reference to a source of data, a dsname, ddname, or path. Additional, optional records may also be present and copied:

- Some records may help you understand the significance of the source.
- Other records may enable IPCS to assist in analysis and formatting of its contents.
- A few records may serve dual roles, symbols allowing you to refer to important data by name and allowing IPCS to locate the same important data and check its validity just once in the course of analyzing a dump.

COPYDDIR can perform three similar types of operations:

1. Copy operations transcribe records to the current session dump directory from another dump directory. You designate the source directory via INDDataset, INDDNAME, or aliases of those keywords. Multiple descriptions may be selected for transcription in a single operation.

2. Import operations transcribe records to the current session dump directory from a RECFM=VB data set. You designate the source RECFM=VB data via INDDataset, INDDNAME, or aliases of those keywords. No selectivity is supported. One description is copied.

3. Export operations transcribe records from either the current session dump directory or another dump directory to a RECFM=VB data set.
   - You imply the use of the current session dump directory by omitting INDDataset, INDDNAME, and their aliases.
   - You designate the source directory via INDDataset, INDDNAME, or aliases of those keywords.
COPYDDIR subcommand

You designate the target RECFM=VB data set via the EXPORT keyword. The same selection options supported for COPY may be used to select a single description to be exported.

The main purpose of the COPYDDIR subcommand is to place the source description of a dump or trace into your current user dump directory, so that you can format and analyze the dump or trace.


Syntax

COPYDDIR [ INDATASET(dsname)|INDSNAME(dsname) ]
[ INFILE(ddname)|INDDNAME(ddname) ]
[ EXPORT {(DSNAME(dsname))|(DATASET(dsname))} ]
{ (FILE(ddname))|(DDNAME(ddname)) ]
[ SUMMARY | NOSUMMARY ]
[ DSNAME(dslist)|DATASET(dslist) ]
[ FILE(ddname-range-list)|DDNAME(ddname-range-list) ]
[ PATH(path-name ...) ]

Note: You can override the following SETDEF parameters. See “SETDEF subcommand — set defaults” on page 5-232

[ TEST | NOTEST ]

Parameters

Use a DSNAME, DATASET, FILE, DDNAME, or PATH parameter to specify the source for the source description to be copied. You can request copying of more than one source description. Note that you can also use a SCREEN keyword with INDATASET or INFILE while the IPCS dialog is active in order to display the COPYDDIR inventory panel for the input dump directory selected.

INDATASET(dsname) or INDSNAME(dsname)
INFILE(ddname) or INDDNAME(ddname)

Specifies the input for copy or import operations. One of these parameters is required except when the EXPORT option is selected. EXPORT uses the current session directory as a source of records when neither input dsname nor input ddname are specified.

Note: Do not specify your current dump directory. Do not specify IPCSDDIR as the ddname.

INDATASET or INDSNAME specifies the input directory by its data set name.
INFILE or INDDNAME specifies the ddname of the input data set.

EXPORT(DSNAME(dsname)) or EXPORT(DATASET(dsname))
EXPORT(FILE(ddname)) or EXPORT(DDNAME(ddname))

Specifies a RECFM=VB data set to receive dump directory records pertaining to one source. RECFM=VB data sets must have a LRECL of 3076 or larger.
COPYDDIR subcommand

**SUMMARY** or **NOSUMMARY**

SUMMARY indicates that a summary line containing the total number of
dump descriptions copied should be displayed and is the default.

NOSUMMARY suppresses the summary line unless one or more source
descriptions were not copied, for example, if error conditions exist, or if the
description already exists in the output directory. You might use
NOSUMMARY when running COPYDDIR within a CLIST or REXX exec.

**DSNAME(dslist)** or **DATASET(dslist)**

**FILE(ddname-range-list)** or **DDNAME(ddname-range-list)**

Specifies one or more data sets for the source descriptions to be copied. If
one of these parameters is not specified, the default is the SETDEF-defined
default source data set.

DSNAME or DATASET specifies the data set name or a list of data set
names of cataloged data sets. The *dslist* can include a wildcard character (*)
to represent any name. A data set name can contain a single asterisk in
place of any qualifier except the first. For example, DSNAME (A,*C)
specifies all names with 3 qualifiers that have A as the first qualifier and C
as the third qualifier.

FILE or DDNAME specifies the ddname, a list of ddname, or a range of
ddnames for the data sets. For example, FILE(A:C) specifies all ddnames
from A to C, including A, AA, ABC, B, C, and so on.

When specifying more than one data set name or ddname, separate the
names with commas or blanks. When specifying a range of ddnames,
separate the first and last ddname with a colon.

PATH specifies the path-name or list of path-names of a file or directory on
a z/OS UNIX file.

**Return codes**

See "Standard subcommand return codes" on page 5-2 for a description of the
return codes produced by the COPYDDIR subcommand.

**Example 1**

Copy the source description for the dump data set MY.DUMP from the sysplex
dump directory, SYS1.DDIR, to your current user dump directory.

- **Action**
  
  COMMAND ===> COPYDDIR INDSNAME(SYS1.DDIR) DSNAME(MY.DUMP)

- **Result**
  
  IPCS copies the source description for MY.DUMP from SYS1.DDIR
  into your current user dump directory and displays a summary of the
  processing.

**Example 2**

Copy source descriptions for multiple data sets to your current user dump
directory:

- **Action**
  
  COMMAND ===> COPYDDIR FILE(W:X) DSNAME(MY.DUMP2) INDSNAME(DUMPDIR)

- **Result**
  
  IPCS copies the source descriptions from the DUMPDIR directory for all data
  sets beginning with W or X and data set MY.DUMP2 into your current user
dump directory. IPCS displays a summary of the processing.
COPYDUMP subcommand

COPYDUMP subcommand — copy dump data

Use the COPYDUMP subcommand to copy a single unformatted dump from one data set to another. COPYDUMP also allows you to:

- Extract a single dump from a string of dumps in a data set
- Copy the records of multi-volume SADMP data sets, retaining the priority order used during dumping.
- Reunite the portions of dump data that was previously split.
- Obtain a summary of all the dump titles in the data set
- Reduce the size of a dump by copying dump records from a specified list of address spaces

Applications, such as IMS, can write several contiguous SYSMDUMPs in a single data set. COPYDUMP can list the title of each dump in the data set and extract one of the dumps from the data set.

SADMP to DASD uses the volumes of multi-volume data sets in parallel, writing to each as rapidly as it is prepared to accept dump records. COPYDUMP recognizes this and creates a copy in which the first data captured by SADMP appears in the first records written without regard to which volume blocks were written.

SADMP to DASD can exhaust the pre-allocated space associated with the initial data set. You can designate second and subsequent data sets to cause a complete SADMP to be written. COPYDUMP accepts a list of data set names and can create a single dump data set for analysis from the several dump data sets to which SADMP wrote.

You can use filtering options to produce a copy that has less records than the original dump. This is particularly useful with a stand-alone dump. Specify ASIDLIST, JOBLIST, or EASYCOPY to select ASIDs that are useful for your dump analysis, leaving ASIDs that are usually not needed to analyze a problem. The following types of copies may be produced:

- A primary copy, filtered if ASIDLIST, JOBLIST, or EASYCOPY options are specified. (Note that these filtering options will remove available pages of the system dumped by SADMP.)
- A FULL copy.
- A COMPLEMENT copy that contains those dump record removed from the primary copy via filtering.

Each type of copy is optional. See the OUTDSNAME and OUTDDNAME options for more information. See the specific filtering options regarding their use to balance importance against size of the copy.

The output data set, into which the dump is copied, is closed after copying is completed. The input data set, from which the dump was copied, is closed when an end of file is encountered. If COPYDUMP completes without reaching an end of file, an option determines whether the input data set is closed or remains open. If it remains open, the input data set is positioned for another COPYDUMP subcommand to resume processing with the next dump.

Syntax
COPYDUMP subcommand

COPYDUMP
{ OUTDSNAME(outds-spec) | OUTDATASET(outds-spec) | ODS(outds-spec) }
{ OUTFILE(outdd-spec) | OFILE(outdd-spec) | OUTDDNAME(outdd-spec) }
{ INDSNAME(dsname_list) | INDATASET(dsname_list) | IDS(dsname_list) }
{ INFILE(ddname_list) | IPCSINDD | IFILE(ddname_list) | IPCSINDD | INDDNAME(ddname_list) | IPCSINDD }[
 DEFAULT ]
{ SPACE(1500,1500) }[
 CLOSE | LEAVE ]
{ ASIDLIST(ddddd[,dddddd]) }[
 JOBLIST(jj[,jj][,jj][,jj][,jj]) ]
{ NOSKIP | SKIP[(nnn|1)|EOF) ] }[
 NOCLEAR | CLEAR ]
{ EASYCOPY }[
]

where
outds-spec := dsname | [INITIALIZE] | NULLFILE
| [COMPLEMENT(dsname | NULLFILE) ]
| [FULL(dsname | [INITIALIZE]) | NULLFILE]

outdd-spec := ddname | [INITIALIZE] | [COMPLEMENT(ddname) ]
| [FULL(ddname | [INITIALIZE]) ]

----- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
"SETDEF subcommand — set defaults" on page 5-232

[ CONFIRM | NOCONFIRM ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

OUTDSNAME(outds-spec) or OUTDATASET(outds-spec) or ODS(outds-spec)
OUTFILE(outdd-spec) or OFILE(outdd-spec) or OUTDDNAME(outdd-spec)

Specifies the output data set into which the subset dump, complementary
dump and full dump are to be copied. At least one output data set must be
specified, unless SKIP(EOF) is specified; if SKIP(EOF) and any output data
set are both specified, IPCS ignores the output data set. If NULLFILE is
specified for any output dataset, then IPCS ignores that output dataset.
NULLFILE can be specified only for dsnames and not for ddnames.

OUTDSNAME, OUTDATASET, or ODS specifies the name of the output
data set. The COMPLEMENT and the FULL data sets can be specified only
if ASIDLIST or JOBLIST is specified. The COMPLEMENT data set contains
the complement of the subset dump. The FULL data set contains the input
dump specified. If a list of input data sets is specified, the input dump is
merged and written into the FULL data set. After copying, IPCS closes and
deallocates the data set.

If the designated data set exists, it must be cataloged. It is dynamically
allocated and used by COPYDUMP. If the data set resides on a volume that
is not mounted as RESIDENT or RESERVED, MVS MOUNT authorization
is required.

If the designated data set does not exist, the system allocates a new data set
with the specified name and the defaults RECFM=FBS, LRECL=4160, and
system-determined BLKSIZE are used. Use the SPACE parameter to
COPYDUMP subcommand

indicate the amount of space to be allocated. If the SPACE parameter is
omitted, COPYDUMP uses default amounts.

OUTFILE, OFILE or OUTDDNAME specifies the ddname of the output
data set. This data set must be allocated by JCL or the TSO/E ALLOCATE
command before COPYDUMP is entered. The COMPLEMENT and the
FULL data sets can be specified only if ASIDLIST or JOBLIST is specified.
The COMPLEMENT data set contains the complement of the subset dump.
The FULL data set contains the entire dump specified. If a list of input
dumps is specified, the input dump is merged and written into the FULL
data set.

After copying, COPYDUMP closes the data set, but does not directly
deallocate it. You must use the JCL option FREE=CLOSE to release the data
set at the earliest possible moment.

If the INITIALIZE option is specified with any of the output data sets or
ddnames, then IPCS will create the dump directory entries and perform
dump initialization for those output dump data sets. INITIALIZE cannot be
specified when NULLFILE is specified and is not used with
COMPLEMENT data sets.

INDSNAME(dsname_list) or INDATASET(dsname_list) or IDS(dsname_list)
INFILE(ddname_list|IPCSINDD) or IFILE(ddname_list|IPCSINDD) or INDDNAME
(ddname_list|IPCSINDD)

Specifies one or more input data sets from which the dump is copied. If
one of these parameters is not specified, IPCS takes the following actions:
- If an open data set is available, COPYDUMP resumes processing the
  open data set.
- If no open data set is available, COPYDUMP opens the default input
data set, IPCSINDD, and begins processing with the first record.

INDSNAME, INDATASET, or IDS specifies the input data set or a list of
input data sets. The designated data sets must exist and must be cataloged.
After copying, COPYDUMP closes and deallocates the input data sets.

If a prior COPYDUMP subcommand left a designated data set open,
processing of the data set is resumed where it left off. Note that
INDSNAME or INDATASET cannot be used to resume processing of a data
set initially opened using INFILE or IFILE or INDDNAME.

If a designated data set is not open, it is dynamically allocated, opened, and
processed beginning with the first record.

If a designated data set resides on a volume that is not mounted as
RESIDENT or RESERVED, MVS MOUNT authorization is required.

INFILE, IFILE or INDDNAME specifies the ddname, or a list of ddname of
the input data sets. The designated data sets must be allocated by JCL or
the TSO/E ALLOCATE command before COPYDUMP is entered. After
copying, COPYDUMP closes the input data sets, but does not directly
deallocate them. You must use the JCL option FREE=CLOSE to release the
data sets at the earliest possible moment.

If a prior COPYDUMP subcommand left a designated data set open,
processing of the data set is resumed where it left off. Note that INFILE or
IFILE or INDDNAME may not be used to resume processing of a data set
initially opened using INDSNAME or INDATASET.

If a designated data set is not open, it is dynamically allocated, opened, and
processed beginning with the first record.
COPYDUMP subcommand

PATH(path-name ...)
Specifies a path-name or list of path-names of a file or directory on a z/OS UNIX file to be processed.

DEFAULT
Specifies that the output data set is to become the current source. If the subcommand specifies a data set name with a password, the data set name and password become the name of the current source.

IPCS changes the current source in both the local and global defaults. If you omit this parameter, or if the subcommand fails, the current source is not changed in the defaults.

Note: If the output data set is specified by OUTFILE or OFILE or OUTDDNAME, the function of the DEFAULT parameter is nullified.

SPACE(nnnn[,mmmm])
Specifies the primary space allocation, nnnn, and the secondary space allocation, mmmm, if a new data set is created. Space is allocated in units of 4160-byte dump records. Excess space is released at the completion of COPYDUMP processing.

If you omit this parameter, both the primary allocation and the secondary allocation defaults are 1500 records. If only the primary allocation is specified, the secondary allocation defaults to the primary allocation.

CLOSE or LEAVE
CLOSE directs COPYDUMP to close the input data set immediately after the dump has been copied.

LEAVE directs COPYDUMP to allow the input data set to remain open if processing of the subcommand completes before reaching an end of file. The input data set is always closed if COPYDUMP completes after reaching the end of file. If the IPCS session ends, the input data set is automatically closed.

ASIDLIST(asid[:asid])
Specifies ASIDs for the address spaces and their associated data spaces to be copied; dump records for other address spaces and their associated data spaces are not copied.

The asid can be a single ASID or an ASID range. When you specify a range, separate the first and last ASID in the range with a colon.

An ASID can be 1 through 65535. An ASID can be expressed in the notation X'asid' or in decimal. An unqualified number is assumed to be decimal.

Note: No matter what ASID you specify on this parameter, COPYDUMP always copies the dump records for address spaces 1 through 4 to the output data set. Correspondingly, when you analyze a copied dump, you might see common storage for an ASID not specified on the ASIDLIST parameter because common storage is stored in a dump with ASID(X'0001').

JOBLIST(j1[,j2][,j3]...[,jn])
Specifies job names for the address spaces and their associated data spaces to be copied; dump records for other address spaces and their associated data spaces are not copied. The JOBLIST can contain a single job name or a list of job names. When you specify a list, separate the job names with a comma. The job name can be 1 to 8 characters.
COPYDUMP subcommand

**Note:** No matter what job name you specify on this parameter, COPYDUMP always copies the dump records for address spaces 1 through 4 to the output data set. Correspondingly, when you analyze a copied dump, you might see common storage for a job name not specified on the JOBLIST parameter because common storage is stored in a dump with ASID(X'0001').

**SKIP**[(nnn | 1 | EOF)] or **NOSKIP**

SKIP(nnn) specifies the number of dumps, nnn, in the input data set to be skipped before copying begins. Each dump title encountered in the input data set is displayed when it is read.

If you enter SKIP but no number, one dump is skipped.

If you specify SKIP(EOF), COPYDUMP skips to the end of the data set, displaying all dump titles that are encountered during that process; however, no copying is performed. Also, the output data set is not needed if SKIP(EOF) is specified.

NOSKIP specifies that no dumps are to be skipped before copying begins.

**NOCLEAR** or **CLEAR**

NOCLEAR specifies that the input data set should not be cleared after the copy.

CLEAR directs COPYDUMP to clear the input data set after the dump has been copied.

**Note:** Because IPCS allocates the input data set with a disposition of SHR, use caution if the input data set is being used by other users. Do not clear the dump data set while other users are still using it.

**EASYCOPY**

If EASYCOPY is specified, one of following events will occur depending on z/OS release, which produced selected source dump:

- For z/OS V1R10 dump, the JOBLIST and ASID RANGE fields will be ignored, and a JOBLIST entry created with a predefined list of system address space names. The JOBLIST includes the following fourteen job names: ALLOCAS, ANTAS000, ANTMAIN, CATALOG, CONSOLE, DEVMAN, DUMPSRV, IXGLOGR, IOSAS, JESXCF, JES2, JES3, and OMVS.

- For z/OS V1R8 and V1R9 dumps, the JOBLIST and ASID RANGE fields will be ignored, and an ASID entry created with a range of 1 to 20.

**Note:** COPYDUMP always copies the dump records for address spaces 1 through 4 to the output data set. Correspondingly, when you analyze a copied dump, you might see common storage for an ASID not specified above because common storage is stored in a dump with ASID(X'0001').

**CONFIRM** or **NOCONFIRM**

CONFIRM specifies that the subcommand is to request your confirmation before performing the copy operation. The subcommand displays the title of the dump to be copied. It then requests your confirmation.

- If you enter Y, the subcommand copies the dump into the output data set and drops any existing records in the dump directory associated with the output data set.
COPYDUMP subcommand

- If you enter N, the subcommand ends without copying the dump into the output data set, and ignores the DEFAULT parameter, if specified. The LEAVE/CLOSE parameter determines if the input data set is left open.

NOCONFIRM specifies that the subcommand is not to request your confirmation before copying the dump into the output data set and dropping any entries in the dump directory that are associated with the specified dump name.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default (established through SETDEF) for this parameter.

Restriction: When using IPCS in the background or while in the IPCS full-screen dialog, you may not specify CONFIRM. Specify NOCONFIRM either on this subcommand or on the SETDEF subcommand.

Return codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>End of file reached. The input data set has been closed and a dump has been copied to the output data set.</td>
</tr>
<tr>
<td>04</td>
<td>End of dump reached. The input data set has been left open, positioned immediately after the dump copied by this subcommand.</td>
</tr>
<tr>
<td>08</td>
<td>End of file reached before reaching the dump to be copied. (This return code is always produced if SKIP(EOF) is specified and COPYDUMP reaches end of file.)</td>
</tr>
</tbody>
</table>
| 12   | Subcommand processing ended for one of the following reasons:  
|      | • COPYDUMP requested your confirmation and confirmation was not received. The CLOSE option was in effect.  
|      | • The COPYDUMP subcommand cannot be interpreted. No input data set was left open by a prior run of COPYDUMP.  
|      | • You generated an attention interrupt before any COPYDUMP processing. No input data set was left open by a prior run of COPYDUMP.  
|      | • COPYDUMP read an incorrect dump header record as the initial record of the input data set. The CLOSE option was in effect. The input data set has been closed, and the output data set (if any) has not been altered. |
| 16   | Subcommand processing ended after detection of a problem in the IPCS processing environment. |
| 20   | Subcommand processing ended as a result of an attention interruption you generated. The input data set has been closed. The output data set has been loaded with part of a dump. |
| 24   | Subcommand processing ended for one of the following reasons:  
|      | • COPYDUMP requested your confirmation and confirmation was not received. The LEAVE option was in effect.  
|      | • The COPYDUMP subcommand cannot be interpreted. An input data set was left open by a prior run of COPYDUMP.  
|      | • An attention interruption was generated by you during COPYDUMP skip processing. The LEAVE option was in effect.  
|      | • COPYDUMP read an incorrect dump header record as the initial record of the input data set. The LEAVE option was in effect. The input data set has been left open, and the output data set (if any) has not been altered. |
COPYDUMP subcommand

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>An error occurred when COPYDUMP attempted to open the input data set for output with the CLEAR option in effect. The input data set was copied to the output data set, but the input data set was not cleared.</td>
</tr>
</tbody>
</table>

COPYTRC subcommand — copy trace entries or records

Use the COPYTRC subcommand to copy GTF trace records to an output data set from trace data sets or trace buffers in dump data sets. You can also use COPYTRC to copy component trace entries to an output data set from trace data sets or trace buffers in dump data sets.

You can use COPYTRC to:

- Combine trace data sets, or trace entries or records in dump data sets, or both, into a single data set.
- Extract trace entries or records from buffers in SVC and stand-alone dumps.
- Combine trace entries or records from multiple systems. When COPYTRC combines trace entries or records from several systems into a single data set, it marks the system of origin for each trace entry or record in the output data set.
- Extract trace entries or records for a specified list of systems from combined trace entries or records.

You can run COPYTRC by entering the subcommand or using the panels on option 5.3 of the IPCS dialog.

The main function of the COPYTRC subcommand is to aid in processing multiple trace sources. Suppose you have multiple GTF data sets from a run on a single system. Before using GTFTRACE to process all of the trace data, you must combine all GTF trace records into a single data set using COPYTRC.

Notes:
1. To process multiple GTF data sets from multiple systems, you can either:
   - Combine the trace records into a single data set with COPYTRC
   - Keep the trace data sets separate and use the MERGE subcommand to format the traces
2. COPYTRC cannot process GTF trace records and component trace entries at the same time. So, for COPYTRC input sources, specify all GTF trace sources, or all component trace sources, but not a mix of both traces. To see GTF trace records and component trace entries chronologically in a single report, use the MERGE subcommand.

COPYTRC does not have a default input or output data set name or ddname.

After the entries or records are copied, COPYTRC closes both the input and output data sets and displays a summary of the trace entries or records that were copied.

Related subcommands
CTRACE
GTFTRACE
MERGE

Syntax
COPYTRC subcommand

COPYTRC [ TYPE(GTF|CTRACE) ]
{ INDATASET(dslist)|INDSNAME(dslist)|IDS(dslist) }
{ INFILE(ddlist)|INDDNAME(ddlist) }
{ OUTDATASET(dsname)|OUTDSNAME(dsname)|ODS(dsname) }
{ OUTFILE(ddname)|OUTDDNAME(ddname) }
[ SPACE(pppp[,ssss]|50,50) ]

------- Data Selection Parameters -----------------------------
[ OPTIONS((ALL|filters)) ]
[ START(mm/dd/yy,hh.mm.ss.dddddd) ]
[ STOP(mm/dd/yy,hh.mm.ss.dddddd) ]
[ SYSNAME(sysname[,sysname]...) ]

------- SETDEF-Defined Parameters -----------------------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

**TYPE(GTF | CTRACE)**
Specifies the type of trace data to be copied. COPYTRC will copy trace data
generated by either GTF or component traces. These two types of traces
may not be combined.

If the TYPE parameter is omitted, COPYTRC tries to copy GTF trace
records.

**INDATASET(dslist) or INDSNAME(dslist) or IDS(dslist)**
**INFILE(ddlist) or INDDNAME(ddlist)**
Specifies the data sets containing the traces to be copied. Use these
parameters in any combination. All data sets should contain the same type
of trace.

To specify multiple input data sets, use any combination of the following
data sets:
- Trace data sets created by GTF or CTRACE
- Trace data sets created by COPYTRC
- SVC, stand-alone dump, and SYSMDUMP dump data sets

An example of a combination of parameters is:

**COMMAND ===>>** COPYTRC INFILE(GTFDINDD) INDATASET(MY.GTFDATA1,MY.GTFDATA2) ...

**INDATASET, INDSNAME, or IDS** specifies the input data set or sets. When
specifying more than one data set name, separate the names with commas
or blanks. IPCS dynamically allocates each input data set. If a data set is
not open, COPYTRC opens the data set after it is dynamically allocated.
COPYTRC subcommand

Each designated data set must exist and must be cataloged to allow the system to locate it. If a data set resides on a volume that is not mounted as RESIDENT, MVS MOUNT authorization is required.

After copying, IPCS closes and deallocates each data set. When the SETDEF-defined default source is specified as an input data set, IPCS does not close or deallocate the data set.

INFILE or INDDNAME specifies the ddname of the input data set or sets. Before using INFILE or INDDNAME, you must allocate each data set using JCL or the TSO/E ALLOCATE command. IPCS opens the data sets.

When specifying more than one ddname, separate the names with commas or blanks.

When IPCS finishes copying, it closes the data set, but does not directly deallocate it. You can use the JCL FREE=CLOSE to release each data set. When the SETDEF-defined default source is specified as an input data set, IPCS does not close or deallocate it.

OUTDATASET(dsname) or OUTDSNAME(dsname) or ODS(dsname) or OUTFILE(ddname) or OUTDDNAME(ddname) or OFILE(ddname)

Specifies the output data set into which the traces are to be copied. The COPYTRC subcommand must specify an output data set.

OUTDATASET, OUTDSNAME, or ODS specifies the output data set. If the designated data set exists, it is dynamically allocated and used by COPYTRC. The data set must be cataloged. If the data set resides on a volume that is not mounted as RESIDENT or RESERVED, MVS MOUNT authorization is required.

If the designated data set does not exist, the system allocates a new data set with the specified name. Use the SPACE parameter to indicate the amount of space to be allocated. If the SPACE parameter is omitted, COPYTRC uses default amounts.

After the copying, IPCS closes and deallocates the data set.

OUTFILE or OUTDDNAME specifies the ddname of the output data set. Before using COPYTRC, you must allocate this data set using JCL or the TSO/E ALLOCATE command.

After the copying, IPCS closes the data set but does not directly deallocate it.

COPYTRC processing might open and close the output data set more than once. Do not use options on the DD statement, such as RLSE or FREE=CLOSE that conflict with the multiple open and close operations.

SPACE(pppp[,ssss] | 50,50)

Specifies the number of tracks for the primary space allocation, pppp, and the secondary space allocation, ssss for a new data set. The system releases excess space at the completion of COPYTRC processing.

If you omit this parameter, both the primary allocation and the secondary allocation defaults are 50 tracks. If only the primary allocation is specified, the secondary allocation defaults to the primary allocation.

Data Selection Parameters

All data selection parameters are optional. If specified, COPYTRC copies only trace entries or records that meet the specified data selection requirement.
OPTIONS((ALL | filters))
Specifies filtering options for a particular component trace. ALL indicates that COPYTRC is to copy all component traces. filters lists the trace names to be used as filters.

filters has the following syntax:
COMP(name) [SUB(name[name]...)] [...] You may specify complete trace names or partial trace names. Separate each partial or complete trace name by a comma. If you specify a partial trace name, COPYTRC copies each trace that matches the partial trace name.

For example, if you specify OPTIONS((COMP(COMP1) SUB(ASID(200)))), the following traces match this partial trace name:
COMP1.ASID(0200).FUNC2.SVC3
COMP1.ASID(0200).FUNC1.SVC3

Note: You must specify TYPE(CTRACE) to use the OPTIONS parameter.

START(mm/dd/yy,hh.mm.ss.dddddd)
Specifies the beginning date and time for the trace entries or records to be copied. When you do not specify START, IPCS starts at the beginning of the trace entries or records. Specify the date and time in mm/dd/yy, hh:mm:ss.dddddd format, where:

mm represents months
dd represents days
yy represents years
hh represents hours
mm represents minutes
ss represents seconds
ddddddd represents decimal fractions of seconds

These rules apply to the date and time specifications:
- You must specify a date and time on the START parameter.
- The month and day can be specified in either single or double digits.
- Separate the date from the time with a comma.
- The time must be Greenwich mean time (GMT).
- Hours, minutes, and seconds can be specified in single or double digits.
- The time can be truncated anywhere on the right.
- The time can be left off completely, in which case, it will default to 00:00:00.000000 (midnight).

Some examples of valid date formats are:
m/dd/yy
mm/d/yy
m/d/yy
mm/dd/yy

Some examples of valid time formats are:
hh.mm.ss.dddddd
hh.mm.ss.dd
hh.mm.ss
h.m.s
hh.mm
hh

STOP(mm/dd/yy,hh.mm.ss.dddddd)
Specifies the ending date and time for the trace entries or records to be copied. When you do not specify STOP, IPCS stops copying after the last trace entry or record.
COPYTRC subcommand

For guidelines on how to specify the date and time, see the START parameter.

SYSNAME(sysname[, sysname]...)
Requests that the trace entries or records should be copied only if the trace’s system name matches one of the system names in the list. SYSNAME accepts up to 16 system names in the list.

Return codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>End of file reached. The input data set has been closed and all trace entries or records have been copied to the output data set.</td>
</tr>
<tr>
<td>04</td>
<td>No valid trace entries or records meeting the selection criteria were found. No trace data was copied to the output data set.</td>
</tr>
<tr>
<td>08</td>
<td>A processing error occurred. Some, but not all trace entries or records were copied to the output data set.</td>
</tr>
<tr>
<td>12</td>
<td>An error occurred in COPYTRC processing. No trace entries or records were copied to the output data set.</td>
</tr>
<tr>
<td>16</td>
<td>Dynamic allocation of the output data set failed. No trace entries or records were copied to the output data set.</td>
</tr>
<tr>
<td>20</td>
<td>The COPYTRC subcommand has a syntax error.</td>
</tr>
<tr>
<td>24</td>
<td>The COPYTRC subcommand has a semantic error.</td>
</tr>
</tbody>
</table>

COUPLE subcommand — analyze cross-system coupling data

Use the COUPLE subcommand to generate reports about the cross-system coupling facility (XCF). This subcommand provides information about the following:
- Groups and members in the sysplex
- Sysplex couple datasets
- XCF signaling service
- XCF storage use
- Status of systems in the sysplex
- XCF internal diagnostic information
- Coupling Facility Resource Management (CFRM)
- Automatic restart management

The COUPLE subcommand does not process active storage.

The reports generated by the COUPLE subcommand contain information for IBM diagnostic use. IBM might ask you to report this information for use in problem determination.

See the XCF component in z/OS MVS Diagnosis: Reference for COUPLE output.

Syntax
COUPLE Subcommand

COUPLE

-------- Report Type Parameters ------------------------
[ GROUP ]
[ SERIAL ]
[ SIGNAL ]
[ STORAGE ]
[ SYSPLEX ]
[ XCFSTACK ]
[ CFRM ]
[ ARM ]

-------- Data Selection Parameters ---------------------
[ DETAIL ]
[ EXCEPTION ]
[ SUMMARY ]

-------- Address Space Selection Parameters -----------
[ ASID(asidlist) ]
[ JOBNAME(joblist) ]

-------- Additional Filter Parameters -----------------
[ CFNAME(cfname) ]
[ STRNAME(strname) ]
[ SYSNAME(sysname) ]
[ GRPNAME(grpname) ]
[ DEVICE(device) ]
[ TYPE(type) ]
[ ELEMENT(element) ]
[ RSTGROUP(rstgroup) ]
COUPLE Subcommand

------- SETDEF-Defined Parameters ------------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAMES(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

Report type parameters
Use these parameters to select the type of report. You may specify more than
one; COUPLE produces a report for each specified parameter. If you omit these
parameters, the default is to present a report for all parameters listed below.

GROUP
Requests information about the groups in the sysplex and the status of
members within each group.

SERIAL
Requests information about the XCF couple data sets.

SIGNAL
Requests information about XCF signaling services. This report includes
information about signaling paths, transport classes, message buffers, list
structures, and devices in use.

STORAGE
Requests information about XCF storage use.

SYSPLEX
Requests information about the status of each system in the sysplex. This
includes sysplex failure management (SFM) information.

XCFSTACK
Requests internal diagnostic information. This information may be
requested by the IBM Support Center.

CFRM
Requests information about coupling facility resource management.

ARM
Requests information about elements and restart groups for the system
where the dump was taken.

Data selection parameters
Data selection parameters limit the scope of the data in the report. The default
is to present a summary report.

SUMMARY
Requests summary information for each of the requested topics.

EXCEPTION
Requests a list of exceptional or unusual conditions for each topic. The list
of exceptions contains information for IBM diagnostic use.
COUPLE Subcommand

DETAIL
Requests a report showing detailed information for each of the selected topics.

Address Space Selection Parameters
Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs).

ASID(asidlist)
Specifies the ASID for the address space to be included in the report.

The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed in the notation X’nnn’ or decimal, nnn.

JOBNAME(joblist)
Specifies a list of job names whose associated address spaces are to be included in the report. Use commands to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

You may use an asterisk (*) at the end of a job name as a generic character. That will result in a match for any value that begins with the characters preceding the asterisk.

Additional filter parameters
Use these parameters to select the information for the report.

CFNAME(cfname)
Requests that only information about the specified coupling facility be included in the report. cfname may also be a list of coupling facilities. You may use an asterisk (*) at the end of cfname as a generic character. That will result in a match for any value that begins with the characters preceding the asterisk.

STRNAME(strname)
Requests that only information about the specified coupling facility structure be included in the report. strname may also be a list of coupling facility structures. You may use an asterisk (*) at the end of strname as a generic character. That will result in a match for any value that begins with the characters preceding the asterisk.

SYSNAME(sysname)
Requests that only information about the specified system be included in the report. sysname may also be a list of systems. You may use an asterisk (*) at the end of sysname as a generic character. That will result in a match for any value that begins with the characters preceding the asterisk.

GRPNAME(grpname)
Requests that only information about the specified group be included in the report. grpname may also be a list of groups. You may use an asterisk (*) at the end of grpname as a generic character. That will result in a match for any value that begins with the characters preceding the asterisk.

DEVICE(device)
Requests that only information about the specified device be included in the report. device may be a list or range of devices. You must specify hexadecimal values.
COUPLE Subcommand

TYPE(type)
Requests that only information about the specified couple data set be
included in the report. type may also be a list of couple data sets.

ELEMENT(element)
Requests that only information about the specified element be included in
the report. element may also be a list of elements.

RSTGROUP(rstgroup)
Requests that only information about the specified restart group be
included in the report. rstgroup may also be a list of restart groups.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the COUPLE subcommand.

CTRACE subcommand — format component trace entries

Use the CTRACE subcommand to process component trace entries in a dump or
trace data set. CTRACE has two basic functions:
• Identify components and applications that have component trace entries in a
dump or trace data set. The QUERY parameter provides this function.
• Process the trace entries in a dump or trace data set.

To process trace entries, CTRACE allows you to:
• Select the traces to be processed
• View formatted trace entries
• Limit the information displayed for each formatted trace
• List entry identifiers for a trace
• Count the number of occurrences of each trace entry

Additional data selection can be done with a component-supplied or user-written
routine. You can use the OPTIONS parameter to pass parameters to data selection
and formatting routines.

The following books provide more information:
• z/OS MVS Diagnosis: Tools and Service Aids tells how to request and format
IBM-supplied component traces and shows trace output from IBM-supplied
traces.
• z/OS MVS IPCS Customization describes the steps needed to set up formatting for
your application’s traces with CTRACE.

Syntax

CTRACE

{ QUERY[(compname) [SUB((name[.name]...))]]
  { [SYSNAME(name)] COMP(name) [SUB((name[.name]...))] }
}

------- Report Type Parameters ------------------------

[ SHORT  ]
[ SUMMARY ]
[ FULL    ]
[ TALLY   ]
CTRACE subcommand

-------- Data Selection Parameters ---------------
[ GMT|LOCAL ]
[ START(mm/dd/yy,hh.mm.ss.ddddd) ]
[ STOP(mm/dd/yy,hh.mm.ss.ddddd) ]
[ EXCEPTION ]
[ LIMIT(nnnnnnnn) ]
[ ENLIST(entidlist) ]
[ USEREXIT(exitname) ]
[ OPTIONS((component routine parms)) ]

-------- Address Space Selection Parameters --------
[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR ]
[ ASIDLIST(asidlist) ]
[ JOBLIST(joblist)|JOBNAME(joblist) ]

-------- SETDEF-Defined Parameters -----------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Note: The PATH keyword is only intended to refer to a dump data set, not an external trace.

Parameters

QUERY[(compname) [SUB((name,name)...)]]
Requests component trace status information based on the level of the request and the number of traces within an available component.

Specify QUERY with no component name to request a list of the names of components or applications that have traces defined in a dump or trace data set. For multiple-trace components, the report lists each SUB level trace name for that component.
To request various summary trace reports for a component, do the following:

- **For single-trace components**, specify QUERY with a component name. The output lists the date and time of the first and last entries for that trace. If that trace is in a dump data set, specify FULL to list the trace options that were active for the trace at the time of the dump.

- **For multiple-trace components**, you may request a list of traces defined to a HEAD level or summary trace information for a single trace.
  - For a list of traces defined to a HEAD level, specify QUERY either with the HEAD level component name or with the component name and HEAD name on the SUB parameter.
  - For summary trace information for a single trace, specify QUERY with the component name and complete SUB name of the trace. The report lists the date and time of the first and last entries for that trace. If that trace is in a dump data set, specify FULL to list the trace options that were active for the trace at the time of the dump.

GMT, LOCAL and OPTIONS are the only data selection parameters that may be specified with QUERY. GMT is the default.

QUERY is the default parameter on the CTRACE subcommand. If you specify CTRACE with no additional parameters, IPCS will process a general query request.

```
[SYSNAME(name)] COMP(name) [SUB((name[,name]...))]
```

Specifies the trace to be processed. If the trace to be processed comes from a component or application that uses a single trace, use only the COMP parameter to identify that trace. Use the SUB parameter with COMP to identify a trace that is part of a multiple-trace component.

The SYSNAME parameter allows only trace entries from a particular system to be processed for a particular trace.

Do not specify a partial trace name for formatting.

Report type parameters, data selection parameters, and address space selection parameters control the output produced by this parameter.

To identify components for which you can view component trace entries, use QUERY. *z/OS MVS Diagnosis: Tools and Service Aids* identifies the value for the COMP parameter for each component that supports tracing.

### Report type parameters

Use these parameters to select the type of report. If you omit a report type parameter, the default is SHORT.

- **SHORT** Requests that one line of output be produced for each requested trace entry. The line includes the component mnemonic, entry identifier, date and time, and a description of the entry.

- **SUMMARY** Requests that key fields from each qualifying trace entry be printed following the date, time, and entry description.

- **FULL** Requests that all the data in each qualifying trace entry be formatted following the date, time, and entry description line.
TALLY
Requests a list of trace entry definitions for the component and counts how many times each trace entry occurred.

If you need only to format entry identifier definitions, specify a small number in the LIMIT parameter to avoid reading all the trace entries. Otherwise, if you do not place a limit on the number of trace entries processed, TALLY finds the number of occurrences of each trace entry and the average interval, in microseconds, between occurrences.

Data selection parameters
Use these parameters to limit the number of trace entries. All data selection parameters are optional.

GMT or LOCAL
GMT indicates that the time specified is Greenwich mean time. LOCAL indicates that the time specified is local time.

START(mm/dd/yy,hh.mm.ss.dddddd)
Specifies the beginning date and time for the trace entries to be formatted. When you do not specify START, IPCS starts at the beginning of the trace entries. Specify the date and time in mm/dd/yy,hh.mm.ss.dddddd format, where:

- **mm** represents months
- **dd** represents days
- **yy** represents years
- **hh** represents hours
- **mm** represents minutes
- **ss** represents seconds
- **dddddd** represents decimal fractions of seconds

These rules apply to the date and time specifications:
- The date section can be specified as an asterisk (*) to use the date from the first trace entry in the dump or trace data set.
- The month and day can be specified in either single or double digits.
- Separate the date from the time with a comma.
- The time can be GMT, by default or specified in a GMT parameter, or local, if specified in a LOCAL parameter.
- Hours, minutes, and seconds can be specified in single or double digits.
- The time can be truncated anywhere on the right.
- The time can be left off completely, in which case, it will default to 00:00:00.000000 (midnight).

Some examples of valid date formats are:

- 
- *m/dd/yy
- mm/dd/yy
- mm/dd/yy
- m/dd/yy
- mm/dd/yy

Some examples of valid time formats are:

- hh.mm.ss.dddddd
- hh.mm.ss.dd
- hh.mm.ss
- h.m.s
- hh.mm
- hh
STOP(mm/dd/yy,hh.mm.ss.ddddddd)
Specifies the ending date and time for the trace entries to be formatted.
When you do not specify STOP, IPCS stops formatting after the last trace
entry.
For guidelines on how to specify the time and date, see the START
parameter.

EXCEPTION
Requests that qualifying exceptional trace entries be formatted.

Note: Not all components support EXCEPTION processing.

LIMIT(nnnnnnnnn)
Limits the number of trace entries that CTRACE will process. The specified
number (nnnnnnnnn) can range from 1 to 999,999,999.

ENTIDLIST(entidlist)
Specifies a list of format entry identifiers to be used as filters for a trace.
Specify the list of entry identifiers using standard TSO/E notation. For
example:
ENTIDLIST('X'00800020',3,'X'12345678':X'22000000')

Note: To obtain a list of allowable entry identifiers for a component, enter
CTRACE TALLY LIMIT(1).

USEREXIT(exitname)
Specifies an optional user exit routine that gets control:
- When CTRACE begins to process each trace entry
- After CTRACE processes the last trace entry
This exit routine can select, gather, and format entries. See z/OS MVS IPCS
Customization for more information about user exits.

OPTIONS((component routine parms))
Identifies parameters to pass to the component-owned CTRACE filter
analysis routine or CTRACE buffer-find exit routine. These options are
shown in the heading of the report. To determine which parameters the
routine accepts, see z/OS MVS Diagnosis: Tools and Service Aids or the related
product documentation.

Address space selection parameters
Use these parameters to obtain data from specific address spaces, which you
specify by their address space identifiers (ASIDs). If you omit these parameters,
the default is ALL. For more information, see the select ASID service in z/OS
MVS IPCS Customization.

Notes:
1. If both ASIDLIST and JOBNAME or JOBLIST parameters are in effect, then
   a match for either allows the trace entry to be processed.
2. Not all components support ASIDLIST processing.
3. Not all components support JOBNAME or JOBLIST processing.

ALL
Specifies processing of the applicable trace entries for all address spaces in
the dump.

CURRENT
Specifies processing of the trace entries for each address space that is active
when the dump is generated.
ERROR
Specifies formatting of trace entries for any address space with an error indicator or containing a task with an error indicator.

TCBERROR
Specifies formatting of trace entries for any address space containing a task with an error indicator. Entries for address spaces with an error indicator are not formatted.

ASIDLIST(asidlist)
Specifies the list of ASIDs for which you want to process trace entries. The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed in the notation X’nnn’, F’nnn’, or B’nnn’. An unqualified number is assumed to be fixed.

JOBNAME(joblist) or JOBLIST(joblist)
Specifies the list of job names whose associated address spaces are to be processed for trace entries. Use commas or spaces to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the CTRACE subcommand.

Example 1
Request a list of traces defined in a dump.

Action
COMMAND ===> ctrace query

Result
CTRACE produces the following output. The report shows the complete name of all traces defined in a dump, organized by component names. In this example, COMP1 is a HEAD level component name for a multiple trace component. Five traces are defined under COMP1.

COMPONENT TRACE QUERY SUMMARY

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>SUB NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001 COMP1</td>
<td>ASID(0010).FUNC2.SVC2</td>
</tr>
<tr>
<td>0002 COMP1</td>
<td>ASID(0020).FUNC1.SVC3</td>
</tr>
<tr>
<td>0003 COMP1</td>
<td>ASID(0200).FUNC2.SVC3</td>
</tr>
<tr>
<td>0004 COMP1</td>
<td>ASID(0200).FUNC1.SVC3</td>
</tr>
<tr>
<td>0005 COMP1</td>
<td>ASID(0012).FUNC1.SVC1</td>
</tr>
<tr>
<td>0006 COMP2</td>
<td>FUNCA</td>
</tr>
<tr>
<td>0007 COMP2</td>
<td>FUNCB</td>
</tr>
<tr>
<td>0008 COMP3</td>
<td></td>
</tr>
<tr>
<td>0009 COMP4</td>
<td></td>
</tr>
</tbody>
</table>

Example 2
Produce a QUERY report for the COMP1 multiple-trace component trace in Example 1.
CTRACE subcommand

- Action
  COMMAND ===> ctrace query(COM1)

- Result
  CTRACE produces the following output. The report is similar to the general query report, listing only the traces from the COM1 component name.

  COMPONENT TRACE QUERY SUMMARY
  
  COMPONENT   SUB NAME
  ---------------
  0001 COM1  ASID(0010).FUNC2.SVC2
  0002 COM1  ASID(0020).FUNC1.SVC3
  0003 COM1  ASID(0200).FUNC2.SVC3
  0004 COM1  ASID(0200).FUNC1.SVC3
  0005 COM1  ASID(0012).FUNC1.SVC1

Example 3
Produce a QUERY report for the COMP1.ASID(0200) HEAD level.

- Action
  COMMAND ===> ctrace query(COM1) sub((ASID(0200)))

- Result
  CTRACE produces the following output.

  COMPONENT TRACE QUERY SUMMARY
  
  COMPONENT   SUB NAME
  ---------------
  0001 COM1  ASID(0200).FUNC2.SVC3
  0002 COM1  ASID(0200).FUNC1.SVC3

Example 4
Produce a QUERY report for the COMP1.ASID(0200).FUNC2.SVC3 trace.

- Action
  COMMAND ===> ctrace query(COM1) sub((ASID(0200).func2.svc3))

- Result
  CTRACE produces the following output.

Component Trace Query Summary

Component    Subname((ASID(0200).FUNC2.SVC.))
--------------
START = 01/05/90 14:37:48.963576 GMT
STOP = 01/05/90 14:39:21.354861 GMT

Example 5
Produce a QUERY FULL report for the COMP1.ASID(0200).FUNC2.SVC3 trace.

- Action
  COMMAND ===> ctrace query(COM1) sub((ASID(0200).func2.svc3)) full

- Result
  CTRACE produces the following output.
Example 6
Produce a SHORT form report for RSM trace entries.

- Action
  COMMAND ===> ctrace comp(sysrsm) lim(10)

- Result
  CTRACE produces the following output.

Example 7
Produce a SUMMARY form report for RSM trace entries.

- Action
  COMMAND ===> ctrace comp(sysrsm) lim(10) summary

- Result
  CTRACE produces the following output.
Example 8

Produce a FULL form report for RSM trace entries.

- Action
  
  ```
  COMMAND ===> ctrace comp(sysrsm) lim(10) full
  ```

- Result
  
  **CTRACE subcommand**

  COMPONENT TRACE SUMMARY FORMAT
  COMP(SYSRSM)
  **** 01/05/90

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>ENTRY ID</th>
<th>TIME STAMP</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSGSNG</td>
<td>00000006</td>
<td>14:37:48.926973</td>
<td>Get Single Frame</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>VSMGTMN</td>
<td>VSM Getmain Service</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 88084001 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 88084000</td>
</tr>
<tr>
<td>RSEPAG</td>
<td>00000008</td>
<td>14:37:48.927078</td>
<td>Enqueue Pageable Frame</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>VSMGTMN</td>
<td>VSM Getmain Service</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 88040001 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 88040000</td>
</tr>
<tr>
<td>XEPEXIT</td>
<td>00000002</td>
<td>14:37:48.927177</td>
<td>External Entry Point Exit</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>VSMGTMN</td>
<td>VSM Getmain Service</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 80000001 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 80000000</td>
</tr>
<tr>
<td>XEPENTRY</td>
<td>00000001</td>
<td>14:37:48.927734</td>
<td>External Entry Point Entry</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 00000003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 00000000</td>
</tr>
<tr>
<td>RSGSNG</td>
<td>00000006</td>
<td>14:37:48.927853</td>
<td>Get Single Frame</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 88084003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 88084000</td>
</tr>
<tr>
<td>RSEPAG</td>
<td>00000008</td>
<td>14:37:48.927953</td>
<td>Enqueue Pageable Frame</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 88004003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 88004000</td>
</tr>
<tr>
<td>XEPEXIT</td>
<td>00000002</td>
<td>14:37:48.928052</td>
<td>External Entry Point Exit</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 00000003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 00000000</td>
</tr>
<tr>
<td>XEPENTRY</td>
<td>00000001</td>
<td>14:37:48.928554</td>
<td>External Entry Point Entry</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 00000003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 00000000</td>
</tr>
<tr>
<td>RSGSNG</td>
<td>00000006</td>
<td>14:37:48.928668</td>
<td>Get Single Frame</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 88084003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 88084000</td>
</tr>
<tr>
<td>RSEPAG</td>
<td>00000008</td>
<td>14:37:48.928772</td>
<td>Enqueue Pageable Frame</td>
</tr>
<tr>
<td>FUNC1...</td>
<td>FLTAEPA</td>
<td>Enabled Addr Space Page Faults</td>
<td></td>
</tr>
<tr>
<td>JOB1...</td>
<td>CONSOLE</td>
<td>ASID1... 000A</td>
<td>PLOCKS.. 88004003 CPU..... 0001</td>
</tr>
<tr>
<td>JOB2...</td>
<td>CONSOLE</td>
<td>ASID2... 000A</td>
<td>RLOCKS.. 88004000</td>
</tr>
</tbody>
</table>
CTRACE subcommand

COMPONENT TRACE FULL FORMAT
COMP(SYSRSM)
**** 01/05/90

MNEMONIC  ENTRY ID  TIME STAMP  DESCRIPTION
----------  --------  ---------------  ---------------
RSGSNG 00000006  14:37:48.926973 Get Single Frame
FUNC1... VSMGTMN  VSM Getmain Service
JOBN1... CONSOLE ASID1... 000A  PLOCKS.. 88084001 CPU..... 0001
JOBN2... CONSOLE ASID2... 000A  RLOCKS.. 88084000
KEY..... 0036  ADDR.... 01B32DC0 ALET.... 00000000
19001200
KEY..... 0001  ADDR.... 012A6000 ALET.... 00000000
012A26A0 0125FBC FFC00000 03000000 00000000 7FFE4000 01877F00 00000000
RSEPAG 00000008  14:37:48.927078 Enqueue Pageable Frame
FUNC1... VSMGTMN  VSM Getmain Service
JOBN1... CONSOLE ASID1... 000A  PLOCKS.. 80004001 CPU..... 0001
JOBN2... CONSOLE ASID2... 000A  RLOCKS.. 80004000
KEY..... 0036  ADDR.... 01B32DC0 ALET.... 00000000
1900
KEY..... 0001  ADDR.... 012A6000 ALET.... 00000000
01A12AAC 0129A7E0 01C00000 03000000 0000000A 0098A000 01B77F00 00000000
XEPEXIT 00000002  14:37:48.927177 External Entry Point Exit
FUNC1... VSMGTMN  VSM Getmain Service
JOBN1... CONSOLE ASID1... 000A  PLOCKS.. 80000001 CPU..... 0001
JOBN2... CONSOLE ASID2... 000A  RLOCKS.. 80000000
KEY..... 0036  ADDR.... 01B2FDC0 ALET.... 00000000
0400
KEY..... 0016  ADDR.... 00000000 ALET.... 00000000
XEPENTRY 00000001  14:37:48.927734 External Entry Point Entry
FUNC1... FLTAEPAG  Enabled Addr Space Page Faults
JOBN1... CONSOLE ASID1... 000A  PLOCKS.. 00000003 CPU..... 0001
JOBN2... CONSOLE ASID2... 000A  RLOCKS.. 00000000
KEY..... 0036  ADDR.... 01B2FDC0 ALET.... 00000000
0400
KEY..... 002F  ADDR.... 0098A000 ALET.... 00000000
KEY..... 0032  ADDR.... 00F2B088 ALET.... 00000000
070C2000 81E8D81AE
RSGSNG 00000006  14:37:48.927853 Get Single Frame
FUNC1... FLTAEPAG  Enabled Addr Space Page Faults
JOBN1... CONSOLE ASID1... 000A  PLOCKS.. 00004003 CPU..... 0001
JOBN2... CONSOLE ASID2... 000A  RLOCKS.. 00004000
KEY..... 0036  ADDR.... 01B2FDC0 ALET.... 00000000
04001200
KEY..... 0001  ADDR.... 012A60A0 ALET.... 00000000
0129E7E0 0125FBC FFC00000 03000000 00000000 7FFE2000 01B14E80 00000000
RSEPAG 00000008  14:37:48.927953 Enqueue Pageable Frame
FUNC1... FLTAEPAG  Enabled Addr Space Page Faults
JOBN1... CONSOLE ASID1... 000A  PLOCKS.. 00004003 CPU..... 0001
JOBN2... CONSOLE ASID2... 000A  RLOCKS.. 00004000
KEY..... 0036  ADDR.... 01B2FDC0 ALET.... 00000000
0400
KEY..... 0001  ADDR.... 012A60A0 ALET.... 00000000
01A12AAC 0129A7E0 01C00000 03000000 0000000A 0098A000 01B14E80 00000000
Example 9

Produce a TALLY form report.

Action

COMMAND ===> ctrace tally comp(sysrsm) lim(22)

Result

CTRACE produces the following output.

Note: The trace record with mnemonic TRACEB has an average interval greater than or equal to 1000 seconds. IPCS supplies the message ] 16 min. for all trace entries with average intervals greater than or equal to 1000 seconds.
DOCPU subcommand — obtain stand-alone dump data for multiple processors

Use the DOCPU subcommand to gather stand-alone dump data for tasks that need to be repeated for each of the specified processors. For example, to display contents of a processor-related control block for a group of processors. With this command, you can obtain processor-related diagnostic data from a stand-alone dump with one command rather than repeating the command for each processor.

Syntax for DOCPU:

```
{ DOCPU }
```

-------- Data Selection Parameters -------------------------

```
[ ( CPU ( cpu-address-range-list ) ) ]
CPUTYPE ( (ZAAP | ZA) | (ZIIP | ZI) | (STANDARD | CP | S) ) |
CPUMASK ( cpumask ) ]
```

---

**COMPONENT TRACE TALLY REPORT**

<table>
<thead>
<tr>
<th>FMTID</th>
<th>COUNT</th>
<th>INTERVAL</th>
<th>MNEMONIC</th>
<th>DESCRIBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000001</td>
<td>4</td>
<td>855</td>
<td>XEPENTRY</td>
<td>External Entry Point Entry</td>
</tr>
<tr>
<td>00000002</td>
<td>4</td>
<td>944</td>
<td>XEPEXIT</td>
<td>External Entry Point Exit</td>
</tr>
<tr>
<td>00000003</td>
<td>0</td>
<td></td>
<td>FIX</td>
<td>Page Being Fixed</td>
</tr>
<tr>
<td>00000004</td>
<td>0</td>
<td></td>
<td>FREE</td>
<td>Page Being Freed</td>
</tr>
<tr>
<td>00000005</td>
<td>0</td>
<td></td>
<td>RGDBL</td>
<td>Get Double Frame</td>
</tr>
<tr>
<td>00000006</td>
<td>3</td>
<td>847</td>
<td>RGSGN</td>
<td>Get Single Frame</td>
</tr>
<tr>
<td>00000007</td>
<td>0</td>
<td></td>
<td>RSEFIX</td>
<td>Enqueue Fixed Frame</td>
</tr>
<tr>
<td>00000008</td>
<td>3</td>
<td>847</td>
<td>RSEPAG</td>
<td>Enqueue Pageable Frame</td>
</tr>
<tr>
<td>00000009</td>
<td>0</td>
<td></td>
<td>RSESQA</td>
<td>Enqueue SQA Frame</td>
</tr>
<tr>
<td>0000000A</td>
<td>0</td>
<td></td>
<td>RSESBUF</td>
<td>Enqueue Storage Buffer Frame</td>
</tr>
<tr>
<td>0000000B</td>
<td>0</td>
<td></td>
<td>RSEDEFER</td>
<td>Enqueue Deferred Frame</td>
</tr>
<tr>
<td>0000000C</td>
<td>0</td>
<td></td>
<td>RSEVRF</td>
<td>Enqueue V=R Waiting Frame</td>
</tr>
<tr>
<td>0000000D</td>
<td>0</td>
<td></td>
<td>RSDFIX</td>
<td>Dequeue Fixed Frame</td>
</tr>
<tr>
<td>0000000E</td>
<td>3</td>
<td>170</td>
<td>RSDPAG</td>
<td>Dequeue Pageable Frame</td>
</tr>
<tr>
<td>0000000F</td>
<td>0</td>
<td></td>
<td>RSDSQA</td>
<td>Dequeue SQA Frame</td>
</tr>
<tr>
<td>00000010</td>
<td>0</td>
<td></td>
<td>RSDSBUF</td>
<td>Dequeue Storage Buffer Frame</td>
</tr>
<tr>
<td>00000011</td>
<td>0</td>
<td></td>
<td>RSDDEFER</td>
<td>Dequeue Deferred Frame</td>
</tr>
<tr>
<td>00000012</td>
<td>0</td>
<td></td>
<td>RSDVRW</td>
<td>Dequeue V=R Waiting Frame</td>
</tr>
<tr>
<td>00000013</td>
<td>0</td>
<td></td>
<td>RSFDAS</td>
<td>Free Double Frame</td>
</tr>
<tr>
<td>00000014</td>
<td>3</td>
<td>162</td>
<td>RSFSNG</td>
<td>Free Single Frame</td>
</tr>
<tr>
<td>00000015</td>
<td>0</td>
<td></td>
<td>ESGET</td>
<td>Get Expanded Storage</td>
</tr>
<tr>
<td>00000016</td>
<td>0</td>
<td></td>
<td>ESENQ</td>
<td>Enqueue Expanded Storage</td>
</tr>
<tr>
<td>00000017</td>
<td>0</td>
<td></td>
<td>ESDEQ</td>
<td>Dequeue Expanded Storage</td>
</tr>
<tr>
<td>00000018</td>
<td>0</td>
<td></td>
<td>ESFREE</td>
<td>Free Expanded Storage</td>
</tr>
<tr>
<td>00000019</td>
<td>0</td>
<td></td>
<td>PAGER2A</td>
<td>Page Request Real to Auxiliary</td>
</tr>
<tr>
<td>0000001A</td>
<td>0</td>
<td></td>
<td>PAGER2P</td>
<td>Page Request Real to Permanent</td>
</tr>
<tr>
<td>0000001B</td>
<td>0</td>
<td></td>
<td>PAGER2E</td>
<td>Page Request Real to Expanded</td>
</tr>
<tr>
<td>0000001C</td>
<td>0</td>
<td></td>
<td>PAGER2R</td>
<td>Page Request Real to Real</td>
</tr>
<tr>
<td>0000001D</td>
<td>0</td>
<td></td>
<td>PAGEA2R</td>
<td>Page Request Auxiliary to Real</td>
</tr>
<tr>
<td>0000001E</td>
<td>0</td>
<td></td>
<td>PAGEP2R</td>
<td>Page Request Permanent to Real</td>
</tr>
<tr>
<td>0000001F</td>
<td>0</td>
<td></td>
<td>PAGEE2R</td>
<td>Page Request Expanded to Real</td>
</tr>
<tr>
<td>00000020</td>
<td>0</td>
<td></td>
<td>PAGEREL</td>
<td>Page Request Related</td>
</tr>
<tr>
<td>00000021</td>
<td>0</td>
<td></td>
<td>PAGEDEF</td>
<td>Page Request Deferred</td>
</tr>
<tr>
<td>00000022</td>
<td>0</td>
<td></td>
<td>FUNCREQ</td>
<td>Function Request</td>
</tr>
<tr>
<td>00000023</td>
<td>2</td>
<td>16 min.</td>
<td>TRACB</td>
<td>Trace Buffer</td>
</tr>
</tbody>
</table>

Total trace entries: 22
DOCPU subcommand

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters for DOCPU:

Data Selection Parameters
Use these parameters to limit the scope of the data in the report. If you omit these parameters (CPU, CPUTYPE, and CPUMASK), all processors are included as the default.

CPU (cpu-address-range-list)
Specifies the processors (CPU) that are selected for to run the specified IPCS subcommand. The cpu-address-range-list is a processor number, a range of processor numbers, or a combination of both. You can specify the processor number in either decimal or hexadecimal format (X'...'). You can use a colon to indicate a range of processors, and use a space or comma as a delimiter. For example:

- CPU(0)
- CPU(5:10)
- CPU(0 5:10)
- CPU(0,3,5:10)
- CPU(X'A')

Note: You can combine CPU, CPUTYPE, and CPUMASK as a union of sets.

CPUTYPE ((ZAAP|ZA) | (ZIIP|ZI) | (STANDARD|CP|S))
- Specifying ZAAP or ZA selects all ZAAP processors in the configuration.
- Specifying ZIIP or ZI selects all ZIIP processors in the configuration.
- Specifying STANDARD or CP or S selects all standard processors in the configuration.

You can combine the ZAAP, ZIIP, and STANDARD options in any order to select a combination of CPU types. For example, CPUTYPE(ZAAP STANDARD). You can use spaces or commas as a delimiter.

CPUMASK(CPU hexadecimal mask)
Specifies processors in a string of hexadecimal characters. Each hexadecimal character identifies four processors. The maximum number of processors supported by z/OS defines the maximum length of this hexadecimal string. Currently, the maximum number of processors supported by z/OS is 128, so the maximum length of the hexadecimal mask is 32. The leftmost bit designates the lower processor address starting from zero. For example:

- CPUMASK(FFF)
- CPUMASK(FOFO)
- CPUMASK(BO) CPU

You can combine CPUTYPE and CPUMASK as a union of sets. If all of the processors are omitted, the default is to include all processors.

EXEC((ipcs subcommand))
Runs the IPCS subcommand for each CPU you specify by appending...
DOCPU subcommand

CPU(xxx) to the IPCS subcommand. The DOCPU subcommand generates a
return code that consists of its own return code plus the return code from
the IPCS subcommand designated on the EXEC parameter.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the DOCPU subcommand.

Examples
– To display four bytes of storage at 414 in every PSA, enter the following
  command:
  DOCPU EXEC((L 414 LEN(4)))
– To format the PSA of processor 0,1,2,3,8,9,10,11, enter the following
  commands:
  DOCPU CPUMASK(F0F0) EXEC((CBF 0 STR(PSA)))

You can delete symbols when you want to free space in the dump directory.

DIVDATA subcommand — analyze data-in-virtual data

Use the DIVDATA subcommand to request:
• Validation, formatting, and display of the data-in-virtual control blocks
• Formatting and display of the data-in-virtual trace table

DIVDATA produces different diagnostic reports depending on the report type
parameters and the address space selection parameters specified. By specifying one
or more report type and address space selection parameters, you can selectively
display the information you want to see.

Report Type Parameters
– DETAIL displays all data-in-virtual control blocks.
– SUMMARY displays a summary of the data-in-virtual control blocks.
– EXCEPTION displays diagnostic error messages for not valid data-in-virtual
  control blocks.
– TRACE displays the data-in-virtual trace table by the specified address space
  selection parameter(s).
– FULLTRACE displays the entire data-in-virtual trace table.

Address Space Selection Parameters
– ALL processes all address spaces.
– CURRENT processes active address spaces of the dump.
– ERROR processes any address space with an error indicator or containing a
task with an error indicator.
– TCBERROR processes any address space containing a task with an error
  indicator.
– ASIDLIST processes address spaces associated with ASID(s).
– JOBLIST or JOBNAME processes address spaces associated with job names.
  Several address space selection parameters can be specified and an address
  space might meet more than one selection criterion. The selection criterion
  (or criteria) that is met for each address space appears in the output. No
  address space is processed more than once.

Syntax
DIVDATA subcommand

DIVDATA

------- Report Type Parameters ---------------------
[ DETAIL ]
[ SUMMARY ]
[ EXCEPTION ]
[ TRACE {OLDEST(n) } ]
[ FULLTRACE {NEWEST(n) } ]

------- Address Space Selection Parameters --------
[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR ]
[ ASIDLIST(asidlist) ]
[ JOBIST(jobist)|JOBNAME(jobist) ]

------- SETDEF-Defined Parameters -----------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

Report Type Parameters
Use these parameters to select the type of report. If you omit a report type
parameter, the default is EXCEPTION.

DETAIL
Specifies the report type that:
- Validates and formats all of the data-in-virtual control blocks
- Produces a data-in-virtual trace table statistics report, which contains
  information about the trace table and trace table entries

SUMMARY
Specifies the report type that validates certain control blocks and produces
a summary table showing the data-in-virtual object ranges that are mapped
and the virtual storage ranges they are mapped into.
If the DETAIL parameter is not also specified, SUMMARY also produces a data-in-virtual trace table statistics report, which contains information about the trace table and trace table entries. Additionally, IPCS validates, formats, and displays certain control blocks.

**EXCEPTION**

Specifies the report type that validates all of data-in-virtual control blocks and displays diagnostic error messages for incorrect control blocks.

A condensed version of the data-in-virtual trace table statistics report is also produced.

**TRACE**

**FULLTRACE**

Specifies the report type for formatting and displaying the data-in-virtual trace table entries.

TRACE specifies formatting and displaying of trace entries based on the address space selection parameters.

FULLTRACE specifies formatting and displaying the entire data-in-virtual trace table entries regardless of any specified address space selection parameter.

The trace table entries are processed based on the specified order parameters, OLDEST or NEWEST.

**OLDEST(n)**

**NEWEST(n)**

Specifies the order in which the trace table entries are to be formatted and displayed.

OLDEST specifies processing from the oldest entry toward the newest.

NEWEST specifies processing from the newest entry toward the oldest.

The $n$ indicates the number of trace entries to be processed. The $n$ can range from 1 through $2^{31}$ and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...'). If $n$ exceeds the total number of trace table entries or is omitted, the entire trace table is formatted and displayed.

If you omit both OLDEST and NEWEST, the default is OLDEST.

**Address Space Selection Parameters**

Use these parameters to obtain data from particular address spaces, which you specify by the address spaces identifier (ASID). If you omit these parameters, the default is CURRENT. For more information, see the select ASID service in z/OS MVS IPCS Customization.

**ALL**

Specifies processing of data-in-virtual control blocks for all address spaces in the system at the time the dump is generated.

**CURRENT**

Specifies processing of data-in-virtual control blocks for each address space that is active (for example, dispatched on some central processor) when the dump is generated.

**ERROR**

Specifies processing of data-in-virtual control blocks for any address space with an MVS error indicator or containing a task with an error indicator.
DIVDATA subcommand

TCBERROR
Specifies processing of data-in-virtual control blocks for any address space containing a task with an error indicator. Blocks for address spaces with an error indicator are not processed.

ASIDLIST(asidlist)
Specifies a list of ASIDs for the address spaces to be in the report.

The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed using the notation X'nnn', F'nnn', or B'nnn'. An unqualified number is assumed to be fixed.

This subcommand does not process summary dump records (ASID X'FFFA').

JOBLIST(joblist) or JOBNAME(joblist)
Specifies a list of job names whose associated address spaces are to be in the report. Use commas to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

Return Codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the DIVDATA subcommand.

Example
See the data-in-virtual component in z/OS MVS Diagnosis: Reference for examples of the DIVDATA subcommand output.

DLFDATA subcommand — format data lookaside facility data

Use the DLFDATA subcommand to generate diagnostic reports about activity by the data lookaside facility (DLF). Use the report type parameters to choose the information you want to see.

DLFDATA

-------- Report Type Parameters -----------------------------

{ CLASS(classname)[OBJECT(objname)] }

{ EXCEPTION }

{ STATS(classname) }

{ STORAGE(classname) }

{ SUMMARY }

{ USER(classname) }
Parameters

Report Type Parameters
Use these parameters to select the type of report. If you omit a report type parameter, the default is SUMMARY.

Note: In the parameter values, classname is 1 through 7 characters, which are alphanumeric or the following:

$ (X'5B')

# (X'7B')

@ (X'7C')

CLASS(classname)
Produces a report with information pertaining to the DLF class specified by classname.

OBJECT(objname)
Is an optional CLASS report parameter. Specify OBJECT to produce information about an object stored in DLF.

EXCEPTION
Produces messages related to any inconsistencies IPCS finds in the DLF data.

STATS(classname)
Produces a report with statistics about DLF activity. If you specify classname, only statistics for the specified class are produced.

STORAGE
Produces a report with information about the storage management of DLF data spaces. If you specify a classname, only storage management information for the specified class is produced.
DLFDATA subcommand

**SUMMARY**
- Produces a report with overall information for each of the classes known to DLF. This is the default report.

**USER(classname)**
- Produces a report with information relating to an address space that was using DLF facilities. If you specify `classname`, only information related to the specified class is produced.

**Address Space Selection Parameters**
Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs). If you omit these parameters, the default is CURRENT. For more information, see the select ASID service in [z/OS MVS IPCS Customization](https://www.ibm.com/docs/en/zos/2.4.0?topic=customization).

**ASIDLIST(asidlist)**
- Specifies a list of ASIDs for the address spaces to be included in the report.
  - The `asidlist` can be a single ASID or a list of noncontiguous ASIDs. When you specify a list, separate the list members with commas.

**CURRENT**
- Specifies that address spaces considered to be current by the select ASID exit service are to be included in the report.

**ERROR**
- Specifies processing for any address space with an error indicator or containing a task with an error indicator.

**TCBERROR**
- Specifies processing for any address space containing a task with an error indicator. Entries for address spaces with an error indicator are not formatted.

**JOBLIST(list) or JOBNAME(list)**
- Specifies a list of job names whose associated address spaces are to be included in the report. Use commas to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

**Return Codes**
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the DLFDATA subcommand.

**Example**
See the virtual lookaside component in [z/OS MVS Diagnosis: Reference](https://www.ibm.com/docs/en/zos/2.4.0?topic=diagnosis-reference) for examples of the DLFDATA subcommand output.

---

DROPDUMP subcommand — delete source description data

**Use the DROPDUMP subcommand to delete a source description or records in a source description from a dump directory. The description is for an unformatted source that IPCS can format, for example, an SVC dump, a stand-alone dump, an SYMDUMP dump, a trace data set, a data set, or active storage. The directory is allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.**

Some reasons for using DROPDUMP are to:
- Delete the description for a source that is no longer needed
- Delete the description for a partially initialized dump
DROP DUMP subcommand

- Delete source descriptions to free space in the directory
- Delete translation records from one or more source descriptions

**Related subcommands**

- ADD DUMP
- LIST DUMP

**Syntax**

```
{DROP DUMP} \[ RECORDS \{ ALL | ANALYSIS | TRANSLATION \} \]
```

**Parameters**

**RECORDS(ALL)**

Designates the type of records to be deleted from a source description.

- RECORDS(ALL) directs IPCS to delete all of the records in a source description.
- RECORDS(ANALYSIS) directs IPCS to delete only analysis records.
- RECORDS(TRANSLATION) directs IPCS to delete only records generated by an IPCS translation process. Translation records are generated by, for example, the simulation of System/390® prefixing or dynamic address translation.

The following are ways to use RECORDS(TRANSLATION):

- When IPCS first processes storage for a central processor in a stand-alone dump, IPCS locates the prefixed storage area (PSA) for the processor. IPCS constructs a central storage map using the absolute storage record map for the dump.
  
  If IPCS used an incorrect PSA, you may correct the definition of the PSAnn symbol in the symbol table. Then, you can run DROP DUMP RECORDS(TRANSLATION) to delete the incorrect translation records from your user dump directory. When IPCS next processes the storage in the dump, IPCS uses the corrected symbol to build a correct record map.

- When IPCS first processes an address space in a stand-alone dump, IPCS locates the segment table for the address space. IPCS constructs a virtual storage record map for the referenced page using the absolute storage record map or the central storage map for the dump.
  
  If IPCS used an incorrect segment table, you may correct the definition of the SGThnnnn symbol in the symbol table. Then, you can run DROP DUMP RECORDS(TRANSLATION) to delete the incorrect translation records from your user dump directory. When IPCS next processes the address space in the dump, IPCS uses the corrected symbol to build a correct record map.
DROP DUMP subcommand

- When you first enter an ANALYZE or STATUS CPU CONTENTION subcommand, IPCS places the following contention records in the source description:
  - The contention queue (CQ)
  - The contention resource (CR)
  - Program history (PH)

These records are incorrect if the symbols for the control blocks are incorrect or if the ANALYZE exit routines specified by parmlib members embedded in the BLSCFECT parmlib member have been redefined. If you determine that the contention records are incorrect, enter DROP DUMP RECORDS (TRANSLATION) to delete all contention records. Then you can run ANALYZE or STATUS CPU CONTENTION to have IPCS gather the contention records again.

DROP DUMP RECORDS (TRANSLATION) does not edit the symbol table or the storage map. For editing, use DROP MAP, DROPSYM, or EQUATE subcommands.

**SUMMARY or NOSUMMARY**

SUMMARY indicates that a processing summary (a final total line) is to be produced.

NOSUMMARY specifies that a processing summary is to be suppressed. The NOSUMMARY parameter is useful to turn off summary messages when the subcommand is invoked within a CLIST or a REXX exec.

**ACTIVE or MAIN or STORAGE**

DSNAME (dslist) or DATASET (dslist)

FILE (ddlist) or DDNAME (ddlist)

Specifies storage or one or more data sets. IPCS is to delete the source description or records in the source description for the storage or data sets. If one of these parameters is not specified, IPCS deletes the source description or records from the source description for your current source data set.

ACTIVE, MAIN, or STORAGE specifies that the source description is for the active storage that was accessed.

DSNAME or DATASET specifies that the source description is for the cataloged data set or sets named in dslist. When specifying more than one data set name, separate the names with commas or blanks.

FILE or DDNAME specifies that the source description is for a data set or sets with the ddname or ddnames in ddlist. When specifying more than one ddname, separate the names with commas or blanks.

**Return Codes**

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the DROP DUMP subcommand.

**Example 1**

Delete a source description for a specific dump.

- **Action**
  
  COMMAND ===> dropdump dsname('sys1.dump.d930428.t110113.system1.s00000')

- **Result**
  
  IPCS deletes from your user dump directory the source description for the dump in the data set named sys1.dump.d930428.t110113.system1.s00000. IPCS issues the following summary output.
**Example 2**

Delete records generated by translation processes,

- **Action**
  
  **COMMAND ===> dropdump records(translation)**

- **Result**
  
  The contention information from a STATUS CPU CONTENTION subcommand for the current dump data set appears to be incorrect. IPCS deletes this information, displays the following output, and permits the STATUS subcommand to be entered again to obtain new contention data.

**DROPMAP subcommand — delete storage map records**

Use the DROPMAP subcommand to delete records from the storage map in a source description for a dump. The source description is in the dump directory allocated with dname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

**Related subcommands**

LISTMAP

SCAN

**Syntax**

```
{DROPMAP } [RANGE (address:address)] [data-descr]
{DROPM }[/

[ SUMMARY | NOSUMMARY ]
```

**Parameters**

**RANGE(address:address)**

Specifies that the range of addresses in the dump for which map records exist are to be deleted. The range can be specified as an address and a length or as a range of addresses.

If you omit the range parameter, the subcommand deletes all map records for the dump.

If a map record describes an address within the range, the subcommand deletes the map record.

**data-descr**

Specifies the data description parameter, which consists of five parts:

- An address (required with the RANGE parameter and when data-descr is explicitly specified on the subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
DROPMAP subcommand

- A remark parameter (optional)

Chapter 3, "Data description parameter," on page 3-1 explains the use and syntax of the data description parameter.

If you specify address processing parameters (which are optional) but omit the address (which is required), the subcommand deletes all map records for the address space.

**SUMMARY or NOSUMMARY**

SUMMARY indicates that a processing summary (a final total line) is to be produced.

NOSUMMARY specifies that a processing summary is to be suppressed. The NOSUMMARY parameter is useful to turn off summary messages when the subcommand is invoked within a CLIST or a REXX exec.

**Return Codes**

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the DROPMAP subcommand.

**Example 1**

Delete all storage map records.

- Action
  
  COMMAND ===> dropmap

- Result
  
  DROPMAP produces the following summary output.

BLS18114I 42 RECORDS ERASED

**Example 2**

Delete storage map records within an address range for the same ASID.

- Action
  
  COMMAND ===> dropmap range(005d4980.:005d4c88.) asid(x'000b')

- Result
  
  DROPMAP produces the following summary output.

BLS18114I 7 RECORDS ERASED

---

DROPSYM subcommand — delete symbols

Use the DROPSYM subcommand to delete symbols from the symbol table in a source description for a dump. The source description is in the dump directory allocated with dname IPCSDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

You can delete symbols when you want to free space in the dump directory.

**Related subcommands**

EQUATE
LISTSYM
RENUM
STACK

**Syntax**
DROPSYM subcommand

{ DROPSYM } { (symbol-list) | * } { DROPS }

{ DROP | NOPURGE } { NODROP } { PURGE }

{ SUMMARY | NOSUMMARY }

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters. See

"SETDEF subcommand — set defaults" on page 5-232

{ ACTIVE | MAIN | STORAGE } { DSNAME(dsname) | DATASET(dsname) } { FILE(ddname) | DDNAME(ddname) } { PATH(path-name) } { TEST | NOTEST }

Parameters

symbol-list or *

Specifies the symbols to be deleted. You can specify one symbol, a range of
symbols, a list of symbols, a combination of these, or, with an asterisk (*),
all symbols in the symbol table. Enclose more than one symbol or range of
symbols in parentheses. The list can contain up to 31 symbols, ranges, or
both.

The symbols follow the IPCS naming conventions for symbols. See "IPCS
symbols" on page A-1.

If you specify a single symbol or a list of symbols, the subcommand deletes
only the specified symbol or symbols.

If you specify a range of symbols, the symbol name must follow the
naming conventions for symbols. See "IPCS symbols" on page A-1. IPCS
deletes all symbols whose names begin with the first character string
through all symbols whose names begin with the second character string. A
range of symbols is inclusive: the subcommand deletes all the symbols in
the range and at both ends of the range.

DROP or NODROP
NOPURGE or PURGE

Defines which symbols are eligible for deletion. The default is NOPURGE.

DROP and NOPURGE specify that only symbols with the DROP attribute
are to be deleted.

NODROP specifies that only symbols with the NODROP attribute are to be
deleted.

PURGE specifies that the NODROP attribute is ignored and all specified
symbols are deleted.

SUMMARY or NOSUMMARY

SUMMARY indicates that a processing summary (a final total line) is to be
produced.

NOSUMMARY specifies that a processing summary is to be suppressed.
The NOSUMMARY parameter is useful to turn off summary messages
when the subcommand is invoked within a CLIST or a REXX exec.

ACTIVE or MAIN or STORAGE
DSNAME(dsname) or DATASET(dsname)
DROPSYM subcommand

FILE(ddname) or DDNAME(ddname)
Specify the source of the source description containing the symbol. If one of
these parameters are not specified, the source is your current source.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the DROPSYM subcommand.

Example 1
Delete a range of ASCB symbols.
– Action
  COMMAND ===> dropsym (ascb00001 : ascb00050) nodrop
– Result
  DROPSYM deletes the ASCB symbols for ASID 1 through 50.

Example 2
Delete all symbols in the symbol table.
– Action
  COMMAND ===> dropsym * purge
– Result
  DROPSYM deletes every entry in the symbol table, including X, for the
current dump. If you omit the PURGE parameter, this example deletes all
symbols except those with the NODROP attribute.

END subcommand — end an IPCS session

Use the END subcommand to end:
• An IPCS session.
  Any default values specified with the SETDEF subcommand are canceled. The
  subcommand closes and deallocates the data set directory, problem directory,
  and any dumps allocated to the user. The subcommand closes but does not
deallocate your user dump directory and the print output data set.
• A session initiated by entering the IPCS TSO subcommand with no operands.
  During a TSO subcommand session, a command such as LIST causes the TSO/E
  command associated with the command to be processed, not the IPCS
  subcommand associated with it. When END is entered during a TSO
  subcommand session, IPCS resumes its normal interpretation of commands.
• CLIST or REXX exec processing initiated with the EXEC parameter of the
  RUNCHAIN subcommand.
• CLIST or REXX exec processing initiated with the IPCS primary command of the
  IPCS dialog.
• CLIST or REXX exec processing initiated through option 4 of the IPCS dialog.

Related subcommands
  IPCS
  SETDEF

Syntax

END

Return Codes
When the END subcommand ends an IPCS session, IPCS returns the highest
return code that was issued during the session.
Use the EPTRACE subcommand to generate reports on the control flow between programs as indicated by 72-byte save areas. Figure 5-1 describes the syntax of the EPTRACE subcommand.

**Related subcommands**

“SETDEF subcommand — set defaults” on page 5-232

**Syntax**

EPTRACE

-------- Report Selection Parameters --------------------------

[ KEYFIELD  |  SAVEAREA ]

[ ORDER(RETURN  |  ENTRY ) ]

[ DATA( TCBCURRENT  |  symbol ) ]

-------- SETDEF-Defined Parameters --------------------------

Note: You can override the following SETDEF parameters

See “SETDEF subcommand — set defaults” on page 5-232.

[ ACTIVE  |  MAIN  |  STORAGE ]

[ DNAME(dsname)  |  DATASET(dsname) ]

[ FILE(ddname)  |  DDNAME(ddname) ]

[ PATH(hfspath) ]

[ FLAG(severity) ]

[ PRINT  |  NOPRINT ]

[ TERMINAL  |  NOTERMINAL ]

[ TEST  |  NOTEST ]

*Figure 5-1. Syntax of the EPTRACE IPCS subcommand*

**Parameters**

**KEYFIELD or SAVEAREA**

Selects the report formatting to be performed for each entry point.

KEYFIELD is the default.

**Note:** The KEYFIELD report of EPTRACE is enhanced in z/OS V1R8 IPCS to consider additional linkage mechanisms:
- Linkages that employ the linkage stack to save status.
- Linkages that mark the initial word of caller’s save areas to indicate how status is saved.

**ORDER(RETURN) or ORDER(ENTRY)**

Selects the order of processing. ORDER(RETURN) causes the GPR 13 current GPR 13 value to be used to locate the active save area, and displays information to be displayed for calling programs later. ORDER(RETURN) is the default because it provides information needed for problem analysis early in the report. ORDER(ENTRY) causes information about the first entry point entered to be listed first.
EPTRACE subcommand

**DATA(symbol)**
Specifies an IPCS symbol that is associated with one of the structures shown in the following table:

<table>
<thead>
<tr>
<th>Structure</th>
<th>Use by EPTRACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCB</td>
<td>The first program of interest is the highest one active for the task.</td>
</tr>
<tr>
<td>IRB, SVRB, TIRB</td>
<td>The first program of interest is the highest one active for the RB.</td>
</tr>
<tr>
<td>REGSAVE</td>
<td>The first program of interest is the one to which this 72-byte save area was passed. Use of this data type limits EPTRACE processing to 72-byte save areas. No attempt is made to identify a related linkage stack.</td>
</tr>
</tbody>
</table>

The default, DATA(TCBCURRENT), is a symbol for which z/OS R5 support is supplied. An IPCS find routine is supplied that will attempt to determine whether an obvious current task can be identified within the dump. If it can be determined, the symbol TCBCURRENT is defined and associated with that TCB. Otherwise, the symbol is undefined. When the symbol explicitly or implicitly cannot be defined or that symbol is defined but is not associated with a supported data type, EPTRACE will generate an error message and will terminate.

**Return Codes**
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the EPTRACE subcommand.

EQUATE subcommand — create a symbol

The EQUATE subcommand allows you to:
- Create a symbol in the symbol table and to associate an address and storage attributes with the symbol
- Change the attributes of a symbol that is already defined in the symbol table
- Create storage map entries
- Set X, the current address, to a specific address

The symbol is in a symbol table that is part of a source description. The source description is in the dump directory allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

On the EQUATE subcommand, specify the name of the symbol followed by any address and other storage attributes that you want associated with the symbol. If the specified symbol already exists in the symbol table, the new address and storage attributes overlay the previous address and storage attributes.

**Note:** Because the EQUATE subcommand can be used either to create a new symbol or redefine an existing symbol, it can be used to create a symbol for a system control block that has failed the validity check during IPCS processing.

See the [z/OS MVS IPCS User's Guide](#) for information about maintaining symbol tables and storage map entries and about creating and validating your own symbol definitions.

**Related subcommands**
DROPSYM
EQUATE subcommand

LISTSYM
RENUM
STACK

Syntax

{ EQUATE } [ symbol | X ] [ data-descr | X ]
{ EQU } [ DROP | NODROP ]

Parameters

symbol or X

Specifies the symbol being defined. The symbol name is 1 through 31 alphanumeric characters; the first character must be a letter or one of the following characters:

$ (X'5B')

# (X'7B')

@ (X'7C')

If you omit this parameter, the default is X, which is the most recently accessed address.

data-descr or X

 Specifies the address and attributes to be associated with the symbol being defined through the data description parameter. The data description parameter consists of five parts:

– An address (required when data-descr is explicitly specified on the subcommand)
– Address processing parameters (optional)
– An attribute parameter (optional)
– Array parameters (optional)
– A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

If you omit this parameter, the default is X, which is the most recently accessed address.

DROP or NODROP

Specifies how the DROPSYM subcommand can delete the symbol.

DROP specifies that the symbol can be deleted from the symbol table by the DROPSYM subcommand without using the PURGE parameter.

NODROP specifies that the symbol not be deleted from the symbol table by the DROPSYM subcommand. This can be overridden by the PURGE parameter on the DROPSYM subcommand.

Return Codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the EQUATE subcommand.

Example 1

Define a symbol for a TCB that caused a dump.
EQUATE subcommand

- Action
  equate failingtcb 51368 length(360) +
  x remark('tcb that caused the dump')
- Result
  This subcommand defines FAILINGTCB at address X'51368'. It is identified
  as a TCB, and its size is 360 bytes (decimal). If the TCB is displayed or
  printed, it is in hexadecimal format. Because the NODROP parameter is not
  specified, this name can be deleted from the symbol table.

Example 2
Define a symbol table entry at the current address.
- Action
  equate jstcb
- Result
  This subcommand creates a symbol table entry for JSTCB. By default, the
  address and attributes associated with JSTCB are those associated with X,
  which is the current address.

Example 3
Set X to a specific address.
- Action
  equate x 522836
- Result
  This sets X to address X'522836'.

Example 4
Define a symbol, then change its attributes.
- Action
  equate buffer1 55280 length(80) asid(3) drop
  equate buffer1 buffer1 nodrop cpu(2)
- Result
  The first EQUATE creates the symbol BUFFER1 and gives it certain
  attributes. The second EQUATE changes the DROP attribute to NODROP
  and specifies a central processor in the CPU parameter. You can change the
  attributes of any symbol in the symbol table whether you created it or
  whether IPCS subcommands created it for you.

EVALDEF subcommand — format defaults

Use the EVALDEF subcommand to retrieve SETDEF-defined default values and
format the values in CLIST variables, REXX variables, or ISPF function pool dialog
variables. The default values can be for:
- Local defaults. These values are currently in use for an ISPF screen in the IPCS
dialog, for a batch IPCS session, or for an IPCS interactive line-mode session.
- Global defaults. These values are used to establish the local defaults when IPCS
processing starts in an ISPF screen, a batch IPCS session, or an IPCS interactive
line-mode session.

The default values are part of a source description. The source description is in the
dump directory allocated with ddname IPCSDDIR and is your current dump
directory. The current dump directory is your user dump directory or, for users
with write access authority, might be the sysplex dump directory.
EVALDEF subcommand

Note: With TSO/E Release 2 installed, you can use this subcommand to update
GLOBAL CLIST variables. See [z/OS TSO/E CLISTS] for information.

Related subcommands
- EQUATE
- EVALDUMP
- EVALMAP
- EVALSYM

Syntax

```
EVALDEF { LOCAL | GLOBAL }
{ CLIST(var-list) }
{ DIALOG(var-list) }
{ REXX(var-list) }
```

------ SETDEF-Defined Parameter --------------------------
Note: You can override the following SETDEF parameter.
See
"SETDEF subcommand — set defaults" on page 5-232

```
[ TEST | NOTEST ]
```

Parameters

**LOCAL or GLOBAL**
- Identifies the type of default values to be retrieved.
  - LOCAL requests the default values that are currently used.
  - GLOBAL requests the default values to be used when local values are not
    specified.

**CLIST(var-list)**
**DIALOG(var-list)**
**REXX(var-list)**
- Specifies how the default values are to be formatted.

  CLIST(var-list) designates that the values be formatted into CLIST variables.
  DIALOG(var-list) designates that the values be formatted into ISPF function
  pool dialog variables.
  REXX(var-list) designates that the values be formatted into REXX variables.

The syntax for var-list is as follows:
EVALDEF subcommand

[ DECIMAL | F ]
[ HEXADECIMAL | X ]

[ CONFIRM(confirm) ]
[ DISPLAY(display) ]
[ FLAG(flag) ]
[ LENGTH(length) ]
[ PRINT(print) ]
[ PROBLEM(problem) ]
[ QUALIFICATION(qualification) ]
[ SOURCE(var-name)|DATASET(var-name)|DSNAME(var-name) ]
[ TERMINAL(terminal) ]
[ TEST(test) ]
[ VERIFY(verify) ]

DECIMAL or F
HEXADECIMAL or X

Specifies the format of the default length.
DECIMAL or F designates that the default length be formatted using decimal digits.
HEXADECIMAL or X designates that the default length be formatted using hexadecimal digits.

CONFIRM(confirm)
Places the parameter CONFIRM or NOCONFIRM in the variable confirm.

DISPLAY(display)
Places one of each of the following options of the DISPLAY parameter in the variable display:
[NO]MACHINE
[NO]REMARK
[NO]REQUEST
[NO]STORAGE
[NO]SYMBOL

SOURCE(var-name) or DATASET(var-name) or DSNAME(var-name)
Places the parameter SOURCE, DATASET, or DSNAME and the default dump source name or the parameter NODSNAME in the variable var-name.

FLAG(flag)
Places one of the following options of the FLAG parameter, in the variable flag:
INFORMATIONAL
WARNING
ERROR
SERIOUS
TERMINATING

LENGTH(length)
Formats and places the default data length in the variable length. The length is in DECIMAL unless HEXADECIMAL is specified.
EVALDEF subcommand

PRINT(print)
   Places the parameter PRINT or NOPRINT in the variable print.

PROBLEM(problem)
   Places the PROBLEM parameter and the default problem number or the
   parameter NOPROBLEM in the variable problem.

QUALIFICATION(qualification)
   Places the default address qualifiers for the default data set in the variable
   qualification.

TERMINAL(terminal)
   Places the parameter TERMINAL or NOTERMINAL in the variable terminal.

TEST(test)
   Places the parameter TEST or NOTEST in the variable test.

VERIFY(verify)
   Places the parameter VERIFY or NOVERIFY in the variable verify.

Return Codes
   See “Standard subcommand return codes” on page 5-2 for a description of the
   return codes produced by the EVALDEF subcommand.

Example
   The BLSCSETD CLIST formats the current SETDEF-defined defaults for display
   on an ISPF data entry panel. It supports option 0 (DEFAULTS) of the IPCS
   dialog when TSO/E Release 2 (or a later release of that product) is installed.
   The first part of the CLIST uses the EVALDEF subcommand to obtain the
   SETDEF-defined defaults as follows. The defaults shown will, by default, be the
   local defaults.

   EVALDEF CLIST(SOURCE(SRC) CONFIRM(CON) DISPLAY(DSP) +
   FLAG(FLG) PRINT(PRI) TERMINAL(TER) VERIFY(VER))
   SET CONTROL=FLAG(&FLG) &CON &VER
   SET ROUTE=&PRI &TER
   IF &LASTCC=8 THEN EXIT
   EVALDEF CLIST(QUALIFICATION(QUAL))

   See the BLSCSETD member of SYS1.SBLCSCLI0 for the complete listing.

EVALDUMP subcommand — format dump attributes

Use the EVALDUMP subcommand to retrieve information from a source
description and format that information in CLIST variables, REXX variables, or
ISPF function pool dialog variables.

The source description is for an unformatted source that IPCS can format, for
example, an SVC dump, a stand-alone dump, an SYSMDUMP dump, a trace data
set, a data set, or active storage. The source description is in a directory allocated
with ddbname IPCSDDIR and is your current dump directory. The current dump
directory is your user dump directory or, for users with access authority, might be
the sysplex dump directory.

The source description is for a source that IPCS has initialized or for a source IPCS
accessed during processing of an ADDDUMP subcommand.

Note: With TSO/E Release 2 installed, you can use this subcommand to update
GLOBAL CLIST variables. See z/OS TSO/E CLISTs for additional
information.
EVALDUMP subcommand

Related subcommands
EQUATE
EVALDEF
EVALMAP
EVALSYM

Syntax

EVALDUMP [ relational-operator ]
[ CLIST(var-list) ]
[ DIALOG(var-list) ]
[ REXX(var-list) ]
[ INDATASET(dsname) | INFILE(ddname) ]

------ SETDEF-Defined Parameter --------------------------
Note: You can override the following SETDEF parameter.
See
"SETDEF subcommand — set defaults" on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ TEST | NOTEST ]

Parameters

relational-operator
Specifies a symbolic or programming operators to be used with the source
to identify the source description to be retrieved from the dump directory.

The syntax for relational-operator is as follows:
[ < | LT ]
[ <= | LE ]
[ > | GT ]
[ >= | GE ]
[ = | EQ ]
[ ¬ > | NG ]
[ ¬ <= | NL ]

For example, the less than (<|LT) relationship is satisfied by the
highest-collating source name that also collates lower than the source name
specified on the EVALDUMP subcommand.

CLIST(var-list)
DIALOG(var-list)
REXX(var-list)
Specifies how the default values are to be formatted.

CLIST(var-list) designates that the information be formatted into CLIST
variables.

DIALOG(var-list) designates that the information be formatted into ISPF
function pool dialog variables.

REXX(var-list) designates that the information be formatted into REXX
variables.

INDATASET(dsname)
INDSNAME(dsname)
Requests allocation of directory dsname and use of the contents of that
directory by the subcommand.
EVALDUMP subcommand

**INFILE**(ddname)
**INDNNAME**(ddname)

Requests use of a directory that the IPCS user has allocated to `ddname` and use of the contents of that directory by the subcommand.

The syntax for `var-list` is as follows:

```
[ DECIMAL | F ]
[ HEXADECIMAL | X ]
[ BLOCKS(blocks) ]
[ BYTES(bytes) ]
[ QUALIFICATION(qualification) ]
[ SOURCE(var-name)|DATASET(var-name)|DSNAME(var-name) ]
```

**DECIMAL or F**
**HEXADECIMAL or X**

Specifies the format of the number of blocks.

- DECIMAL or F designates that IPCS format the number of blocks using decimal digits.
- HEXADECIMAL or X designates that IPCS format the number of blocks using hexadecimal digits.

The default is DECIMAL.

**BLOCKS**(blocks)

Places the number of blocks contained in the dump to be formatted in the variable `blocks`.

**BYTES**(bytes)

Formats and places the number of bytes contained in the dump in the variable `bytes`. IPCS always uses decimal for the number of bytes.

**QUALIFICATION**(qualification)

Formats and places the address qualifiers that describe the default address space for the dump in the variable `qualification`.

**SOURCE**(var-name) | **DATASET**(var-name) | **DSNAME**(var-name)

Places the name of the retrieved data set in the variable `var-name`.

**SETDEF-Defined Parameters**

**ACTIVE** or **MAIN** or **STORAGE**
**DSNAME**(dsname) or **DATASET**(dsname)
**FILE**(ddname) or **DDNAME**(ddname)

Specifies the source of the source description from which you want to retrieve information. If one of these parameters is not specified, IPCS uses your current source.

**Return Codes**

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the EVALDUMP subcommand.

**Example**

The BLSCEDUM CLIST lists the number of blocks and bytes for each source in the dump directory. It uses the EVALDUMP subcommand to retrieve the information as follows:
EVALDUMP subcommand

EVALDUMP >= ACTIVE CLIST(SOURCE(SRC) BLOCKS(JL) BYTES(JY))

See the BLSCEDUM member of SYS1.SBLSCL10 for the complete listing.

EVALMAP subcommand — format a storage map entry

Use the EVALMAP subcommand to retrieve information associated with an entry in the storage map and to format that information in CLIST variables, REXX variables, or ISPF function pool dialog variables.

The storage map is part of a source description. The source description is for an unformatted source that IPCS can format, for example, an SVC dump, a stand-alone dump, an SYSDUMP dump, a trace data set, a data set, or active storage. The source description is in a directory allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with access authority, might be the sysplex dump directory.

Numeric information may be formatted in decimal or hexadecimal. Default formatting for pointers and data used in conjunction with pointers is hexadecimal. Default formatting for other numeric data is decimal.

Note: With TSO/E Release 2 installed, you can use this subcommand to update global CLIST variables. For information about using global variables and writing your own CLISTs, see z/OS TSO/E CLISTs and z/OS MVS IPCS User’s Guide.

Related subcommands
EQUATE
EVALDEF
EVALDUMP
EVALSYM

Syntax

EVALMAP [ relational-operator ]

data-descr

[ SELECT([AREA] [MODULE] [STRUCTURE]) ]

[ CLIST(var-list) ]
[ DIALOG(var-list) ]
[ REXX(var-list) ]

------- SETDEF-Defined Parameter "-------------------
Note: You can override the following SETDEF parameter.
See "SETDEF subcommand — set defaults" on page 5-232

[ TEST | NOTEST ]

Parameters
The DIMENSION, ENTRY, HEXADECIMAL, LENGTH, MULTIPLE, POSITION, and X parameters may appear in both the data-descr and var-list variables.

relational-operator
Specifies one of the following symbolic or programming operators to be used in conjunction with the data description to identify which map entry is to be retrieved.
The syntax for relational-operator is as follows:

- `<|LT`  
- `<=|LE`  
- `>|GT`  
- `>=|GE`  
- `>|NG`  
- `<|NL`  
- `|=|EQ`

For example, the less than(<|LT) relationship is satisfied by the highest-collating map entry that collates lower than the byte addressed by the data description.

data-descr

Specifies the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

Note: The qualification, address, and data type are all part of the key of a map entry. To write a CLIST or dialog that moves from one map entry to another, you must specify all three arguments in your data description.

SELECT([AREA][MODULE][STRUCTURE])

Specifies the data types to be returned as results of the EVALMAP command.

AREA

Allows EVALMAP to associate the location of interest with AREAs.

MODULE

Allows EVALMAP to associate the location of interest with MODULEs.

STRUCTURE

Allows EVALMAP to associate the location of interest with STRUCTUREs.

When no selection is specified or all selections are chosen, EVALMAP can associate the location of interest with AREAs, MODULEs, or STRUCTUREs.

CLIST(var-list)
DIALOG(var-list)
REXX(var-list)

Specifies how the information is to be formatted.

CLIST(var-list) designates that the information be formatted into CLIST variables.

DIALOG(var-list) designates that the information be formatted into ISPF function pool dialog variables.

REXX(var-list) designates that the information be formatted into REXX variables.

The syntax for var-list is as follows:
EVALMAP subcommand

[ DECIMAL | F ]
[ HEXADECIMAL | X ]

[ ADDRESS(address) ]
[ ANALYSIS(analysis) ]
[ DATATYPE(type[,group]) ]
[ DIMENSION(dimension)|MULTIPLE(dimension) ]
[ ENTRY(entry) ]
[ FLAG(flag) ]
[ LENGTH(length) ]
[ POSITION(position) ]
[ QUALIFICATION(qualification) ]

DECIMAL or F  
HEXADECIMAL or X

Specifies the format of the numeric information.

DECIMAL or F designates that the numeric information be formatted using decimal digits.

HEXADECIMAL or X designates that the numeric information be formatted using hexadecimal digits.

The following table summarizes the effect of DECIMAL and HEXADECIMAL on the other parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Specifying DECIMAL changes the default</th>
<th>Specifying HEXADECIMAL changes the default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>HEXADECIMAL</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DIMENSION</td>
<td>DECIMAL</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>ENTRY</td>
<td>DECIMAL</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>LENGTH</td>
<td>DECIMAL</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>POSITION</td>
<td>HEXADECIMAL</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

ADDRESS(address)

Requests that the address associated with the map entry be formatted and placed in the variable address. Unless DECIMAL is specified, the address is formatted in hexadecimal; if DECIMAL is specified, decimal digits are used.

ANALYSIS(analysis)

The degree of validation completed for the block is placed in the variable analysis:

NOCHECKER
NONE
PARTIAL
COMPLETE

DATATYPE(type[,group])

Requests that the data type associated with the map entry be formatted and placed in the variable type.
EVALMAP subcommand

If you specify group, EVALMAP formats the group data type and places it in the variable group. For example, if type is set to STRUCTURE(UCBDA) for an MVS dump, group is set to STRUCTURE(UCB).

**DIMENSION(dimension) | MULTIPLE(dimension)**
Requests that the dimension, or replication factor, for the map entry be formatted and placed in the variable dimension. Unless HEXADECIMAL is specified, the dimension is formatted in decimal; if HEXADECIMAL is specified, hexadecimal digits are used.

If the map entry is defined as a SCALAR, a zero dimension is supplied. The return code is set to 4 unless a more serious condition is also detected.

**ENTRY(entry)**
Requests that the subscript associated with the initial array entry described by the map entry be formatted and placed in the variable entry. Unless HEXADECIMAL is specified, the subscript is formatted in decimal; if HEXADECIMAL is specified, hexadecimal digits are used.

If the map entry is defined as a SCALAR, a zero subscript is supplied. The return code is set to 4 unless a more serious condition is also detected.

**FLAG(flag)**
Requests that the most severe condition detected when the validity of the block was checked be placed in the variable flag:

- INFORMATIONAL
- WARNING
- ERROR
- SERIOUS

**LENGTH(length)**
Requests that the length associated with the map entry be formatted and placed in the variable length. Unless HEXADECIMAL is specified, the length is formatted in decimal; if HEXADECIMAL is specified, hexadecimal digits are used.

If the data described is an array, length is for one entry in the array. To calculate the length of the array, multiply the length by the dimension.

**POSITION(position)**
Requests that the signed offset associated with the map entry be formatted and placed in the variable position. The offset is the number of bytes skipped between the address of the data and the first physical byte described.

Unless DECIMAL is specified, the address is formatted in hexadecimal; if DECIMAL is specified, decimal digits are used.

**QUALIFICATION(qualification)**
Requests that the address qualifiers be formatted and placed in the variable qualification. The address qualifiers are for the address space described by the map entry.

**Return Codes**
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the EVALMAP subcommand.

**Example**
The BLSCEMAP CLIST counts all the task control blocks (TCBs) in the storage map for the default data set and displays the sum. It uses the EVALMAP subcommand to retrieve the information as follows:
EVALMAP subcommand

EVALMAP >> 0. ABSOLUTE STRUCTURE CLIST(QUALIFICATION(Q) +
ADDRESS(A) DATATYPE(T))

See the BLSCEMAP member of SYS1.SBLSCLI0 for the complete listing.

EVALPROF subcommand — format PROFILE subcommand options

Use the EVALPROF subcommand values to format the values in CLIST variables,
REXX variables, or ISPF function pool dialog variables.

The default values are established from the dump directory during IPCS session
initialization. You can modify the defaults using the PROFILE subcommand during
the course of your session, which will cause the values to become effective
immediately and recorded as defaults for a subsequent session where the same
directory is used.

Related subcommands

EVALDEF
EVALDUMP
EVALMAP
EVALSYM
PROFILE

Syntax

VERB OPERANDS
EVALPROF { CLIST(variable-list) } { DIALOG(variable-list) } { REXX(variable-list) }

-------- SETDEF-Defined Parameter --------------------------
Note: You can override the following SETDEF parameter.
See "SETDEF subcommand — set defaults" on page 5-232

Parameters

CLIST(var-list)
DIALOG(var-list)
REXX(var-list)

Specifies how the information is to be formatted.

CLIST(var-list) designates that the information be formatted into CLIST
variables.

DIALOG(var-list) designates that the information be formatted into ISPF
function pool dialog variables.

REXX(var-list) designates that the information be formatted into REXX
variables.

The syntax for var-list is as follows:

EXCLUDE(variable-name)
LINESIZE(variable-name)
PAGESIZE(variable-name)
STACK(variable-name)
EVALSYM subcommand — format the definition of a symbol

Use the EVALSYM subcommand to retrieve information associated with a symbol and format that information in CLIST variables, REXX variables, or ISPF function pool dialog variables.

The symbol is in a symbol table that is part of a source description. The source description is in a directory allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with access authority, might be the sysplex dump directory.

Numeric information can be formatted in decimal or hexadecimal. Default formatting for pointers and data used in conjunction with pointers is hexadecimal. Default formatting for other numeric data is decimal.

**Note:** With TSO/E Release 2 installed, you can use this subcommand to update global CLIST variables. For information about using global variables and writing your own CLISTS, see [z/OS TSO/E CLISTS](https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.zos/tso_jfcclists.html) and [z/OS MVS IPCS User’s Guide](https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.zos/tso_jfmsipm.html).

**Related subcommands**
- EQUATE
- EVALDEF
- EVALDUMP
- EVALMAP
- EVALUATE

**Guideline:** EVALUATE does not handle log streams nor does it deal with dumps or traces in added status within the dump directory. The ability to format the value of a literal symbol was added to EVALSYM to enable command procedures to access such values in these circumstances.

**Syntax**
EVALSYM subcommand

EVALSYM  [ relational-operator ]

symbol

[ CLIST(var-list) ]
[ DIALOG(var-list) ]
[ REXX(var-list) ]

[ INDATASET(dsname) | INFILE(ddname) ]

------- SETDEF-Defined Parameter -----------------------------
Note: You can override the following SETDEF parameter. See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ TEST | NOTEST ]

Parameters

relational-operator
Specifies one of the following symbolic or programming operators to be
used in conjunction with the data description to identify which map entry
is to be retrieved.

The syntax for relational-operator is as follows:

[ <  LT ]
[ <= LE ]
[ >  GT ]
[ >= GE ]
[ => NG ]
[ =  NL ]
[ ¬ >  NLT ]
[ ¬ <= NL ]
[ =  EQ ]

For example, the less than (<|LT) relationship is satisfied by the
highest-collating map entry that collates lower than the byte addressed by
the data description.

symbol
Specifies a symbol to be used with a relational operator. The definition of
the symbol is to be retrieved.

CLIST(var-list)
DIALOG(var-list)
REXX(var-list)

Specifications how the information is to be formatted.

CLIST(var-list) designates that the information be formatted into CLIST
variables.

DIALOG(var-list) designates that the information be formatted into ISPF
function pool dialog variables.

REXX(var-list) designates that the information be formatted into REXX
variables.

INDATASET(dsname)
INDSNNAME(dsname)

Requests allocation of directory dsname and use of the contents of that
directory by the subcommand.
EVALSYM subcommand

**INFILE(ddname)**
**INDDNAME(ddname)**

Requests use of a directory that the IPCS user has allocated to *ddname* and use of the contents of that directory by the subcommand.

The syntax for *var-list* is as follows:

```
DECIMAL or F
HEXADECIMAL or X
ADDRESS(address)
DATATYPE(type[,group])
DIMENSION(dimension)|MULTIPLE(dimension)
DROP(drop)
ENQUOTE|UNQUOTE|NOQUOTES
ENTRY(entry)
FLAG(flag)
LENGTH(length)
NOBLANKS
POSITION(position)
QUALIFICATION(qualification)
REMARK(remark)
SYMBOL(symbol)
VALUE(value)
```

**DECIMAL or F**
**HEXADECIMAL or X**

Specifies the format of the numeric information:
- DECIMAL or F for decimal
- HEXADECIMAL or X for hexadecimal

**ADDRESS(address)**

Places in the variable *address* the address associated with the symbol. Unless DECIMAL is specified, the address is formatted in hexadecimal; if DECIMAL is specified, decimal is used.

**DATATYPE(type)**

Places in the variable *type* the data type for the symbol. The preferred representations for the data type are:
- BIT (rather than HEXADECIMAL or X)
- CHARACTER (rather than C)
- SIGNED (rather than F)
- POINTER (rather than PTR)

**DIMENSION(dimension) or MULTIPLE(dimension)**

Places in the variable *dimension* the dimension, or replication factor, associated with the symbol. Unless HEXADECIMAL is specified, the dimension is in decimal; if HEXADECIMAL is specified, hexadecimal is used.
EVALSYM subcommand

If the symbol is defined as a SCALAR, a zero dimension is supplied. The return code is set to 4 unless a more serious condition is also detected.

DROP(drop)
Places in the variable drop the value DROP or NODROP.

ENQUOTE | UNQUOTE | NOQUOTES
Specifies how REMARK text is to be formatted:
– ENQUOTE requests a quoted string.
– UNQUOTE and NOQUOTES request that apostrophes (X'7D') translated to periods.

ENTRY(entry)
Places in the variable entry the subscript associated with the initial array entry described by the symbol. Unless HEXADECIMAL is specified, the subscript is in decimal; if HEXADECIMAL is specified, hexadecimal is used.

If the symbol is defined as a SCALAR, a zero subscript is supplied. The return code is set to 4 unless a more serious condition is also detected.

FLAG(flag)
Places in the variable flag the most severe condition detected when the validity of the block was checked:
INFORMATIONAL
WARNING
ERROR
SERIOUS

LENGTH(length)
Places in the variable length the length associated with the symbol. Unless HEXADECIMAL is specified, the length is decimal; if HEXADECIMAL is specified, hexadecimal is used.

If the data described is an array, the length describes one entry in the array. The length of the array may be computed by multiplying the length of one entry by the dimension.

NOBLANKS
Requests that blanks (X'40') in REMARK text be translated to periods.

POSITION(position)
Places in the variable position the signed offset associated with the symbol. The offset is the number of bytes skipped between the address of the data and the first physical byte described. Unless DECIMAL is specified, the address is in hexadecimal; if DECIMAL is specified, decimal is used.

QUALIFICATION(qualification)
Places in the variable qualification the address qualifiers for the address space described by the symbol.

REMARK(remark)
Places in the variable remark the remark associated with the symbol. The remark text is edited for use in CLISTS, REXX execs, or ISPF dialogs:
– EBCDIC lower case alphabetic characters (a-z) are always replaced by uppercase characters (A-Z), and EBCDIC superscript decimal digits (X'B0'-X'B9') are always replaced by common decimal digits (X'F0'-X'F9').
– Characters not present on either the IBM 1403 TN print chain or the IBM 3211 T11 print train are always replaced by periods.
– Ampersands are always replaced by periods.
EVALSYM subcommand

- Blanks are replaced by periods if the NOBLANKS option is selected. Otherwise, blanks are not edited.
- Apostrophes (X’7D’) are left alone if you do not specify ENQUOTE, UNQUOTE, or NOQUOTES. The string placed in the variable is the same length as that of the string in the dump. However, the following parameters affect this option:

ENQUOTE
   One leading apostrophe and one trailing apostrophe are supplied. Apostrophes found in dump data are paired.

UNQUOTE|NOQUOTES
   Apostrophes found in dump data are replaced by periods. The string placed in the variable is the same length as that of the string in the dump.

SYMBOL(symbol)
   Places in the variable symbol the name of the symbol retrieved.

VALUE(value)
   Places in the literal value the value associated with a literal symbol. The following formatting is performed:
   1. If the symbol is not associated with a literal value, a single blank is stored.
   2. Unless HEXADECIMAL is specified, SIGNED and UNSIGNED data are formatted using decimal digits. If HEXADECIMAL is specified, hexadecimal digits are used.
   3. Unless DECIMAL is specified, POINTER data is formatted using hexadecimal digits. If DECIMAL is specified, decimal digits are used.
   4. CHARACTER data is formatted subject to the same criteria used for REMARK text.
   5. All other types of data are formatted using hexadecimal digits.

SETDEF-Defined Parameters

ACTIVE or MAIN or STORAGE
   DSNAMES(dsnames) or DATASETS(dsname)
   FILE(ddname) or DDNAMES(ddname)
   Specifies the source of the source description that contains the symbol. If one of these parameters is not specified, IPCS uses your current source.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the EVALSYM subcommand.

Example
The BLCESYM CLIST counts all the symbols representing task control blocks (TCBs) in the symbol table for the default data set and displays the sum. It uses the following EVALSYM subcommand to retrieve the information:

EVALSYM >> $ CLIST(SYMBOL(SYM) DATATYPE(T))

See the BLCESYM member of SYS1.SBLSCLI0 for the complete listing.

EVALUATE subcommand — retrieve dump data for a variable

Use the EVALUATE subcommand to retrieve information from a dump and format that information in CLIST variables, REXX variables, or ISPF function pool dialog variables.
EVALUATE subcommand

“Default option” on page 5-113 discusses the processing of the EVALUATE subcommand when the CHECK, CLIST, REXX, and DIALOG parameters are all omitted. This is an archaic form of the EVALUATE subcommand that should not be used in new CLISTs, REXX execs, or dialogs. When existing CLISTs and REXX execs are updated, the old subcommand should be replaced with an EVALUATE subcommand using a CLIST, REXX, or DIALOG parameter. See “CLIST, REXX, or DIALOG option” on page 5-112.

Notes:
1. EVALUATE might modify X, the current address.
2. With TSO/E Release 2 installed, you can use this subcommand to update global CLIST variables. For information about using global variables and writing your own CLISTs, see z/OS TSO/E CLISTs and z/OS MVS IPCS User’s Guide.

Related subcommands
– EVALSYM

Syntax

{ EVALUATE } data-descr
{ EVAL }

[ CLIST(var-list) [ MASK(mask) ] ]
[ DIALOG(var-list) [ MASK(mask) ] ]
[ REXX(var-list) [ MASK(mask) ] ]

[ CHECK ]

SETDEF-Defined Parameter

Note: You can override the following SETDEF parameter.

See “SETDEF subcommand — set defaults” on page 5-232

[ TEST | NOTEST ]

Parameters

data-descr
   Specifies the data description parameter, which consists of five parts:
   – An address (required)
   – Address processing parameters (optional)
   – An attribute parameter (optional)
   – Array parameters (optional)
   – A remark parameter (optional)

   Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

MASK(mask)
   Defines a value that is logically ANDed with the retrieved information. The AND operation occurs before the retrieved information is formatted into a variable. The mask must be the same length as the retrieved information. The mask value must be a general value. See Chapter 2, “Literal values,” on page 2-1 for more information about specifying a general value.

CHECK
   Directs IPCS to inform a CLIST, REXX exec, or ISPF dialog whether 1 to 4 bytes of storage can be accessed in a dump. “CHECK option” on page 5-113 below discusses this option further.

CLIST(var-list)
DIALOG(var-list)
EVALUATE subcommand

REXX(var-list)
  Specifies how to format the information.
  CLIST(var-list) designates that the information be formatted into CLIST variables.
  DIALOG(var-list) designates that the information be formatted into ISPF function pool dialog variables.
  REXX(var-list) designates that the information be formatted into REXX variables.

The syntax for var-list is as follows:

[ ENQUOTE|UNQUOTE|NOQUOTES ]
[ NOBLANKS ]
[ PROTECTION(protection) ]
[ STORAGE(storage) ]
[ FORMATTED|UNFORMATTED ]

ENQUOTE or UNQUOTE or NOQUOTES
  Specifies how CHARACTER data is to be formatted:
  – ENQUOTE requests a quoted string.
  – UNQUOTE and NOQUOTES request that apostrophes (X’7D’) translated to periods.

NOBLANKS
  Requests that blanks (X’40’) in CHARACTER data be translated to periods.

PROTECTION(protection)
  Specifies the name of the CLIST, REXX, or ISPF dialog variable into which IPCS places the formatted protection key.

Note: When no storage key is known for a block of storage, IPCS supplies the value X’FF’ This occurs when IPCS processes DOMAIN(SUMDUMP) records and active storage. The following topic, "CLIST, REXX, or DIALOG option" on page 5-112 discusses the processing performed.

STORAGE(storage)
  Specifies the name of the variable into which IPCS places the formatted storage.

FORMATTED or UNFORMATTED
  Specifies how the information is to be returned:
  – FORMATTED
    Formatted data is returned. This is the default.
  – UNFORMATTED
    Unformatted data is returned. This option is mutually exclusive with the following var-list keywords:
    - ENQUOTE | UNQUOTE | NOQUOTES
    - NOBLANKS
    The UNFORMATTED keyword causes the storage variable, if specified, to receive an image of the data requested. The storage that can be processed is 32760 bytes.
EVALUATE subcommand

CLIST, REXX, or DIALOG option

EVALUATE processing is divided into four parts:

1. The data description is edited, if necessary:
   - If the length of data is more than 512 bytes, LENGTH(512) is substituted.
   - If an array containing multiple entries is described, DIMENSION(1) is substituted.
   - If a data type other than bit, character, pointer, signed, or unsigned is specified, BIT is substituted.

   Return code 4 is set when editing occurs.

2. The storage described by the edited data description is retrieved.

   If the storage is not available, EVALUATE processing ends with return code 12.

3. If storage formatting was requested, the data is formatted and stored in a variable. Formatting is primarily controlled by the type of data retrieved:

   **BIT|POINTER** — Bit string and pointer data is formatted using 2 hexadecimal digits for each byte retrieved.

   **CHARACTER** — Character string data is edited for use in CLISTs, REXX execs, or ISPF dialogs:
   - EBCDIC lower case alphabetic characters (a-z) are replaced by uppercase characters (A-Z), and EBCDIC superscript decimal digits (X'B0'-X'B9') are replaced by common decimal digits (X'F0'-X'F9').
   - Characters not present on either the IBM 1403 TN print chain or the IBM 3211 T11 print train are replaced by periods.
   - Ampersands are replaced by periods.
   - Blanks are replaced by periods if the NOBLANKS option is selected. Blanks are not changed otherwise.
   - Editing of apostrophes (X'7D') is governed by the subcommand option selected:

     **ENQUOTE**
     One leading and one trailing apostrophe are supplied. Apostrophes found in dump data are paired.

     **UNQUOTE|NOQUOTES**
     Apostrophes found in dump data are replaced by periods. The string placed in the variable is the same length as that of the string in the dump.

     If no subcommand option is specified, apostrophes are not edited. The string placed in the variable is the same length as that of the string in the dump.

   **SIGNED** — Signed binary integers are formatted using decimal digits. Leading zeros are removed. A minus sign is supplied for negative integers.

   **UNSIGNED** — Unsigned binary integers are formatted using decimal digits. Leading zeros are removed.

4. If the protection key was requested, it is formatted and stored in a variable.

   The protection key is formatted using 2 hexadecimal digits.
   - If no storage key was provided by the dumping program or multiple inconsistent storage keys (different fetch-protection or reference key values) apply to the storage, the value stored is X'FF'.
   - Otherwise, the value is formatted using the fetch-protection and reference key bits that apply to all storage described. The reference and change bits are represented as on if they are on for any block of storage described.
EVALUATE subcommand

5. If no storage formatting was requested with UNFORMATTED, the data requested is returned in the area specified by STORAGE. The amount of data retrieved can be up to 32760 bytes. When UNFORMATTED is specified, the use of ENQUOTE | UNQUOTE | NOQUOTES and NOBLANKS is not allowed.

If the CLIST, REXX, or DIALOG option is specified, EVALUATE uses its return code to indicate whether the requested operation was successful.

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion</td>
</tr>
<tr>
<td>04</td>
<td>Description of data was edited.</td>
</tr>
<tr>
<td>12</td>
<td>Data not available or not defined. The variables are not changed.</td>
</tr>
</tbody>
</table>

Default option

The default option of the EVALUATE subcommand retrieves an unsigned binary number from a dump and uses that number as its return code. The number in the dump may span 1 to 4 bytes.

Note: If a 4-byte number is used as a return code, EVALUATE translates the high-order byte of the number to zeros after retrieving it from the dump and before using it as a return code. This reduces the actual precision of the value from 32-bits (0 to 2^31-1) to 24-bits (0 to 2^23-1) because the latter is the precision used for TSO command and subcommand return codes.

In a CLIST, the subcommand following EVALUATE can refer to the return code with the CLIST variable &LASTCC. EVALUATE has little use other than in CLISTS because the return code is made available by the CLIST variable &LASTCC.

Each subcommand in a CLIST resets &LASTCC. Thus, the data retrieved by EVALUATE must be examined or moved from &LASTCC before another subcommand in the CLIST overlays it.

Use caution in using the contents of &LASTCC after this subcommand. It may contain data or a return code; however, there is no way of determining which. For example, if the specified storage cannot be retrieved, EVALUATE generates return code 12. This is, in fact, a return code indicating the failure to retrieve the data, but it can be interpreted as data.

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Severe, requested storage cannot be retrieved.</td>
</tr>
<tr>
<td>16</td>
<td>Terminating, an error condition from a called service routine forced an early termination.</td>
</tr>
<tr>
<td>other</td>
<td>Successful completion, uses the requested data as a return code.</td>
</tr>
</tbody>
</table>

CHECK option

If the CHECK option is specified, EVALUATE uses its return code to indicate whether diagnostic data can be retrieved. It is also used to indicate other concerns if the same data description is used with the default form of EVALUATE.

Return codes for the CHECK option

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Severe, requested storage cannot be retrieved.</td>
</tr>
<tr>
<td>16</td>
<td>Terminating, an error condition from a called service routine forced an early termination.</td>
</tr>
<tr>
<td>other</td>
<td>Successful completion, uses the requested data as a return code.</td>
</tr>
</tbody>
</table>
**EVALUATE subcommand**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful completion</td>
</tr>
<tr>
<td>04</td>
<td>Description of data was edited</td>
</tr>
<tr>
<td></td>
<td>• If the length is more than 4 bytes, LENGTH(4) is substituted.</td>
</tr>
<tr>
<td></td>
<td>• If an array containing multiple entries is described, DIMENSION(1) is substituted.</td>
</tr>
<tr>
<td></td>
<td>• Only the UNSIGNED data type is supported. If another data type is described, UNSIGNED is substituted.</td>
</tr>
<tr>
<td>08</td>
<td>Four bytes of data were retrieved but the initial byte does not contain X'00'.</td>
</tr>
<tr>
<td></td>
<td>Significance is lost if the first byte of a fullword is removed. That byte does not contain X'00'.</td>
</tr>
<tr>
<td>12</td>
<td>Data not available or not defined.</td>
</tr>
</tbody>
</table>

**FIND subcommand — locate data in a dump**

Use the FIND subcommand to locate literal values in a dump.

**Search argument and options**

You must specify a search argument the first time you use FIND. FIND saves the search argument and any options you specify:

- The data type of the search argument allows you to request signed binary comparisons or logical (bit by bit) comparisons.
- A relational operator allows you to indicate whether the data sought is less than, equal to, or greater than the search argument, and so on.
- The BOUNDARY option allows you to search only for data aligned on storage boundaries, such as doubleword boundaries.
- The BREAK option allows you to stop when storage is missing for a comparison or continue the search beyond the missing storage.
- The MASK option allows you to ignore selected bits when the search argument is compared with storage.

If you omit a search argument later, the subcommand uses the saved argument and options. If you override options, the new options are merged with those saved earlier and all options are saved.

If you respecify a search argument, the saved options are discarded.

**Storage searched**

You can limit the search by specifying the range of addresses to be searched. FIND uses the symbol FINDAREA (recorded in the symbol table) to describe the beginning address and the length of the area.

The FIRST, LAST, NEXT, and PREVIOUS options allow you to control the direction of a search and to force a search to be resumed at either end of FINDAREA.

Before the search begins, FIND sets X to the first address to be searched. If it locates a match, FIND sets X to the address of the match. Otherwise, FIND leaves X set to the first address searched. If no range of addresses is explicitly set on the initial invocation of the FIND subcommand, IPCS searches an entire address space.

After the subcommand sets the search range (FINDAREA and its length), if you request another search without specifying a new range and if X is outside the
current search range, FIND ends immediately, without modifying X. (X can be outside the current search range only if you have modified FINDAREA, X, or both between the two searches.)

If you do not specify a beginning address for the search range but you do specify a search argument, FIND begins the search at X. If you do not specify a beginning address for the search range or a search argument, FIND begins the search at:
- X + 1 if FIND FIRST or FIND NEXT processing is being resumed.
- X - 1 if FIND LAST or FIND PREVIOUS processing is being resumed.

In either case, the end point of the search range remains the same.

Note: This subcommand may modify X, the current address.

Related subcommands
FINDMOD
FINDUCB

Syntax

{ FIND } [ relational-operator ]
{ F } [ value ]
[ data-descr ]
[ BOUNDARY(bdy [,index-range]) ]
[ BREAK | NOBREAK ]
[ FIRST ]
[ LAST ]
[ NEXT ]
[ PREVIOUS ]
[ MASK(mask) ]

----- SETDEF-Defined Parameters ------------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232

[ DISPLAY[(display-options) ] ]
[ NODISPLAY[(display-options) ] ]
[ FLAG(severity) ]

[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]

Parameters

relational-operator
Specifies one of the following symbolic or programming operators to be used with the value parameter and the BOUNDARY, BREAK, and MASK parameters to establish the search criterion:
[<|LT|<=|LE|>|GT|¬|NG|=|EQ|>=|GE|¬<|NL|>|GT|¬|NE]

value
Specifies a general value. See Chapter 2, “Literal values,” on page 2-1 for
**FIND subcommand**

information, syntax, and examples. If the BOUNDARY, BREAK, and MASK parameters are not specified in the FIND subcommand, the default options are:

- **BOUNDARY(1,1)**
- **BREAK**
- **NOMASK**

**data-descr**

Specifies the data description parameter, which consists of five parts:

- An address (required when `data-descr` is explicitly specified on the subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter. However, the following exception applies to the FIND subcommand only:

- The address is not a positional parameter. You must use the ADDRESS parameter to specify an address.

**BOUNDARY(bdy[,index-range])**

Requests that storage be partitioned into strings `bdy` bytes in length. The address of each string is divisible by `bdy`. FIND performs only one comparison with data whose first byte lies within any string. The abbreviation BDY is accepted for this parameter. The index value designates which byte FIND is to select:

- **BDY(1) or BDY(1,1) or BDY(1,1:1)**
  - FIND examines each byte.
- **BDY(2) or BDY(2,1) or BDY(2,1:1)**
  - FIND performs comparisons with strings originating at even-numbered addresses.
- **BDY(2,2) or BDY(2,2:2)**
  - FIND performs comparisons with strings originating at odd-numbered addresses.
- **BDY(5,5) or BDY(5,5:5)**
  - FIND performs comparisons only with strings originating at addresses 4 bytes past an address divisible by 5.
- **BDY(7,6:7)**
  - FIND performs comparisons only with strings originating at addresses 5 or 6 bytes past an address divisible by 7.
- **BDY(8) or BDY(8,1) or BDY(8,1:1)**
  - FIND performs comparisons only with strings aligned on doubleword boundaries.

Both `bdy` and `index-range` can be 1 through $2^{31}$ and can be specified in decimal, hexadecimal (`X'xxx...'`), or binary (`B'bbb...'`).

When you specify this option, it remains in effect until you specify a new search argument or override this option. The default, BDY(1,1), is used only when a new search argument is entered and this option is omitted.
BREAK or NOBREAK
Indicates if FIND is to continue processing if IPCS cannot retrieve storage from the dump.

BREAK specifies that FIND is to stop processing if it cannot retrieve storage from the dump to continue the search. This happens if the required storage was not obtained by IPCS or the required storage is not contained in the dump.

NOBREAK specifies that FIND is to continue processing if it cannot retrieve storage from the dump. FIND continues the search with the next available address in the dump.

When you specify BREAK or NOBREAK, it remains in effect until you specify a new search argument or you override this option. The default of BREAK is used only when a new search argument is entered and this option is omitted.

FIRST
LAST
NEXT
PREVIOUS
Specifies where the search is to begin.
FIRST specifies that the search is to begin at the lowest address in FINDAREA and is to proceed from low-numbered addresses to higher addresses.
LAST specifies that the search is to begin at the highest address in FINDAREA and is to proceed from high-numbered addresses to lower addresses.
NEXT specifies that the search is to proceed from low-numbered addresses to higher addresses.
PREVIOUS specifies that the search is to proceed from high-numbered addresses to lower addresses.

MASK(mask) | NOMASK
Requests or suppresses a mask.
MASK defines a value that is logically ANDed with both operands before performing the comparison. The mask must be the same size as the data items being compared.

The mask value must be a general value. See "General values" on page 2-3 for more information.

NOMASK suppresses masking.

Return codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the FIND subcommand.

Example 1
Search for a character string in the first 10 columns of an 80-byte record in a buffer pool. The first 10 columns contain a character string.

- Action
  COMMAND ===> find c'ABC' addr(bufferpool) bdy(80,1:10)
- Result
**FIND subcommand**

X is set to describe the 3 bytes of storage in which the data was found. If the VERIFY parameter is in effect, FIND displays where the match was found. The actual content of the display is controlled by the DISPLAY parameters in effect.

**Example 2**

Search for a fullword pointer that is present in the storage searched.

- **Action**

  COMMAND ===> find a'fdfd' bdy(4)

- **Result**

  X is set to describe the 4 bytes of storage in which the data was found. If the VERIFY parameter is in effect, FIND displays where the match was found. The actual content of the display is controlled by the DISPLAY parameters in effect.

**Example 3**

Search the NUCLEUS CSECT table for the entry containing a requested address. The table is aligned on a page boundary and contains a series of 16-byte entries. For example:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Name of NUCLEUS CSECT in EBCDIC</td>
</tr>
<tr>
<td>08</td>
<td>Address of NUCLEUS CSECT</td>
</tr>
<tr>
<td>0c</td>
<td>Length of NUCLEUS CSECT</td>
</tr>
</tbody>
</table>

The entries in the table are sorted in ascending order by the address of the NUCLEUS CSECT.

- **Action**

  COMMAND ===> find [= a'requested-address' address(table-origin :table-end) bdy(16,9) last

- **Result**

  This command updates X to describe the ninth through the twelfth bytes of the table entry. That is, X describes the field that contains the address of the NUCLEUS CSECT.

Here is a breakdown of each parameter's function in this example:

- The relational-operator, [=, causes the search to fail for all table entries associated with CSECTS whose addresses are greater than the requested-address.
- The fullword pointer, requested-address, is the value sought.
- ADDRESS(table-origin :table-end) limits the search within the bounds of the table. No address processing parameters are included because it is assumed that the table is visible from the default address space in the dump.
- bdy(16,9) causes comparisons to be made with strings originating at addresses 8 bytes past an address divisible by 16.
- LAST causes the search to begin from the end of the table and proceed to its beginning.

**FINDMOD subcommand — locate a module name**

Use the FINDMOD subcommand to locate a module in the dump. IPCS searches as follows, in order:

1. Searches the symbol table for the specified symbol name with the attribute MODULE
FINDMOD subcommand

2. Searches the active link pack area (LPA) queue in the dump for the module in the MLPA/EMLP and FLPA/EFLPA
3. Searches the LPA directory in the dump for the module in the PLPA/EPLPA

If FINDMOD finds the requested module in the symbol table, it does not create new symbols. If it finds the requested module on the CDE chain, it creates the symbols:
   CDEmodulename
   XLmodulename
   modulename

If it finds the requested module on the LPDE chain, it creates the symbols:
   LPDEmodulename
   modulename

Note: This subcommand can modify X, the current address.

Related subcommands
   FIND
   FINDUCB

Syntax

{FINDMOD } modulename
{FMOD}

[ CHARACTER ]
[ HEXADECIMAL ]

--------------- SETDEF-Defined Parameters -------------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ DISPLAY[(display-options)] ]
[ NODISPLAY[(display-options)] ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]

Parameters

modulename
   Specifies the module name to be located.

CHARACTER

HEXADECIMAL
   Indicates how the module name is specified in modulename.
   CHARACTER indicates a string of 1 to 8 EBCDIC characters.
FINDMOD subcommand

HEXADECIMAL indicates a string of 2 to 16 hexadecimal digits.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the FINDMOD subcommand.

FINDSWA subcommand — locate a scheduler work area (SWA) block

Use the FINDSWA subcommand to locate a Scheduler Work Area (SWA) block, including a SWA block prefix, in a dump.

Note: This subcommand can modify X, the current address.

Related subcommands
“CBFORMAT subcommand — format a control block” on page 5-29
“FIND subcommand — locate data in a dump” on page 5-114
“FINDMOD subcommand — locate a module name” on page 5-118
“FINDUCB subcommand — locate a UCB”

Syntax

{ FINDSWA } data-descr
{ FSWA } CONTEXT ( JSCBACTIVE | symbol )

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232.

Parameters

data-descr
Describes the location of a 3-byte SWA virtual token (SVA) for the SWA block of interest.

CONTEXT(JSCBACTIVE)
CONTEXT(symbol)
Describes the context in which the SVA is to be interpreted. If a symbol other than JSCBACTIVE is designated, it must describe either a STRUCTURE(JSCB) or a STRUCTURE(TCB).

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the FINDSWA subcommand.

FINDUCB subcommand — locate a UCB

Use the FINDUCB subcommand to locate the unit control block (UCB) for a specified device. When the subcommand finds the control block, it creates an entry in the symbol table for UCBddddd, where dddd is the device number.

FINDUCB processes the specified device number as follows:
1. Searches the symbol table for the symbol UCBddddd. If found, IPCS displays the storage associated with that symbol.
2. Verifies that the device was defined during system initialization.
3. Locates the device's UCB.

Notes:
1. This subcommand may modify X, the current address.
2. Casual use of the FINDUCB subcommand is not recommended because FINDUCB's processing requires a great deal of time.
FINDUCB subcommand

Related subcommands
- FIND
- FINDMOD

Syntax

{FINDUCB } device-number
{FINDU}

----- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ DISPLAY[(display-options)] ]
[ NODISPLAY[(display-options)] ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]

Parameters
device-number
Specifies the device number of the device whose UCB is to be found. The
number is 1 to 4 hexadecimal digits; leading zeros are optional.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the FINDUCB subcommand.

Example
Locate the UCB for device number 8000.
- Action
  COMMAND ===> FINDUCB 8000
- Result

UCB8000 - UNIT CONTROL BLOCK FOR CHANNEL TO CHANNEL ADAPTER
LIST 01D0E028 ASID(X'0001') POSITION(X'0008') LENGTH(48) STRUCTURE(UCBCTC)

Even if you are using captured UCBs, FINDUCB returns the address of the
actual UCB. In this example, the actual UCB address is 01D0E028.

GO subcommand — resume IPCS trap processing

Use the GO subcommand to resume trap processing after the STOP trap option is
encountered on the TRAPON subcommand. See “TRAPON subcommand —
activate IPCS traps” on page 5-290 for more information.
GO subcommand

The GO subcommand is valid only during STOP processing for an exit debugging trap. When GO is used and STOP processing is not in effect, IPCS issues message BLS21006I.

Note: The GO subcommand can be entered only in line mode. It cannot be entered while in the IPCS dialog.

Related subcommands
   TRAPON
   TRAPOFF
   TRAPLIST

Syntax

GO

Return codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the GO subcommand.

GRSDATA subcommand — format Global Resource Serialization data

Use the GRSDATA subcommand to format reports showing serialization effected by the ENQ, DEQ, ISGENQ, RESERVE, and latch service interfaces.

Note that when the GRS is running in STAR mode, the output of the GRSDATA subcommand is dependent on the GRSQ option setting of the parmlib member GRSCNFxx. For more information about the GRSCNFxx GRSQ setting, see the z/OS MVS Planning: Global Resource Serialization.

Related subcommands
   ANALYZE
   STATUS

Syntax
GRSDATA subcommand

GRSDATA
The parameters are:

Data Selection Parameters:

[DETAIL]
[SUMMARY]

Additional Filter Parameters:

[SYSNAME(sysname)]
[QNAME(qname)]
[RNAME(rname)]
[STEP] [ SYSTEM] [ SYSTEMS]
[JOBNAME(jobname)]
[ASID(asid)]
[TCB(tcb)]
[RESERVE]
[CONTENTION]

------ SETDEF-Defined Parameters ---------------
Note: You can override the following SETDEF parameters.
See

"SETDEF subcommand — set defaults" on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters
Data selection parameters

DETAIL
  Provides a detailed GRSTRACE report.

SUMMARY
  Provides a summary GRSTRACE report.

Note: DETAIL and SUMMARY with GRSDATA produce the same report.

Additional Filter Parameters
Use these parameters to limit the scope of the data in the report. If no data
selection parameter is selected, the default is DETAIL.

SYSNAME(sysname)
  Displays all ENQ resources with the given specified system name. Note in
  GRS=STAR, if the specified GRSQ option is LOCAL, only resource requests
  from the dumped system will be displayed.

QNAME(qname)
  Displays all ENQ resources with the specified QNAME (major name).

RNAME(rname)
  Displays all ENQ resources with the specified RNAME (minor name).

[STEP] [ SYSTEM] [ SYSTEMS]
  Displays all ENQ resources with a scope of STEP, SYSTEM, or SYSTEMS.
GRSDATA subcommand

**JOBNAME( jobname )**
Displays all ENQ resources associated with the specified job name.

**ASID( asid )**
Displays all ENQ resources associated with the specified address space ID.

**TCB( tcb )**
Displays all ENQ resources associated with the specified task

**RESERVE**
Displays only RESERVE requests that have not been converted to global ENQs.

**CONTENTION**
Displays only ENQ resources that are in ENQ contention. Device RESERVE contention is not taken into consideration.

**Return codes**
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the GRSDATA subcommand.

**Example**
Format a global resource serialization report.
- Action
  COMMAND ===> GRSDATA
- Result
  IPCS produces the following output when SDATA=GRSQ information is found in a dump.
Resources are presented in the following order:
1. ASID(X'xxxx') (STEP) resources (ordered by ASID)
2. Local (SYSTEM) resources
3. Global (SYSTEMS) resources

This is consistent with the order used by verb exit QCBTRACE in prior releases and with the order used by the GRSDATA subcommand in the current release when GRS control blocks are used instead of the data collected with the SDATA=GRSQ option of SDUMP.

Major resource names are presented using notation similar to that used by assembler language coders. GRSDATA expects that uppercase letters, including national characters, decimal digits, blanks and a small number of punctuation characters are printable on all media. If there is reason to believe that the major name cannot be accurately shown on all media, a comma is placed after the EBCDIC representation and a precise hexadecimal representation is added. For example,
Minor resource names are presented using notation familiar to assembler language coders with trailing blanks, a common occurrence not shown literally. The same test is made of minor names for printability that is made for major names. If there is reason to believe that the minor name cannot be accurately shown on all media, the hexadecimal representation of the minor name is shown directly after the EBCDIC representation.

The line beginning with the SCOPE caption introduces each paragraph that discusses a TCB that owns or is awaiting ownership of a resource. If the resource is associated with RESERVE processing on a system other than the one dumped, the word RESERVE is added by itself at the end of this line. If the resource is associated with RESERVE processing on the dumped system, RESERVE is used as a caption for a device address.

The line beginning with the ASID caption adds system internal status to what was provided on the line beginning with the SCOPE caption. The following status values shown in Table 5-3 may appear:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCLUSIVE</td>
<td>Exclusive status held</td>
</tr>
<tr>
<td>MCEXC</td>
<td>Exclusive must-complete status held</td>
</tr>
<tr>
<td>MCSHR</td>
<td>Shared must-complete status held</td>
</tr>
<tr>
<td>SHARED</td>
<td>Shared status held</td>
</tr>
<tr>
<td>WAITEXC</td>
<td>Awaiting exclusive status</td>
</tr>
<tr>
<td>WAITMCE</td>
<td>Awaiting exclusive must-complete status</td>
</tr>
<tr>
<td>WAITMCS</td>
<td>Awaiting shared must-complete status</td>
</tr>
<tr>
<td>WAITSHR</td>
<td>Awaiting shared status</td>
</tr>
</tbody>
</table>

When the status value begins with a 'WAIT', either the SVRB or the ECB address used by GRS for notification is also presented.

Paragraphs that discuss a TCB may also contain a line beginning with a MASID caption, showing the MASID ENQ ASID and TCB address for those resource requests using the MASID option. Similarly, paragraphs that discuss a TCB may also contain a line beginning with a SASID caption when a server address space has performed an ENQ or RESERVE operation on behalf of a requester address space.

---

**GTFTRACE subcommand — format GTF trace records**

Use the GTFTRACE subcommand to format generalized trace facility (GTF) records contained in a dump or in a trace data set. The GTF records must be in a single source. If you have multiple GTF trace data sets, use the COPYTRC subcommand to combine the trace records into one data set.

**Syntax**
GTFTRACE subcommand

\{GTF\} \{GTFTRACE\} [ ASCB(ascb-address-list) ]
\{GTF\} [ ASID(asidlist) ]
\{GTF\} [ JOBNAME(joblist) | JOBLIST(joblist) ]

\{BEGINFIRST\}
\{BEGINOLD\}

\{CICS((text))\}
\{CPU(cpu-address)\}
\{DEBUG\}
\{EOF\}
\{EXIT(pgmname)\}

\{START(ddd, hh.mm.ss)\}
\{STOP(ddd, hh.mm.ss)\}

\{STARTLOC(ddd, hh.mm.ss)\}
\{STOPLOC(ddd, hh.mm.ss)\}

\{SYSNAME(name-list)\}

----------- Data Selection Parameters ---------------
\{CCW[(record-type)]\}
\{DSP\}
\{EXT\}

\{IO[(device-number-list)]\}
\{SSCH[(device-number-list)]\}
\{IOSSCH|SSCHIO[(device-number-list)]\}

\{PI[(codelist)]\}
\{RNIO\}
\{RR\}
\{SLIP\}
\{SRM\}
\{SVC[(svclist)]\}
\{SYS\}

\{USR [(symbol-list)]\}
\{[(idvalue-list)]\}
\{[(idrange-list)]\}
\{[(ALL)]\}

-------- SETDEF-Defined Parameters ---------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232

\{ACTIVE | MAIN | STORAGE\}
\{DSNAME(dsname) | DATASET(dsname)\}
\{FILE(ddname) | DDNAME(ddname)\}
\{PATH(path-name)\}

\{PRINT | NOPRINT\}
\{TERMINAL | NOTERMINAL\}
\{TEST | NOTEST\}
\{FLAG(severity)\}
GTFTRACE subcommand

Note: The PATH keyword is only intended to refer to a dump data set, not an external trace.

Parameters

If you need more than one physical line to enter the GTFTRACE subcommand, continue it with a plus or minus sign as you do with any TSO/E command.

Command =*=*=GTFTRACE DD(SYSTRACE) 10(D34,D0C,E8, + FFF,2A0,2E4)

Standard TSO/E continuation techniques apply to all GTFTRACE subcommand parameters.

ASCB(ascb-address-list)

Specifies ASCB addresses corresponding to the trace entries and user records you want to format. Specify the ASCB address list as one or more 1- to 8-digit hexadecimal addresses, separated by commas.

ASID(asidlist)

Specifies a list of ASIDs for the address spaces for which trace entries and user records are to be formatted.

The asidlist can be a single ASID or a list of noncontiguous ASIDs. When you specify a list, separate the list members with commas. The ASID can be 1 through 65535.

Note: ASID is ignored when processing data from a trace data set.

JOBNAME(joblist) or JOBLIST(joblist)

Specifies one or more job names for which trace entries and user records are formatted. Each job name can be up to 8 characters long. Job names specified for SYSMDUMP data sets are ignored. SYSMDUMPS do not contain the job name field.

Both generic and specific job names may be used in the joblist. A generic job name may use the following wildcards:

- Asterisks to denote any string of valid characters, including no characters. You may use one or more asterisks in any position.
- Percent signs to denote one valid character. Use one percent sign for each character position.

For example, given the following job names:
MPA  MPPA  MPP1A  MAP1A
M00PA  MPP01A  MPPABA  MPPABCA
- MPP*A will match these job names: MPPA, MPP1A, MPPABA, MPPABCA
- M*P*A will match all job names in the list.
- MPP%A will match these job names: MPP01A, MPPABA

Note: *MASTER* represents the master address space.

BEGINFIRST

Requests that formatting start with the first block of records in a trace data set, regardless of TAPE/DASD or wrapping.

BEGINFIRST is the default for tape data sets; it is ignored for dumps. BEGF may be used as the short form of this parameter.

BEGINOLD

Requests that formatting start with the oldest block of records in a trace data set. The command determines the oldest time stamp record, regardless of where the data set resides (TAPE/DASD). GTFTRACE creates the symbol GTFWRAP to save the number of the oldest block across IPCS sessions.
GTFTRACE subcommand

However, the GTFWRAP symbol will not be created if both of the following are true:
- The trace data set has been placed in IPCS fast path access mode (that is, normal initialization of the trace data set has been bypassed).
- The trace data set is wrapped (the first trace record in the data set is not the oldest trace record in the data set).

BEGINOLD is the default for DASD data sets; it is ignored for dumps.
BEGO may be used as the short form of this parameter.

CICS((text))
Specifies that the entered text be placed in a buffer, preceded by a fullword-length field, and that the address of this text buffer be placed in the work area list entry corresponding to the format identification disk (X'EF') assigned to the Customer Information Control System (CICS®). This processing makes the text string addressable by the CICS formatting appendage, AMDUSREF.

CPU(cpu-address)
Specifies that events occurring on the central processor whose physical identifier is cpu-address be formatted. The cpu-address can be any CPU address supported by the current release. And you can use decimal, hexadecimal (X'xxx...'), or binary (B'bbb...') notations to specify the cpu-address.

CPU filtering is only effective with IO-related trace records. Records which are subject to CPU filtering are SSCH, CSCH, HSCH, MSCH, RSCH, IO, EOS, PCI, and CCW.

DEBUG
Specifies the display of the internal control table after parsing the parameters entered on the GTFTRACE subcommand.

EOF
Specifies that the exit routine identified by the EXIT parameter is to receive control on all GTFTRACE normal and abnormal ending conditions.

EXIT(pgmname)
Specifies the program name of a user-written exit routine that inspects all trace data records. When the EOF parameter is specified, IPCS also passes control to this routine at the logical end of the trace data.

If the routine does not exist or if IPCS cannot successfully load it, GTFTRACE processing ends and IPCS processes the next subcommand.

START(ddd,hh.mm.ss) or STARTLOC(ddd,hh.mm.ss)
STOP(ddd,hh.mm.ss) or STOPLOC(ddd,hh.mm.ss)
Specifies that the blocks for processing lie between times. The times for START and STOP are GMT; STARTLOC and STOPLOC indicate local time. IPCS formats only those records that you request with trace data selection parameters. When you do not specify START or STARTLOC, GTFTRACE starts at the beginning of the data set, or at the first block in a dump. When you do not specify STOP or STOPLOC, GTFTRACE completes processing after it reads the end of the data set, or the last block in a dump. The record timestamps are not used, and can have times greater than the block timestamp. ‘ddd’ is Julian day, and ‘hh.mm.ss’ is the hours, minutes and seconds as set in the TOD clock.

Note: You do not need to specify leading zeros.
SYSNAME(name-list)
Filters the GTF data merged from several data sets. When SYSNAME is specified, the GTF data will be formatted only if its system name agrees with one of the values in the name-list. SYSNAME will accept up to 16 names in the name-list.

Data Selection Parameters
Use these parameters to limit the kinds of trace records processed. For these parameters, the phrase "base record" means the first record of the many records that form one logical record. If you omit data selection parameters, the default is SYS.

CCW(record-type)
Requests that channel program trace records be formatted. To format CCW trace records, IPCS formats either SSCH base records or I/O base records, or both. For record-type, you can specify:

I Requests formatting of all the CCW trace records for I/O events, and, if present, program-controlled interrupt (PCI) events. IPCS formats I/O base records even if you do not specify the IO parameter. When you specify both the IO parameter and CCW(I), IPCS formats only the CCW trace records for events on the devices identified on the IO parameter.

S Requests formatting of all CCW trace records for start subchannel and resume subchannel operations. IPCS formats SSCH base records even if you do not specify the SSCH parameter. When you specify both the SSCH parameter and CCW(S), IPCS formats only the CCW trace records for events on the devices identified by the SSCH parameter.

SI Requests formatting of all CCW, I/O, start subchannel, and resume subchannel trace records in the specified data set. IPCS formats SSCH and I/O base records even if you do not specify the SSCH and IO parameters. When you specify the SSCH and IO parameters, with either CCW or CCW(SI), IPCS formats only the CCW trace records for events on the devices identified by the SSCH and IO parameters.

DSP Requests that IPCS format all dispatching event trace records.

EXT Requests that IPCS format all trace records for external interruptions.

IO(device-number-list)
SSCH(device-number-list)
IOSSCH|SSCHIO(device-number-list)
Request formatting of I/O trace records, SSCH trace records, or both. Supplied alone, the IO parameter specifies formatting of IO, PCI, HSCH, CSCH, and MSCH trace records. The SSCH parameter tells IPCS to format start and resume subchannel trace records.

SSCHIO and IOSSCH are synonymous. Either one requests formatting of both I/O and start and resume subchannel records.

device-number-list can contain from 1 to 50 device numbers, for which you want either or both types of trace records formatted. IPCS formats trace records only for the specified devices. If you do not specify any device numbers, IPCS formats trace records for all devices.

IOX(device-number-list)
Requests formatting of I/O Summary trace records. device-number-list can contain from 1 to 50 three-digit device numbers, for which you want
records formatted. IPCS formats trace records only for the specified devices. If you do not specify any device numbers, IPCS formats trace records for all devices.

PI[(codelist)]
Specifies formatting of program interruption trace records, for the interruption codes in codelist. codelist can contain 0 to 255 decimal interruption codes of one to three digits each. If you do not specify any codes, IPCS formats trace records for all the program interruption codes found in the dump.

RNIO
Requests formatting of all the records for VTAM remote network activities.

RR  Requests formatting of all recovery routine event records.

SLIP
Requests formatting of all SLIP trace records.

SRM
Requests formatting of system resources manager (SRM) event records.

SVC[(svclist)]
Requests display of the formatted trace records associated with the numbers specified in svclist.

svclist can contain 0 to 255 decimal SVC numbers of 1 to 3 digits each.

SYS
Requests formatting of all system event trace records. SYS, the default, formats all the GTF trace records that were recorded in a dump or trace data set except for USR records.

USR ((symbol-list | idvalue-list | idrange-list | ALL))
Requests formatting of user/subsystem trace records created by the GTRACE macro.

symbol-list or idvalue-list denote trace records belonging to one component or subsystem. GTRACE data consists of user event trace records or IBM subsystem event records from these subsystems:

- OPEN/CLOSE/EOV
- SAM/PAM/DAM
- VTAMVSAM

symbol-list contains 1 through 20 symbols, with multiple symbols separated by commas. When id values are assigned to a subsystem, the component defines the symbol that is used. The following table shows valid symbols and their corresponding ids and subsystems:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>ID</th>
<th>Subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM01</td>
<td>FF5</td>
<td>VSAM</td>
</tr>
<tr>
<td>APTH</td>
<td>FE2</td>
<td>TSO/VTAM TGET/TPUT trace</td>
</tr>
<tr>
<td>APTR</td>
<td>FE3</td>
<td>VTAM reserved</td>
</tr>
<tr>
<td>CL01</td>
<td>FF1</td>
<td>VTAM buffer contents trace (USER)</td>
</tr>
<tr>
<td>CL02</td>
<td>FF0</td>
<td>VTAM SMS (buffer use) trace</td>
</tr>
<tr>
<td>DB2V</td>
<td>F5F</td>
<td>DB2/VSAM transparency</td>
</tr>
<tr>
<td>DMA1</td>
<td>FFF</td>
<td>OPEN/CLOSE/EOV</td>
</tr>
<tr>
<td>FSI4</td>
<td>F54</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI5</td>
<td>F55</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI6</td>
<td>F56</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI7</td>
<td>F57</td>
<td>FSI trace</td>
</tr>
</tbody>
</table>
**GTFTTRACE subcommand**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>ID</th>
<th>Subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSI8</td>
<td>F58</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI9</td>
<td>F59</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI A</td>
<td>F5A</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI B</td>
<td>F5B</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI C</td>
<td>F5C</td>
<td>FSI trace</td>
</tr>
<tr>
<td>FSI D</td>
<td>F5D</td>
<td>FSI trace</td>
</tr>
<tr>
<td>INT1</td>
<td>FE1</td>
<td>VTAM internal table</td>
</tr>
<tr>
<td>OSIC</td>
<td>F53</td>
<td>OSI Communication Subsystem</td>
</tr>
<tr>
<td>SPD1</td>
<td>FF3</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD2</td>
<td>FF4</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD3</td>
<td>FF6</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD4</td>
<td>FF7</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD5</td>
<td>FF8</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD6</td>
<td>FF9</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD7</td>
<td>FFA</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD8</td>
<td>FFB</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPD9</td>
<td>FFC</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPDA</td>
<td>FFD</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>SPDB</td>
<td>FFE</td>
<td>SAM/PAM/DAM</td>
</tr>
<tr>
<td>TPIO</td>
<td>FEF</td>
<td>VTAM buffer contents trace</td>
</tr>
</tbody>
</table>

**idvalue-list** contains 1 through 20 values, which are 3-digit hexadecimal identifiers assigned to a subsystem. If more than one value is specified, separate them with commas. The following table shows valid identifiers and their corresponding subsystems:

<table>
<thead>
<tr>
<th>ID</th>
<th>Issued by</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-3FF</td>
<td>GTF user program</td>
</tr>
<tr>
<td>400-5F0</td>
<td>Reserved for IBM Use</td>
</tr>
<tr>
<td>5F1</td>
<td>PVM</td>
</tr>
<tr>
<td>5F2-5F3</td>
<td>Reserved for IBM Use</td>
</tr>
<tr>
<td>5F4-5F5</td>
<td>NetView® System Monitor</td>
</tr>
<tr>
<td>5F6-F47</td>
<td>Reserved for IBM Use</td>
</tr>
<tr>
<td>F48</td>
<td>IOS</td>
</tr>
<tr>
<td>F49</td>
<td>BDT</td>
</tr>
<tr>
<td>F4F</td>
<td>OSAM</td>
</tr>
<tr>
<td>F50-F52</td>
<td>Reserved for IBM Use</td>
</tr>
<tr>
<td>F53</td>
<td>OSI Communications Subsystem</td>
</tr>
<tr>
<td>F54-F5D</td>
<td>FSI</td>
</tr>
<tr>
<td>F5E</td>
<td>Reserved for IBM Use</td>
</tr>
<tr>
<td>F5F</td>
<td>DB2®</td>
</tr>
<tr>
<td>F60</td>
<td>JES3</td>
</tr>
<tr>
<td>F61</td>
<td>VSAM Buffer Manager</td>
</tr>
<tr>
<td>F62</td>
<td>Dynamic output SVC installation exit</td>
</tr>
<tr>
<td>F63</td>
<td>Converter/Interpreter installation exit</td>
</tr>
<tr>
<td>F64</td>
<td>APPC/VM VTAM Support (AVS)</td>
</tr>
<tr>
<td>F66-F6A</td>
<td>VTAM</td>
</tr>
<tr>
<td>F6C</td>
<td>CICS</td>
</tr>
<tr>
<td>FAA</td>
<td>VTAM VM/SNA Console Services (VSCS)</td>
</tr>
<tr>
<td>FAB-FAE</td>
<td>VTAM buffer contents trace</td>
</tr>
</tbody>
</table>

**idrange-list** contains 1 through 20 id value ranges, which are the first and last 3-digit values of the id range, separated by a hyphen. If more than one range is specified, separate them with a comma.
GTFTRACE subcommand

ALL requests formatting of all user and subsystem trace records. ALL overrides any idvalue, idrange, or symbol specification.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the GTFTRACE subcommand.

Example
For examples of GTFTRACE output, see the GTF trace in z/OS MVS Diagnosis: Tools and Service Aids.

HELP subcommand — get information about subcommands

Use the HELP subcommand to obtain information about the function, syntax, and parameters of the IPCS subcommands. If you enter HELP with no parameters, all the IPCS subcommands are listed.

Note: In the IPCS dialog, use only the abbreviated form, H, of this subcommand. See the z/OS MVS IPCS User's Guide for more information.

Syntax

{ HELP|H } [subcommand [ALL | FUNCTION | SYNTAX| OPERANDS[(list)]] ]

Parameters

subcommand
Specifies the name of the IPCS subcommand about which you want information. If you omit this parameter, the subcommand displays information about all IPCS subcommands.

ALL
Specifies that you want all the information available about the specified subcommand.

If you omit the FUNCTION, SYNTAX, and OPERANDS parameters, ALL provides information about all IPCS subcommands.

FUNCTION
Specifies that you want to know more about the purpose and operation of the specified subcommand.

SYNTAX
Specifies that you want to know more about the syntax of the specified subcommand.

OPERANDS[(list)]
Specifies that you want to know more about the parameters of the specified subcommand. If you specify a list of parameters, HELP displays information about those parameters. If you specify no parameters, HELP displays information about all the parameters of the specified subcommand.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the HELP subcommand.

INTEGER subcommand — format or list a number

Use the INTEGER subcommand to:

• Convert a number from decimal to hexadecimal representation or vice versa.
INTRODUCTION

- Format a value having a specified length for CLIST, REXX, or ISPF dialog usage. The formatted values may be used to compose tabular reports or to construct symbols such as those generated by the RUNCHAIN subcommand.

Syntax

```
INTEGER  integer
          [ CLIST (STORAGE(storage)) ]
          [ DIALOG (STORAGE(storage)) ]
          [ REXX (STORAGE(storage)) ]
          [ LIST ]
          [ CHARACTER ]
          [ OFFSET [(precision)] ]
          [ POINTER [(precision)] ]
          [ SIGNED [(precision)] ]
          [ UNSIGNED [(precision)] ]
```

--- SETDEF-Defined Parameters ---

Note: You can override the following SETDEF parameters. See “SETDEF subcommand — set defaults” on page 5-232

```
[ LENGTH(length) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
```

Parameters

integer

Specifies the integer to be converted. The integer must be signed and between \(-2^{31}\) and \(2^{31}-1\). The notation of the integer can be:

- Decimal: [+|-]nnn
- Hexadecimal: X'[+|-]xxx'
- Binary: B'[+|-]bbb'

CLIST(STORAGE(storage))
DIALOG(STORAGE(storage))
REXX(STORAGE(storage))

Specifies where IPCS is to store the value of the converted integer. CLIST directs that the value be stored in CLIST variable storage. DIALOG directs that the value be stored in ISPF function pool dialog variable storage. REXX directs that the value be stored in REXX variable storage.

LIST

Specifies that the value is to be displayed. If CLIST, DIALOG, or REXX is omitted, the default is LIST.

CHARACTER

OFFSET [(precision)]
POINTER [(precision)]
SIGNED [(precision)]
UNSIGNED [(precision)]

Specifies the notation into which the integer is to be converted.
**INTEGER subcommand**

CHARACTER specifies that the 4 bytes of a signed binary fullword containing a number integer are to be formatted as 4 EBCDIC characters. Characters present on neither the 1403 TN print chain nor the 3211 T11 print train are to be translated to EBCDIC periods.

OFFSET specifies that the number integer is to be formatted using a leading plus or minus sign plus hexadecimal digits.

POINTER specifies that the 4 bytes of a signed binary fullword containing a number integer are to be formatted as an unsigned binary fullword using hexadecimal digits.

SIGNED specifies that the number integer is to be formatted using a leading blank or minus sign plus decimal digits.

UNSIGNED specifies that the 4 bytes of a signed binary fullword containing a number integer are to be formatted as an unsigned binary fullword using decimal digits.

**precision** is the number of digits in the formatted result. If no precision is specified, all leading zero digits are removed from the result.

**LENGTH**(length)

Specifies the number of characters for the formatted result. Leading blanks are supplied to attain the specified length. If length is not specified, no leading blanks are supplied.

**Return codes**

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the INTEGER subcommand.

---

**IOSCHECK subcommand — format I/O supervisor data**

Use the IOSCHECK subcommand to format the contents of specific I/O supervisor (IOS) control blocks and related diagnostic information.

You request diagnostic information about a captured unit control block (UCB) with the CAPTURE parameter on IOSCHECK. IOSCHECK produces different diagnostic reports for captured UCBs with the address space selection parameter(s) (ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, and JOBLIST).

**Address Space Selection Parameters**

- ALL processes all address spaces.
- CURRENT processes active address spaces of the dump.
- ERROR processes any address space with an error indicator or containing a task with an error indicator.
- TCBERROR processes any address space containing a task with an error indicator.
- ASIDLIST processes address spaces associated with ASID(s).
- JOBLIST or JOBNAME processes address spaces associated with job names.

If you do not specify an address space selection parameter, CURRENT is the default. Address space selection parameters only apply with the CAPTURE parameter.

**Syntax**
IOSCHECK subcommand

{ IOSCHECK } [ ACTUCBS ]
{ IOSK   }

[ ALLUCBS ]
[ CAPTURE ]
[ CHAR(device-number-list) ]
[ CHPR ]
[ COMM(device-number-list) ]
[ CTC(device-number-list) ]
[ DASD(device-number-list) ]
[ DISP(device-number-list) ]
[ EXCEPTION ]
[ HOTIO ]
[ MIH ]
[ RECOVERY ]
[ SMGRBLKS ]
[ TAPE(device-number-list) ]
[ UCB(device-number-list) ]
[ UREC(device-number-list) ]
[ VALIDATE ]

-------- Address Space Selection Parameters ----------
[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR ]
[ ASIDLIST(asidlist) ]
[ JOBLIST(joblist)|JOBNAME(joblist) ]
--- SETDEF-Defined Parameters ---

Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232.

[ACTIVE | MAIN | STORAGE ]
[DSNAME(dsname) | DATASET(dsname) ]
[FILE(ddname) | DDNAME(ddname) ]
[PATH(path-name) ]

[PRINT | NOPRINT]

[Terminal | NOTERMINAL]

[TEST | NOTEST]

Parameters

In the parameters, device-number-list is one of the following:

- A single hexadecimal device number of up to four digits.
  - Parentheses are accepted but are not required.
  - Leading zero digits are accepted but are not required.
- A range of device numbers defined by the lowest and highest device
  numbers separated by a colon.
  - Parentheses are accepted but are not required.
  - Leading zeros are accepted but are not required.
  - The second device number must be equal to or greater than the first, for
    example, 193:198.
- A list containing either single device numbers or ranges of device numbers.
  Parentheses are required. In the list, separate list members with blanks,
  commas, or horizontal tabulation (X'05') characters. The separators are
  permitted, but not required, between the left parenthesis and the first
  member and between the last member and the right parenthesis.

Report Type Parameters

Use these parameters to select the type of report.

ACTVUCBS

Validates I/O control blocks, formats active UCBs and these associated
control blocks:

- IOQ
- IOSB
- SRB
- EWA
- CRWQ
- SRWQ

ALLUCBS

Validates the I/O control blocks and formats all UCBs, along with these
associated control blocks:

- IOQ
- IOSB
- SRB
- EWA
- CRWQ
- SRWQ

CAPTURE

Formats the captured UCB pages in an address space (based on the address
space selection parameters) along with these associated control blocks:

- IOQ
An application program can access an above 16 megabyte UCB with a 24-bit address through a view of the UCB captured in the program's address space.

The report also displays the captured UCB pages in common storage, if any exist. The report gives you the address space identifier (ASID) and information about each captured page. The report provides the following information for each captured page:
- Actual page address
- Captured page address
- Captured UCB count

The captured UCB count is the number of captures of UCBs, these can be captures of the same UCB.

**CHAR(device-number-list)**
Requests formatting of selected channel-to-channel attention routine (CHAR) UCBs.

**CHPR**
Requests formatting of the installation channel path table (ICHPT), the channel recovery block (CHRB), and the global channel report word queue (CRWQ) elements.

**COMM(device-number-list)**
Requests formatting of selected communication (COMM) UCBs.

**CTC(device-number-list)**
Requests formatting of selected channel-to-channel (CTC) UCBs.

**DASD(device-number-list)**
Requests formatting of selected direct access storage device (DASD) UCBs.

**DISP(device-number-list)**
Requests formatting of any dispatcher (DISP) UCBs that you have selected (using device-number-list).

**EXCEPTION**
Specifies that IPCS check the validity of the IOS control blocks and print diagnostic error messages for blocks that are not valid. This parameter formats these control blocks:
- I/O communications area (IOCOM)
- I/O communications writeable area (IOCW)
- IOS level definitions
- I/O work area (IOWA) for each processor, and the IOS module work areas for each IOWA
- I/O prevention table (IOPT), if accessible

EXCEPTION is the default.

For additional information about IOS level definitions see [IOSCHECK ACTVUCBS subcommand Output](z/OS MVS Diagnosis: Reference).
**HOTIO**
Requests formatting of the hot I/O detection table (HIDT) and the associated status collector data areas (SCDs).

**MIH**
Requests formatting of the missing interrupt handler work area (MIHA) and the associated time interval control blocks (TICBs).

**RECOVERY**
Requests formatting of the control blocks for the HOTIO, MIH, and CHPR parameters.

**SMGRBLKS**
Requests formatting of entries in the IOS storage manager page table for IOQ, RQE, and large blocks, and formatting of the queue of pages for each entry. The string LGA will appear in the formatted output instead of LGE to distinguish between a below the line large block and above the line large block.

SMGR: 0188CB70
+0000 BLKID.... LGEB   PGID..... LGAPool IOS SMGR
+0014 PGESEZ.... 00001000 PGFLG1.... 00   PGFLG2.... 00
+001A PGPPOOL.... 00E2   SYMCA.... 0188C1D0 BLKCNT.... 000F
+0022 TBLKLN.... 0100   BLKLEN.... 00F8   POSET.... 0100
+0028 PTOLE.... 00E2   LINKOF.... 00F4   HDROF.... 00F8
+002E BIDOF.... 00F0   PGEINC.... 0010000F ALLOCW.... 00000076
...

PAGE: 02FCC000
+0000 RCNT..... 0000000F BLKP..... 00FCC100 FLG1..... 80
+0009 FLG2..... 00   MCNT..... 000F   CHN..... 02FCB000
+0010 ID....... LGAPool IOS SMGR WKR..... 00000000
+0024 EXTP..... BACK..... 00000000

LGAB at 01FCC100
+0000 00000000 C5E77D97 00FCBF68 00000000 | ....EXPR........
+0010 00000000 00000000 00000000 00000000 | ................
+0020 00000000 00000000 00000000 00000000 | ................
...

**TAPE(device-number-list)**
Requests formatting of selected TAPE UCBs and ranges.

**UCB(device-number-list)**
Requests formatting of selected unit control blocks (UCBs).

**UREC(device-number-list)**
Requests formatting of selected unit record (UREC) UCBs.

**VALIDATE**
Requests validity checking of the following IOS control blocks:
- Device class queue chain (DCQ)
- Unit control blocks (UCB) queued off the DCQ
- I/O request blocks (IOQ) chained off the UCB and the associated IOQ chain
- I/O supervisor block (IOSB) pointed to from each IOQ
- Service request block (SRB) pointed to from each IOSB
- IOS error recovery procedure (ERP) work area (EWA) pointed to from the IOSB

When IOS detects a control block that is not valid, IOS formats the control block, and prints a diagnostic error message.
IOSCHECK subcommand

Note: For SVC dumps, not all the data pertaining to IOSCHECK is saved at the time of error. As a consequence, many control blocks may be reused before the data is dumped. Informational messages indicate that the data is not from the time of error. For example, the following message indicates that the IOQ has been reused:

IOS10107I IOQ AT xxxxxxxx does not point to UCB at yyyyyyy

Address Space Selection Parameters
Use these parameters to obtain captured page data from particular address spaces, which you specify by their ASIDs. These parameters only apply with the CAPTURE parameter. If you specify CAPTURE but omit these parameters, the default is CURRENT. For more information, see the select ASID service in z/OS MVS IPCS Customization.

ALL
Specifies processing of captured pages for all address spaces in the system at the time the dump is generated.

CURRENT
Specifies processing of captured pages for each address space that is active (for example, dispatched on some central processor) when the dump is generated.

ERROR
Specifies processing of captured pages for any address space with an MVS error indicator or containing a task with an error indicator.

TCBERROR
Specifies processing of captured pages for any address space containing a task with an error indicator. Blocks for address spaces with an error indicator are not processed.

ASIDLIST(asidlist)
Specifies a list of ASIDs for the address spaces to be in the report.

The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed using the notation X'nnn', F'nnn', or B'nnn'. An unqualified number is assumed to be fixed.

This subcommand does not process summary dump records (ASID X'FFFA').

JOBLIST(joblist) or JOBNAME(joblist)
Specifies a list of job names whose associated address spaces are to be in the report. Use commas to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

Return codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the IOSCHECK subcommand.

Example 1
Display IOS-related control blocks.

– Action
COMMAND ===> IOSCHECK UCB(2D0,2E0,410:440,620)
– Result
**IOSCHECK subcommand**

This example formats UCBs for 3 device numbers and one range.

For an example of IOSCHECK output, see the IOS component in [z/OS MVS Diagnosis: Reference](#).

Example 2
Display captured UCB information for address spaces that are active.

- Action
  
  COMMAND ===> IOSCHECK CAPTURE

- Result
  
  This example formats the captured UCB information for any address space that is active. The output looks similar to the following for each address space:

  * * * ADDRESS SPACE CAPTURE DATA * * *

  ASID 000F

<table>
<thead>
<tr>
<th>ACTUAL PAGE ADDRESS</th>
<th>CAPTURE PAGE ADDRESS</th>
<th>CAPTURE UCB COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01D0E000</td>
<td>006F8000</td>
<td>00000005</td>
</tr>
<tr>
<td>01D0F000</td>
<td>006F7000</td>
<td>00000003</td>
</tr>
</tbody>
</table>

  Two pages were captured in address space 000F. The first page had five captures of UCBs and the second had three.

**IPCSDATA subcommand — request a report about IPCS activity**

Use the IPCSDATA subcommand to generate reports about data maintained by IPCS in a dump:

- IPCS sessions may have been active in various ASIDs dumped. If not and IPCSDATA is asked to process an ASID, a very brief report will be generated saying:

  No IPCS session data was found in ASID(X'xxxx')

  If you do not specify an address space selection parameter, CURRENT is the default.

- Most dumps include the ECSA storage in which BLSJPRMI stores tables that identify the sysplex dump directory name and enumerate materials available for use during SNAP/ABDUMP formatting. Ask IPCSDATA to process COMMON storage to format this data.

  Address space selection and data selection parameters limit the scope and extent of the information that appears in the report.

- **Syntax**
IPCSDATA subcommand

--- Data Selection Parameters ------------------------------
[ COMMON | NOCOMMON ]

[ PRIVATE | NOPRIVATE ]

[ PARMLIB | NOPARMLIB ]

[ OPEN | NOOPEN ]

[ TASK | NOTASK ]

--- Address Space Selection Parameters ---------------------
[ ALL ]

[ ASIDLIST(asidlist) ]

[ CURRENT ]

[ ERROR ]

[ JOBLIST(joblist) | JOBNAME(joblist) ]

[ TCBERROR ]

--- SETDEF-Defined Parameters -----------------------------
Note: You can override the following SETDEF parameters.

See “SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]

[ DSNAMEx(dsname) | DATASET(dsname) ]

[ FILE(ddname) | DDNAME(ddname) ]

[ PATH(path-name) ]

[ FLAG(severity) ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ TEST | NOTEST ]

- Data selection parameters

COMMON or NOCOMMON
Requests or suppresses a report pertaining to common storage data
maintained to support SNAP and ABDUMP formatting in the dumped
system.

PRIVATE or NOPRIVATE
Requests or suppresses reports pertaining to IPCS sessions, if any, in the
address spaces selected.

PARMLIB or NOPARMLIB
Requests or suppresses reports showing information obtained from parmlib
members.

OPEN or NOOPEN
Requests or suppresses reports pertaining to open data sets.

Note: Dump directory performance statistics are only produced by
IPCSDATA when it is run against ACTIVE storage.1
**TASK or NOTASK**  
Requests or suppresses reports pertaining to tasks associated with IPCS session activity.

- **Address Space Selection Parameters**  
  Request address spaces for which IPCSDATA private storage reports should be produced. See “SELECT subcommand — generate address space storage map entries” on page 5-229.

- **SETDEF-Defined Parameters**  
  Overrides defaults established through the SETDEF subcommand or the Defaults option of the IPCS dialog. See “SETDEF subcommand — set defaults” on page 5-232.

- **Diagnosis — Sample IPCSDATA Reports**
  - **Example 1. Sample IPCSDATA Common Storage Report**  
    The following sample includes parmlib information. Use of the NOPARMLIB option eliminates all lines of the report following the one beginning “BLSQXBT”.

---

1. Statistics are acquired through the VSAM SHOWCB ACB programming interface, and no equivalent interface is supported for ACB images retrieved from a dump.
IPCSDATA subcommand

Common storage report

BLSQXBT at OD35CC0 LENGTH(4927)

SYSDDIR 'MVSSPT.SYSPLEX.DMPDIR'

DATA STRUCTURE(ALE) MODEL(IEAALEP)
DATA STRUCTURE(ASC) FIND(BLSSASC) MODEL(IEAAASCBP) SCAN(BLSVASC)
DATA STRUCTURE(ASS) MODEL(IEAASSBP) SCAN(BLSVASS)
DATA STRUCTURE(AST) FIND(BLSSAST) GROUP(ASTE) MODEL(IEAASTEPE) SCAN(+ BLSVASTE)
DATA STRUCTURE(ASTE) FIND(BLSSASTE) MODEL(IEAASTEPE) SCAN(BLSVASTE)
DATA STRUCTURE(ASX) FIND(BLSSASX) MODEL(IEAASXB) SCAN(BLSVASX)
DATA STRUCTURE(CDE) FIND(BLSSCDE) MODEL(CSVFMCDE) SCAN(BLSVCDE)
DATA STRUCTURE(CDEMAJOR) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(CDEMINOR) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(CVT) FIND(BLSSCVT) MODEL(IEACVTP) SCAN(BLSVCVT)
DATA STRUCTURE(IRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)
DATA STRUCTURE(JSAB) FIND(IAZJSABF) MODEL(IAZJSABM) SCAN(IAZJSABV)
DATA STRUCTURE(LLE) MODEL(CSVFMLLE)
DATA STRUCTURE(LPDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(LPDEFINAL) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(LPDEMAJOR) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(LPDEMINOR) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(LPDENULL) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFMCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(LS) FORMAT(IEAVD3A)
DATA STRUCTURE(LSE) MODEL(IEALSEP)
DATA STRUCTURE(LSEH) MODEL(IEALSEHP)
DATA STRUCTURE(LSET) MODEL(IEALSETP)
DATA STRUCTURE(LSSA) MODEL(IEALSSAP)
DATA STRUCTURE(LSSD) MODEL(IEALSSGP) SCAN(IEACLSSD)
DATA STRUCTURE(LSSG) MODEL(IEALSSGP) SCAN(IEACLSSG)
DATA STRUCTURE(PRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)
DATA STRUCTURE(PSW) FIND(BLSSPSW) FORMAT(BLSQPSWF,JBB2125)
DATA STRUCTURE(REGACC) FIND(BLSSREGA) MODEL(BLSBREGA)
DATA STRUCTURE(REGCTL) FIND(BLSSREGC) MODEL(BLSBREGC)
DATA STRUCTURE(REGFLT) FIND(BLSSREFG) MODEL(BLSBREFG)
DATA STRUCTURE(REGGEN) FIND(BLSSREGG) MODEL(BLSBREGG)
DATA STRUCTURE(REGS) MODEL(BLSBREGS)
DATA STRUCTURE(REGSAVIM) MODEL(BLSBREGI)
DATA STRUCTURE(RTM2WA) MODEL(IEAVTRP2) SCAN(IEAVTRV2)
Example 2. Sample IPCSDATA Private Storage Report

The following sample includes parmlib, open data set and task information.

- Use of the NOPARMLIB option eliminates all lines of the report starting with the line beginning “SYSDDIR” and ending with the line beginning “SYMBOL PREFIX(Z)”.  
- Use of the NOOPEN option eliminates the paragraphs starting with lines beginning “Dump directory” and “BLSRZZ6 at”.  
- The lines in the “Dump directory” paragraph starting with the line beginning “NLOGR” only appear when IPCSDATA is run against ACTIVE storage. Most of these statistics are also maintained by VSAM in the catalog and can be formatted by LISTCAT. SHRPOOL, BFRFND and BUFDRS are accumulated within a single session and can only be obtained through IPCSDATA against ACTIVE storage.  
- Use of the NOTASK option eliminates the report lines starting with the line beginning “Master BLSUZZ2”.  

```
DATA STRUCTURE(SCB) FORMAT(IEAVTRF4,JBB2125) SCAN(IEAVTRVS)
DATA STRUCTURE(SDWA) MODEL(IEAMSDWA)
DATA STRUCTURE(SIRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)
DATA STRUCTURE(SSAT) MODEL(IEASSATP)
DATA STRUCTURE(STCB) MODEL(IEASTCBP) SCAN(BLSVSTCB)
DATA STRUCTURE(SVVRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)
DATA STRUCTURE(TCB) FIND(BLSSTCB) MODEL(IEATCBP) SCAN(BLSVTCB)
DATA STRUCTURE(TIOT) FORMAT(BLSQTIOT)
DATA STRUCTURE(TIRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)
DATA STRUCTURE(UCB) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) SCAN(IOSVUCBV)
DATA STRUCTURE(UCBCTC) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(+UCB) SCAN(IOSVUCBV)
DATA STRUCTURE(UCBDCA) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(UCB) SCAN(IOSVUCBV)
DATA STRUCTURE(UCBGFX) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(+UCB) SCAN(IOSVUCBV)
DATA STRUCTURE(UCBTAPE) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(UCB+) SCAN(IOSVUCBV)
DATA STRUCTURE(UCBTSP) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(UCB) SCAN(IOSVUCBV)
DATA STRUCTURE(UCBROW) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(UCB) SCAN(IOSVUCBV)
DATA STRUCTURE(UCB3270) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(UCB+) SCAN(IOSVUCBV)
DATA STRUCTURE(VF) FORMAT(IEAVSSA2)
DATA STRUCTURE(XSB) MODEL(IEAXSBP)
DATA STRUCTURE(XTLS) FIND(BLSSXTLS) MODEL(CSVFMXTL) SCAN(BLSVXTLS)
```
IPCSDATA subcommand

ASID('X'0305'), JOBNAME(RLW)

BLSUZZI at 00038800

Dump directory BLSUZZ4 at 00050E00
  FILE(IPCSDIR) DSNAMES('RLW.DDIR')
  NLOGR(6135) NRETR(52452) NINSR(13209) NUPDR(253) NDELR(19792)
  CINV(22528) NCIS(208) NSSS(6) SHRPPOOL(15)
  BFRFDN(39103) BUFROS(7) NEXCP(4744)

BLSQXBT at 0DAE20C0 LENGTH(61245)

SYSDDIR 'MVSSPT.SYSPLEX.DMPDIR'

DATA STRUCTURE($CADDR) MODEL(HASMCADR)
DATA STRUCTURE($CKB) MODEL(HASMCKB)
DATA STRUCTURE($CKG) MODEL(HASMCKG)

DATA STRUCTURE(ACE) MODEL(ILRMACE)
DATA STRUCTURE(AFT) FIND(BLSSAFT) GROUP(AFTE) SCAN(BLSVAFT)
DATA STRUCTURE(AFTE) FIND(BLSSAFT) SCAN(BLSVAFT)
DATA STRUCTURE(AIA) MODEL(ILRMAIA)
DATA STRUCTURE(ALE) MODEL(ILRMALEP)
DATA STRUCTURE(BMDCMAP) MODEL(BLSBCPST)
DATA STRUCTURE(AR) FORMAT(IEAVX02)
DATA STRUCTURE(ASEI) MODEL(ILRMASEIP)
DATA STRUCTURE(ASEMHD) MODEL(ILRMAHMD)
DATA STRUCTURE(ASMV) FIND(ILRFASMV) MODEL(ILRMAsvm)
DATA STRUCTURE(ASPCA) FORMAT(IEAVSPC)
DATA STRUCTURE(ASSB) FIND(BLSSASSB) MODEL(ILRMASSBP) SCAN(BLSVASBB)

EXIT FORMAT(ASCB) EP(IEASRBQ2)
DATA STRUCTURE(ASCN) MODEL(ILRMAIN)
DATA STRUCTURE(ASVRQ) MODEL(ILRMAVRQ)
DATA STRUCTURE(AST) FIND(BLSSASTE) GROUP(ASTE) MODEL(ILRMAstep) SCAN(+
  BLSVaste)
DATA STRUCTURE(ASTE) FIND(BLSSASTE) MODEL(ILRMAstep) SCAN(BLSVaste)
DATA STRUCTURE(ASYV) FIND(BLSSASYV) SCAN(BLSVASYT)
DATA STRUCTURE(ASXV) FIND(BLSSASXV) MODEL(ILRMAxv) SCAN(BLSVASXV)
DATA STRUCTURE(BLSLNTRC) SCAN(BLSVTRC)
DATA STRUCTURE(BLSQXBT) FIND(BLSSXBT) SCAN(BLSVXBT)
DATA STRUCTURE(BLSVXBT) SCAN(BLSVARQ)

IPCSDATA subcommand

DATA STRUCTURE(CACHE) MODEL(ILRMCCACH)
DATA STRUCTURE(CDE) FIND(BLSSCDE) MODEL(CSVFCDE) SCAN(BLSVCDE)
DATA STRUCTURE(CDEMAJOR) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(CDEMINOR) FIND(BLSSCDE) GROUP(CDE) MODEL(CSVFCDE) SCAN(+ BLSVCDE)
DATA STRUCTURE(CIBAL) FIND(IATIFBAL) MODEL(IATIPBAL)
.
.
DATA STRUCTURE(STORESTATUS)

    EXIT CBSTAT(STORESTATUS) EP(IEAVNPW)
    EXIT CBSTAT(STORESTATUS) EP(IXCFMCBS)

DATA STRUCTURE(SUPVT) MODEL(IEASVTP)
DATA STRUCTURE(SVRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)
DATA STRUCTURE(SVT) MODEL(IEASVTP)
DATA STRUCTURE(SVTX) MODEL(IEASVTXP) SCAN(IEACSVTX)
DATA STRUCTURE(TCB) FIND(BLSSTCB) MODEL(IEATCBP) SCAN(BLSVTCB)

    EXIT CBSTAT(TCB) EP(IEAVFGL)
    EXIT CBSTAT(TCB) EP(IEAVTC)
    EXIT CBSTAT(TCB) EP(IEAVG701)
    EXIT CBSTAT(TCB) EP(IEAVXO)

    EXIT FORMAT(TCB) EP(IECDAFMT)
    EXIT FORMAT(TCB) EP(IECIOFMT)
    EXIT FORMAT(TCB) EP(IEAVTFMT)
    EXIT FORMAT(TCB) EP(IEAVD30)
    EXIT FORMAT(TCB) EP(IEAVDC)

DATA STRUCTURE(DCM) MODEL(IEEMB904)
.
.
DATA STRUCTURE(XTLST) FIND(BLSSXTLS) MODEL(CSVFMXTL) SCAN(BLSVXTLS)
DATA AREA(COMMON) FIND(BLSSCOMM)
DATA AREA(CSA) FIND(BLSSCSA)
DATA AREA(DATOFFNUCLEUS) FIND(BLSSDONU)
DATA AREA(ECSA) FIND(BLSECSA)
DATA AREA(FLPA) FIND(BLSSEFLP)
DATA AREA(EMLPA) FIND(BLSSEMPL)
DATA AREA(ENUCLEUS) FIND(BLSSENUC)
DATA AREA(EPLPA) FIND(BLSSEPLP)
DATA AREA(EQA) FIND(BLSESQA)
DATA AREA(FLPA) FIND(BLSSFPLA)
**IPCSDATA subcommand**

```
DATA AREA(MLPA) FIND(BLSSMLPA)
DATA AREA(NUCLEUS) FIND(BLSSNUC)
DATA AREA(PPLPA) FIND(BLSSLPLPA)
DATA AREA(PRIVATE) FIND(BLSSPRIV)
DATA AREA(PRIVATEX) FIND(BLSSPRIX)
DATA AREA(ROLNUCLEUS) FIND(BLSSLNRO)
DATA AREA(QQA) FIND(BLSSQA)
EXIT ANALYZE EP(IARZANAL)
EXIT ANALYZE EP(IEAVESLX)
EXIT ANALYZE EP(IEFAB4W)
EXIT ANALYZE EP(IESVFMT)
EXIT ANALYZE EP(ISGDCONT)
EXIT ANALYZE EP(IXCFMLAN)
EXIT VERB(ALCWAUT) EP(IEFAB4W) HELP(IEFAB4WP) ABSTRACT('Allocation wait + summary')
EXIT VERB(AOMDATA) EP(AOIMIPCS) ABSTRACT('AOM analysis')
DIALOG NAME(APPCDATA) HELP(ATBH999) ABSTRACT('APPC/MVS Data Analysis') + PARM('PANEL(ATBH000)')
DIALOG NAME(ASCHDATA) HELP(ASBH999) ABSTRACT('APPC/MVS Scheduler Data + Analysis') PARM('PANEL(ASBH000)')
DIALOG NAME(ASMHCHECK) HELP(ILRASCMCH) ABSTRACT('Auxiliary storage paging + activity') PARM('PGM(BLSSGMCMD) PARM(ASMHCHECK TERMINAL NOPRINT)')
EXIT VERB(ASMDATA) EP(ILRFTMAN) HELP(ILRASMDH) ABSTRACT('ASM control + block analysis')
EXIT VERB(AVMDATA) EP(AVFRDFMT) HELP(AVHELP) ABSTRACT('AVM control + block analysis')
EXIT VERB(CICSDATA) EP(DFHPDX) ABSTRACT('CICS analysis')
EXIT VERB(CICST212) EP(DFHPD212) ABSTRACT('CICS Version 2 Release 1.2 + analysis')
EXIT VERB(CICST321) EP(DFHPD321) ABSTRACT('CICS Version 3 Release 2.1 + analysis')
EXIT VERB(CICST330) EP(DFHPD330) ABSTRACT('CICS Version 3 Release 3 + analysis')
EXIT VERB(CICST410) EP(DFHPD410) ABSTRACT('CICS Version 4 Release 1 + analysis')
.
.
EXIT VERB(VTAMMAP) EP(ISTRAFD1) ABSTRACT('VTAM control block analysis')
DIALOG NAME(XESDATA) HELP(IIXLHDIA) ABSTRACT('XES analysis') PARM('PANEL(+IXLFPMMN)')
```
IPLDATA subcommand — request IPL reports

Use the IPLDATA subcommand to request reports about the IPL process and options.

Syntax

IPLDATA

------- Report Selection Parameters ---------------------------------
 [ INFORMATION | STATUS ]
IPLDATA subcommand

------ SETDEF-Defined Parameters ------------------------

Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAM(dsnname) | DATASET(dsnname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

INFORMATION
Selects the INFORMATION report, the default. This report has nearly the same format as the output of the DISPLAY IPLINFO system command.

STATUS
Selects the STATISTICS report. This is the same report produced by verb exit BLSAIPST. The report contains status data collected during IPL, NIP, and Master Scheduler Initialization (MSI) during system initialization.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the IPLDATA subcommand.

Example
Select the INFORMATION report.
- Action
  COMMAND ===> IPLDATA INFORMATION
- Result

System IPLed at 10:38:23.552 on 10/03/2011
Release z/OS 02.01.00
Used LOADTH in SYS0.IPLPARM on 0343
IEASYM LIST=(TH,L)
IEASY LIST=TE (OP)
IODF device 0343
IPL device: original 0980 current 0810 volume B00810

ISPEXEC subcommand — request an ISPF dialog service

Use the ISPEXEC subcommand to request services supplied by the Program Development Facility (PDF) Program Product and the ISPF Dialog Manager Program Product. The function of the IPCS ISPEXEC subcommand is the same as the ISPF ISPEXEC command.

Before requesting PDF services, make sure your installation has installed PDF.

ISPEXEC can be entered only in the IPCS dialog. If you enter the ISPEXEC subcommand outside the IPCS dialog, ISPEXEC abnormally ends with a return code of 16.

Syntax
ISPEXEC subcommand

The syntax of the IPCS ISPEXEC subcommand is the same as the syntax of the
ISPF ISPEXEC command. The ISPEXEC command is documented in z/OS ISPF
Reference Summary.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the ISPEXEC subcommand.

LIST subcommand — display storage

Use the LIST subcommand to display storage from the current dump. You can
display storage from one or several dump locations. Specify the amount of storage
and its format with the appropriate data description parameters.

Note: This subcommand might modify X, the current address.

Related subcommands
  EQUATE
  FIND
  FINDMOD
  FINDUCB
  LISTMAP
  LISTSYM
  STATUS

Syntax

{ LIST } { data-descr } }
{ L } { (data-descr...) }

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters. See
“SETDEF subcommand — set defaults” on page 5-232
[ DISPLAY[(display-options)] ]
[ NODISPLAY[(display-options)] ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

data-descr or (data-descr...)
  Specifies that either one data description or a list of data descriptions be
evertered. A list of data descriptions consists of multiple address expressions
and one group of data description parameters that apply to all addresses in
the list.

The data description parameter consists of five parts:
  – An address (required)
  – Address processing parameters (optional)
  – An attribute parameter (optional)
  – Array parameters (optional)
  – A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and
syntax of the data description parameter.
LIST subcommand

Use the following data description parameters to obtain particular information:
- TITLE to obtain the title of an SVC dump.
- COMPDATA(IEASLIP) to obtain the SLIP command parameters in EBCDIC for an SVC dump requested by a SLIP command. If the SLIP command parameters are not available, the following appears:
  SLIP TRAP TEXT NOT AVAILABLE

DISPLAY[(display-options)]
NODISPLAY[(nodisplay-options)]

Specifies if IPCS is to display or not display the storage identified in the data-descr parameter. For the LIST subcommand, the default is DISPLAY(STORAGE). See the SETDEF subcommand for other values for DISPLAY.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the LIST subcommand.

Example 1
Display the title of the dump.
- Action
  COMMAND ===> list title
- Result
  Using the special symbol TITLE, the LIST subcommand generates the following output, including the dump title, “Hang After Hotstart”. IPCS also displays the dump title during dump initialization.

Example 2
Display all PSAs when running the 3090 model 400.
- Action
  COMMAND ===> list (psa0, psa1, psa2, psa4) structure(psa)
- Result
  LIST displays the PSA for each central processor that is online.

Example 3
Display SQA storage.
- Action
  COMMAND ===> list sqa
- Result
  LIST displays SQA storage.

Example 4
Display multiple system storage areas.
- Action
  Specify the appropriate symbols with LIST, enclosing them in parentheses:
  COMMAND ===> list (sqa csa private)
- Result
  LIST displays the storage for the areas.

Example 5
LIST subcommand

Display central storage. There are several ways to do this. One way is to request a range of absolute addresses, like this:

- Action
  COMMAND ===> list 0:7fffffff absolute
- Result
  LIST displays all of ABSOLUTE storage, without performing storage prefixing.

Example 6
Another way to display central storage is to request a range of central storage for a given central processor.

- Action
  COMMAND ===> list 0:7fffffff CPU(0) real
- Result
  LIST displays the same storage as Example 5, replacing the ABSOLUTE PSA (the storage at 0:0FFF) with the PSA of central processor CPU(0). The ABSOLUTE PSA appears where the PSA for CPU(0) appeared in the Example 5.

Note: If you want to print the dump quickly, you can break your request into pieces; for example:

<table>
<thead>
<tr>
<th>To Get This Result</th>
<th>Make This Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute PSA</td>
<td>list 0:0fff absolute</td>
</tr>
<tr>
<td>Real PSAs for each central processor</td>
<td>list 0:0fff cpu(n) real</td>
</tr>
<tr>
<td>Absolute storage above the PSA</td>
<td>list 1000:fffffff absolute</td>
</tr>
</tbody>
</table>

LISTDUMP subcommand — list dumps in dump directory

Use the LISTDUMP subcommand to:
- Display the names of the sources described in a dump directory
- Produce a dumped storage summary report

A source description is for an unformatted source that IPCS can format, for example, an SVC dump, a stand-alone dump, an SYSMDUMP dump, a trace data set, a data set, or active storage. The source descriptions are in the dump directory allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

Related subcommands
COPYDUMP
DROPDUMP
EVALDUMP
SUMMARY

Syntax
LISTDUMP subcommand

{ LISTDUMP } [ SUMMARY | NOSUMMARY ]
{ LDMP } [ ]

[ SELECT [ (ATTRIBUTES ) ] ]
[ [ (BACKING ) ] ]
[ [ (DUMPED ) ] ]
[ [ (TRANSLATION ) ] ]
[ ]
[ NOSELECT ]

[ SYMPTOMS | NOSYMPTOMS ]
[ INDATASET(dsname) | INFOLE(ddname) ]

------- SETDEF-Defined Parameters ------------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

SUMMARY or NOSUMMARY

SUMMARY indicates that a processing summary (a final total line) is to be produced.

NOSUMMARY suppresses the processing summary. The NOSUMMARY parameter is useful for turning summary messages off when the subcommand is invoked within a CLIST or a REXX exec.

SELECT[(options)]

NOSELECT

Specifies whether dumped storage is to be provided.

SELECT provides dumped storage; NOSELECT provides only a list of the sources for the source descriptions and the number of storage locks and bytes for the source.

The options control the amount of information included in the summary. When specifying more than one option, separate options with a blank and enclose the list of options in parentheses. The options are:

ATTRIBUTES

Requests that the attributes of each range of storage in the report be included on the output line for that range. Where applicable, one or more of these attributes appear in the generated report:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSOLUTE</td>
<td>Represents a storage frame that was in processor storage when a stand-alone dump was requested.</td>
</tr>
<tr>
<td>COMMON</td>
<td>Represents common virtual storage.</td>
</tr>
<tr>
<td>MISSING</td>
<td>Represents storage not available in the dump.</td>
</tr>
</tbody>
</table>
LISTDUMP subcommand

PREFIXED
Represents storage to which access is affected by central storage prefixing.

RECLAIMED
Represents storage that was marked not valid in the page table but was located in a reclaimable storage frame.

SUMLIST
Represents storage recorded in response to the summary dump options (SUMLIST, SUMLISTA, and so on) of the SDUMP macro.

TRANSLATED
Represents storage located using an IPCS translation algorithm and retained in the dump directory to avoid repeated translation. These translation processes use the following mechanisms:
- Simulation of dynamic address translation when IPCS processes a stand-alone dump.
- Simulation of central storage prefixing when IPCS processes a stand-alone dump.
- Simulation of the page reclamation process performed by the RSM component.

BACKING
Specifies that the dump storage summary report indicate where the dumped information is backed in the dump records. In other words, it provides record numbers of, and offsets into, the records where the storage can be found.

For example, the following portion of a line of the report output indicates that 4096 consecutive dump records, beginning with RECORD(5), each contain 4096 bytes of consecutive storage:
RECORD(5:4100) POSITIONS(48:4143)

This option is most useful for diagnosing problems within the dump records.

Note: For data sets that are not RECFM=F or RECFM=FBA, the relative track address (TTR) will appear instead of RECORD.

DUMPED
Requests that the storage summary report include storage explicitly described by the dumping program.

TRANSLATION
Specifies that the storage summary report include translation results that IPCS retained in the dump directory. TRANSLATION suppresses the output from the DUMPED option unless both options are explicitly specified.

Note: IPCS can record storage that cannot be accessed in the dump. In the report output for requests that produce only storage ranges — such as LISTDUMP SELECT (DUMPED TRANSLATION) — the only way to distinguish accessible storage from missing storage is by checking the separators between the first and last addresses in the range. Accessible storage ranges use colons as separators:
00F0C000:00F0EFFF
while missing storage range addresses are separated by a dash:

LISTDUMP subcommand

SYMPTOMS
NOSYMPTOMS
Specifies whether LISTDUMP is to add two lines of information to that displayed for each dump selected:
- The first line shows the dump title (symbol TITLE) or indicates that none is available from the dump directory.
- The second line show symptoms in addition to the title or indicates that none are available from the dump directory. The symptoms chosen are indicated by the caption and are, in order of preference:
  - Trap — SLIP trap text (symbol SLIPTRAP)
  - Psym — Primary symptom string (symbol PRIMARYSYMPTOMS)
  - Ssym — Secondary symptom string (symbol SECONDARYSYMPTOMS)

If an output medium is selected that is too narrow to display the dump directory data available for either line, as much data is shown as will fit on one line.

The default NOSYMPTOMS keyword suppresses this output.

INDASET(dsname)
INDNAME(dsname)
Requests allocation of directory dsname and use of the contents of that directory by the subcommand.

INFILE(ddname)
INDDNAME(ddname)
Requests use of a directory that the IPCS user has allocated to ddname and use of the contents of that directory by the subcommand.

ACTIVE or MAIN or STORAGE
DATASET(dsname-list) or DSNAME(dsname-list)
FILE(ddname-range-list) or DDNAME(ddname-range-list)
Specifies the source or sources of the source descriptions to be selected from the dump directory. Use these parameters with the SELECT parameter. If these parameters are omitted, the report is for all sources in the user dump directory.

ACTIVE, MAIN, or STORAGE specifies central storage as the source.

DATASET or DSNAME specifies the names of one or more data set as the sources.

FILE or DDNAME specifies one, several, or a range of ddnames for data sets as the sources.

When specifying more than one data set name or ddname, separate the names with commas or blanks. When specifying a range of ddnames, separate the first and last ddname with a colon.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the LISTDUMP subcommand.

Example 1
List the dump sources described in the dump directory.
- Action
  COMMAND === listdump
LISTDUMP subcommand

This command has the defaults of SUMMARY, NOSELECT, and NOSYMPTOMS.

- Result

The following output is produced. Notice that the last line, which is produced by the SUMMARY parameter, provides a total number of the displayed dump data sets.

<table>
<thead>
<tr>
<th>Source of Dump</th>
<th>Blocks</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>2</td>
<td>8,208</td>
</tr>
<tr>
<td>FILE(CLIC)</td>
<td>10</td>
<td>2,640</td>
</tr>
<tr>
<td>FILE(MENUS)</td>
<td>10</td>
<td>2,640</td>
</tr>
<tr>
<td>DSNAME('D46RLW1.LOG.MISC')</td>
<td>79</td>
<td>187,809</td>
</tr>
<tr>
<td>DSNAME('D46RLW1.RLW.CLIST')</td>
<td>25</td>
<td>6,600</td>
</tr>
<tr>
<td>DSNAME('D46RLW1.SYSDUMP')</td>
<td>1,218</td>
<td>4,998,672</td>
</tr>
<tr>
<td>DSNAME('D46RLW1.XMIT.NAMES')</td>
<td>2</td>
<td>3,520</td>
</tr>
<tr>
<td>DSNAME('D46RLW1.THIS.IS.A.LONG.DSNAME.FOR.TESTING(TESTMEMB)')</td>
<td>1,040</td>
<td>31,200</td>
</tr>
<tr>
<td>DSNAME('D83DUMP.DUMPC.PB06511')</td>
<td>1,346</td>
<td>5,523,984</td>
</tr>
<tr>
<td>DSNAME('D83DUMP.DUMPC.PB07251')</td>
<td>24,142</td>
<td>99,078,768</td>
</tr>
</tbody>
</table>

10 Dumps described

Example 2

Obtain a dumped storage summary report with retained translation data, attributes, and storage described by the dumping program for a particular dump data set, MY.DUMP.

- Action

COMMAND ===> listdump select(dumped attributes translation) dsname(my.dump)

- Result

The following output is produced.
LISTDUMP subcommand

Source of Dump Blocks Bytes
DSNAME('RLW.HBB5520.SAMPLE.SVCDUMP') . . 994 . . 4,135,040

ABSOLUTE
00A95000:00A95FFF ABSOLUTE
00CD3000:00CD3FFF ABSOLUTE
010A7000:010A7FFF ABSOLUTE
012A0000:012A8FFF ABSOLUTE
01539000:01539FFF ABSOLUTE
01756000:01757FFF ABSOLUTE
0175A000:0175CFFF ABSOLUTE
40,960, X'0000A000', bytes described in ABSOLUTE

ASID(X'0001')
00000000:00000FFF COMMON PREFIXED TRANSLATED
006F6000:006FBFFF
00AF2000:00B0CFFF COMMON TRANSLATED
00FA0000:00F4DFFF COMMON TRANSLATED
00FB8000:00F90FFF COMMON TRANSLATED
00F9C000:00FA0FFF COMMON TRANSLATED
00FAC000:00FA6FFF COMMON TRANSLATED
00FB0000:00FC9FFF COMMON TRANSLATED
00FCF000:00FD2FFF COMMON TRANSLATED
00FD3000-00FD5FFF COMMON MISSING TRANSLATED
00FD6000:00FD6FFF COMMON TRANSLATED

1,589,248, X'00184000', bytes described in ASID(X'0001')

ASID(X'0004')
7F350000:7F745FFF
69,632, X'00011000', bytes described in ASID(X'0004')
Example 3
List the dump sources described in the dump directory with additional title and symptom information.

- Action
  COMMAND ===> listdump symptoms
  This command has the defaults of SUMMARY and NOSELECT.

- Result
  The following output is produced.
LISTDUMP subcommand

<table>
<thead>
<tr>
<th>Source of Dump</th>
<th>Blocks</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td>2</td>
<td>8,320</td>
</tr>
<tr>
<td>No title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No symptoms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source of Dump</th>
<th>Blocks</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNAME('C89.BLSRMVCL.SOC4DUMP')</td>
<td>26,544</td>
<td>110,423,040</td>
</tr>
<tr>
<td>Title=JOBNAME C89 STEPNAME SMPROC SMPROC SYSTEM 0C4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Psym=RIDS/BLSRVEC3#L RIDS/BLSRMVL PIDS/5752SC132 AB/S00C4 RIDS/BLSUSTAI#R V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source of Dump</th>
<th>Blocks</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNAME('H441PCS.R38A.PMR00137.B379.EH603')</td>
<td>12,762</td>
<td>53,089,920</td>
</tr>
<tr>
<td>Title=COMPON=COMPONENT TRACE,COMPID=SCTRC,ISSUER=ITTAWRIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Psym=RIDS/NUCLEUS#L RIDS/ITTAWRIT PIDS/5752SCTRC AB/S001D RIDS/ITTAWRIT#R VA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source of Dump</th>
<th>Blocks</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNAME('H441PCS.R38A.PMR00218.B677.DUMP1')</td>
<td>10,574</td>
<td>43,987,840</td>
</tr>
<tr>
<td>Title=SLIP DUMP ID=0005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trap=SLIP SET,COMP=01D,NUCMOD=1ARDS,DN=(3.*,15.SYSLGR0),SD=(ALLNUC,PSA,SQA,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source of Dump</th>
<th>Blocks</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNAME('NHAN.FBS29K.DUMP')</td>
<td>438,123</td>
<td>1,822,591,680</td>
</tr>
<tr>
<td>Title='MVSPROD1 02/27/97'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No symptoms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Dumps described
IPCS

The output medium to which the preceding output was directed was 78 characters wide. This caused the lines beginning "Psym=RIDS/BLSRVEC3#L", "Psym=RIDS/NUCLEUS#L", and "Trap=SLIP" to be truncated.

LISTEDT subcommand — format the eligible device table (EDT)

Use the LISTEDT subcommand to display information from the eligible device table (EDT). You can access the EDT in a dump data set or in active storage.

The system can have two EDTs during a dynamic configuration change. You must distinguish between formatting a primary EDT and a secondary EDT.

Each EDT is divided into subtables, which you can format separately with LISTEDT.

See the allocation/unallocation component in z/OS MVS Diagnosis: Reference for information about primary and secondary EDTs. Also, see z/OS MVS Data Areas in z/OS Internet Library at http://www.ibm.com/systems/z/os/zos/bkserv/ for information about the EDT.

Syntax
LISTEDT subcommand

LISTEDT
  [ PRIMARY | SECONDARY ]

------- Data Selection Parameters ----------------------
  [ COMPGENS[[index-number-list]] ]
  [ DETAIL ]
  [ DEVNUM[[index-number-list]] ]
  [ DEVPool[[index-number-list]] ]
  [ GENERIC[[index-number-list]] ]
  [ GROUP[[index-number-list]] ]
  [ GRPMSK[[index-number-list]] ]
  [ GRPPTT[[index-number-list]] ]
  [ GRPCONV[[index-number-list]] ]
  [ HEADER ]
  [ LIBRARY[[index-number-list]] ]
  [ LUv[[index-number-list]] ]
  [ PREF[[index-number-list]] ]
  [ SHOWDEVN(device-number-list) ]
  [ SHOWGRPN[[group-number-list]] ]
  [ SUMMARY[[unit-name-list]] | SHOWUNIT[[unit-name-list]] ]
  [ TAPE ]

------- SETDEF-Defined Parameters ------------------------
Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232
  [ ACTIVE | MAIN | STORAGE ]
  [ DSNAME(dsname) | DATASET(dsname) ]
  [ FILE(ddname) | DDNAME(ddname) ]
  [ PATH(path-name) ]

  [ FLAG(severity) ]
  [ PRINT | NOPRINT ]
  [ TERMINAL | NOTERMINAL ]
  [ TEST | NOTEST ]

Parameters

PRIMaRY or SECONDARY
  Specifies the EDT that is to be formatted. The types of EDTs are:
  - Primary EDT: processes all current and new allocation requests.
  - Secondary EDT: processes all allocation requests issued before a dynamic configuration change.
LISTEDT subcommand

PRIMARY is the default. If you specify SECONDARY and no secondary EDT exists in the source storage or dump, IPCS displays message IEF10010I in the report.

Data Selection Parameters

Use these parameters to limit the scope of the data in the report. If you omit a data selection parameter, the default is HEADER.

In the data selection parameter descriptions, index-number-list is one or more 1- to 4-digit hexadecimal numbers, ranges of numbers, or both. Each index number corresponds to an index for a sub-table entry. If you omit index-number-list, IPCS formats the entire sub-table.

The index-number-list can be a single number, a range of numbers, or a list of numbers. When you specify a range, separate the first and last numbers in the range with a colon. When you specify a list, separate the numbers with commas. The number or numbers are enclosed in parentheses.

**COMPGENS**

Specifies that the compatible-generic section of the EDT appears in the output. Generics are compatible when a data set can be allocated to any generic.

**DETAIL**

Specifies that all the subtables in the EDT appear in the output.

**DEVNUM**(index-number-list)

Specifies that the device number section appears in the output.

**DEVPOOL**(index-number-list)

Specifies that the system-managed type library device pool entries in the EDT appear in the output. Each pool represents a set of tape drives within a library. In the output, look-up-value entry indexes refer to the output of the LUV parameter of the LISTEDT subcommand.

**GENERIC**(index-number-list)

Specifies that the generic section of the EDT appears in the output.

**GROUP**(index-number-list)

Specifies that the group section of the EDT appears in the output.

**GRPCONV**(index-number-list)

With Version 4.2.0 or a later release, specifies that the group mask conversion table appears in the output. This table exists only after a dynamic configuration change.

**GRPMSK**(index-number-list)

Specifies that the group mask table appears in the output.

**GRPPTTR**(index-number-list)

Specifies that the group pointer table of the EDT appears in the output.

**HEADER**

Specifies that the EDT header appears in the output.

**LIBRARY**(index-number-list)

Specifies that the system-managed tape library entries in the EDT appear in the output. The entries include indexes for the related system-managed tape library device pool entries.

**LUV**(index-number-list)

Specifies that the look-up value section of the EDT appears in the output.
LISTEDT subcommand

PREF[(index-number-list)]
Specifies that the preference table appears in the output.

SHOWDEVN(device-number-list)
Lists the group number to which each device number in the
device-number-list belongs. device-number-list must be specified and should
consist of one or more 1- to 4-digit hexadecimal device numbers, ranges of
numbers, or both.

SHOWGRPN[(group-number-list)]
Lists the unit names associated with each of the group numbers in the
group-number-list.
group-number-list is one or more 1- to 4-digit hexadecimal numbers, ranges
of numbers, or both. If you do not supply group-number-list, IPCS formats
information for all the device groups in the system.

SUMMARY[(unit-name-list)] | SHOWUNIT[(unit-name-list)]
Produces a summary report for all the unit names in the unit-name-list.
unit-name-list is one or more 1- to 8-character alphanumeric unit names.
Separate multiple list items with one or more commas, blanks, or tab
characters (X'05'). If you do not supply unit-name-list, IPCS formats
information for all unit names in the system.

TAPE
Requests formatting of the tape maximum eligibility table. The output
includes tape device information such as density and device type.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the LISTEDT subcommand.

Example
Display information for device numbers 0001 through 0006 and 0021 through
0028 in the secondary EDT.
– Action
  COMMAND ===> listedt secondary devnum(0001:0006,0021:0028)
– Result
  See the allocation/unallocation component in z/OS MVS Diagnosis: Reference
  for an example of LISTEDT output.

LISTMAP subcommand — list storage map entries

Use the LISTMAP subcommand to produce output using the storage map:
• Generate dump displays of blocks within a range of addresses (VERIFY option).
• Repeat diagnostic messages pertaining to blocks within a range of addresses
  (RESCAN option).

The storage map is part of a source description. A source description is for an
unformatted source that IPCS can format, for example, an SVC dump, a
stand-alone dump, an SYSDUMP dump, a trace data set, a data set, or active
storage. The source description is in the dump directory allocated with ddname
IPCSDDIR and is your current dump directory. The current dump directory is your
user dump directory or, for users with write access authority, might be the sysplex
dump directory.

For information about using the storage map, see the z/OS MVS IPCS User’s Guide
LISTMAP subcommand

Related subcommands
DROP MAP
SCAN

Syntax

\{ LISTMAP \} \[ RANGE(address:address) \] \[ data-descr \]
\{ LMAP \}

\[ RESCAN | NORESCAN \]
\[ SUMMARY | NOSUMMARY \]

------ SETDEF-Defined Parameters ---------------
Note: You can override the following SETDEF parameters.
See
"SETDEF subcommand — set defaults" on page 5-232
\[ DISPLAY[(display-options)] \]
\[ NODISPLAY[(display-options)] \]

\[ FLAG(severity) \]
\[ PRINT | NOPRINT \]
\[ TERMINAL | NOTERMINAL \]
\[ TEST | NOTEST \]
\[ VERIFY | NOVERIFY \]

Parameters

RANGE(address:address)
Specifies a range of addresses in the dump for which map entries are to be listed.

data-descr
Specifies the data description parameter, which consists of five parts:
- An address (specified with the RANGE parameter and required when data-descr is explicitly specified on the subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

If you specify address processing parameters (which are optional) but omit the address (which is required), the subcommand lists all map records for the address space.

If you omit the range parameter, the subcommand lists all map records for the dump.

RESCAN or NORESCAN
Requests or suppresses retransmission of diagnostic messages pertaining to blocks in the range selected, subject to the restriction imposed by the FLAG parameter.

RESCAN requests retransmission.

NORESCAN suppresses retransmission.
LISTMAP subcommand

SUMMARY or NOSUMMARY

SUMMARY indicates that a processing summary (a final total line) is to be produced.

NOSUMMARY specifies that a processing summary is to be suppressed. The NOSUMMARY parameter is useful to turn off summary messages when the subcommand is invoked within a CLIST or a REXX exec.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the LISTMAP subcommand.

Example
Display storage map entries for a range of addresses.
– Action
listmap range(5000.:10000.) terminal noprint
– Result
The subcommand requests a display, at the terminal only, of the storage map entries that originate between the addresses X’5000’ and X’10000’.

LISTSYM subcommand — list symbol table entries

Use the LISTSYM subcommand to display the definitions of symbols for a source or to produce a display using symbols for a source.

The symbols are in a symbol table that is part of a source description. A source description is for an unformatted source that IPCS can format, for example, an SVC dump, a stand-alone dump, an SYSMDUMP dump, a trace data set, a data set, or active storage. The source description is in the dump directory allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

Related subcommands
DROPSYM
EQUATE
RENUM
STACK

Syntax
LISTSYM subcommand

{ LISTSYM } [ (symbol-list) | * ]
{ LSYM }

[ SELECT [(ALL | DROP | NODROP)] ]

[ SUMMARY | NOSUMMARY ]

------ SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ DISPLAY[(display-options)] ]
[ NODisplay[(display-options)] ]

[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

symbol-list or *

Specifies the symbols to be displayed:
- symbol-list specifies one or more particular symbols.
- * specifies all the symbols in the symbol table. If you omit this
  parameter, the default is *.

The symbol-list can be a single symbol, a range of symbols, a list of symbols,
or any combination of these. When you specify a range, separate the first
and last symbols in the range with a colon. When you specify a list,
separate the symbols with commas. If you specify more than one symbol or
range, enclose them in parentheses. The list can contain a maximum of 31
symbols, ranges, or both.

The symbols must follow the IPCS naming conventions for symbols if a
range is specified. See “IPCS symbols” on page A-1.

For a range, IPCS displays all symbols whose names begin with the first
character string through all symbols whose names begin with the second
character string. A range of symbols is inclusive; IPCS displays all the
symbols in the range and at both ends of the range.

SELECT(ALL | DROP | NODROP)

Specifies a selection criterion for symbols to be displayed:
- ALL specifies that all symbols are to be displayed.
- DROP specifies that only symbols with the DROP attribute are to be
displayed.
- NODROP specifies that only symbols with the NODROP attribute are to be
displayed.

If you omit ALL, DROP, or NODROP, the default is ALL.

SUMMARY or NOSUMMARY

SUMMARY indicates that a processing summary (a final total line) is to be
produced.
LISTSYM subcommand

NOSUMMARY specifies that a processing summary is to be suppressed. The NOSUMMARY parameter is useful to turn off summary messages when the subcommand is invoked within a CLIST or a REXX exec.

**DISPLAY((display-options))**
**NODISPLAY((display-options))**
Specifies the display options. The defaults are:

```
DISPLAY(NOMACHINE NOREMARK REQUEST NOSTORAGE SYMBOL)
```

LISTSYM uses a special, tabular display format unless you specify one of the following display options:

```
DISPLAY(MACHINE NOREQUEST STORAGE NOSYMBOL)
```

If you specify none of these options, IPCS uses the general-purpose dump display format.

In addition, the archaic REMARKS parameter can be specified as a separate parameter. REMARKS is the equivalent of DISPLAY(REMARK). It causes the display to include any remarks associated with a symbol.

**ACTIVE or MAIN or STORAGE**
**DSNAME(dsname) or DATASET(dsname)**
**FILE(ddname) or DDNAME(ddname)**
Specifies the source of the source description containing the symbol. If one of these parameters is not specified, the default is your current source.

**Return Codes**
See [“Standard subcommand return codes” on page 5-2](#) for a description of the return codes produced by the LISTSYM subcommand.

**Example 1**
List a range of symbols.

- **Action**
  
  COMMAND ===> listsym (my:my title acvt)

- **Result**

  The following output is produced.

```
SYMBOL ADDRESS ATTRIBUTES
ACVT  ID418. ASID('0001') POSITION(-24) LENGTH(1248) STRUCTURE(CVT) DROP
MY#LONG#SYMBOLIC
  0. ASID('0078') LENGTH(96) AREA DROP
MYARRAY F0000. ASID('0078') POSITION(+64) LENGTH(4) ENTRIES(52:77) SIGNED DROP
MYCVT  ID418. ASID('0001') POSITION(-24) LENGTH(1248) STRUCTURE(CVT) DROP
TITLE  0. HEADER POSITION(20) LENGTH(53) CHARACTER NODROP
5 DEFINITIONS LISTED
```

- **Explanation**

  - Symbols are always processed alphabetically. Specifying “acvt” after the other selection criteria produces the same result as moving it to the beginning of the list.

  - A caption line is provided for the special, tabular format of the LISTSYM display. Symbol and address captions describe the values that will appear beneath. Attributes are shown in a self-describing format using standard IPCS parameters plus decimal or hexadecimal values. Underscores are added to the caption line when transmitted to a print data set.

  - The entire definition of a symbol is typically displayed on one line. The format resembles that of the EQUATE subcommand parameters.
LISTSYM subcommand

- When the symbol and the address overlap, if both are displayed on a single line, the symbol will appear alone on the initial line, and the address and attributes will begin on a second line.
- When the full complement of attributes will not fit on one line, they may overflow onto an additional line.

Example 2
List a range of ASCB symbols.
- Action
  COMMAND ===> listsym (ascb00001 : ascb00050)
- Result
  LISTSYM displays the ASCB symbols for ASID 1 through 50.

Example 3
List a range of TCB symbols.
- Action
  COMMAND ===> listsym (tcb00001aaaaa : tcb00001baaaa)
- Result
  LISTSYM displays the specified range of TCBs.

LISTTOD subcommand — list TOD clock image

Use the LISTTOD subcommand to translate a hexadecimal GMT TOD clock value to the specified time stamp. The LISTTOD command supports three types of STCK or STCKE time stamps using the time-zone adjustments from your dump:
- ABSOLUTE time stamps are produced by the STCK or STCKE instructions directly.
- UTC time stamps are produced by adjusting the STCK or STCKE time stamps using a leap second adjustment factor maintained by the z/OS timer services.
- LOCAL time stamps are produced by adjusting the UTC time stamps using a time zone adjustment factor maintained by the z/OS timer services.

An INPUT option is now supported to allow the IPCS user to say which interpretation applies to the time stamp being entered, and LISTTOD now formats 26-character values corresponding to all three interpretations.

Use the INPUT option to specify the interpretation type for the time stamp. If you omit this option, the default value is ABSOLUTE. The system translates the TOD clock value to the time stamp as you specified, and it also formats other time stamps if the corresponding adjustment factors can be retrieved from the current dump. The first one to be displayed is for the specified option, and the other two are to be shown in the following order: ABSOLUTE, UTC, and LOCAL.

Syntax
LISTTOD subcommand

{ LISTTOD|LTOD } (gmt-tod-value) [ EXTENDED ]
[ INPUT ( { ABSOLUTE | UTC | LOCAL } ) ]

------- SETDEF-Defined Parameters ------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME (dsname) | DATASET (dsname) ]
[ FILE (ddname) | DDNAME (ddname) ]
[ PATH (hfs path) ]
[ FLAG (severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

gmt-tod-value
Specifies the first 1-32 hexadecimal digits of a TOD clock value associated
with a dump.

EXTENDED
Specifies that the value should be treated as a 16-byte STCKE value rather
than an 8-byte TOD clock value that are stored by the STCK instruction. In
the output line for each time stamp, the 8-byte STCK or 16-byte STCKE
value used to format the time stamp is also displayed.

INPUT ( { ABSOLUTE | UTC | LOCAL } )
Specifies the interpretation appropriate for the STCK or STCKE value
entered.

ABSOLUTE
Specifies that the value is the direct product of STCK or STCKE
instructions. If you omit the INPUT option, ABSOLUTE is the default
value.

UTC
Specifies that the value is adjusted with leap second factor.

LOCAL
Specifies that the value is adjusted with both leap second and local time
zone factors.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the LISTTOD subcommand.

Example 1
Translate the TOD clock value from a dump.

– Action
  listtod BDC613404B435A0A

  The command treats the value as STCK value, translates it to the absolute
time stamp and then formats the other two time stamps.
– Result
LISTTOD subcommand

Example 2
Translate the TOD clock value to local time stamp.

- Action
  listtod BDC613404B435A0A input(local)

  The command treats the value as STCK value, translates it to the local time stamp and then formats the other two time stamps.

- Result
  10/17/2005 17:46:48.479157 STCK X'BDC67135 DE1B5A0A'
  10/17/2005 17:46:26.479157 UTC X'BDC67120 E3035A0A'

Example 3
Translate the TOD clock value that is stored by the STCKE instruction.

- Action
  listtod 00BDC613404B435A0A extended

  The command treats the value as STCKE value, translates it to the absolute time stamp and then formats the other two time stamps.

- Result
  10/17/2005 10:46:26.479157 STCKE X'00BDC613 404B435A 0A000000 00000000'
  10/17/2005 10:46:04.479157 UTC X'00BDC613 2B502B5A 0A000000 00000000'
  10/17/2005 03:46:04.479157 LOCAL X'00BDC5B5 4AB86B5A 0A000000 00000000'

Example 4
Translate the TOD clock value to local time stamp, treating the value as stored by the STCKE instruction.

- Action
  listtod 00BDC613404B435A0A extended input(local)

  The command treats the value as STCKE value, translates it to the local time stamp and then formats the other two time stamps.

- Result
  10/17/2005 10:46:26.479157 LOCAL X'00BDC613 404B435A 0A000000 00000000'
  10/17/2005 17:46:48.479157 STCKE X'00BDC671 3DE18B5A 0A000000 00000000'
  10/17/2005 17:46:26.479157 UTC X'00BDC671 2DE3035A 0A000000 00000000'

LISTUCB subcommand — list UCBs

Use the LISTUCB subcommand as a convenient means to request the display of one or more unit control blocks (UCBs).

Syntax
LISTUCB subcommand

\{ LISTUCB \} (device-number-list)
\{ LISTU \}

-------- SETDEF-Defined Parameters -----------------
Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ DISPLAY((display-options)) ]
[ NODISPLAY((display-options)) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

device-number-list
  Specifies the device number for one or more devices for which UCBs are to be displayed. device-number-list can be:
  - A single hexadecimal device number of up to 4 digits with a subchannel set identifier digit specified on qualified devices.
    - Parentheses are accepted but are not required.
    - Leading zero digits are accepted but are not required.
  - A range of device numbers defined by the lowest and highest device numbers separated by a colon.
    - Parentheses are accepted but are not required.
    - Leading zeros are accepted but are not required.
    - The second device number must be as large as the first.
  - A list containing either single device numbers or ranges of device numbers. Parentheses are required. In the list, separate list members with blanks, commas, or horizontal tabulation (X'05') characters. The separators are permitted, but not required, between the left parenthesis and the first member and between the last member and the right parenthesis.

IPCS processes the list from the left to the right, displaying UCBs in that order. IPCS displays UCBs in a range starting with the lowest device number. An individual UCB can be specified as often as you want and is displayed again each time it is specified.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the LISTUCB subcommand.

Example
Format the device for device 0410.
  - Action
    COMMAND ===> LISTUCB 0410
  - Result
LITERAL subcommand

Use the LITERAL subcommand to assign a general value to a literal, which you identify with a symbol. IPCS stores the symbol and its value in the symbol table that is in a source description in your user dump directory.

If the source is a dump, IPCS does not initialize it. If the source has not been added to your user dump directory when you enter LITERAL, IPCS performs ADDDDUMP processing for it, then stores the symbol and its value in the newly created source description.

Syntax
LITERAL subcommand

LITERAL symbol general-value
   [ DROP | NODROP ]
   [ NOREMARK | REMARK('text') ]

------- SETDEF-Defined Parameters ---------------
Note: You can override the following SETDEF parameters.
See
"SETDEF subcommand — set defaults" on page 5-232
   [ ACTIVE | MAIN | STORAGE ]
   [ DSNAME(dsname) | DATASET(dsname) ]
   [ FILE(ddname) | DDNAME(ddname) ]
   [ PATH(path-name) ]

   [ TEST | NOTEST ]

Parameters
symbol
   Specifies the symbol that is to represent a literal. When specifying symbol,
do not include the ampersand (&) or the period (.) that are normally part of
symbolic notation. The symbol is 1 through 31 alphanumeric characters; the
first character must be a letter or one of the following characters:
$ (X'5B')
# (X'7B')
@ (X'7C')

general-value
   Specifies the value of the literal. See "General values" on page 2-3 for the
types of values and for how to specify them.

DROP
NODROP
   Specifies whether the created symbol can be deleted or not from the symbol
table by a DROPSYM subcommand without a PURGE parameter:
   - DROP specifies that the symbol can be deleted. The default is DROP.
   - NODROP specifies that the symbol cannot be deleted. However,
     NODROP can be overridden by a PURGE parameter on the DROPSYM
      subcommand.

REMARK('text')
NOREMARK
   Specifies or suppresses a remark associated with a symbol:
   - REMARK specifies the remark. The text of the remark must be enclosed
      in parentheses and apostrophes.
   - NOREMARK suppresses the remark.

ACTIVE or MAIN or STORAGE
DSNAME(dsname) or DATASET(dsname)
FILE(ddname) or DDNAME(ddname)
   Specifies the source of the source description that is to contain the symbol.
   If one of these parameters is not specified, IPCS stores the symbol in the
   source description for your current source.

Return Codes
See "Standard subcommand return codes" on page 5-2 for a description of the
return codes produced by the LITERAL subcommand.

Example
LITERAL subcommand

Create a literal and place it in the symbol table of your current user dump directory.
- Action
  literal data2 x'ff34a' nodrop
- Result
  IPCS places the literal X'FF34A' into the symbol table and identifies it the symbol DATA2.

LOGGER subcommand — format system logger address space data

The LOGGER subcommand formats data in the system logger address space in a dump. Status is provided about the state of the address space, coupling facility structures in use by system logger, logstreams and logstream connections.

The LOGGER command can help in diagnosing errors in the system logger address space, when the dump includes system logger private storage.

The LOGGER subcommand has no parameters.

Syntax

LOGGER

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAMEm-name ]
[ DATASET(dsname) ]
[ FILE(ddname) ]
[ DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the LOGGER subcommand.

LPAMAP subcommand — list link pack area entry points

Use the LPAMAP subcommand to list the entry points in the active link pack area (LPA) and pageable link pack area (PLPA), including the modified link pack area (MLPA). IPCS flags duplicate entry points in the modified link pack area (MLPA).

Related subcommands
  FINDMOD
  WHERE

Syntax
Parameters

EPA
Requests a report containing an entry point listing that is sorted by entry point address.

MODNAME
Requests a report containing an entry point listing that is sorted alphabetically.

ALL
Requests both the MODNAME and the EPA entry point reports.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the LPAMAP subcommand.

Example
Obtain the LPA entry points.
– Action
  LPAMAP
– Result
  The output follows.
The LPAMAP output continues with data like the above.

**MERGE and MERGEEND subcommands — merge multiple traces**

Use the MERGE subcommand to merge multiple component traces and generalized trace facility (GTF) traces chronologically. MERGE combines formatted trace entries produced by CTRACE subcommands, GTFTTRACE subcommands, or both, into chronological order in a single report. Use the MERGEEND subcommand to stop merging traces.
MERGE and MERGEEND subcommands

Start the merging by entering MERGE in IPCS line mode. Next, format the traces to be merged by entering, one at a time, CTRACE and GTFTRACE subcommands. You can enter up to 16 subcommands. To mark the end of the merging, enter MERGEEND.

**Note:** It is recommended that you use the MERGE option in the IPCS Dialog. See z/OS MVS IPCS User’s Guide for more information.

MERGE can process any of the dump or trace data sets that CTRACE and GTFTRACE can process; however, MERGE has one restriction. Only one of the trace sources may be on tape. The rest must be on direct access storage device (DASD).

Do not specify different output locations on the CTRACE and GTFTRACE subcommands. Each subcommand must contain the same output specifications. For example, do not specify PRINT on one subcommand and TERMINAL on another.

Any syntax errors on the CTRACE and GTFTRACE subcommands will result in unsuccessful processing of MERGE.

**Syntax**

```
MERGE
  ...
  1 to 16 CTRACE and GTFTRACE subcommands
  ...
MERGEEND
```

**Return Codes**

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the MERGE subcommand.

**Example**

Merge a component trace and GTF trace.

- **Action**
  
  ```
  MERGE
  CTRACE COMP(SYSRSM) FULL LIMIT(1) DSN('MYDUMP1')
  GTFTRACE DSN('COPY.TRACE1')
  MERGEEND
  ```

- **Result**

  MERGE produces a report similar to the following.
**MERGE and MERGEEND subcommands**

********** MERGED TRACES **********
01. GTF dsn(copy.trace1)
02. CTRACE dsn(rsm.ctrace) limit(5) comp(sysrsm) summary

**** GTFTRACE DISPLAY OPTIONS IN EFFECT ****
SSCH=ALL IO=ALL CCW=S1
SVC=ALL PI=ALL
EXT RNIO SRM RR DSP SLIP

**** GTF DATA COLLECTION OPTIONS IN EFFECT: ****
System resource manager events traced

**** GTF TRACING ENVIRONMENT ****
Release: SP4.2.0 FMID: HBB4420 System name: SYSTEM42
CPU Model: 3090 Version: FF Serial no. 170067

COMPONENT TRACE SUMMARY FORMAT
SYSNAME(SYSTEM41)
COMP(SYSRSM)

**** 07/23/90 ****

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>ENTRY ID</th>
<th>TIME STAMP</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>XEPEXIT</td>
<td>00000002</td>
<td>14:18:40.0000</td>
<td>External Entry Point Exit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010005</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
</tr>
</tbody>
</table>
MERGE and MERGEEND subcommands

02. XEPEXIT 00000002 14:18:40.952534 External Entry Point Exit
    FUNC1... GENIOCMP General I/O Completion
    JOBN1... JES2 ASID1... 0012 PLOCKS.. 00004081 CPU.... 0001
    JOBN2... *ALL* ASID2... FFFE RLOCKS.. 00004000
02. XEPENTRY 00000001 14:18:40.964644 External Entry Point Entry
    FUNC1... PGFIX Page Fix
    JOBN1... EDWTR1 ASID1... 0014 PLOCKS.. 80000001 CPU.... 0001
    JOBN2... EDWTR1 ASID2... 0014 RLOCKS.. 80000000

   Explanation
   The output from the MERGE subcommand begins with a numbered list of
   CTRACE and GTFTRACE subcommands that were input to MERGE. In the
   trace output, these numbers appear in the first two columns to identify each
   formatted trace entry with the trace subcommand that produced it. In the
   example:
   - 01. identifies a GTF trace entry
   - 02. identifies an RSM component trace entry
   The number for a component trace entry is on the first line of the entry. The
   number for a GTF entry is on the time-stamp line at the end of the entry.

NAME subcommand — translate an STOKEN

Use the NAME subcommand to identify the address space, data space, or subspace
related to an STOKEN, and return the ASID and name associated with the space.

IPCS can identify the data space for an STOKEN if the data space is accessible in
the dumped environment; storage from the data space does not need to be
dumped to enable the identification.

Related subcommands
    SELECT

Syntax
NAME subcommand

NAME STOKEN(value)

[ LIST | NOLIST]
[ CLIST (QUALIFICATION(variable-name)) ]
[ DIALOG (QUALIFICATION(variable-name)) ]
[ REXX (QUALIFICATION(variable-name)) ]

-------- SETDEF-Defined Parameters  -----------------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAMES(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameter

STOKEN(value)
Specifies the 8-byte STOKEN value of the address space, data space, or
subspace you want to identify. When you specify STOKEN, use the IPCS
rules for expressing general values; see "General values" on page 2-3.

LIST or NOLIST
LIST indicates that a report is to be generated. LIST is the default.
NOLIST suppresses the generation of a report.

CLIST(QUALIFICATION(variable-name))
DIALOG(QUALIFICATION(variable-name))
REXX(QUALIFICATION(variable-name))
Specifies where IPCS is to store the unedited value of STOKEN.
variable-name specifies the name of the variable into which the information
is stored. If the token cannot be successfully resolved by the NAME
subcommand, no change is made to the specified command procedure
variable.

CLIST directs that the value be stored in CLIST variable storage.
DIALOG directs that the value be stored in ISPF function pool dialog
variable storage.
REXX directs that the value be stored in REXX variable storage.

Return Codes
See "Standard subcommand return codes" on page 5-2 for a description of the
return codes produced by the NAME subcommand.

Example
Obtain the name of the address space, data space, or subspace associated with
the hexadecimal STOKEN value, 11223344 55667788.

-- Action
COMMAND ==> name stoken(x'11223344 55667788')
NAME subcommand

- Result

NAME produces a listing that displays the address space, data space, or subspace associated with the hexadecimal STOKEN value, 1122334455667788.

NAMETOKN subcommand — display the token from a name/token pair

Use the NAMETOKN subcommand to obtain the token from a name/token pair in a dump. Specify the name and the level of the name/token pair; in response, NAMETOKN returns the following:
- The token data
- Whether the name/token pair is persistent
- Whether an authorized program created the name/token pair
- The address space identifier (ASID) for the address space associated with the name/token pair

Syntax

NAMETOKN data-descr

{ NAME((name)) }

[ LIST | NOLIST]

[ CLIST (TOKEN(variable-name) ) ]
[ DIALOG (TOKEN(variable-name) ) ]
[ REXX (TOKEN(variable-name) ) ]

---------------- SETDEF-Defined Parameters ------------------

Note: You can override the following SETDEF parameters.

See

“SETDEF subcommand — set defaults” on page 5-232

[ FLAG(severity) ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ TEST | NOTEST ]

Parameters

data-descr

Describes the level of the name/token pair. The data description parameter consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

To retrieve the token from a task-level name/token pair, specify a TCB on data-descr. For example:

NAMETOKN TCB65A NAME((TASKLEV_NAME_003))
NAMETOKN 0F8640. STRUCTURE(TCB) ASID(65) NAME((TASKLEV_NAME_003))
NAMETOKN subcommand

To retrieve an primary- or home-address-space-level name/token pair, specify an ASCB on data-descr. For example:

```
NAMETOKN ASCB65 NAME((ASCBLEV_NAME_003))
NAMETOKN 0F2200. STRUCTURE(ASCB) NAME((ASCBLEV_NAME_003))
```

If you specify a data-descr other than an ASCB or TCB, NAMETOKN assumes the token you want to retrieve is from a system-level name/token pair. For example:

```
NAMETOKN & NAME((SYSTLEV_NAME_003))
NAMETOKN CVT NAME((SYSTLEV_NAME_003))
```

If you do not specify a data-descr parameter, NAMETOKN assumes the token you want to retrieve is from a system-level name/token pair.

NAME((name))

Specifies the name to be translated. NAMETOKN treats all text inside the parentheses, including blanks, literally. Enclose the name in double parentheses.

If the name contains non-printing hexadecimal characters or lowercase EBCDIC characters, then specify the name using hexadecimal characters. For example:

```
NAMETOKN NAME((X'007D3A23'))
```

In this case, NAMETOKN does not treat the apostrophes and the letter X literally.

LIST or NOLIST

LIST indicates that a report is to be generated. LIST is the default.

NOLIST suppresses the generation of a report.

CLIST(TOKEN(variable-name))
DIALOG(TOKEN(variable-name))
REXX(TOKEN(variable-name))

Specifies where IPCS is to store the unedited value of the token associated with the name. variable-name specifies the name of the variable into which the information is stored. If the token cannot be successfully resolved by the NAMETOKEN subcommand, no change is made to the specified command procedure variable.

CLIST directs that the value be stored in CLIST variable storage.
DIALOG directs that the value be stored in ISPF function pool dialog variable storage.
REXX directs that the value be stored in REXX variable storage.

Note: Many binary values can produce unintended results when placed into a CLIST variable. Only names associated with fully-printable EBCDIC tokens should be handled by a CLIST. Command procedures that need to handle arbitrary token values should be written using ISPF DIALOG or REXX services.

Return Codes

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the NAMETOKN subcommand.

Example 1

Retrieve a system-level token from the name/token pair SYSTLEV_NAME_003.

- Action

```
COMMAND ===> NAMETOKN CVT NAME((SYSTLEV_NAME_003))
```
Example 2
Obtain the logrec data set name by retrieving a system-level token from the name/token pair DSNLOGREC. This example has 5 actions.

Action 1
In the IPCS dialog, specify your dump data set and options.

Action 2
In the IPCS primary menu, choose the COMMAND option. In the COMMAND panel, enter:

```===> NAMETOKN 0 NAME((DSNLOGREC))```

Results
The following NAMETOKN output is produced.

```
System level
TOKEN.... 01CE0020 0100002C 00000000 00000000
NAME...... DSNLOGREC
ASID.... 0010
Persistent
Created by authorized program
```

Explanation
The fields in the output contain:
- Field 1: Address of area that contains the name of the logrec data set. The data set name field is 44 bytes.
- Field 2: 
  - Byte 1: Version
  - Byte 2: Reserved
  - Bytes 3 and 4: Length of data area pointed to by field 1
- Field 3: Reserved
- Field 4: Reserved

Action 3
Browse your dump data set to look at the address in the NAMETOKN output.

Result

ASID(X'0010') is the default address space

```
PTR   Address   Address space   Data type
000001 00000000   ASID(X'0010')   AREA
```

Remarks:

Action 4
Add a pointer entry that has the address from field 1 in the NAMETOKN output.

Results
NAMETOKN subcommand

ASID(X'0010') is the default address space
PTR Address Address space Data type
00001 00000000 ASID(X'0010') AREA
Remarks:
s0002 01CE0020 ASID(X'0010') AREA
Remarks:

- Action 5
  Select a new pointer to obtain a display of the logrec data set name.
- Results

01CE0020 01CE0020 ASID(X'0010') AREA
Remarks:

NOTE subcommand — generate a message

Use the NOTE subcommand to direct messages to the IPCSPRNT data set, IPCSPDS data set, your terminal, or all three, and to control spacing and pagination.

The maximum length of the message depends on its destination:
- Terminal display: The message is truncated to 250 characters.
- Print output data set: The message is truncated to the data set's logical record length, minus 5.
- Print output partitioned data set: The message is truncated to the data set's logical record length, minus 5.

Thus, a message may be truncated to a different length for each destination.

NOTE directs the message to the IPCSPRNT data set, IPCSPDS data set, your terminal, or all three, depending on the PRINT, PDS, and TERMINAL parameters. If you omit the PRINT, PDS, and TERMINAL parameters, NOTE uses the current local defaults for these parameters.

You can also assign a message severity level, which determines whether the message is sent to its destination. If the assigned message level is below the user's current default FLAG setting (see the SETDEF subcommand), the NOTE subcommand does not send the message. If the message level assigned to a message equals or exceeds the default FLAG setting, the subcommand sends the message.

Syntax
NOTE subcommand

{ NOTE } [ 'text' ]
{ N }

[ CAPS | ASIS ]

[ PAGE | NOPAGE ]

[ SPACE[ (count) ] ]

[ NOSPACE ]

[ OVERTYPE ]

[ TOC ( [indentation | 1 ] ) [toc-text ]] | NOTOC

------- SETDEF-Defined Parameters  --------------------------
Note: You can override the following SETDEF parameters.
See
 SETDEF subcommand — set defaults” on page 5-232

[ FLAG( severity ) ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ PDS | NOPDS ]

[ TEST | NOTEST ]

Parameters

'text'

Specifies the text of the message, enclosed in apostrophes. If the message is
directed to a terminal, it is truncated to 250 characters. If it is directed to
the IPCSPRNT or IPCSPDS data set, it is truncated to that data set's logical
record length, minus 5. If you specify a null line in this parameter, IPCS
assumes a blank line.

If you omit this parameter, IPCS transmits no message but performs the
specified spacing or paging relative to the previous line on the terminal or
in the IPCSPRNT and/or IPCSPDS data sets.

CAPS or ASIS

Specifies if the message text is to be in uppercase or in its present form,
which may be in uppercase, lowercase, or a mix.

CAPS specifies that IPCS translate the message text to uppercase.

ASIS specifies that IPCS not translate the message text, but transmit it in its
present form.

If you use this subcommand in a CLIST, the message text is normally
translated to uppercase by the editor or by CLIST processing before the
message text is available to IPCS, regardless if you specify ASIS. If you
want to use the ASIS option on the NOTE subcommand:
  – Ensure that the editor that you use stores mixed uppercase and
    lowercase text in your CLIST data set.
  – Ensure that your installation has installed TSO/E support for the
    CONTROL ASIS statement. Insert CONTROL ASIS in your CLIST before
    the first NOTE subcommand with ASIS. This allows the text that you
    entered in the CLIST to be passed to the IPCS NOTE subcommand
    without editing lowercase to uppercase.

If you omit both CAPS and ASIS, the default is CAPS.
NOTE subcommand

**PAGE or NOPAGE**
Specifies if the message is to be printed on a new page or the current page.

PAGE specifies a new page. PAGE affects printed output only. If the message is printed, NOTE precedes the message with a page eject. If the message is displayed on a user's terminal, NOTE ignores the PAGE parameter.

NOPAGE specifies that a new page not be forced before printing the message.

If you omit both PAGE and NOPAGE, the default is NOPAGE.

**SPACE[(count)]**
NOSPACE
OVERTYPE
Specifies if blank lines are to be added before printing the message or if the message is to overlay the previous message.

SPACE specifies the number of blank lines to be inserted before the message. The count may be specified as a decimal number. If you specify a count greater than PAGESIZE - 2 (as specified in the session parameters member), IPCS uses PAGESIZE - 2. If this parameter causes a page eject, you may lose 1 or 2 blank lines.

If you specify SPACE but omit the count, it defaults to 1.

NOSPACE inserts no blank lines before the message. The message becomes the next line in the output.

OVERTYPE overlays this message on the previous message. For example, you may use this parameter to underscore all or part of the previous message. The subcommand ignores this parameter if you specify no text or if the output is directed to a terminal.

If you omit SPACE, NOSPACE, and OVERTYPE, the default is NOSPACE.

**TOC [([indentation] [toc-text])]**
NOTOC
Specifies if a table of content entry is to be generated when the message associated with NOTE is routed to the IPCSPRNT data set.

TOC specifies that a table of contents entry is to be generated.

indentation
Indicates that the entry in the table of contents is to be indented.
Indentation is an integer from 1 through 4 and can be specified in decimal (n), binary (B'n'), or hexadecimal(X'n') notation. The default indentation is 1.

toc-text
One to 40 bytes of text that is to be associated with the table of contents entry. The text can be enclosed in single quotation marks if you want. The default toc-text is the text of the note, truncated to 40 characters where necessary.

NOTOC specifies that no table of contents entry is to be generated. NOTOC is the default.

Return Codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the NOTE subcommand.
OMVSDATA subcommand — format z/OS UNIX data

Use the OMVSDATA subcommand to generate diagnostic reports about z/OS UNIX System Services (z/OS UNIX) users and resources.

Syntax

OMVSDATA

-------- Data Selection Parameters --------------------------

[ COMMUNICATIONS ]

[ FILE ]

[ IPC ]

[ PROCESS ]

[ STORAGE ]

-------- Report Type Parameters ----------------------------

[ DETAIL ]

[ EXCEPTION ]

[ SUMMARY ]

-------- Address Space Selection Parameters ----------------

[ ASIDLIST(asidlist) ]

[ USERLIST(userlist) ]

-------- SETDEF-Defined Parameters --------------------------

Note: You can override the following SETDEF parameters. See "SETDEF subcommand — set defaults" on page 5-232

[ ACTIVE | MAIN | STORAGE ]

[ DSNNAME(dsname) | DATASET(dsname) ]

[ FILE(ddname) | DDNAME(ddname) ]

[ PATH(path-name) ]

[ FLAG(severity) ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ TEST | NOTEST ]

Parameters

Data Selection Parameters

Use these parameters to limit the scope of the data in the report. IPCS produces a report for each data selection parameter. If you omit a data selection parameter, the default is PROCESS.

COMMUNICATIONS

Specifies that communication services information appears in the report.
OMVS DATA subcommand

**FILE**
Specifies that file systems information appears in the report.

**IPC**
Specifies that the report is to contain information about interprocess communication for shared memory, message queues, and semaphores.

**PROCESS**
Specifies that information about all dubbed processes appears in the report. The report includes information about serialization, signaling, and, if the DETAIL parameter is also specified, open files.

**STORAGE**
Specifies that storage services information appears in the report.

**Report Type Parameters**
Use these parameters to select the type of report. If you omit a report type parameter, the default is SUMMARY.

**DETAIL**
Requests the detail report, which includes detailed information about the data area selected.

**EXCEPTION**
Requests the exception report, which contains exceptional or unusual conditions for the data area selected. The exception report contains diagnostic information for IBM use.

**SUMMARY**
Requests a summary report for the data area selected.

**Address Space Selection Parameters**
Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs) or by the TSO/E user IDs associated with the address space.

**ASIDLIST(asidlist)**
Specifies a list of ASIDs for the address spaces for which you want IPCS to process the requested data.

The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

An ASID can be 1 through 65535. An ASID can be expressed in the notation X'nnn', F'nnn', or B'nnn'. An unqualified number is assumed to be fixed.

**USERLIST(userlist)**
Specifies a list of TSO/E user IDs associated with the address spaces for which you want IPCS to process the requested data.

The userlist can be a single user ID or a list of user IDs. When you specify a list, separate the list members with commas. For example:

USERLIST(userid)

USERLIST(userid,userid...,userid)

**Return Codes**
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the OMVS DATA subcommand.

**Example**
OMVSADATA subcommand

See z/OS MVS Diagnosis: Reference for examples of the OMVSADATA subcommand and its output.

**OPCODE subcommand — retrieve operation code**

Use the OPCODE subcommand to retrieve the mnemonic operation code associated with an instruction.

**Syntax**

```
OPCODE
```

**Parameters**

**search-argument**
- The first 2-12 hexadecimal digits of the instruction of interest. If less digits are entered than needed to complete an instruction, trailing zero digits are supplied. Excess digits are ignored.

**LIST(var-list)**
- Requests that the information retrieved be made available to a command procedure or ISPF dialog.

**NOLIST**
- Specifies whether the information retrieved is to be displayed and, if it is, whether it is to appear as part of a line mode report or as an ISPF message on the logical screen.

**Examples**

In z/OS V1R4, IPCS enhances the display of the multi-byte operation codes associated with z/Architecture. The split-opcode instructions beginning with
OPCODE subcommand

E3, EB, or ED are displayed as follows:

Command ===> opcode e303

00000000 000A0000 000130E1 00000000 00000000 | ..............
00000010 00FC6FC0 00000000 00000000 00000000 | ..?{...........
00000020.:3F.--All bytes contain X'00'

The response to the command opcode e303 is shown below.

BLS18350I Split operation code X'E303' occupies bytes 0 and 5
Mnemonic for X'E303' is LRAG

OPEN subcommand — prepare resources for use by IPCS

Use the OPEN subcommand to prepare one or more resources for use by IPCS.
You can prepare:
• One or more source data sets containing dumps or traces
• Active storage, to be used as the source for IPCS processing
• A print data set with the ddname IPCSPRNT or a substitute name
• A table of contents (TOC) data set with the ddname IPCSTOC or a substitute name

See z/OS MVS IPCS User's Guide for information about using the OPEN subcommand for the print and TOC data sets.

Syntax

OPEN

[ ACTIVE | MAIN | STORAGE
  [ DSNAMES(dslist) | DATASETS(dslist) ]
  [ FILES(dslist) | DDNAMES(dslist) ]
  [ PATHS(path-name ...) ]
  [ DEFAULT ]
  [ CONDITIONALLY | UNCONDITIONALLY ]
  [ PRINT [(options) ] ]

-------- SETDEF-Defined Parameter -------------------------
Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232

[ TEST | NOTEST ]
[ CONFIRM | NOCONFIRM ]

Parameters

ACTIVE or MAIN or STORAGE
DSNAME(dslist) or DATASET(dslist)
OPEN subcommand

**FILE(ddlist | IPCSDDIR) or DDNAME(ddlist)**
Specifies the source to be prepared for use. If one of these parameters is not specified, IPCS opens the current source. IPCS opens the data sets in the order in which they are specified in the OPEN subcommand.

ACTIVE, MAIN, or STORAGE directs IPCS to prepare to access central storage as the source.

DSNAME or DATASET specifies the name of one or more cataloged data sets to be opened.

FILE or DDNAME specifies the ddname of one or more data sets to be opened.

When specifying more than one data set or ddname, separate the names with commas or blanks. When specifying a range of ddnames, separate the first and last ddname with a colon.

OPEN FILE(IPCSDDIR) indicates that you want to open the data set for your dump directory. You have to specify IPCSDDIR explicitly; specifying a range of ddnames does not include the dump directory. For further information about default values and restrictions for dump directories, see the CLOSE subcommand.

**PATH(path-name ...)**
Specifies one or more z/OS UNIX file paths to be processed. The **PATH(path-name ...)** option permits a list of path names to be processed in addition to any **ddnames** and **dsnames** listed on the subcommand. Partly-qualified path names may be used.

**DEFAULT**
Specifies that the final source listed in the subcommand is to become the current source. If the subcommand specifies a data set name with a password, the data set name and password become the name of the current source.

IPCS changes the current source in both the local and global defaults. If you omit this parameter, or if the subcommand fails, the current source is not changed in the defaults.

**CONDITIONALLY or UNCONDITIONALLY**
Determines how IPCS should handle a data set that is already open when the OPEN subcommand is processed.

For CONDITIONALLY, IPCS does not issue messages about the data being open.

For UNCONDITIONALLY, IPCS issues messages about the data set being open. UNCONDITIONALLY is the default.

**PRINT[(options)]**
Specifies the IPCS print data set.

The syntax for **options** is as follows:
OPEN subcommand

```
[ FILE(ddname|IPCSPRNT ]
[ DDNAME(ddname|IPCSPRNT ]
[ TITLE('text' ['time-stamp']) ]
[ TOC(FILE(ddname|IPCSTOC)) ]
[ CAPS ]
[ ASIS ]
[ CHARS(DUMP) ]
[ DISP|EXTEND|REUSE ]
```

**FILE(ddname|IPCSPRNT) or DDNAME(ddname|IPCSPRNT)**

Specifies that the designated ddname be opened as the IPCS print data set. If this parameter is omitted, FILE(IPCSPRNT) is used.

**TITLE(text[time-stamp])**

Specifies the title of the dump. The text appears on each page produced from the IPCS print data set.

Enclose the text in single quotation marks.

If *text* is omitted, IPCS uses the title extracted from the default dump data set. When processing multiple dumps during a single session, IPCS uses the default titles for each new dump encountered.

If IPCS cannot use the title from the default data set, but a userid is available, IPCS places on each page “IPCS PRINT LOG FOR userid” and the date and time that IPCS began problem analysis. If the userid is unavailable, “IPCS PRINT LOG” appears.

**Restriction:** When using IPCS in the background, the title will not contain the phrase “FOR userid” unless you use the TSO/E TMP and specify a USER parameter in the JCL JOB statement.

The *time-stamp* is the time that a problem occurred rather than the time that the problem analysis started.

Enclose the time stamp in single quotation marks.

If *time-stamp* is omitted, IPCS provides a date and time on the first line of each printed page indicating the time that the problem analysis started.

**TOC(FILE(ddname|IPCSTOC))**

Specifies that the data set be opened as the IPCS table of contents (TOC). If TOC is omitted, FILE(IPCSTOC) is used.

**Note:** The TOC data set must be different from the PRINT data set in order for both data sets to contain the correct data.

**CAPS**

Directs IPCS to change lowercase EBCDIC letters to uppercase before writing each line to the print and table of contents data sets.

**ASIS**

Directs IPCS to write text exactly as entered (uppercase and lowercase letters) to the data sets.

**CHARS(DUMP)**

Directs IPCS to format any text transmitted to the data sets in the IBM 3800 CHARS(DUMP) font. Use this option only for:
- Data sent to the print or TOC data sets, or both
- Data that has a data-type attribute of AREA
OPEN subcommand

Note: AREA is the IPCS default attribute parameter when a literal storage address is used and is the data-type associated with IPCS-defined symbols such as CSA.

**DISP|EXTEND|REUSE**
Permits an IPCS user, tailored dialogs, or command procedures to defer decision to overlay or extend a print file until a transaction that will use the file is requested.

**DISP**
Open the print and table of contents files with no attempt to influence positioning.

**EXTEND**
Requests that data management add additional records to the end of the print and table of contents files.

**REUSE**
Requests that data management reuse the print and table of contents files to contain new reports.

If you omit CAPS, ASIS or CHARS(DUMP), ASIS is the default.

If the logical record length for the IPCS print data set will not accommodate the text of the title plus a time stamp and a page number, the text is truncated.

**Return Codes**
See [“Standard subcommand return codes” on page 5-2](#) for a description of the return codes produced by the OPEN subcommand.

**Example 1**
Open the IPCS TOC data set.

- **Action**
  COMMAND ===> open print (toc(file(mytoc)) caps)

- **Result**
  File mytoc contains entries, which are written in uppercase.

**Example 2**
Open a print data set and give it a title.

- **Action**
  COMMAND ===> open print (title (‘A Troubled Dump’ ’12-07-81’))

- **Result**
  ‘A Troubled Dump 12-07-81’ appears on each page of the IPCS default print data set (IPCSPRNT).

PATCH subcommand

Use the PATCH subcommand to repair data residing in a RECFM=F or RECFM=FBS data set or to manage the list of patches in effect for a dump.

Patching may impact IPCS performance and is intended to be used very sparingly. The reason that a patching capability has been included is the following scenario:
1. You attempt to run a high level report against a dump. The report is important for your analysis.
2. The report writer encounters a block that appears to be damaged. Rather than using the contents of the damaged block and risking the production of a misleading report, the report writer identifies the block and the damage detected.
PATCH subcommand

3. You examine the damaged block, verify that its damage is not the root problem
   that you sought, and are able to determine values that repair damage to it.

4. You use the PATCH subcommand to identify the repairs to IPCS. IPCS does not
   alter the dump data set in any way. The alterations are stored in your dump
directory.

Patching storage that IPCS knows can be seen from multiple perspectives, such
as both common virtual storage and real storage visible to each CPU in the
dumped system, affects all perspectives.

Restrictions
- IPCS may access dump data before application of a patch, recording
  conclusions regarding that data in the dump directory before application of a
  patch. The PATCH subcommand does not attempt to locate and alter any
  such data. Some of this data may be affected using other subcommands such as
  DROPDUMP RECORDS(TRANSLATION)
  DROPMAP
  DROPSYM
- The current implementation of PATCH support directly uses data in dump
  records for most information associated with DISPLAY(MACHINE) output
  and the related data that may be extracted from a dump using the
  EVALUATE subcommand. Processing of storage by EVALUATE does honor
  PATCH requests.
- Storage may be added to what was dumped, such as from ASID(75), through
  PATCH processing, but PATCH will not attempt to identify the absolute or
  real storage locations where that storage would have resided in the dumped
  system. If this is important to your analysis, you must use PATCH to add it
  from all perspectives important to your analysis.

Qualifier
The following qualifiers distinguish the functions performed by the PATCH
subcommand:

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Causes the PATCH subcommand to store a new patch. See “Adding or replacing a patch” on page 5-195 for more information. Existing, overlapping patches are considered to be an error and cause the new patch to be rejected.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Causes the PATCH subcommand to delete patches. See “Deleting patches” on page 5-195 for more information.</td>
</tr>
<tr>
<td>LIST</td>
<td>Causes the PATCH subcommand to list patches. See “Listing patches” on page 5-196 for more information.</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Causes the PATCH subcommand to store a patch, replacing one or more existing ones whose descriptions overlap the new one. See “Adding or replacing a patch” on page 5-195 for more information. At least one existing, overlapping patch is expected. If there is none, it is considered to be an error, and the new patch is rejected.</td>
</tr>
<tr>
<td>STORE</td>
<td>Causes the PATCH subcommand to store a patch, replacing any existing ones whose descriptions overlap the new one. See “Adding or replacing a patch” on page 5-195 for more information.</td>
</tr>
</tbody>
</table>
PATCH subcommand

**Adding or replacing a patch**

Syntax

```
PATCH  { ADD | REPLACE | STORE }

general-value

[ data-descr ]
```

---- SETDEF-Defined Parameters
Note: You can override the following SETDEF parameters.

```
[ TEST | NOTEST ]
```

Parameters

**ADD**

**REPLACE**

**STORE**

Indicates whether the patch may replace existing patches that describe overlapping storage.

**general-value**

Specifies the patch using general value notation.

**data-descr**

Specifies the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

The following applies to PATCH ADD and PATCH REPLACE only:
- Patch uses the address space, address and offset to determine the origin of the storage to be patched. The number of bytes affected by the patching request are indicated by the general value entered.
- If you omit the ADDRESS parameter, the default for the ADD and REPLACE options of the PATCH subcommand is ADDRESS(X), the most recently accessed address.

**Deleting patches**

Syntax

```
PATCH       DELETE

[ data-descr ]
```

------ SETDEF-Defined Parameters
Note: You can override the following SETDEF parameters.

```
[ TEST | NOTEST ]
```

Parameters

**DELETE**

Indicates that patches affecting the storage described by `data-descr` are to be deleted.
PATCH subcommand

**data-descr**
Specifies the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

The following applies to PATCH DELETE only:
- All patches affecting the range of storage described are deleted.
- If you omit the ADDRESS parameter, the default for PATCH DELETE is ADDRESS(X), the most recently accessed address.

### Listing patches

**Syntax**

```plaintext
PATCH LIST
  [ data-descr ]
  [ DETAIL ]
```

**Parameters**

**LIST**
Indicates that patches affecting the storage described by `data-descr` are to be listed.

**data-descr**
Specifies the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

The following applies to PATCH LIST only:
- All patches affecting the range of storage described are listed.
- If you omit the ADDRESS parameter, the default for PATCH LIST is all patches.

**DETAIL**
Requests a detailed description of the data supporting patches.

**Return Codes**
The PATCH subcommand generates standard IPCS return codes.

**Code** | **Meaning**
---|---
X'00'  | Normal completion of the request.
X'0C'  | Request not completed for reasons related to user actions. Examples of such actions are:
- Specifying PATCH ADD processing for a location where a patch has already been applied.
**PATCH subcommand**

- Use of the TSO attention mechanism to terminate PATCH processing when IKJPARS solicits operand correction.

**X'10'**  
Request not completed because of problems with the IPCS execution environment. Examples of such problems are:
- Insufficient virtual storage to complete the request.
- An I/O error when accessing the dump directory.

IPCS transmits error messages, when possible, to identify the underlying cause of this return code.

**PROFILE subcommand — set preferred line and page size defaults**

Use the PROFILE subcommand to establish defaults for reports generated under IPCS:
- A preferred line size
- Preferred lines per printed page

The defaults you specify with PROFILE are recorded in your dump directory and remain in effect until you change them. You can issue PROFILE at any time during an IPCS session to view your default values. To change one or more of your defaults, enter the PROFILE subcommand with the parameters for the defaults.

Except for NOPAGESIZE, a newly established default is used for both the current session and any subsequent sessions in which you use the same dump directory. NOPAGESIZE does not become effective until the beginning of your next IPCS session.

Unlike the defaults set by a SETDEF subcommand, the PROFILE defaults cannot be overridden by parameters on other IPCS subcommands. The defaults can be changed only by entering a PROFILE subcommand.

The PROFILE-defined defaults shipped with IPCS are:

```bash
/*---------------------- IPCS Profile Data ------------------------*/
PROFILE NOEXCLUDE /* No dump analysis excluded */
PROFILE NOLINESIZE /* Limit for variable-width reports */
PROFILE NOPAGESIZE /* Line limit for print file pages */
PROFILE STACK(NODUPLICATES) /* Duplicate stack entry screening */
```

*Figure 5-3. PROFILE-Defined Defaults*

**Notes:**

1. The NOLINESIZE parameter is the equivalent to a line size of 250 characters per line. Variable-width reports can appear somewhat different when the output is directed to the terminal or the IPCS print data set.
2. The NOPAGESIZE parameter causes IPCS to use the PAGESIZE supplied in the IPCS session parameters member. If PAGESIZE is not supplied in the session parameters member, IPCS uses a default of 60 lines per page.

See [z/OS MVS IPCS User's Guide](#) for information about using the PROFILE subcommand to set print data set report defaults.

**Related subcommands**

- ANALYZE
- EVALPROF
- OPEN
### PROFILE subcommand

- WHERE

**Syntax**

```
{ PROFILE } [ EXCLUDE(name[ :name]...) | NOEXCLUDE ]
{ PROF } [ LINESIZE(nnn) | NOLINESIZE ]
    [ PAGESIZE(nnn) | NOPAGESIZE ]
[ LIST | NOLIST ]
[ STACK {{DUPLICATES | NODUPLICATES} ]
```

**Parameters**

**EXCLUDE(name[:name]...) or NOEXCLUDE**

Controls optional analysis performed by IPCS.

Using a single name explicitly designates a single type of analysis. Names can be 1-31 characters in length. They must begin with a letter or the characters $, @, or #. The same characters can be used in the remaining positions and decimal digits.

You are not limited to the names specified in Table 12. If you designate a name that is not supported by the current release, the name is recorded but has no effect on processing by IPCS.

Using name:name describes all types of analysis that collate within the range described. For example, the range A:B, excludes all types of analysis for which the name begins with either the letter A or the letter B.

Any list that you enter will be edited before being displayed by the LIST option of this subcommand or by the EVALPROF subcommand. The edited list is shown after it has been sorted and edited for efficient searching incorporating merging overlapping ranges. The implementation limits this list to 48 ranges.

Table 5-4 describes the naming conventions for the names supported by z/OS R7 MVS IPCS.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZEexit-name</td>
<td>The combination of the prefix ANALYZE and a suffix matching the name of an ANALYZE exit excludes that exit from the process of gathering contention data. This pertains to all places within IPCS where contention analysis may be performed, not only the ANALYZE subcommand.</td>
</tr>
<tr>
<td>WHERECSVCOMMON</td>
<td>Excludes WHERE processing that forces common area modules into the IPCS storage map before searching for associations.</td>
</tr>
</tbody>
</table>

---

Note: You can override the following SETDEF parameter. See "SETDEF subcommand — set defaults" on page 5-232.
Table 5-4. EXCLUDE parameter naming conventions (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERECSVPRIVATE</td>
<td>Excludes WHERE processing that forces private area modules into the IPCS storage map before searching for associations.</td>
</tr>
<tr>
<td>WHEREIGVPRIVATE</td>
<td>Excludes WHERE processing that forces private area pages for virtual storage manager subpools into the IPCS storage map before searching for associations.</td>
</tr>
</tbody>
</table>

**LINESIZE(nnn) or NOLINESIZE**

Controls the width of variable-width reports generated by IPCS. IPCS.

LINESIZE limits the width to nnn. Specify nnn in decimal ([+]nnn), hexadecimal (X'[X'X'X']xxx'), or binary (B'[B'B'B']bbb') notation. The minimum line size is 78 and the maximum is 250.

If variable-width reports are sent to any medium that is narrower than nnn characters, IPCS limits the output lines of the report to the width of the medium or 78 characters, whichever is larger.

NOLINESIZE specifies that variable-length reports use the full width of the medium to which they are written.

NOLINESIZE is equivalent to LINESIZE(250). NOLINESIZE is the default.

**PAGESIZE(nnn) or NOPAGESIZE**

Controls the number of lines per page in reports generated by IPCS.

PAGESIZE specifies the number of lines per page as nnn. Specify nnn in decimal ([+]nnn), hexadecimal (X'[X'X'X']xxx'), or binary (B'[B'B'B']bbb') notation. A nnn less than 3 is equivalent to NOPAGESIZE. The maximum page size is 2^{31}-1.

IBM recommends that you specify the number of lines that will fit on the forms typically used at your installation.

IPCS can generate normal, ascending page numbers if the printed output consumes less than 2^{32} lines of output medium. If you use a large PAGESIZE, the page number will wrap back to zero once the maximum is reached.

IPCS obtains the number of lines per page for the IPCS print output data set by checking the following in order:

1. The PAGESIZE specified on the PROFILE subcommand.
2. The PAGESIZE specified in the session parameters member for the IPCS session. (If PROFILE NOPAGESIZE is in effect, IPCS checks here first.)
3. When neither of the preceding is available, IPCS uses a default of 60 lines per page.

NOPAGESIZE specifies that a default not be established for the number of lines per page for the IPCS print data set. IPCS uses the PAGESIZE specified in the session parameters member or a default of 60 lines per page.

**Note:** Entering PROFILE NOPAGESIZE does not alter the default for your current IPCS session. It becomes effective at the beginning of your next IPCS session.
PROFILE subcommand

NOPAGESIZE is the default.

**LIST or NLIST**
Specifies if IPCS is to display your current PROFILE defaults on your terminal regardless of the current value for the TERMINAL parameter.

LIST specifies that the subcommand is to display all of the default values and parameters that are in effect. For an example, see Figure 5-3 on page 5-197.

NOLIST specifies that the subcommand not display the default values and parameters.

If you enter PROFILE without any parameters, the default is LIST. If you omit LIST and NOLIST but specify any other parameter, the default is NOLIST.

**STACK(DUPLICATES | NODUPLICATES)**
Controls duplication of stack entries for your current IPCS session and for future IPCS sessions that use the same dump directory.

STACK(DUPLICATES) allows stack entries to be duplicated.

STACK(NODUPLICATES) suppresses duplication of stack entries.

**Notes:**
1. To be considered a duplicate, a stack entry must have all the same attributes, including remarks, as an existing entry.
2. Specifying NODUPLICATES will not affect duplicate entries created as a result of:
   - The EQUATE subcommand and primary commands
   - The RUNCHAIN subcommand
   - The I and R line commands issued from the IPCS dialog BROWSE option pointer panel
   - From the BROWSE option pointer panel, editing that overstrikes a pointer stack entry
3. No messages result when duplicate entries are suppressed. The request is considered satisfied without action if the entry already exists.

**Return Codes**
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the PROFILE subcommand.

**Example**
Change your line, page, and stack defaults.

- **Action**
  
  COMMAND ===> profile linesize(78) pagesize(90) stack(duplicates) list

- **Result**
  
  You normally use a graphics terminal with a physical screen width of 80 characters but with an actual display screen of 78 characters. LINESIZE (78) tells IPCS to produce variable-width reports with a line length of 87, regardless of whether the report output is directed to your terminal or to the print data set.
  Each printed page contains 90 lines of data.
  By specifying STACK(DUPLICATES), you authorize IPCS to add entries to the pointer stack that have exactly the same attributes as other entries in the pointer stack.
The LIST parameter displays the following:

```c
/---------------------- IPCS Profile Data ------------------------*/
PROFILE LINESIZE(78) /* Limit for variable-width reports */
PROFILE PAGESIZE(90) /* Line limit for print file pages */
PROFILE STACK(DUPLICATES) /* Duplicate stack entry screening */
```

**RENUM subcommand — renumber symbol table entries**

Use the RENUM subcommand to renumber all address pointer entries in the
symbol table in your dump directory. IPCS renumbers the entries in ascending
order, from Z1 to Z99999.

The symbol table is part of a source description. The source description is in the
dump directory allocated with ddname IPCSDDIR and is your current dump
directory. The current dump directory is your user dump directory or, for users
with write access authority, might be the sysplex dump directory.

**Related subcommands**

EQUATE
DROPSYM
LISTSYM
STACK

**Syntax**

```c
{RENUM }   [ SUMMARY | NOSUMMARY ]
{REN }      
```

**Parameters**

**SUMMARY or NOSUMMARY**

SUMMARY specifies that a summary of RENUM’s processing is to be
produced. If so, IPCS issues one of the following comments (where \( n \) is a
number):

- The stack contains no entries.
- The stack contains 1 entry, none was renumbered.
- The stack contains 1 entry, 1 was renumbered.
- The stack contains \( n \) entries, 1 was renumbered.
- The stack contains \( n \) entries, \( n \) of which was renumbered.
- The stack contains \( n \) entries, none of which was renumbered.

NOSUMMARY specifies that a processing summary is to be suppressed.
The NOSUMMARY parameter is useful to turn off summary messages
when the subcommand is invoked within a CLIST or REXX exec.

**ACTIVE or MAIN or STORAGE**

DATASET(dsname) or DSNAME(dsname)
RENUM subcommand

FILE(ddname) or DDNAME(ddname)

Specifies the source of the source description containing the symbols. If one of these parameters is not specified, the source is your current source.

ACTIVE, MAIN, or STORAGE specifies central storage as the source.

DSNAME or DATASET specifies the name of a cataloged data set as the source.

FILE or DDNAME specifies the ddname for a data set as the source.

Return codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the RENUM subcommand.

Example

Renumber the address pointer entries in the symbol table.

- Action
  COMMAND ===> renum

- Result
  The subcommand produces the following summary output line:

  The stack contains 4 entries, 3 of which were renumbered

RSMDATA subcommand — analyze real storage manager data

Use the RSMDATA subcommand to generate reports about the attributes and status of the real storage manager (RSM) at the time of a dump. This subcommand produces the following types of reports:

- Address spaces report
- Common Pools
- Data-in-virtual mapped range report
- Data space report
- Exception report
- Execution status report
- Expanded storage report
- High virtual common
- High virtual page report
- High virtual shared data report
- Real frames report
- RSM requests report
- RSM shared data report
- Subspace report
- Summary report
- Trace
- Virtual pages report

Address space selection, data selection, and report type parameters limit the scope and extent of the information that appears in a report.

Syntax
RSMDATA subcommand

RSMDATA

-------- Report Type Parameters -----------------------------
  [ ADDRSPACE ]
  [ DIVMAP ]
  [ DSPACE ]
  [ EXCEPTION ]
  [ EXECUTION ]
  [ HIGHVIRTUAL]
  [ HVSHRDATA ]
  [ HVCOMMON ]
  [ REALFRAME ]
  [ RSMREQ ]
  [ SHRDATA ]
  [ SUBSPACE ]
  [ SUMMARY ]
  [ VIRTPAGE ]

-------- Data Selection Parameters ----------------------------
  [ COMMON ]
  [ DATASPACES ]
  [ DETAIL ]
  [ HVCOMM ]
  [ HVSHARED ]
  [ PERMCOMM ]
  [ RANGE(rangelist) ]
  [ SAVEAREA(address) ]
  [ SHARED ]
  [ SHORT ]
  [ STATUS(statuslist) ]
  [ TOKEN(token) ]
  [ TOTONLY ]

-------- Address Space Selection Parameters -------------------
  [ ALL ]
  [ ASIDLIST(asidlist) ]
  [ CURRENT ]
  [ ERROR ]
  [ JOBLIST(joblist) | JOBNAME(joblist) ]
RSMDATA subcommand

------- SETDEF-Defined Parameters ----------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF subcommand — set defaults” on page 5-232

Parameters
Report type parameters
Use these parameters to select the type of report. Specify only one; if you
specify more than one, RSMDATA processes only the right-most parameter. If
you omit a report type parameter, the default is SUMMARY.
Some of the selection parameters do not apply to all reports. "Matrix of report
type parameters and other parameters" on page 5-206 summarizes the
parameters you can specify with a given report.

ADDRSPACE
Requests the RSM address spaces report. This report summarizes real
storage usage for specified address spaces. The report is sorted by ASID.
Usage note: The only data selection parameters that apply to this report are
STATUS, SHORT, and TOTONLY.

DIVMAP
Requests the data-in-virtual mapped range report. This report displays
information relating to areas of storage that are identified to data-in-virtual
and that have been mapped. The information is sorted by address space
identifier (ASID) and by the status of each data-in-virtual mapped range.
Usage note: The only data selection parameters that apply to this report are
STATUS and TOTONLY.

DSPACE
Requests the data space report. This report displays information about all
data spaces in the system. All installation-defined and RSM-defined data
spaces are summarized.
Usage note: The only data selection parameter that applies to this report is
TOTONLY.

EXCEPTION
Requests the RSM diagnostics report. This report verifies RSM global data
structures and generates information about areas that are in error. You can
also request verification of local data structures for specific address spaces
using address space selection parameters.
Usage note: The only data selection parameters that apply to this report are
DATASPACES and SAVEAREA.

Note: The EXCEPTION report might take an excessive amount of time to
run when one or both of these conditions is true:
RSMDATA subcommand

- You specify more than 3 address spaces.
- You have specified DATASPACES and any of the specified address space owns more than 3 data spaces.

You might consider submitting a batch job to obtain an EXCEPTION report under these circumstances.

EXECUTION
Requests the RSM execution status report. This report contains information for IBM internal use. IBM might ask you to run this report for use in problem determination.

Usage note: The only data selection parameter that applies to this report is SAVEAREA. Address space selection parameters do not apply to this report.

HVCOMMON
Requests the high virtual common report. This report displays the status of high virtual common memory objects including owner, size, and status.

Usage note: The only selection parameter that applies to this report is RANGE.

HIGHVIRTUAL
Requests the high virtual page report. This report identifies the page owner, the location and status for virtual pages in the system that are above 2 Gigabytes, and a summary of the memory objects.

Usage note: The only data selection parameters that apply to this report are RANGE, STATUS, and TOTONLY.

Note: The VIRTPAGE report might take an excessive amount of time to run when large ranges are specified.

HVSHRDATA
Requests the high virtual shared data report. This report provides information about virtual storage above 2 gigabytes that is shared using the IARV64 macro.

Usage note: The only data selection parameters that apply to this report are RANGE and DETAIL.

REALFRAME
Requests the real frame report. This report displays information about each frame's status, location, and current/most recent owner. The information is sorted by the ASID of the current/most recent owner unless you specify the ALL address space selection parameter. In this case the information is sorted by frame number.

Usage note: The only data selection parameters that apply to this report are COMMON, PERMCOMM, RANGE, SHARED, HVCOMM, HVSHARED, STATUS, and TOTONLY.

RSMREQ
Requests the RSM requests report. This report summarizes asynchronous RSM activity in the system or for a particular job. It identifies the requester, lists the request's status, and identifies the requested pages for asynchronous requests.

Usage note: The only data selection parameters that apply to this report are COMMON, SHARED, HVCOMM, HVSHARED, STATUS, and TOTONLY.
RSMDATA subcommand

SHRDATA
Requests the RSM shared data report. This report provides information about the virtual storage locations that are defined as shared through the IARVSERV macro.

Usage note: The only data selection parameters that apply to this report are COMMON, STATUS, TOKEN, and TOTONLY.

SUBSPACE
Requests the subspace report. This report displays information about subspaces in an address space. The information is sorted by ASID and, within the address space, by the address at the lower limit of the range.

Usage note: The only data selection parameters that apply to this report are RANGE and STATUS.

SUMMARY
Requests the RSM summary report and is the default. This report provides statistics about system-wide real and auxiliary storage usage. It also contains information about any unusual RSM conditions that exists in the dump.

Usage note: Data selection and address space parameters do not apply to this report.

VIRTPAGE
Requests the virtual page report. This report identifies the page owner and its location and status for virtual pages in the system.

Usage note: The only data selection parameters that apply to this report are COMMON, DATASOURCES, PERMCOMM, RANGE, STATUS and TOTONLY.

Note: The VIRTPAGE report might take an excessive amount of time to run when one or both of these conditions is true:
- You specify more than 3 address spaces.
- You have specified DATASOURCES and any of the specified address space owns more than 3 data spaces.

You might consider submitting a batch job to obtain a VIRTPAGE report under these circumstances.

Matrix of report type parameters and other parameters

The following two tables summarize for each report type use of address space selection parameters and data selection parameters.

<table>
<thead>
<tr>
<th>Report Type Parameter</th>
<th>ALL ASIDLIST</th>
<th>CURRENT</th>
<th>JOBLIST/JOBNAME</th>
<th>COMMON</th>
<th>DATASOURCES</th>
<th>DETAIL</th>
<th>HVCOMMON</th>
<th>HVSHARED</th>
<th>PERMCOMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRSPACE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVMAP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSPACE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXECUTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGHLIMIT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVCOMMON</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVSHRDATA</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REALFRAME</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSMREQ</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHRDATA</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RSMDATA subcommand

<table>
<thead>
<tr>
<th>Report Type Parameter</th>
<th>ALL ASIDLIST CURRENT JOBLIST/JOBNME</th>
<th>COMMON</th>
<th>DATASPACES</th>
<th>DETAIL</th>
<th>HVCOMM</th>
<th>HVSHARED</th>
<th>PERMCOMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSPACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SUMMARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIRTPAGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Report Type Parameter</th>
<th>RANGE</th>
<th>SAVE AREA</th>
<th>SHARED</th>
<th>STATUS</th>
<th>TOKEN</th>
<th>TOTONLY</th>
<th>SHORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRSPACE</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>DIVMAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSPACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCEPTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXECUTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGHVIRTUAL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>HVCOMMON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVSHRDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REALFRAME</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>RSMREQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHRDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSPACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>VIRTPAGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data selection parameters

Use these parameters to limit the scope of the data in the report.

**Note:** Common area data is not included when you specify ASIDLIST, JOBNAME, or JOBLIST. You need to specify COMMON or PERMCOMM with the report parameters that accept them if you want to see common area resources in the report. High virtual shared data is not included when you specify ASIDLIST, JOBNAME, or JOBLIST. You need to specify HVSHARED with the report parameters that accept them if you want to see high virtual shared resources in the report.

**COMMON**
Requests that any non-permanently-assigned common area page found in CSA, SQA, PLPA, MLPA, or common disabled reference storage appear in the report. Use COMMON to select data in the REALFRAME, RSMREQ, SHRDATA, and VIRTPAGE reports.

**DATASPACES**
Requests information about data spaces for the VIRTPAGE and EXCEPTION reports. (For these reports, data space-related information will not appear unless you explicitly request it.)

**DETAIL**
Requests that more detailed information be reported. For the HVSHRDATA report this information includes the view of segments from each address space sharing the memory object. Use DETAIL with the HVSHRDATA report.

**HVCOMM**
Requests that the report contain information about data defined as high virtual common. Use HVCOMM to select data in the REALFRAME or RSMREQ reports.
RSMDATA subcommand

**HVSHARED**
Requests that the report contain information about data defined as high virtual shared (shared storage above two gigabytes). Use HVSHARED to select data in the REALFRAME or RSMREQ reports.

**PERMCOMM**
Requests that permanently assigned pages in the nucleus, absolute frame zero, PSAs, HSA, or FLPA appear in the report. Use PERMCOMM to select data in the REALFRAME and VIRTPAGE reports.

**RANGE(rangelist)**
Specifies a range of real frames or virtual pages to include in the report. Use RANGE with the REALFRAME, SUBSPACE, VIRTPAGE, HIGHVIRTUAL, HVCOMMON, and HVSHRDATA reports.

The rangelist is one or more ranges. In each range, the lower and upper limits are separated by a colon character (:).

The value to specify for rangelist depends on the report:

<table>
<thead>
<tr>
<th>Report Parameter</th>
<th>Value for rangelist</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGHVIRTUAL</td>
<td>Hexadecimal virtual addresses from 80000000 to FFFFFFFF_FFFFFFFF. The default range for this report is 1_00000000:1_80000000.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Each range limit can be 17 characters each and may contain underscores.</td>
</tr>
<tr>
<td>HVCOMMON</td>
<td>Hexadecimal virtual addresses from 80000000 to FFFFFFFF_FFFFFFFF. The default range for this report is the defined common area for the system which is dumped.</td>
</tr>
<tr>
<td>HVSHRDATA</td>
<td>Hexadecimal virtual addresses from 80000000 to FFFFFFFF_FFFFFFFF. The default range for this report is the defined shared area for the system which is dumped.</td>
</tr>
<tr>
<td>REALFRAME</td>
<td>Hexadecimal real frame numbers from 0 to the number of real frames in the system (up to 8 hexadecimal digits).</td>
</tr>
<tr>
<td>VIRTPAGE</td>
<td>Hexadecimal virtual addresses from 0 to 7FFFFFFF.</td>
</tr>
<tr>
<td>SUBSPACE</td>
<td>Hexadecimal virtual addresses from 0 to 7FFFFFFF.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Hexadecimal notation (X'n...') is optional, that is, 7FF as opposed to X'7FF'.</td>
</tr>
</tbody>
</table>

**SAVEAREA(address)**
Requests that the report contain information about the RSM module save area at the specified address. Use SAVEAREA for the EXCEPTION and EXECUTION reports.
RSMDATA subcommand

**SHARED**
Requests that the report contain information about data defined as shared. Use SHARED to select data in the REALFRAME and RSMREQ reports.

**SHORT**
Requests that the report contain abbreviated information that can be obtained quickly. Use SHORT to select data in the ADDRSPACE report.

**STATUS(statuslist)**
Requests that the report include the status of each object.

The *statuslist* is a list of one or more object states, separated by blanks or commas. The following is a list of report parameters and the object states for each report. If you do not specify STATUS, the report will contain information about all possible states for a given object.

- **Object states for ADDRSPACE report:**
  
  **NONSWAP**
  Displays the address spaces that are non-swappable.

  **RESWPIP**
  Displays the address spaces that are in the process of in-real swap (real swap).

  **SWAUX**
  Displays the address spaces that are swapped to auxiliary storage.

  **SWAUXIP**
  Displays the address spaces that are in the process of being swapped to auxiliary storage.

  **SWEXP**
  Displays the address spaces that are swapped to expanded storage.

  **SWEXPIP**
  Displays the address spaces that are in the process of being swapped to expanded storage.

  **SWIN**
  Displays the address spaces that are swapped in.

  **SWINIP**
  Displays the address spaces that are in the process of being swapped in.

  **TERM**
  Displays the address spaces that are in the process of terminating.

- **Object states for DIVMAP report:**
  
  **MAPIP**
  Displays the data-in-virtual mapped ranges that are involved in a DIV MAP request.

  **MAPRPIP**
  Displays the data-in-virtual mapped ranges that are involved in a DIV MAP-reprime request.

  **UNMAPIP**
  Displays the data-in-virtual mapped ranges that are involved in a DIV UNMAP request.
RSMDATA subcommand

SAVEIP
Displays the data-in-virtual mapped ranges that are involved in a
DIV SAVE request

RESETIP
Displays the data-in-virtual mapped ranges that are involved in a
DIV RESET request

MAPPED
Displays the data-in-virtual mapped ranges that are not involved in
a DIV request

Object states for HIGHVIRTUAL report:

AUX
Displays pages that have their most recent copies on a DASD paging
data set or on storage-class memory (SCM).

DASD
Displays pages that have their most recent copies on a DASD paging
data set.

FREF
Displays all 4K and 1M pages that are in first-reference state. That is,
one of the following conditions is true for a given 4K or 1MB page:
- It was never referenced.
- It was released through the PGASER macro.
- It was released through the IARV64 macro.

FRFM
Displays all 4K pages that are in a first-reference state. That is, one of
the following conditions is true for a given 4K page:
- It was never referenced.
- It was released through the IARV64 macro.

GUARD
Displays pages that are in the guard area of a memory object.

HIDE
Displays pages that are hidden.

Note: Hidden pages that are part of globally shared memory objects
may not show up as hidden in this report. Run the
HVSHRDATA report to see the global view of those memory
objects.

REAL
Displays all pages that reside in real storage. They are either valid or
have output paging I/O in progress.

RL_M
Displays 1MB pages that reside in real storage. They are either valid
or have output paging I/O in progress.

RL2G
Displays 2GB pages that reside in real storage.

SCM
Displays pages that have their most recent copies on storage-class
memory (SCM).
RSMDATA subcommand

SCMM
Displays 1MB pages that have their most recent copies on
storage-class memory (SCM).

SIAI
Displays pages that are in the process of being swapped in from
auxiliary storage.

SOAI
Displays pages that are in the process of being swapped out to
auxiliary storage.

SWAX
Displays pages that have their most recent copies swapped to
auxiliary storage.

- Object states for REALFRAME report:

ALLOC
Displays the frames that are allocated.

ALLOC1M
Displays the 1MB pages that are allocated.

ALLOC2G
Displays the 2GB pages that are allocated.

ALLOCSM
Displays only frames backing pages of shared segments.

ALLOCVR
Displays frames allocated to V=R jobs that are either running or
waiting for additional frames.

AVAIL
Displays available frames.

AVAIL1M
Displays the 1MB pages that are not allocated.

AVAIL2G
Displays the 2GB pages that are not allocated.

OFFINT
Displays the frames that will be taken offline when freed from the
current owner.

OFFINTPL
Displays the frames that are offline intercepted and currently in use
by a job that is polluting the V=R area with a long term resident
page.

OFFINTVR
Displays frames that are offline intercepted and allocated to a V=R
job.

OFFLINE
Displays frames that are offline.

POLLUTE
Displays frames that are part of the V=R area, but are allocated to a
long-term resident page that is not V=R.
RSMDATA subcommand

**VRINT**
Displays frames that will be assigned to a waiting V=R job when freed from the current owner.

*Object states for RSMREQ report:*

**CANCEL**
Displays any canceled requests.

**COMPLETE**
Displays non-fast path PGSER FIX requests that have completed and are awaiting the corresponding PGSER FREE request.

**DBLFRAME**
Displays requests that are waiting for a real frame pair.

**FAIL**
Displays requests that had failures other than I/O or cross memory access failures.

**FRAMEAA**
Displays requests that are waiting for any type of real frame.

**FRAMEAB**
Displays requests that are waiting for a real storage frame that resides below 16 megabytes.

**FRAMEPA**
Displays requests that are waiting for a real frame that resides in the preferred area.

**FRAMEPB**
Displays requests that are waiting for a real frame that resides in the preferred area below 16 megabytes.

**INPROGR**
Displays requests that are in progress. These requests may or may not be waiting for a frame or I/O. The presence or absence of other entries in this report for the same request indicates if a wait for a frame or I/O exists.

**IOFAIL**
Displays requests that had I/O failures.

**PGREAD**
Displays requests that are waiting for a page to be read in from a paging data set, or some other data set.

**PGWRITE**
Displays requests that are waiting for a page to be written to a paging data set or some other data set.

**XMFAIL**
Displays requests that had cross memory access errors.

*Object states for SHRDATA report:*

**AUX**
Displays pages that have their most recent copies on a DASD paging data set or in storage-class memory (SCM).

**DASD**
Displays pages that have their most recent copies on a DASD paging data set.
**RSMDATA subcommand**

**DSN**
Displays pages that have their most recent copies on a data set containing the data-in-virtual object of which the pages are a part.

**FREF**
Displays all 4K-pages that were in a first-reference state. That is, one of the following conditions is true for a given page:
- It was never referenced.
- It was released through the PGSER macro.
- It was released through the DSPSERV macro.

**REAL**
Displays all 4K-pages that reside in real storage. They are either valid or have output paging I/O in progress.

**SCM**
Displays 4K-pages that have their most recent copies on storage-class memory (SCM).

- **Object states for SUBSPACE report:**

  **GLOBAL**
  Displays the storage that is addressable by all subspaces within this address space.

  **ASSIGN**
  Displays the storage in this address space that is assigned to subspaces. In the report, the names of the subspaces to which the storage is assigned appear in the SSP NAME column.

  **UNASSIGN**
  Displays the storage in the address space that is not assigned to any subspace.

- **Object states for VIRTPAGE report:**

  **AUX**
  Displays pages that have their most recent copies on a DASD paging data set or in storage-class memory (SCM).

  **DASD**
  Displays pages that have their most recent copies on a DASD paging data set.

  **DSN**
  Displays pages that have their most recent copies on a data set containing the data-in-virtual object of which the pages are a part.

  **FREF**
  Displays all (4K or 1M) pages that were in a first-reference state. That is, one of the following conditions is true for a given page:
  - It was never referenced.
  - It was released through the PGSER macro.
  - It was released through the DSPSERV macro.

  **FRFM**
  Displays 1M pages that were in a first-reference state. That is, one of the following conditions is true for a given 1M page:
  - It was never referenced.
  - It was released through the PGSER macro.
RSMDATA subcommand

MIG
Displays pages for which both of the following conditions are true:
- The most recent copies are migrated to auxiliary storage from expanded storage.
- The most recent copies reside in incorrect segments.

REAL
Displays all (4K or 1M) pages that reside in real storage. They are either valid or have output paging I/O in progress.

RL_M
Displays 1M pages that reside in real storage. They are either valid or have output paging I/O in progress.

SCM
Displays pages that have their most recent copies on storage-class memory (SCM).

SCMM
Displays 1M pages that have their most recent copies on storage-class memory (SCM).

SMEG
Displays pages that are part of a shared segment.

VIO
Displays pages that have their most recent copies on a VIO data set.

Note: All of the following swap states apply only to working set pages.

SIAI
Displays pages that are in the process of being swapped in from auxiliary storage.

SIEI
Displays pages that are in the process of being swapped in from expanded storage.

SOAI
Displays pages that are in the process of being swapped out to auxiliary storage.

SOEI
Displays pages that are in the process of being swapped out to expanded storage.

SWAX
Displays pages that have their most recent copies swapped to auxiliary storage.

SWEX
Displays pages that have their most recent copies swapped to expanded storage.

SWMG
Displays pages that are in the process of migrating from expanded storage to auxiliary storage.

TOKEN(token)
Requests that the SHRDATA report be run only for the input token.

Usage note: The system ignores all other data selection parameters when you specify TOKEN.
RSMDATA subcommand

TOTONLY
Requests that for tabular reports, only the totals should be produced. All other output is suppressed. If you do not specify TOTONLY, RSMDATA prints all report data. Use TOTONLY for the ADDRSPEC, DIVMAP, DSPACE, REALFRAME, RSMREQ, SHRDATA, and VIRTPAGE tabular reports.

Address space selection parameters
Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs). Use these parameters for ADDRSPEC, DIVMAP, DSPACE, EXCEPTION, REALFRAME, RSMREQ, SHRDATA, SUBSPACE, and VIRTPAGE reports. In these reports, if you omit an address space selection parameter, the defaults are CURRENT and ERROR. For more information, see the select ASID service in z/OS MVS IPCS Customization.

ALL
Specifies processing of RSM control blocks for all address spaces in the system at the time the dump is generated.

ASIDLIST(asidlist)
Specifies the list of address space identifiers for which you want to process RSM control blocks.

The asidlist can be specified as a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed in the notation X'nnn', F'nnn', or B'nnn'. An unqualified number is assumed to be fixed.

CURRENT
Specifies processing of RSM control blocks for each active address space (that is, address spaces dispatched on some central processor, or bound by cross memory to an address space dispatched on some central processor) at the time of the dump.

ERROR
Specifies processing of RSM control blocks for the error address space(s).

JOBLIST(joblist) or JOBNAME(joblist)
Specifies the list of job names whose associated address spaces are to be processed for RSM control blocks. Use commas to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the RSMDATA subcommand.

Examples
See z/OS MVS Diagnosis: Reference for detailed descriptions and examples of RSMDATA output.

When viewing RSMDATA output through the IPCS dialog, you can enter the HELP primary command (or PF key). Choosing option 6 from the HELP selection panel will display full help text on the contents of the RSMDATA report.

Example 1
RSMDATA subcommand

Generate a report on virtual pages, including data space pages, residing on expanded storage for job MYJOB.

Action

COMMAND ==> RSMDATA VIRTPAGE JOBNAME(MYJOB) DATASPACES STATUS(AUX)

Example 2

Generate a report showing all real frames (not just CURRENT and ERROR) in the V=R region that are intercepted for use by a V=R job, or are polluting the V=R region.

Action

COMMAND ==> RSMDATA REALFRAME ALL STATUS(VRINT,POLLUTE) RANGE(5:86)

Notes:
1. Determine the range of the V=R region using RSMDATA SUMMARY.
2. In this case, specify ALL to override the default CURRENT address space selection parameters, so that the report will contain all the real frames that satisfy the selection criteria.

Example 3

Generate a report showing all RSM requests for the CURRENT address space.

Action

COMMAND ==> RSMDATA RSMREQ

Example 4

Generate a report showing real storage usage summary for every address space in the dump.

Action

COMMAND ==> RSMDATA ADDRSPACE ALL

Example 5

Generate a report showing the storage in address space X'023' that is assigned to a subspace, not assigned to a subspace, or available to all subspaces.

Action

COMMAND ==> RSMDATA SUBSPACE STATUS(GLOBAL,ASSIGN,UNASSIGN) ASIDLIST(X'023')

RUNARRAY subcommand — process an array of control blocks

Use the RUNARRAY subcommand to process an array of control blocks. You can specify the order that subscripts should be processed.

RUNARRAY optionally displays each control block.

You can specify additional subcommand, CLIST, or REXX exec processing with the EXEC parameter. For each entry in the array, RUNARRAY will display the storage, set the value of X to describe the entry, and then process the EXEC parameter for that entry.

Related subcommands
RUNCHAIN
RUNCPPOOL

Syntax
RUNARRAY subcommand

RUNARRAY

[ data-descr | ADDRESS(X) ]
[ ASCENDING | DESCENDING ]
[ EXEC((clist|rexx-exec|subcommand)) ]
[ SUMMARY | NOSUMMARY ]

-------- SETDEF-Defined Parameters  -------------------------
Note: You can override the following SETDEF parameters.

[ DISPLAY[[display-options]] ]
[ NODISPLAY[[display-options]] ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]

Parameters

**data-descr**

**ADDRESS(X)**

Specifies the data description parameter, which consists of five parts:
- An address (required when data-descr is explicitly specified on the subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

[Chapter 3, “Data description parameter,” on page 3-1](#) explains the use and syntax of the data description parameter. However, the following applies to RUNARRAY only:
- The address is **not** a positional parameter. You must use the ADDRESS parameter to specify an address.
- If you omit the ADDRESS parameter, the default for the RUNARRAY subcommand is ADDRESS(X), the most recently accessed address.
- If you describe a block that is not an array, RUNARRAY treats it as an array containing one entry, ENTRY(1).

**ASCENDING**

**DESCENDING**

Specifies the order in which subscripts are to be processed.

**EXEC((clist))**

**EXEC((rexx-exec))**

**EXEC((subcommand))**

Specifies that a CLIST, a REXX exec, or an IPCS subcommand is to be appended to the RUNARRAY subcommand invocation. The appended CLIST, REXX exec, or subcommand runs for each control block in the chain. Parameters or keywords can accompany the CLIST, REXX exec, or IPCS subcommand. The symbol X will point to the current array entry before each EXEC invocation.
**RUNARRAY subcommand**

The RUNARRAY subcommand generates a return code that consists of its own return code plus the return code from the CLIST, REXX exec, or IPCS subcommand designated on the EXEC parameter. If the CLIST, REXX exec, or IPCS subcommand returns with a serious condition, RUNARRAY processing ends with the current array entry.

**SUMMARY**

**NOSUMMARY**

Controls the formatting of a processing summary after normal completion of RUNARRAY processing. A processing summary is always produced if abnormal conditions force termination of RUNARRAY.

**Return codes**

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the RUNARRAY subcommand.

The RUNARRAY subcommand generates a return code that consists of its own return code plus the return code from a CLIST, REXX exec, or IPCS subcommand if designated by the EXEC parameter. If the CLIST, REXX exec, or IPCS subcommand returns with a serious condition, RUNARRAY processing ends with the current control block.

**RUNCHAIN subcommand — process a chain of control blocks**

Use the RUNCHAIN subcommand to process a chain of control blocks. You can specify the links to follow and a mask to apply to the links. You can also limit the length of the chain to prevent infinite loops. With z/OS Release 3 and higher, you can also specify that attributes and data within a chain of data areas is to determine their order of processing by the RUNCHAIN subcommand.

RUNCHAIN displays each control block and creates entries for each control block in the symbol table that is part of the source description for your current source. You can specify a control block name for each symbol.

You can specify additional subcommand, CLIST, or REXX exec processing with the EXEC parameter. For each control block in the chain, RUNCHAIN will display the storage, set the value of X to the address of the control block, and then process the EXEC parameter for that control block.

You can also process multiple levels of control block chains by specifying another RUNCHAIN subcommand on the EXEC parameter.

**Related subcommands**

- DROPSYM
- EQUATE
- LISTSYM
- RUNCPOOL
- RUNARRAY

**Syntax**
RUNCHAIN subcommand

{ RUNCHAIN | RUNC }

[ data-descr | ADDRESS(X) ]
[ AMASK(mask) ]
[ CHAIN [nnn|999] ]
[ DROP | NODROP ]
[ EXEC((clist|rexx-exec|subcommand)) ]
[ LINK(range[LENGTH(integer)]) ] [ NAME(prefix) ]
[ NULL [(value|0)] ]
[ SORTBY(sort-key [ ASCENDING | DESCENDING ] ...) ]

-------- SETDEF-Defined Parameters ---------------------------------
Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232

[ DISPLAY[(display-options)] ]
[ NODISPLAY[(display-options)] ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]

Parameters

data-descr or ADDRESS(X)
  Specifies the data description parameter, which consists of five parts:
  – An address (required when data-descr is explicitly specified on the
    subcommand)
  – Address processing parameters (optional)
  – An attribute parameter (optional)
  – Array parameters (optional)
  – A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and
syntax of the data description parameter. However, the following exceptions
apply to RUNCHAIN only:
  – The address is not a positional parameter. You must use the ADDRESS
    parameter to specify an address.
  – If you omit the ADDRESS parameter, the default for the RUNCHAIN
    subcommand is ADDRESS(X), the most recently accessed address.

AMASK(mask)
  Specifies an unsigned integer mask that RUNCHAIN is to AND to the link
  field before using that field as the address of the next block in the chain.
RUNCHAIN subcommand

IPCS accepts 64-bit values and interprets all values entered as having 64-bit precision. If the chain originates below \(2^{24}\), the default is X'00FFFFFF'. If the chain originates above \(2^{24}\), the default is X'7FFFFFFF'. If the chain originates above the bar, the default is X'FFFFFFFF_FFFFFFFF'.

**CHAIN\[(nnn|999)\]**

Specifies the maximum number of blocks the subcommand is to process. The number can be a maximum of 16,777,215 and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

If you omit this parameter, the default is CHAIN(999).

**DROP or NODROP**

Specifies the DROP or NODROP attribute for the names RUNCHAIN places in the symbol table. RUNCHAIN places the names of the control blocks it finds in the symbol table when you specify the NAME parameter.

DROP specifies the DROP attribute. This attribute allows the symbols to be deleted from the symbol table by a DROPSYM subcommand.

NODROP specifies the NODROP attribute. This attribute prevents the symbols from being deleted from the symbol table by a DROPSYM subcommand, unless DROPSYM contains a PURGE parameter.

**EXEC((clist|rexx-exec|subcommand))**

Specifies that a CLIST, a REXX exec, or an IPCS subcommand is to be appended to the RUNCHAIN subcommand invocation. The appended CLIST, REXX exec, or subcommand runs for each control block in the chain. Parameters or keywords can accompany the CLIST, REXX exec, or IPCS subcommand. The symbol X will point to the current control block on the chain before each EXEC invocation.

The EXEC parameter also accepts another RUNCHAIN invocation to process multiple levels of control blocks. See the BLSRNCC2 CLIST in SYS1.SBLSCLI0 for an example.

The RUNCHAIN subcommand generates a return code that consists of its own return code plus the return code from the CLIST, REXX exec, or IPCS subcommand designated on the EXEC parameter. If the CLIST, REXX exec, or IPCS subcommand returns with a serious condition, RUNCHAIN processing ends with the current control block.

**LINK(range[LENGTH(integer)])**

Defines a range of offsets that contain a 1-8 byte pointer from one block in the chain to the next.

- LINK(0:3) 4-byte pointer at the origin of the block
- LINK(8:15) 8-byte pointer at displacement 8 in the block
- LINK(8:4) Error. Descending range

Range consists of one or two an unsigned integers. The end of the range may be omitted or can be designated using LENGTH(integer). For compatibility with earlier releases, RUNCHAIN treats this as a description of a 4-byte pointer.

The link pointer is always extended to 8-bytes before masking, nullity checking, and use for access to the next block on the chain.

If you omit this parameter, the default is LINK(0).

**MASK\[(mask)\]**

Specifies an unsigned integer mask that RUNCHAIN is to AND to the link
RUNCHAIN subcommand

field before comparing it to the value specified with the NULL parameter. IPCS accepts 64-bit values and interprets all values entered as having 64-bit precision.

The length of the mask must be eight bytes. If it is less than eight bytes, the subcommand right-justifies it and pads it on the left with zeros. If it exceeds eight bytes, the subcommand rejects it.

You can specify the mask in decimal, hexadecimal (X‘xxx...’), or binary (B‘bbb...’). If you specify it in decimal or binary, the value is converted to its hexadecimal equivalent and padded if needed.

If you omit this parameter, the default for all chains is 
MASKX’FFFFFFFF_FFFFFFFF’.

NAME(prefix)

Specifies the prefix RUNCHAIN uses to generate names for each control block it finds. The subcommand places the generated names in the symbol table. The generated name can be 1 to 31 alphanumeric characters and the first character must be a letter or the characters “$”, “@”, or “#”.

RUNCHAIN appends a sequence number to the prefix to produce a unique control block name. The sequence number starts at 1 and is limited by the value specified with the CHAIN parameter.

The prefix for any control block may not exceed 30 characters.

If you omit this parameter, RUNCHAIN does not generate names for the control blocks it finds.

NULL[(value|0)]

Specifies the unsigned integer doubleword value that indicates the end of the chain. IPCS accepts 64-bit values and interprets all values entered as having 64-bit precision.

For each control block on the chain, RUNCHAIN:
- Locates the link field at the offset specified in the LINK parameter.
- ANDs the mask with the contents of the link field.
- Compares the result of the AND with the NULL value.
- When the result of the comparison is equal, chaining ends.
- When the result of the comparison is not equal, chaining continues.

SORTBY(sort-key [ASCENDING|DESCENDING]...]
Controls the order of processing for chain elements.

sort-by
A list of sort-keys directs RUNCHAIN to make two passes over the chain. The first pass internally enumerates the blocks on the chain and collects up to 256 bytes of aggregate sort key data.

If any data described as a sort key cannot be retrieved, the chain is logically terminated at the preceding block during the first pass.

Each sort-key may be designated in one of the following ways:

signed-integer[:signed-integer]
Designates a range of offsets from the origin of the block. A string or unsigned binary number at those locations is used as a sort key.
If the end of the range is not specified, four bytes are selected.

ADDRESS
DIMENSION
LENGTH
RUNCHAIN subcommand

MULTIPLE
These keywords designate an unsigned attribute of the block. Each of these attributes uses 8 bytes of the 256 available.

ENTRY
POSITION
These keywords designate a signed attribute of the block. A signed comparison between these attributes is performed. Each of these attributes uses 8 bytes of the 256 available.

DATATYPE
The DATATYPE keyword designates the type of block, for example, STRUCTURE(UCBDASD) versus STRUCTURE(UCBTape). Each of these attributes uses 34 bytes (see Data Area BLSRDATT) of the 256 available.

ASCENDING
DESCENDING
These keywords designate the sort order for the preceding key. Ascending sort order is the default.

Return codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the RUNCHAIN subcommand.

The RUNCHAIN subcommand generates a return code that consists of its own return code plus the return code from a CLIST, REXX exec, or IPCS subcommand if designated by the EXEC parameter. If the CLIST, REXX exec, or IPCS subcommand returns with a serious condition, RUNCHAIN processing ends with the current control block.

Example
The BLSCRNCH CLIST runs the chain of task control blocks (TCB) for an address space. It displays the following information:
- The current TCB
- The TCBs that are lower on the priority chain in that address space
- The currently dispatched RB for each of the TCBs

This CLIST, written for SVC dumps, uses the RUNCHAIN subcommand as follows:

PROC 0 TCB(21C.%)
RUNCHAIN ADDRESS(&TCB) STRUCTURE(TCB) /* Process TCBs */+
LINK(X'74') /* Connected by field TCBTCB */+
VERIFY DISPLAY /* Maximum display for each TCB */+
EXEC((LIST X+0% STRUCTURE(RB) DISPLAY))/* Show RB for TCB*/

The logic of this CLIST is as follows:

PROC 0 TCB(21C.%)
This line indicates that the default path to the first TCB is the fullword pointer at location X'21C'.

RUNCHAIN ADDRESS(&TCB) STRUCTURE(TCB)
This line processes the first TCB that can be found by using the default path or an alternate path to a TCB, described when the CLIST is invoked. IPCS validates the TCB and creates a storage map entry for it. The STRUCTURE attribute parameter identifies that a TCB is being processed.

Note: If SDUMP writes the dump, IPCS does not require address processing parameters. IPCS establishes the dumped ASID as the default address space.
RUNCHAIN subcommand

LINK(X'74')
This line establishes addressability to the TCBTCB field at offset X'74' for each TCB, thereby providing the address of the next TCB on the chain to be processed.

VERIFY DISPLAY
This line lists all TCBs found on the chain and displays the maximum amount of information for each TCB.

The VERIFY and DISPLAY parameters each override the defaults established by the SETDEF subcommand for the corresponding parameter.

EXEC((LIST X+0% STRUCTURE(RB) DISPLAY))
This line updates the current TCB that is currently being processed, establishes addressability to the TCBRBP field at offset X'0' within the current TCB, and accesses the RB related to the current TCB.

RUNCPOOL subcommand — process a CPOOL

Use the RUNCPOOL subcommand to process a cell pool created and managed by the CPOOL macro. Cells are partitioned into the following categories:

- Used cells are those that contained current data when a dump was produced.
- Available cells are those that were not currently in use when a dump was produced. CPOOL services use the first four bytes in each such cell, but residual data useful for analysis may remain in the other part of such a cell.
- Indeterminate cells are those that IPCS cannot place in either of the preceding categories.

The most common reason for this is that the pool was actively being changed during the dumping process, producing a “blurred picture” of this part of the dumped system. Storage overlays and storage missing from a dump may also produce indeterminate cells.

You can specify which categories of cells should be processed.

Establishing categories of cells is done before processing the cells themselves, and an optional report may be formatted that identifies data areas used to manage the cell and data extracted from those data areas.

RUNCPOOL optionally displays each cell.

You can specify additional subcommand, CLIST, or REXX exec processing with the EXEC parameter. For each cell, RUNCPOOL will display the storage, set the value of X to the address of the cell, and then process the EXEC parameter for that cell.

Related subcommands
RUNARRAY
RUNCHAIN

Syntax
RUNCPOOL subcommand

RUNCPOOL cpid-general-value
[ ASID(asid) ]
[ DATABLKS | NODATABLKS ]
[ USED | NOUSED ]
[ INDETERMINATE | NOINDETERMINATE ]
[ AVAILABLE | NOAVAILABLE ]
[ EXEC((clist|rexx-exec|subcommand)) ]
[ SUMMARY | NOSUMMARY ]

---- SETDEF-Defined Parameters  ---------------
Note: You can override the following SETDEF parameters.
[ ACTIVE | MAIN | STORAGE ]
[ DSNNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ DISPLAY((display-options)) ]
[ NODisplay((display-options)) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]

Parameters

cpid-general-value
Specifications a fullword cell pool identifier (CPID used in conjunction with the
CPOOL macro).

ASID(asid)
Specifications the ASID of a CPOOL in private storage as a positive integer.
This may be omitted if the default IPCS address processing parameters
specify an ASID.

DATABLKS
NODATABLKS
Controls the formatting of a report that identifies data areas used to control the
cell pool and extracts information from them regarding the status of the
cell pool.

USED
NOUSED
Specifications whether cells in the pool that are in use are to be included in
RUNCPOOL processing.

INDETERMINATE
RUNCPOOL subcommand

NOINDETERMINATE
Specifies whether cells known to be in the pool but whose status as used or available cannot be determined are to be included in RUNCPOOL processing.

AVAILABLE
NOAVAILABLE
Specifies whether cells in the pool that are available are to be included in RUNCPOOL processing.

EXEC(,clist))
EXEC((rexx-exec))
EXEC((subcommand))
Specifies that a CLIST, a REXX exec, or an IPCS subcommand is to be appended to the RUNCPOOL subcommand invocation. The appended CLIST, REXX exec, or subcommand runs for each control block in the chain. Parameters or keywords can accompany the CLIST, REXX exec, or IPCS subcommand. The symbol X will point to the current cell on the chain before each EXEC invocation.

The RUNCPOOL subcommand generates a return code that consists of its own return code plus the return code from the CLIST, REXX exec, or IPCS subcommand designated on the EXEC parameter. If the CLIST, REXX exec, or IPCS subcommand returns with a serious condition, RUNCPOOL processing ends with the current control block.

SUMMARY
NOSUMMARY
Controls the formatting of a processing summary after normal completion of RUNCPOOL processing. A processing summary is always produced if abnormal conditions force termination of RUNCPOOL.

Return codes
The RUNCPOOL subcommand generates a return code that consists of its own return code plus the return code from a CLIST, REXX exec, or IPCS subcommand if designated by the EXEC parameter. If the CLIST, REXX exec, or IPCS subcommand returns with a serious condition, RUNCPOOL processing ends with the current control block.

Examples

Example 1 - Small private area CPOOL
Example 1 shows a small private area CPOOL in which all of the cells are currently unused.

runpool x'0F188300'
PPD at 7F7E8F88
  ASID(X'036E') CPID(X'0F188300') in loc(any,any) subpool(78)
  Csize(3,072) primary(5) secondary(40)
PXT at 0F188300
SPD at 7F7E8FC0
Cells(5) used(0)
IGV18094I No cells processed

Example 2 - Larger private area CPOOL
Example 2 shows (part of) a larger private area subpool, one that has expanded into a secondary extent. Slightly more than half of the cells are currently in use and are displayed.
Example 3 - Common area CPOOL

Example 3 shows a summary of a common area CPOOL.

```
runcpool a'2D37000'
```

```
PDP at 02EBF068
  CPID('X'02D37000') in loc(any) subpool(248)
  Csize(32,640) primary(1) secondary(1)
PXT at 02D37000

SPD at 02Ebf0A0
SXT at 0412B000
SXT at 045B2000
SXT at 049CF000
SXT at 0273B000

Cells(5) used(0)

IGV180941 126 cells processed
```

**SCAN subcommand — validate system data areas**

Use the SCAN subcommand to validate system data and make storage map entries for that data. “Control blocks and data areas scanned, mapped, and formatted” on page C-1 lists the data areas that IPCS scans, maps, and formats.

SCAN validates a control block by checking:

- Boundary alignment. (Certain control blocks must begin on word, doubleword, or other special boundaries.)
SCAN subcommand

- Standard fields in the control block, such as:
  - Acronyms
  - Count fields
  - Masks or bit maps
- Pointers that address other system data

SCAN initiates its processing from your storage map and validates control blocks listed in the storage map that are within the address range you specified. As it does this, SCAN makes new map entries for control blocks pointed to by the block being validated. Depending on the DEPTH and PASSES parameters, new entries (control blocks) in the map may or may not be validated; however, if the new control blocks are found to be not valid, their entries remain in the map.

The process of validating one control block and following its pointers to other control blocks to the indicated depth is called a scan probe. If you specify a large number for DEPTH, the scan probe of one control block can add many entries to the map. If this control block is the CVT or an ASCB, one scan probe can map all the AREAs and STRUCTUREs in the dump. Dump initialization provides entries in the map for the current dump. SCAN requires at least one entry to begin its processing.

If a control block does not appear valid, IPCS issues a message that gives the control block name, its address, and the apparent error; the control block's entry remains in the storage map.

If SCAN, in validating a control block, follows a pointer to a new control block, and finds that the new control block is not valid, IPCS issues two messages. The first message has a severity level of ERROR to inform you that the original control block contains a bad pointer. The second message has a severity level of SEVERE to inform you that the (alleged) new control block is not valid.

Syntax

```
SCAN [ limit|100 ]
    [ RANGE(address:address) ] [data-descr]
    [ DEPTH(n|2) ]
    [ PASSES(n|1) ]
    [ SUMMARY | NOSUMMARY ]
```

--------- SETDEF-Defined Parameters ---------------------

Note: You can override the following SETDEF parameters.

See

“SETDEF subcommand — set defaults” on page 5-232

```
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
```

Parameters

limit

Specifies the maximum number of scan probes that SCAN is to perform.
SCAN subcommand

The limit can range from 1 through $2^{31}$ and can be specified in decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’).

This parameter, if specified, it must precede any parameters.

If you omit this parameter, the default is 100.

RANGE(address:address)

Specifies the range of addresses, the types of entries, or both, in the storage map from which SCAN is to perform scan probes. When validating a control block, SCAN may access other control blocks outside the specified range. The RANGE parameter specifies the addresses from which the SCAN probes start. When the RANGE parameter is omitted, SCAN validates all control blocks that have not been validated.

data-descr

Specifies the data description parameter, which consists of five parts:
- An address (required with the RANGE parameter and when data-descr is explicitly specified on the subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

If you specify the STRUCTURE attribute parameter with a data type, it causes the subcommand to create a map record. This new map record does not otherwise change the results of this subcommand.

If you omit this parameter, SCAN validates all storage map entries not previously validated. A control block may be only partially validated because of limits on DEPTH and PASSES on previous scans.

DEPTH(n|2)

Specifies the maximum level of indirection for each scan probe. For example, the new control blocks that a given control block points to are at depth 1. The control blocks that the new control blocks point to are at depth 2, and so on.

The $n$ can be 1 through 65535. The number can be specified in decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’). An unqualified number is decimal.

If you omit this parameter, the default is DEPTH(2).

PASSES(n|1)

Specifies the number of times SCAN processes the storage map entries in the specified address range. As SCAN reprocesses the storage map, it does not revalidate control blocks previously validated.

The $n$ can be 1 through $2^{31}$. The number can be specified in decimal, hexadecimal (X’xxx...’), or binary (B’bbb...’). An unqualified number is decimal.

If you omit this parameter, the default is PASSES(1).

SUMMARY or NOSUMMARY

SUMMARY indicates that a processing summary (a final total line) is to be produced.
SCAN subcommand

NOSUMMARY specifies that a processing summary is to be suppressed. The NOSUMMARY parameter is useful to turn off summary messages when the subcommand is invoked within a CLIST or a REXX exec.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the SCAN subcommand.

SELECT subcommand — generate address space storage map entries

Use the SELECT subcommand to:

- Create storage map entries that describe address spaces. Storage map entries include the address space address, address space identifier (ASID), length, and AREA data type.
- Produce a report that displays the ASID, associated job name, ASCB address, and selection criteria for each address space selected.

The storage map is part of a source description. A source description is for an unformatted source that IPCS can format, for example, an SVC dump, a stand-alone dump, an SYSMDUMP dump, a trace data set, a data set, or active storage. The source description is in the dump directory allocated with ddname IPCSDDIR and is your current dump directory. The current dump directory is your user dump directory or, for users with write access authority, might be the sysplex dump directory.

Related subcommands
EVALMAP
LISTMAP
LIST
SUMMARY

Syntax
SELECT subcommand

SELECT [ LIST | NOLIST ]

-------- Address Space Selection Parameters --------------
[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR|ANOMALY ]
[ ASIDLIST(asidlist) ]
[ JOBLIST(joblist)|JOBNAME(joblist) ]

-------- SETDEF-Defined Parameters ----------------------
Note: You can override the following SETDEF parameters.

See "SETDEF subcommand — set defaults" on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

LIST or NOLIST
 specifies if IPCS should generate a report.
 LIST specifies a report.
 NOLIST specifies no report. NOLIST is provided mainly for CLIST processing, for example, when a CLIST might want to generate a storage map entry without creating a report. When NOLIST is specified, NOPRINT and NOTERM are assumed.

Address Space Selection Parameters
 Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs). If you omit these parameters, the defaults is CURRENT.
 These parameters also control the name portion for the AREA attribute of the storage map entries. (For a refresher on the AREA attribute parameter, see "ATTRIBUTE parameters" on page 3-13.) This table shows what to specify for name:

<table>
<thead>
<tr>
<th>When You Specify This Address Space Parameter</th>
<th>You Get This AREA(name) Storage Map Entry Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT</td>
<td>AREA(CURRENT)</td>
</tr>
<tr>
<td>ERROR</td>
<td>AREA(ERROR)</td>
</tr>
<tr>
<td>TCBERROR</td>
<td>AREA(TCBERROR)</td>
</tr>
<tr>
<td>JOBLIST</td>
<td>AREA(JOBxxxx)</td>
</tr>
</tbody>
</table>
**SELECT subcommand**

Notes:
1. Storage map entries are created when you specify the CURRENT, ERROR, TCBERROR, and JOBNAME/JOBLIST address space selection parameters.
2. For an address space to be mapped when you select it with JOBLIST, it must have a standard alphanumeric job name.
3. When you use JOBLIST to select the master scheduler address ("MASTER") space, IPCS maps it with an AREA name of JOBMASTER.

For more information, see the select ASID service in [z/OS MVS IPCS Customization](#).

**ALL**
- Specifies processing of all address spaces in the dump.

**CURRENT**
- Specifies the processing of each address space that is active when the dump is generated.

**ERROR**
- Specifies processing of control blocks for any address space with an MVS error indicator or containing a task with an error indicator.

**TCBERROR or ANOMALY**
- Specifies processing of control blocks for any address space containing a task with an error indicator. Blocks for address spaces with an error indicator are not processed.

**ASIDLIST(asidlist)**
- Specifies a list of ASIDs for the address spaces to be processed.
  - The *asidlist* can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.
  - The ASID can be 1 through 65535. An ASID can be expressed in the notation X'nnn', F'nnn', or BB'nnn'. An unqualified number is assumed to be fixed.

**JOBLIST(joblist) or JOBNAME(joblist)**
- Specifies a list of job names whose associated address spaces are to be processed. Use commas to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

**SETDEF-Defined Parameters**

**ACTIVE or MAIN or STORAGE**

**DATASET(dsname) or DSNAME(dsname)**

**FILE(ddname) or DDNAME(ddname)**
- Specifies the source of the source description containing the storage map. If one of these parameters is not specified, the source is your current source.
  - ACTIVE, MAIN, or STORAGE specifies central storage as the source. When active storage is specified, the SELECT subcommand can process only current address spaces.
  - DSNAME or DATASET specifies the name of a cataloged data set as the source.
  - FILE or DDNAME specifies the ddname for a data set as the source.

**Return Codes**
SELECT subcommand

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the SELECT subcommand.

Example

Generate a report containing information for the current, error, master scheduler, and JES3 address spaces.

- Action

  COMMAND ===> SELECT CURRENT ERROR JOBLIST(*MASTER* JES3) LIST

- Result

  SELECT produces this output:

<table>
<thead>
<tr>
<th>ASID</th>
<th>JOBNAME</th>
<th>ASCBADDR</th>
<th>SELECTION</th>
<th>CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td><em>MASTER</em></td>
<td>00123456</td>
<td>CURRENT</td>
<td>JOBNAME</td>
</tr>
<tr>
<td>0010</td>
<td>JES3</td>
<td>00234567</td>
<td>JOBNAME</td>
<td>CURRENT</td>
</tr>
<tr>
<td>01BB</td>
<td>USERJOB</td>
<td>00789ABC</td>
<td>ERROR</td>
<td>Current</td>
</tr>
</tbody>
</table>

  It also generates these storage map entries describing the selected address spaces:

  LIST 00001000. ASID(X'0001') LENGTH(11530240) AREA(CURRENT)
  LIST 00001000. ASID(X'0001') LENGTH(11530240) AREA(JOBMASTER)
  LIST 00001000. ASID(X'0010') LENGTH(11530240) AREA(CURRENT)
  LIST 00001000. ASID(X'0010') LENGTH(11530240) AREA(JOBJES3)
  LIST 00001000. ASID(X'01BB') LENGTH(11530240) AREA(ERROR)

  You can access these entries with the EVALMAP subcommand.

SETDEF subcommand — set defaults

Use the SETDEF subcommand to set, change, and display your default values for certain parameters on IPCS subcommands. You can run SETDEF at any time during an IPCS session to display your default values.

To set or change the value for a default, enter a SETDEF subcommand with the parameter and its new value. IPCS uses the new default value for both your current session and any subsequent sessions in which you use the same user dump directory, until you change the value. SETDEF sets two types of default values:

- Local defaults. These values are currently in use for an ISPF screen in the IPCS dialog, for a batch IPCS session, or for an IPCS interactive line-mode session.
- Global defaults. These values are used to establish the local defaults when IPCS processing starts in an ISPF screen, a batch IPCS session, or an IPCS interactive line-mode session.

Your global defaults are obtained from the dump directory being used. IPCS uses the global defaults the following, in this order:

1. The last value specified as a global default in a SETDEF subcommand or on the IPCS Default Values panel in the IPCS dialog.
2. The value in the IPCSPRxx parmlib member
3. The IBM-supplied value

The IBM-supplied values for global SETDEF-defined defaults are:
ASID and CPU, the address processing parameters, are not listed and are null until you specify a source data set or storage. SETDEF rejects any attempt to set these values before you specify a source. When you specify a source and access it with any of the analysis subcommands, that subcommand sets your local default address processing value to describe an address space contained in that data set or storage.

When you specify a source data set or storage on a SETDEF subcommand, your next analysis subcommand causes IPCS to initialize the specified source data set or storage.

If all parameters on a SETDEF subcommand are valid, IPCS sets the specified values. However, if IPCS rejects any parameter, the subcommand ends without IPCS changing any values.

Many subcommands can override a current local default by specifying a SETDEF parameter and value. For each subcommand, the SETDEF-defined parameters are grouped in the syntax diagram, thereby identifying the SETDEF-defined parameters that apply specifically to the subcommand. These overriding values apply only to the subcommand, are not saved in your user dump directory, and are not retrieved by an EVALDEF subcommand.

**Syntax**
SETDEF subcommand

```
{SETDEF } [ LIST | NOLIST ]
{SETD } [ LOCAL ]
   [ GLOBAL ]

-------- SETDEF-Defined Parameters ----------------------------
[ address-processing-parameters ]
[ ACTIVE | MAIN | STORAGE ]
[ DSNAM(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ NODSNAM | NODATASET ]
[ PATH(path-name) ]
[ CONFIRM | NOCONFIRM ]
[ DISPLAY[(display-options)] ]
[ NODISPLAY[(display-options)] ]
[ FLAG(severity) ]
[ LENGTH(length) ]
[ PRINT | NOPRINT ]
[ PDS | NOPDS ]
[ TERMINAL | NOTERMINAL ]
   [ TEST | NOTEST ]
[ VERIFY | NOVERIFY ]
```

Parameters

**LIST or NOLIST**

Specifies whether IPCS is to display all of your local and global default values.

LIST requests IPCS to display the values at your terminal, regardless of the current value for the TERMINAL parameter.

NOLIST specifies that IPCS is not to display the values.

If you enter SETDEF without any parameters, the default is LIST. If you omit LIST and NOLIST but specify any other parameter, the default is NOLIST.

**LOCAL**

Specifies local default values:
- If LIST is also specified, IPCS lists your local default values.
- If LIST is not also specified, IPCS changes any local default to the value specified on this SETDEF subcommand. Your global default values are not changed.

**GLOBAL**

Specifies global default values:
- If LIST is also specified, IPCS lists your global default values.
- If LIST is not also specified, IPCS changes any global default to the value specified on this SETDEF subcommand. Your local default values are not changed; also, these new global values do **not** override any local default values currently being used.
If you omit or specify both LOCAL and GLOBAL, IPCS lists or changes both local and global default values.

**SETDEF-Defined Parameters**

Default values for the following parameters are defined and shipped with IPCS. Your default values are kept in your dump directory. To change your defaults, enter a SETDEF subcommand with your own values for the parameters.

**address-processing-parameter**

Specifies address processing values, which are a part of the data description (data-descr) parameter. “Address processing parameters” on page 3-7 explains how to specify address processing parameters. Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

ASID(X'0000') and CPU(0) are the IPCS-defined defaults.

**CONFIRM or NOCONFIRM**

Specifies if certain subcommands are to request confirmation before performing their function.

CONFIRM requests your confirmation before:
- Deleting a problem
- Dissociating and scratching a data set
- Modifying a data set’s attributes, if the data set is associated with more than one problem
- Accessing summary dump data during dump initialization

The subcommands affected by CONFIRM are:
- Any subcommand that starts initializing a dump that contains summary dump data.

NOCONFIRM does not request your confirmation before running these subcommands. When NOCONFIRM is specified, IPCS uses summary dump data.

CONFIRM is the IPCS-defined default.

**ACTIVE or MAIN or STORAGE**

**DSNAME(dsname) or DATASET(dsname)**

**FILE(ddname) or DDNAME(ddname)**

**NODATASET or NODSNAME**

Specifies the source. If one of these parameters is not specified, the IPCS-defined default is NODSNAME.

**ACTIVE, MAIN, or STORAGE** specifies the central storage for the address space in which IPCS is currently running and allows you to access that active storage as the dump source. You can access private storage and any common storage accessible by an unauthorized program.

You might use one of these parameters to, for instance:
- Display individual control blocks and examine how they are chained within the executing IPCS address space
- Compare system control blocks (such as the CVT) that were formatted in a dump data set with system control blocks that are currently being used in the IPCS address space
- Examine a field in the read-only nucleus that does not appear in a dump report
- Diagnose an error in IPCS processing

Chapter 5. IPCS subcommands 5-235
SETDEF subcommand

You should not use these parameters for:
- Volatile common or private storage
- Prefixed storage

If IPCS is running as an MVS system migration aid, IPCS rejects these parameters.

IPCS does not create a storage map when this parameter is entered. IPCS does maintain a symbol table but limits its automatic creation of symbols into the table.

**DSNAME or DATASET** specifies the source with the name of a cataloged data set. If the data set is password protected, also specify the password. If you omit the password and it is required, IPCS prompts you for it.

IPCS dynamically allocates and opens the data set when it is first accessed. When an IPCS session completes, IPCS dynamically closes and releases the data set, restoring the data set to its status before being accessed.

**FILE or DDNAME** specifies the source with the ddname of a data set. The data set can reside on tape or a direct access storage device (DASD). If the data set is password protected, IPCS ignores the password.

The data control block (DCB) attributes (BLKSIZE, DSORG, KEYLEN, LRECL, and RECFM) designated when the data set was defined override the following:
- For DASD data sets, these attributes in the data set control block (DSCB)
- For data sets on standard-labeled tapes, these attributes on the tape label

IPCS opens the data set when it is first accessed and closes the data set, restoring it to its original status. However, IPCS does not allocate or deallocate (release) the data set. The data set must be allocated before being requested in a FILE or DDNAME parameter on an IPCS subcommand. To allocate the data set, enter a TSO/E ALLOCATE command or the appropriate JCL statement before using the subcommand. To deallocate the data set, enter a TSO/E FREE UNALLOC command or use the parameter FREE=CLOSE on the JCL DD statement.

**Note:** IPCS processing does not allow the concatenation of data sets.

**NODATASET or NODSNAME** specifies that the subcommand is to set the source name in the local or global defaults to a null value. If you do not specify a source, the null value remains in effect.

**DISPLAY[(display-options)]**
**NODISPLAY[(nodisplay-options)]**

Specifies if the source is to be displayed or not.

DISPLAY, entered alone, requests that all parts of a dump be displayed. It is equivalent to entering

DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL)

DISPLAY, entered with one or more display-options, selects parts of a source to be displayed.

NODISPLAY, entered alone, is the same as DISPLAY(REQUEST). It is equivalent to entering:

DISPLAY(NOMACHINE NOREMARK REQUEST NOSTORAGE NOSYMBOL)

NODISPLAY entered with one or more values, suppresses (or selects) parts of a display.
SETDEF subcommand

Note: If VERIFY is specified or defaulted, and the NODISPLAY parameter is also specified, a conflict exists. In this case, IPCS responds as if you had entered DISPLAY(REQUEST).

DISPLAY(NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL) are the IPCS-defined defaults.

The DISPLAY and NODISPLAY parameter options and their meanings are:

{ DISPLAY } [ ( MACHINE | NOMACHINE ) ]
{ NODISPLAY } [ ( REMARK | NOREMARK ) ]
[ ( REQUEST | NOREQUEST ) ]
[ ( STORAGE | NOSTORAGE ) ]
[ ( SYMBOL | NOSYMBOL ) ]

MACHINE or NOMACHINE
MACHINE displays the address processing parameters, address, storage key, and absolute address of the data area being displayed. DISPLAY(MACHINE) and NODISPLAY(NOMACHINE) request this data.

For information about storage key values, see z/Architecture Principles of Operation.

NOMACHINE suppresses the address processing parameters, address, storage key, and absolute address of the data area being displayed. DISPLAY(NOMACHINE) and NODISPLAY(MACHINE) suppress it.

REMARK or NOREMARK
REMARK displays the remark associated with a symbol requested by the SYMBOL value. DISPLAY(REMARK) and NODISPLAY(NOREMARK) request this data.

NOREMARK suppresses the remark associated with a symbol requested by the SYMBOL value. DISPLAY(NOREMARK) and NODISPLAY(REMARK) suppress it.

Note: If both NOREMARK and SYMBOL are selected, IPCS displays as much of the remark text as possible on the same line as the symbol with which the remark is associated.

REQUEST or NOREQUEST
REQUEST displays a model LIST subcommand that is used to display the information you requested. The LIST subcommand parameters include the data description parameters you specify and other relevant default parameters (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

To modify the attributes of the displayed data, modify the parameters on the model LIST subcommand and run it. DISPLAY(REQUEST) and NODISPLAY(NOREQUEST) request this data.

NOREQUEST suppresses the model LIST subcommand. DISPLAY(NOREQUEST) and NODISPLAY(REQUEST) suppress it unless no data is requested. In that case, IPCS forces the DISPLAY(REQUEST) option into effect.
SETDEF subcommand

**STORAGE or NOSTORAGE**

STORAGE displays the storage at the specified or default address, for the specified or default length. The subcommand displays the storage as in a printed dump: four words in hexadecimal followed by the EBCDIC equivalent. DISPLAY(STORAGE) and NODISPLAY(NOSTORAGE) request this data.

NOSTORAGE suppresses the storage display. DISPLAY(NOSTORAGE) and NODISPLAY(STORAGE) suppress it.

**SYMBOL or NOSYMBOL**

SYMBOL displays the symbol (if any) associated with the dump data displayed. DISPLAY(SYMBOL) and NODISPLAY(NOSYMBOL) request this storage.

NOSYMBOL suppresses the symbol associated with the dump data displayed. DISPLAY(NOSYMBOL) and NODISPLAY(SYMBOL) suppress it.

**FLAG(severity)**

Specifies that IPCS subcommands eliminate some problem analysis diagnostic messages based upon the severity of the problem indicated by the message. Use FLAG to make a report easier to read by eliminating some messages.

The following messages can be suppressed with FLAG:

- Messages produced by IPCS services during the production of a report, but are not part of the report itself. For example, you can suppress the following message with FLAG(TERMINATING):
  
  BLS22020I ASCBASCB not equal C'ASCB'

  Although FLAG can make a report easier to read, it may eliminate useful information. For example, message BLS22020I may help you to understand why a report does not contain information you expected and may help you locate a storage overlay condition that requires further analysis.

- Messages produced by an IPCS CLIST or REXX exec. For example, you can suppress the following message:
  
  BLS18104I Symbol xxx not found

  Again FLAG can make a report easier to read, but it may eliminate useful information. The author of a CLIST or REXX exec may use FLAG on FIND and NOTE subcommands to make message suppression and transmission conditional.

Messages that do not detract from the legibility of a report are generally not affected by the FLAG value.

The **FLAG severity parameters and the messages transmitted follow.** WARNING is the IPCS-defined default.

```
FILE { (ERROR) } { (INFORMATIONAL) } { (SEVERE) } { (TERMINATING) } { (WARNING) }
```

**ERROR**

Transmits ERROR, SERIOUS (SEVERE), and TERMINATING messages
and suppresses INFORMATIONAL and WARNING messages. Error messages describe control blocks or data that point to incorrect control blocks or data.

INFORMATIONAL
Transmits all messages to your terminal.

SERIOUS or SEVERE
Transmits SERIOUS (SEVERE) and TERMINATING messages and suppresses INFORMATIONAL, WARNING, and ERROR messages. Serious or severe messages describe control blocks or data that are not valid.

TERMINATING
Transmits only TERMINATING messages and suppresses INFORMATIONAL, WARNING, ERROR, and SERIOUS (SEVERE) messages.

WARNING
Transmits WARNING, ERROR, SERIOUS (SEVERE), and TERMINATING messages and suppresses INFORMATIONAL messages. WARNING messages describe unusual conditions that are not necessarily wrong but might indicate errors.

LENGTH(length)
Specifies the length of the storage area to be used by dump analysis subcommands. The length may be 1 through \(2^{31}\) bytes and may be specified in decimal (nnn), hexadecimal (X'nnn'), or binary (B'nnn') notation.

LENGTH(4) is the IPCS-defined default.

PRINT or NOPRINT
Specifies whether a subcommand's output is to be sent to the print data set, IPCSPRNT.

PRINT sends the subcommand's output to the print data set. Note that IPCS always sends certain non-report type messages to your terminal or the TSO/E SYSTSPRT data set.

NOPRINT suppresses sending output to the print data set.

NOPRINT is the IPCS-defined default.

PDS or NOPDS
Specifies whether a subcommand output is to be sent to a member of the defined partitioned data set (PDS), allocated by ddname IPCSPDS.

PDS sends the subcommand output to the defined member of PDS. The defined member of PDS means that the name of this member will be equivalent to the name of the used IPCS subcommand. Note that IPCS always sends certain non-report type messages to your terminal or the TSO/E SYSTSPRT data set.

NOPD suppresses sending output to the PDS.

NOPD is the IPCS-defined default.

TERMINAL or NOTERMINAL
Specifies whether a subcommand's output is to be sent to your terminal or, for a batch job, to the TSO/E SYSTSPRT data set.

TERMINAL sends the subcommand's output to your terminal in an interactive IPCS session and to the TSO/E SYSTSPRT data set if IPCS is being run in a batch job.
**SETDEF subcommand**

NTERMINAL suppresses sending output. However, if NOPRINT is also in effect, all IPCS subcommands, except the SUMMARY subcommand, override the NOTERMINAL option and send their output as if the TERMINAL option had been specified.

NOTERMINAL is the IPCS-defined default.

**Note:** You may want to use the SETDEF subcommand to set the defaults to NOTERMINAL and NOPRINT. When these defaults are in effect, you need to specify only the PRINT parameter on a subcommand to send its output to the print data set, but not to the terminal. In contrast, with the standard defaults of NOPRINT and TERMINAL, the same subcommand with PRINT sends its output to both destinations. Both PRINT and NOTERMINAL are needed to selectively send output to only the print data set.

See [Table 1-1 on page 1-3](#) for a summary of the output possibilities.

**TEST or NOTEST**

Specifies if IPCS is supporting testing of IPCS code or is being used to analyze problem data.

TEST places IPCS in a mode designed to support interactive testing of code that operates in the IPCS environment. It is not recommended that you use this mode for any other purpose.

If you anticipate an abnormal ending while testing a new exit routine written to function in the environment provided by the ASCBEXIT, TCBEXIT, or VERBEXIT subcommands and you want to use TSO/E TEST facilities to isolate the cause of any problems, you should specify the TEST parameter. When TEST is in effect, IPCS allows the TMP, the TSO/E TEST ESTAI functions, or both, to gain control when an abnormal ending occurs.

TEST mode also activates error-detection functions that have been developed to isolate dump data examination problems. Detected errors cause IPCS to abend, so that problems may be trapped close to the point of error.

NOTEST places IPCS in the production mode of operation. Automatic error recovery is attempted should errors occur in the IPCS environment.

When the NOTEST parameter is in effect, IPCS automatically recovers from most abnormal endings without permitting TSO/E TEST to gain control.

NOTEST is the IPCS-defined default.

**VERIFY or NOVERIFY**

Specifies whether subsequent subcommands are to produce output and send it to the destination or destinations specified by the PRINT and TERMINAL parameters.

VERIFY specifies that subcommands should produce output and send it.

NOVERIFY specifies that subsequent subcommands are not to produce output or send it anywhere, regardless of the PRINT and TERMINAL parameters.

VERIFY is the IPCS-defined default.

**Return Codes**

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the SETDEF subcommand.

**Example**
SETDEF subcommand

Change the IPCS-defined defaults.

- Action
  COMMAND ===> setd dsn('d4.dmp.svc20') asid(X'0008') list

- Result
  IPCS produces the following output:

/*---------------- Default Values for IPCS Subcommands ----------------*/
SETDEF NOPRINT TERMINAL NOPDS /* Routing of displays */
SETDEF FLAG(WARNING) /* Optional diagnostic messages */
SETDEF CONFIRM /* Double-checking major acts */
SETDEF NOTEST /* IPCS application testing */
SETDEF DSNAMED('D4.DMP.SVC20') /* Default data length */
SETDEF LENGTH(4) /* Optional dumping of data */
SETDEF DISPLAY(NOMACHINE) /* Include storage keys, .... */
SETDEF DISPLAY( REMARK) /* Include remark text */
SETDEF DISPLAY( REQUEST) /* Include model LIST subcommand */
SETDEF DISPLAY( NOPRSTORAGE) /* Include contents of storage */
SETDEF DISPLAY( SYMBOL) /* Include associated symbol */
SETDEF ASID(X'0008') /* Default address space */

SMFDATA subcommand — obtain system management facilities records

Use the SMFDATA subcommand to recover system management facilities (SMF) records from buffers in the dump and transfer them to a pre-allocated SMF (VSAM) data set or a log stream if RECORDING(LOGSTREAM) had been in use at the time of the dump.

The output data set must be:
- Pre-allocated to the data set with a ddname of SMFDATA
- Using the same control interval size as the defined SMF data sets
- Large enough to accommodate all the SMF data in the dump
- Allocated and used for only this purpose
- Defined with a low offload threshold (for example HIGHOFFLOAD(10)) to account for heavy utilization of the coupling facility structure.

The output log stream must be:
- Defined with the administrative data utility (IXCMIAPU or IXCM2APU) with a log stream name of IFASMF.DUMP00
- Defined with a MAXBUFSIZE that matches or exceeds the defined MAXBUFSIZE value of the logstream data that resides in the dump.
- Accessible from the local system
- Sized large enough to hold the data in the dump
- Allocated and used for only this purpose.

Syntax

SMFDATA

Return Codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the SMFDATA subcommand.
SSIDATA subcommand

SSIDATA subcommand — display subsystem information

Use the SSIDATA subcommand to display information about subsystems defined to the subsystem interface (SSI), including:
- The number of subsystems defined to the SSI
- The subsystem name
- Whether the subsystem is the primary subsystem
- Whether the subsystem is dynamic
- The status of the subsystem
- Whether the subsystem accepts or rejects the SETSSI operator command
- The address of the subsystem request router
- The function routines that the subsystem supports

Syntax

SSIDATA

------- SETDEF-Defined Parameters -----------------------------

Note: You can override the following SETDEF parameters.
See “SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[FLAG(severity)]
[PRINT | NOPRINT]
[TERTIAL | NOTERMINAL]
[TEST | NOTEST]

Return Codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the SSIDATA subcommand.

Example

The SSI component chapter in z/OS MVS Diagnosis: Reference shows an example of SSIDATA output.

STACK subcommand — create a symbol in the stack

Use the STACK subcommand to add a symbol to the symbol table for the current source in your dump directory. The STACK subcommand adds a created symbol in the form Znnnnn to the end of the stack in the symbol table. To determine the number nnnnn, IPCS uses the smallest numeric suffix that is greater than the suffix currently in use. See the z/OS MVS IPCS User’s Guide for more information about stack symbols.

Related subcommands
- EQUATE
- DROPSYM
- LISTSYM
- RENUM

Syntax
STACK subcommand

STACK [ data-descr | X ]

[ DROP | NODROP ]

------- SETDEF-Defined Parameter --------------------------
Note: You can override the following SETDEF parameter.
See
“SETDEF subcommand — set defaults” on page 5-232

[ TEST | NOTEST ]

Parameters

data-descr or X

Specifies the address and attributes to be associated with the symbol being
defined. The data description parameter consists of five parts:
- An address (required when data-descr is explicitly specified on the
  subcommand)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and
syntax of the data description parameter. However, the following exception
applies only to STACK:
- If you omit the data description parameters, the default for the STACK
  subcommand is X, which is the most recently accessed address.

DROP or NODROP

Specifies whether the created symbol can be deleted or not from the symbol
table by a DROPSYM subcommand without a PURGE parameter:
- DROP specifies that the symbol can be deleted.
- NODROP specifies that the symbol cannot be deleted. However,
  NODROP can be overridden by a PURGE parameter on the DROPSYM
  subcommand.

Return Codes

See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the STACK subcommand.

STATUS subcommand — describe system status

Use the STATUS subcommand to display data that are typically examined during
the initial part of the problem determination process.

STATUS produces different diagnostic information depending on the report type
parameter or parameters entered: SYSTEM, CPU, WORKSHEET, and FAILDATA.

The information displayed by STATUS for each central processor is helpful in
problem analysis for most dumps. However, the ANALYZE or SUMMARY
subcommands can be more helpful:
- If a dump is taken as a result of operator intervention, such as an SVC dump
  from a DUMP operator command or a stand-alone dump. In these dumps, IPCS
  might not be able to identify appropriate units of work from which analysis can
  proceed. In fact, by the time the operator has recognized the need for a dump
  and requested one, the unit of work that caused the problem might no longer
  exist.

Chapter 5. IPCS subcommands 5-243
Some problems involve the interaction of multiple units of work. If one of the units of work detects a problem and requests a dump, the analysis of the STATUS subcommand focuses primarily on the unit of work that requested the dump.

Related subcommands
ANALYZE
CBSTAT
LIST
SUMMARY

Syntax

```{ STATUS }
{ ST }
```

-------- Report Type Parameters -------------------------------

```[ SYSTEM | NOSYSTEM ]
[ CPU[(cpu)] ]
[ registers | NOREGISTERS ]
[ VECTOR | NOVECTOR ]
[ CONTENTION | NOCONTENTION ]
[ DATA | NODATA ]
[ NOCPU ]
[ WORKSHEET | NOWORKSHEET ]
[ FAILDATA | NOFAILDATA ]
```

-------- SETDEF-Defined Parameters ------------------------------

Note: You can override the following SETDEF parameters. See "SETDEF subcommand — set defaults" on page 5-232

```[ ACTIVE | MAIN | STORAGE ]
[ DSNAMES dsname ]
[ DATASETS dsname ]
[ FILES ddname ]
[ DDNAMES ddname ]
[ PATHS path-name ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
```

Parameters

Report Type Parameters

Use these parameters to select the type of report. If you omit a report type parameter, the default is SYSTEM, CPU, WORKSHEET, and FAILDATA. For more information about defaults, see 5-246

SYSTEM or NOSYSTEM

Specifies or suppresses the system status information. The SYSTEM parameter displays:
- The nucleus member name
- I/O configuration data
- The sysplex name
STATUS subcommand

- Time-of-day (TOD) clock in both local and Greenwich mean time (GMT)
- The name of the program that produced the dump
- The name of the program that requested the dump

SYSTEM specifies the information. See 5-247 for an example of the SYSTEM report.

NOSYSTEM suppresses the information.

CPU[cpu] or NOCPU
Specifies or suppresses the CPU status information. The CPU parameter displays for each central processor:
- The PSW and its analysis
- A description of the current unit of work by its type of control block, for example, the address space control block (ASCB), the task control block (TCB), or the system request block (SRB)
- A list of locks held
- A summary of the current function recovery routine (FRR) stack
- The contents of the general purpose registers and control registers
- The contents of the access registers
- The contents of the vector registers for each central processor that has a Vector Facility installed
- A breakdown of resources held by the unit of work
- A table containing CPU start and end times.

CPU specifies the information and, optionally, limits CPU status messages to those applicable to the designated central processor address, cpu. If cpu is not specified, the default is for all central processors that were online when the dump was generated. See 5-248 for an example of the CPU report. Part of this report can be obtained through an ANALYZE ASID subcommand.

NOCPU suppresses the information.

The following parameters modify the CPU report. If any of these parameters are specified and CPU is not specified, CPU is the default.

REGISTERS or NOREGISTERS
Specifies or suppresses the formatting of the general purpose and control registers for the specified central processors.

REGISTERS specifies the register data. The abbreviation REGS can be used for REGISTERS.

NOREGISTERS suppresses register data and is the default.

VECTOR or NOVECTOR
Specifies or suppresses the formatting of the vector registers for the specified central processors.

VECTOR specifies the vector register data.

NOVECTOR suppresses vector register data and is the default.

CONTENTION or NOCONTENTION
Specifies or suppresses the formatting of contention information for the unit of work that was active on the central processor(s) at the time of the dump.

Note: If you want to format contention information for the entire dumped system, use the ANALYZE subcommand instead of STATUS.

CONTENTION requests contention information.
### STATUS subcommand

**NOCONTENTION** suppresses contention information and is the default.

**DATA or NODATA**

Specifies or suppresses formatting of central processor-related control blocks and global system control blocks.

DATA requests the control blocks. Global system control blocks that are not central processor-related appear before individual central processor-related information. If you specify a particular central processor number, global system control blocks are not formatted.

The central processor-related control blocks for this subcommand are:
- Logical configuration communication area (LCCA)
- Physical configuration communication area (PCCA)
- Prefixed save area (PSA)
- Supervisor control FLIH save area (SCFS)
- The linkage stack for the active unit of work

The global system control blocks for this subcommand are:
- Common system data (CSD)
- System verification table (SVT)

NODATA suppresses the control blocks and is the default.

**WORKSHEET or NOWORKSHEET**

Specifies or suppresses the diagnostic worksheet, which contains central processor information. The WORKSHEET diagnostic report describes the state of the system and each central processor in the system, and includes:
- The CPU serial number
- The CPU version
- The CPU address
- The SDUMP parameter list, if the dump is an SVC dump or a SYSMDUMP
- Current wait state messages

WORKSHEET specifies the diagnostic worksheet. All central processors in the system are in the report. For stand-alone dumps, IPCS obtains much of the information from the store status records. For SVC dumps, the processor-related data does not contain the store status data. The WORKSHEET parameter displays the SDUMP parameter list for SVC dumps. See [5-249](#) for an example of the WORKSHEET report.

NOWORKSHEET suppresses the diagnostic worksheet.

**FAILDATA or NOFAILDATA**

Specifies or suppresses formatting of the system diagnostic work area (SDWA), which is in the SVC dump header.

FAILDATA specifies formatting of the SDWA. See [5-251](#) for an example of the FAILDATA report.

NOFAILDATA suppresses formatting of the SDWA.

**Defaults**

The defaults for the STATUS report type parameters follow.
### STATUS subcommand

<table>
<thead>
<tr>
<th>Parameters on the STATUS subcommand</th>
<th>Reports Requested</th>
<th>For Example</th>
</tr>
</thead>
</table>
| No report type parameter           | SYSTEM, CPU, WORKSHEET, and FAILDATA | COMMAND ===>
|                                    |                   | status     |
|                                    |                   | STATUS displays SYSTEM, CPU, WORKSHEET, and FAILDATA reports. |
| One or more of the report type parameters: SYSTEM, CPU, WORKSHEET, FAILDATA | The requested report or reports: SYSTEM, CPU, WORKSHEET, or FAILDATA | COMMAND ===>
|                                    |                   | status system cpu(1) |
|                                    |                   | STATUS displays SYSTEM and CPU(1) reports. |
| One or more of the negative report type parameters: NOSYSTEM, NOCPU, NOWORKSHEET, NOFAILDATA | Not specifying the suppressed reports | COMMAND ===>
|                                    |                   | status nosystem |
|                                    |                   | STATUS displays CPU and WORKSHEET reports. |
| No report type parameter, but one or more CPU parameters: REGISTERS, NOREGISTERS, VECTOR, NOVECTOR, CONTENTION, NOCONTENTION, DATA, NODATA | CPU report | COMMAND ===>
|                                    |                   | status noregisters |
|                                    |                   | STATUS displays a CPU report. |

### Return Codes

See [“Standard subcommand return codes” on page 5-2](#) for a description of the return codes produced by the STATUS subcommand.

**Example 1**

Produce a system status report.

- **Action**
  COMMAND ===>
  status system

- **Result**

  The following output is produced.

```plaintext
SYSTEM STATUS
Nucleus member name: IEANUC01
I/O configuration data:
  IODF data set name: SYS0.IODF52
  IODF configuration ID: CONFIG00
  EDT ID: 00
Sysplex name: PLEX01
TIME OF DAY CLOCK: A9B7540D 54AD1405 08/12/1994 10:54:28.305617 local
Program Producing Dump: SVC03
Program Requesting Dump: DATSVY02
Incident token: LOCAL 5520 08/12/1994 14:54:23.888770 GMT
```

1. Identifies the STATUS report type, SYSTEM.
2. Identifies the nucleus member name, IEANUC01, that was initialized at system installation.
3. Gives information about the I/O configuration that was active when the dump was produced. IPCS identifies the name of the IODF data set, the configuration identifier, and the eligible device table (EDT) definition.
4. Identifies the sysplex name, PLEX01, specified in the COUPLExx parmlib member.
STATUS subcommand

5 Displays a TOD clock value placed in the dump to indicate when the dump was produced. The TOD clock value is in hexadecimal and in a date and time of day for local time and Greenwich mean time (GMT). To determine local time, the system uses field CVTTZ in the CVT.

6 Identify the programs requesting and producing the dump.

Note: A list of the SVC dump options follow the output shown above.

Example 2
Produce a CPU status report.
- Action
  COMMAND ===> status cpu registers contention
- Result
  The following output is produced.

CPU STATUS:
PSW=07BC1000 83D00B72 (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
DISABLED FOR PER
ASID(X'0015') 03D00B72. DATSVY02+03CA IN EXTENDED PRIVATE
ASCB21 at F9D800, JOB(DAESVY01), 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

Identifies the STATUS report type, CPU. The CPU address is omitted because a virtual dump is being processed.

Displays the program status word (PSW) followed by a description of what the PSW indicates.

IPCS extracts the current PSW from the dump header record for virtual storage dumps and from the store status record for absolute storage dumps.

One of the following descriptions providing PSW status might appear after the PSW:

- **NO WORK WAIT**
- **DISABLED WAIT STATE CODE xxx SUPPLEMENT CODE yyyyy** where:
  - xxx is the wait state code in hexadecimal
  - yyyyy is supplemental information in hexadecimal for the wait state code. The format is dependent on the particular wait state. See **z/OS MVS System Codes** for more information.
- **RUNNING IN mode, KEY k, AMODE aa, datmode**
  - mode is the address space addressability of either primary or secondary.
  - k is the current storage key of 0 through F.
STATUS subcommand

- *aa* is the current addressing mode of either 24 or 31 bit.
- *datmode* is either DAT-ON or DAT-OFF

**- ENABLED | DISABLED**

When the PSW is enabled or disabled, a list of the interrupts is displayed.

**Note:** For dumps generated by a stand-alone dump, the system operator must perform the store status operation before IPLing the stand-alone dump program. If the store status operation is not done, the PSW will not be accurate.

3 Displays the current ASCB, ASXB, or TCB.

The output might also display the processor status. One of the following descriptions can appear:

- **HOME ASID: hhhh PRIMARY ASID: pppp SECONDARY ASID: ssss**
  IPCS identifies the applicable address spaces (in hexadecimal) relevant to the unit of work running on the CPU at the time of the dump.
  - *hhhh* is the home address space identifier
  - *pppp* is the primary address space identifier
  - *ssss* is the secondary address space identifier

- **HOLDING LOCK(S): lockname1 lockname2 ...**
  IPCS identifies the locks that are held by the unit of work that is running on the CPU at the time of the dump. See [z/OS MVS Diagnosis: Reference](https://www.ibm.com/support/knowledgecenter/SSG5G2_2.4.0/com.ibm.zos.mvs.r24.doc/rgbmsvsc0000.htm) for the list of locks.

- **CURRENT FRR STACK IS: stack-name**
  **PREVIOUS FRR STACK(S): stack-name1 stack-name2 ...**
  STATUS identifies the current FRR stack and the previous FRR stack names and displays the previous FRR stack names in the order that the stack will get control.

  **Note:** If the CURRENT stack is the NORMAL stack, the **PREVIOUS FRR STACK(S) is not displayed.**

4 Displays the general register (GPR) contents in hexadecimal.

5 Displays the access register contents in hexadecimal or displays a message that all ALETs are zero.

6 Displays the control register contents in hexadecimal.

**Not shown**

If the VECTOR parameter is specified and if a Vector Facility is installed on the processor, the vector registers are displayed in hexadecimal following the control registers.

**Not shown**

If this dump had contention data, the contention report follow the register information. The contention data report lists the held resources, resources being waited on, and any contention data related to other units of work.

**Example 3**

Produce a diagnostic worksheet.

- **Action**
  COMMAND ===> status worksheet
STATUS subcommand

- Result
  The following output is produced.

MVS Diagnostic Worksheet

Dump Title: SERV2_DUMP1

CPU Model 3090 Version FF Serial no. 176280 Address 01
Date: 05/05/1994   Time: 16:59:35.414381 Local

Original dump dataset: SERV2.DMP0001

Information at time of entry to SVCDUMP:

HASID 0001   PASID 0001   SASID 0001   PSW 070C1000   83930882
CML ASCB address 00000000 Trace Table Control Header address 7F731000
System reset nondispatchability Trace Table Control header address 7F585000

Dump ID: 001
Error ID: N/A
SDWA address N/A

SYSTEM RELATED DATA

CVT SNAME (154) S520   VERID (-18) HBB5520 DRIVER08

CUCB (64) 00FCF8F8   PVTX (164) 00FDDED0   GDA (230) 01D551A0

RTMCT (23C) 00FC0780   ASMVT (2C0) 00FD3250   RCEP (490) 014D9F80
CSD Available CPU mask: 4000   Alive CPU mask: 40000000   00000000
Number of active CPUs: 00000001
System set non-dispatchable by SVC dump

PROCESSOR RELATED DATA

<table>
<thead>
<tr>
<th>NAME</th>
<th>OFFSET</th>
<th>CPU 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHR1</td>
<td>208</td>
<td>00</td>
</tr>
<tr>
<td>SPN1/2 Spin</td>
<td>20C</td>
<td>0000</td>
</tr>
<tr>
<td>CPU5 CPU WSAVT</td>
<td>218</td>
<td>00F8AA4B</td>
</tr>
<tr>
<td>DSFI/2 Dispatcher</td>
<td>21C</td>
<td>0080</td>
</tr>
<tr>
<td>CRFL</td>
<td>2B4</td>
<td>00000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>OFFSET</th>
<th>CPU 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOLD</td>
<td>21C</td>
<td>00000000</td>
</tr>
<tr>
<td>AOLD</td>
<td>224</td>
<td>00F4A580</td>
</tr>
<tr>
<td>SUPER</td>
<td>22B</td>
<td>00000000</td>
</tr>
<tr>
<td>CLHT</td>
<td>2B0</td>
<td>00FCDA1B</td>
</tr>
<tr>
<td>LOCAL</td>
<td>2EC</td>
<td>00000000</td>
</tr>
<tr>
<td>CLHS</td>
<td>2F8</td>
<td>80000000</td>
</tr>
<tr>
<td>CSTK</td>
<td>380</td>
<td>000000C0</td>
</tr>
<tr>
<td>SMPSW SB Disp PSW</td>
<td>420</td>
<td>070C0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>424</td>
</tr>
<tr>
<td>PSWSV PSW Save</td>
<td>468</td>
<td>070C0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>46C</td>
</tr>
<tr>
<td>MODE</td>
<td>49F</td>
<td>04</td>
</tr>
</tbody>
</table>
STATUS subcommand

SDUMP Parameter List

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0000</td>
<td>FLAG0..</td>
<td>1E</td>
</tr>
<tr>
<td>+0001</td>
<td>A1</td>
<td>SDATA.. 9920</td>
</tr>
<tr>
<td>+0002</td>
<td>SDAT4..</td>
<td>00</td>
</tr>
<tr>
<td>+000C</td>
<td>HDRAD..</td>
<td>01E298E0</td>
</tr>
<tr>
<td>+0011</td>
<td>ASIDP..</td>
<td>00000000</td>
</tr>
<tr>
<td>+002A</td>
<td>TYP1..</td>
<td>10</td>
</tr>
<tr>
<td>+0040</td>
<td>STRAL..</td>
<td>00000000</td>
</tr>
<tr>
<td>+0068</td>
<td>PSWRP..</td>
<td>00000000</td>
</tr>
<tr>
<td>+007C</td>
<td>IDA....</td>
<td>00000000</td>
</tr>
<tr>
<td>+0090</td>
<td>RMADR..</td>
<td>00000000</td>
</tr>
<tr>
<td>+00A4</td>
<td>JLAALT..</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Note: The SDUMP parameter list appears if this is an SVC dump.

1. Identifies the STATUS report type, WORKSHEET.
2. Displays the title, date, and time from the dump header record.
3. This section identifies the CPU model, version, serial number, and address. The end of this section will also display wait state messages, if they are current.
4. Displays the Trace Table Control Header address of the SNAPTRC, which was issued if the system was reset to be dispatchable because the system has been kept non-dispatchable longer than the MAXSNDSP value.
5. The identifiers of the dump and the error.
6. Lists system-related data by displaying key fields and their hexadecimal offsets in the CVT and by displaying information about the processors in the system that appears in the CSD.

The SYSTEM RELATED DATA section:
- Provides information for both SVC dumps and stand-alone dumps.
- Displays “N/A” for any missing data.
- May display the following texts after the CSD data:
  System set non-dispatchable by SVC Dump
  ACR in progress

7. Lists processor-related data. For each CPU, IPCS displays the contents of the PSW, control registers (CR) 0 and 6, and selected fields from the LCCA and PSA.

The PROCESSOR RELATED DATA section:
- Does not display the store status data for SVC dumps
- Fills in a CPU header and column for each nonzero PCCAVT entry
- Displays “N/A” for any missing data.
- Repeats the PROCESSOR RELATED DATA section as many times as necessary to include all processor-related data that was dumped. The number of CPU columns depends on the recommended display width that is set by IPCS to be the lesser of the terminal width and the print data set LRECL.

Example 4

Produce an SDWA report.
- Action
  COMMAND ===> status faildata
STATUS subcommand

- Result
The following output is produced.

1
* * * DIAGNOSTIC DATA REPORT * * *

2 SEARCH ARGUMENT ABSTRACT

RIDS/DMPSD998#L RIDS/DMPSD998 AB/S00C1 PRCS/00000001 REGS/0B5CA
RIDS/DMPESTAE#R

Symptom Description
------- -----------
RIDS/DMPSD998#L Load module name: DMPSD998
RIDS/DMPSD998 Csect name: DMPSD998
AB/S00C1 System abend code: 00C1
PRCS/00000001 Abend reason code: 00000001
REGS/0B5CA Register/PSW difference for R0B: 5CA
RIDS/DMPESTAE#R Recovery routine csect name: DMPESTAE

3 SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

Program id
Recovery Routine Label
Date Assembled
Module Level
Subfunction

4 Time of Error Information

PSW: 070C2000 81E0D616 Instruction length: 02 Interrupt code: 0001
Failing instruction text: 920A1005 00000000 5870A1F8

Registers 0-7
GR: 40000004 00C13300 00000000 00C13300 00C13300 00C13300 00C13300 00C136EB 00C1349C
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Registers 8-15
GR: 00C13350 00C13300 01E01260 81E0D04C 01E0E04B 01E01260
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Home ASID: 000C Primary ASID: 000C Secondary ASID: 000C
PKM: 0080 AX: 0000 EAX: 0000

RTM was entered because of a program check interrupt.
The error occurred while an enabled RB was in control.
No locks were held.
No super bits were set.

5 STATUS FROM THE RB WHICH ESTABLISHED THE ESTAE EXIT

PSW and registers are the same as those from the time of error.

6 RECOVERY ENVIRONMENT

Recovery routine type: ESTAE recovery routine
Recovery routine entry point: 01E0D8A8
FRR parameter area on entry to FRR:
+00 00C13350 00C13300 01E01260 81E0D04C 01E0E04B 01E01260
There were no outstanding I/O operations to purge.

7 NO DATA EXISTS IN THE VARIABLE RECORDING AREA
Identifies the report type, DIAGNOSTIC DATA REPORT.

The search argument abstract is generated from the error-related information in the SDWA. It is useful for problem searches against customer or IBM problem-reporting data bases.

Note: If you report the problem to IBM, include symptoms from this abstract in the problem report.

Indicates information that was not available because the recovery routine did not provide it. When this information is available, it appears in section 2 under the title “Other Serviceability Information”.

Provides PSW, register, and ASID-related error information, along with failure reasons and environments and, if applicable, super or spin bit settings.

Note: The locks that were held at the time of error might have been released by RTM, thus resulting in the statement of No locks were held in the Time of Error Information report.

Presents second-level status information as indicated by the second set of registers and their corresponding PSW, which are located in the SDWA.

Provides details about the recovery environment for the error. This section may include one or more of the following items:
- Recovery routine type
- PSW at entry to functional recovery routine (FRR)
- Recovery routine entry point (ESTAE/ESTAI/ARR)
- FRR parameter area contents
- Information relevant to the previous recovery environment
- Error entry information
- Status of I/O operations

Indicates that the variable recording area is empty. If the area contained data, it is displayed here in hexadecimal and EBCDIC format. When this area is in key-length-data format, each key-length-data structure is individually formatted.

STRDATA subcommand — format coupling facility structure data

Use the STRDATA subcommand to format coupling facility structure data. Depending on the parameters you specify, you can obtain information at the summary or detail level and about one or more coupling facility structures.

If duplexing rebuild is supported for a structure, duplexing control information is returned in addition to the dump header information for each structure instance. The control information is returned regardless of whether duplexing is currently active for the structure.

For more information about the reports generated by the STRDATA subcommand, see the XES chapter of z/OS MVS Diagnosis: Reference.

Note: To diagnose problems related to XES, you may also want to use the XESDATA and COUPLE subcommands.

Syntax
STRDATA subcommand

STRDATA

------- Data Selection Parameters -------
{ DETAIL   }
{ SUMMARY  }

------- Report Type Parameters -------
{ ALLSTRS  }
{ STRNAME(strname,strdumpid),...  }

------- Additional Filter Parameters -------
[ ALLDATA    ]
[ ARB         ]
[ COCLASS(coclass) ]
[ EMCONTROLS(emcontrols) ]
[ ENTRYID(entryid) ]
[ ENTRYNAME(entryname) ]
[ EVENTQS(conid) ]
[ LISTNUM(listnum) ]
[ LOCKENTRIES(lockentries) ]
[ STGCLASS(stgclass) ]
[ USERCNTLS(usercntls) ]

------- Cache Specifier Parameters -------
[ ENTRYPOS(entrypos) ]
[ ORDER            ]

------- List Specifier Parameters -------
[ ENTRYPOS(entrypos) ]
[ ORDER            ]
[ ENTRYKEY(entrykey) ]
STRDATA subcommand

------ SETDEF-Defined Parameters -----------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAMES(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[FLAG(severity)]

[PRINT | NOPRINT]

[TERMINAL | NOTERMINAL]

[TEST | NOTEST]

Parameters
If you omit all parameters, the defaults are SUMMARY and ALLSTRS.

Data Selection Parameters
Use these parameters to limit the scope of the data in the report. If you omit these parameters, the default is SUMMARY.

SUMMARY
Requests summary information for each report you specify. The report output is STRDATA ALL STRUCTURES SUMMARY REPORT. The output fields for each structure are:
- Structure name
- Structure type
- Structure dump ID
- Coupling facility information
- Facility name
- A summary of coupling facility structure controls

An example is:
COMMAND ====> STRDATA SUMMARY

DETAIL
Requests detailed information for each report you specify. The report output is STRDATA ALL STRUCTURES DETAIL REPORT The output fields for each structure are:
- Structure name
- Structure type
- Structure dump ID
- Coupling facility information
- Facility name
- All of the coupling facility structure controls
- List of assigned users
- If applicable, duplexing control information including
  - Duplexing-active indicator
  - Remote facility node descriptor (ND) and system identifier (SYID)
  - Remote structure identifier (SID) and structure authority (SAU)

An example is:
COMMAND ====> STRDATA DETAIL

Report Type Parameters
Use these parameters to select the type of report. If you omit a report type parameter, the default is ALLSTRS.
STRDATA subcommand

**ALLSTRS**
Requests information about all coupling facility structures found in the dump. The report output is STRDATA ALL STRUCTURES SUMMARY REPORT. The output fields for each structure are:
- Structure name
- Structure type
- Structure dump ID
- Coupling facility information
- Facility name
- A summary of coupling facility structure controls

**COMMAND ==> STRDATA ALLSTRS**

**STRNAME (strname,strdumpid),(strname,strdumpid),...**
Requests information about the coupling facility structures listed. Structures may be list, cache, or any combination of list and cache.

**Note:** Lock structures are not dumped.

The report output is CACHE STRUCTURE SUMMARY REPORT. The output fields for each structure name specified are:
- Structure name
- Structure type
- Structure dump ID
- Coupling facility information
- Facility name
- A summary of coupling facility cache structure controls

The *strname* specifies the name of a structure. For example:

**COMMAND ==> STRDATA STRNAME((CACHE01))**

**Note:** If you specify a list structure in *strname*, the report output is a List Structure Summary Report.

At the end of a *strname*, an asterisk (*) may be used as a generic character to include in the report all structure names having the specified characters in common. The following subcommand specifies all structure names beginning with the characters 'LIST' and the report includes structures LIST01, LIST02, LIST03, and so forth.

**COMMAND ==> STRDATA STRNAME((LIST*))**

The *strdumpid* specifies an instance of the structure in the dump. A reason you may have more than one instance of a structure in a dump is if a structure is in rebuild processing or is in the Duplex Established phase, when the dump is captured. If a structure dump ID is not provided, information for all the structures in the dump with the same name are displayed. The *strdumpid* is specified in hexadecimal and without quotation marks, as this example shows:

**COMMAND ==> STRDATA STRNAME((CACHE01,0101))**

The STRDATA STRNAME parameter is associated with the STRNAME parameter of the IXLCONN macro.

**ALLDATA**
Requests the display of all data found in the dump for the specified structures. When ALLDATA is specified with STRNAME, all the data regarding the specified structure is presented. When ALLDATA is specified with ALLSTRS, all the data found for all the structures in the dump is presented. The report output is:

LIST STRUCTURE ALLDATA SUMMARY REPORT
For the output fields in the report, see the output fields for ARB, ENTRYPOS, LOCKENTRIES, and USERCNTLS. If a cache structure had been specified, then all reports pertaining to cache structures would have been displayed.

An example is:

```
COMMAND ===> STRDATA STRNAME((LIST02)) ALLDATA
```

Additional data selection parameters

**COCLASS (ALL | coclass,coclass:coclass,...)**

Requests information by cast-out class for a coupling facility cache structure.

The *coclass* can be a single cast-out class, a range of classes, or a list of noncontiguous classes. When you specify a range, separate the first and last classes in the range with a colon. When you specify a list, separate the list members with commas.

The report output is:

```
STRDATA ALL STRUCTURES SUMMARY REPORT
CASTOUT CLASS SUMMARY REPORT
```

The output fields for each *coclass* specified are:

- Class type
- Class
- Class status
- Cast-out class controls

The STRDATA COCLASS parameter is associated with:

- The NUMCOCLASS parameter of the IXLCONN macro
- The COCLASS parameter of the IXLCACHE macro

An example is:

```
COMMAND ===> STRDATA COCLASS(01)
```

**STGCLASS (ALL | stgclass,stgclass:stgclass,...)**

Requests information by storage class for a coupling facility cache structure.

The *stgclass* can be a single storage class, a range of classes, or a list of noncontiguous classes. When you specify a range, separate the first and last classes in the range with a colon. When you specify a list, separate the list members with commas.

The report output is:

```
STRDATA ALL STRUCTURES SUMMARY REPORT
STORAGE CLASS SUMMARY REPORT
```

The output fields for each storage class specified are:

- Class type
- Class
- Class status
- Class control information

The STRDATA STGCLASS parameter is associated with:

- The NUMSTGCLASS parameter of the IXLCONN macro
STRDATA subcommand

- The STGCLASS parameter of the IXLCACHE macro

An example is:

COMMAND ===> STRDATA STGCLASS(01)

LISTNUM (ALL | listnum,listnum:listnum,...)

Requests information by list number in a coupling facility list structure.

The listnum can be a single list number, a range of numbers, or a list of noncontiguous numbers. When you specify a range, separate the first and last numbers in the range with a colon. When you specify a list, separate the list members with commas.

The report output is:

STRDATA ALL STRUCTURES SUMMARY REPORT
LIST NUMBER SUMMARY REPORT

The output fields for each list number specified are:

- List number
- List number status
- Summary of the list controls

The STRDATA LISTNUM parameter is associated with:
- The LISTHEADERS parameter of the IXLCONN macro
- The LISTNUM parameter of the IXLLIST macro

An example is:

COMMAND ===> STRDATA LISTNUM(01)

EMCONTROLS(ALL | listnum,listnum:listnum,...)

Requests information about event monitor controls (EMCs) associated with a list structure identified by its list number.

The listnum can be a single list number, a range of list numbers, or a list of noncontiguous list numbers. When you specify a range, separate the first and last identifiers in the range with a colon. When you specify a list number, separate the list numbers with commas.

The report output is:

STRDATA ALL STRUCTURES SUMMARY/DETAIL REPORT
EVENT MONITOR CONTROLS SUMMARY/DETAIL REPORT

The output fields for each list number are:

- Event monitor controls list number
- Event monitor controls status
- For each EMC associated with the list number, the following EMC Detail Report information:
  - Connection ID
  - List number
  - List entry key
  - Event queue status
  - User notification controls.

An example is:

COMMAND ===> STRDATA EMCONTROLS(01)

EVENTQS(ALL | conid,conid:conid,...)

Requests information about event monitor controls (EMCs) on the event queue associated with a list structure connector.

The conid can be a single connection identifier, a range of connection identifiers, or a list of noncontiguous connection identifiers. When you
specify a range, separate the first and last identifiers in the range with a colon. When you specify a connection identifier, separate the connection identifiers with commas.

The report output is:
- STRDATA ALL STRUCTURES SUMMARY/DETAIL REPORT
- EVENT QUEUE CONTROLS SUMMARY/DETAIL REPORT

The output fields for each connection ID are:
- Connection ID
- Number of EMCs dumped
- Event queue controls status
- Event queue transition exit status
- Event queue monitoring status
- Event notification vector index
- Number of EMCs queued
- Number of state transitions
- For each EMC on the event queue:
  - EMC Detail Report information as described above for EMCONTROLS

An example is:
COMMAND ===> STRDATA EVENTQS(1)

**USERCNTLS (ALL | conid,conid:conid,...)**
Requests information by user connection identifier about the user of a structure.

The conid can be a single connection identifier, a range of identifiers, or a list of noncontiguous identifiers. When you specify a range, separate the first and last identifiers in the range with a colon. When you specify a list, separate the list members with commas.

The report output is:
- STRDATA ALL STRUCTURES SUMMARY REPORT
- USER CONTROLS REPORT

The output fields for each connection identifier (ID) specified are:
- Connection ID status
- Connection name
- Connection ID
- Connection status
- User authority
- User control information

An example is:
COMMAND ===> STRDATA USERCNTLS(01)

**LOCKENTRIES (ALL | lockentry,lockentry:lockentry,...)**
Requests information by the entries specified for the lock table entries of a coupling facility list structure.

The lockentry can be an entry, a range of entries, or a list of noncontiguous entries. When you specify a range, separate the first and last entries in the range with a colon. When you specify a list, separate the list members with commas.

The report output is:
- STRDATA ALL STRUCTURES SUMMARY REPORT
- LOCK ENTRIES REPORT

The output fields for each entry into the lock table are:
- Lock entries status
STRDATA subcommand

- Lock entries
- Owners connection ID
- Held By system indicator

The STRDATA LOCKENTRIES parameter is associated with:
- The LOCKENTRIES parameter of the IXLCONN macro
- The LOCKINDEX parameter of the IXLLIST macro

An example is:
COMMAND ====> STRDATA LOCKENTRIES(ALL)

ENTRYID (entryid,X'entryid',...)
Requests the display of information by list entry identifiers for a coupling facility list structure.

The entryid can be expressed in decimal or in hexadecimal (X'nnn').

The report output is:
STRDATA ALL STRUCTURES SUMMARY REPORT
LIST ENTRY IDENTIFIER SUMMARY REPORT

The output fields for each entry ID specified are:
- List entry identifier
- List entry controls
- Adjunct data
- Structure serialization indicator

The STRDATA ENTRYID parameter is associated with the ENTRYID parameter of the IXLLIST macro.

An example is:
COMMAND ====> STRDATA ENTRYID(X'000000000000000100000009')

ENTRYNAME (entryname,entryname...)
Requests information by list entry names in a coupling facility list structure or by data entry names in a coupling facility cache structure.

The report output is:
STRDATA ALL STRUCTURES SUMMARY REPORT
ENTRY NAME SUMMARY REPORT

The output fields for each entry name specified are:
- Entry name
- Directory information (for cache)/ list entry controls (for list)
- Adjunct data
- Structure serialization indicator

The STRDATA ENTRYNAME parameter is associated with:
- The ENTRYNAME parameter of the IXLLIST macro
- The NAME parameter of the IXLCACHE macro

An example is:
COMMAND ====> STRDATA ENTRYNAME(ELEMENT2)

ARB
Requests formatting of the associated request block (ARB), which contains a list of all the valid ranges specified on the STRLIST option of the DUMP, CHNGDUMP, or SLIP operator command. If the dump was taken by a recovery routine, the ARB contains the data derived from the IHABLDP macro.
**STRDATA subcommand**

*Note:* The actual dump parameters may have been modified to be consistent with the structure specifications. For example, if castout classes 1 to 2000 were requested to be dumped, but only castout classes 1 to 10 were valid, the ARB input were modified before the dump was taken.

The report output is:

```
STRDATA ALL STRUCTURES SUMMARY REPORT
ASSOCIATED REQUEST BLOCK REPORT
```

The output fields are:
- Total ranges requested in ARB
- Last range dumped
- Range number
- Dump object type for each range requested. For example, list number or lock entries.

An example is:

```
COMMAND ===> STRDATA ARB
ENTRYPOS (ALL | entrypos,entrypos:entrypos,...)
```

Requests information about an entry in a particular position, or range of positions. This parameter is valid only with COCLASS, STGCLASS, or LISTNUM. The position of an entry is counted from the head or tail of the queue, depending on the ORDER parameter.

The `entrypos` can be a single position, a range of positions, or a list of noncontiguous positions. When you specify a range, separate the first and last positions in the range by a colon. When you specify a list, separate the list members with commas.

The report output is:

```
STRDATA ALL STRUCTURES SUMMARY REPORT
LIST NUMBER ENTRY POSITION SUMMARY REPORT
```

*Note:* If STGCLASS is also specified, IPCS also displays the STORAGE CLASS ENTRY POSITION SUMMARY REPORT. If STGCLASS or COCLASS is specified, IPCS also displays the CASTOUT CLASS ENTRY POSITION SUMMARY REPORT.

The output fields for each entry specified are:
- List number
- List number status
- Summary of the list controls
- Entry key, if requested
- Order indicator
- For each entry requested:
  - Entry position
  - List entry controls
  - Adjunct data
  - Serialization indicator

The STRDATA ENTRYPOS parameter is associated with:
- The LISTDIR parameter of the IXLLIST macro
- The COCLASS and STGCLASS parameters of the IXLCACHE macro

An example is:

```
COMMAND ===> STRDATA LISTNUM(ALL) ENTRYPOS(2)
```

**ORDER (HEAD | TAIL)**

Specifies the order for entries to be displayed. Specify ORDER only with
**STRDATA subcommand**

ENTRYPOS. The position number specified in ENTRYPOS depends on whether you are counting from the head or the tail of the queue.

HEAD is the default and specifies that entries be located from at the top of a list or the head of a queue. For a storage class, the head of a queue is the least recently referenced entry. For a cast-out class, the head of a queue is the least recently changed entry.

TAIL specifies that entries be located from the end of a list or the tail of a queue. For a storage class, the tail of a queue is the most recently referenced entry. For a cast-out class, the tail of a queue is the most recently changed entry.

For example, if there are 35 entries on list number 2, and you want the 30th entry from the start of the queue, specify either of the following to display the same entry:

```
COMMAND ===> STRDATA LISTNUM(2) ENTRYPOS(30) ORDER(HEAD)
COMMAND ===> STRDATA LISTNUM(2) ENTRYPOS(6) ORDER(TAIL)
```

**ENTRYKEY(entrykey,entrykey...)**

Requests the display of a list entry with the specified key or the event monitor controls (EMCs) associated with a list entry and the specified key. This parameter can be used only for LISTNUM (when ENTRYPOS is specified) and EMCONTROLS processing.

The report output is:

```
STRDATA ALL STRUCTURES SUMMARY REPORT
LIST NUMBER ENTRYKEY ENTRY POSITION SUMMARY REPORT
```

The output fields are:
- List number
- List number status
- Summary of the list controls
- Entry key
- Order indicator
- For each entry requested:
  - Entry position
  - List entry controls
  - Adjunct data
  - Serialization indicator

The STRDATA ENTRYKEY parameter is associated with:
- The LISTCNTLTYPE=ENTRY and REFOPTION=KEY parameters of the IXLCONN macro
- The ENTRYKEY parameter of the IXLLIST macro

For example, the entry positions are in an order that is relative to the entry key. The following shows queue 1, which is a list with 5 entries:

<table>
<thead>
<tr>
<th>entry 1</th>
<th>key 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry 2</td>
<td>key 2</td>
</tr>
<tr>
<td>entry 3</td>
<td>key 2</td>
</tr>
<tr>
<td>entry 4</td>
<td>key 2</td>
</tr>
<tr>
<td>entry 5</td>
<td>key 3</td>
</tr>
</tbody>
</table>

**LIST 2**

<table>
<thead>
<tr>
<th>Head of Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry 1</td>
</tr>
<tr>
<td>entry 2</td>
</tr>
<tr>
<td>entry 3</td>
</tr>
<tr>
<td>entry 4</td>
</tr>
<tr>
<td>entry 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tail of Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry 1</td>
</tr>
<tr>
<td>entry 2</td>
</tr>
<tr>
<td>entry 3</td>
</tr>
</tbody>
</table>
STRDATA subcommand

To display the second and third entries for key 2 from the head of list 2, enter the following command:

COMMAND ==> STRDATA LISTNUM(2) ENTRYPOSITIONS(2,3) ENTRYKEY(02) ORDER(HEAD)

The following shows how entries with the same key are considered a separate queue, queue 2, so you get back entry 3 as entry position 2 and entry 4 as entry position 3.

<table>
<thead>
<tr>
<th>LIST 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry 1 key 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Head of Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry 2 key 2 position 1</td>
</tr>
<tr>
<td>entry 3 key 2 position 2</td>
</tr>
<tr>
<td>entry 4 key 2 position 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tail of Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry 5 key 3</td>
</tr>
</tbody>
</table>

Another example is:

COMMAND ==> STRDATA ENTRYKEY(02) LISTNUM(ALL) ENTRYPOSITIONS(ALL)

SUMMARY subcommand — summarize control block fields

Use the SUMMARY subcommand to display or print dump data associated with one or more specified address spaces.

SUMMARY produces different diagnostic reports depending on the report type parameter, FORMAT, KEYFIELD, JOBSUMMARY, and TCBSUMMARY, and the address space selection parameters, ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, and JOBLIST. Specify different parameters to selectively display the information you want to see.

Note: Installation exit routines can be invoked at the system, address space, and task level for each of the parameters in the SUMMARY subcommand.

Related subcommands
LISTSYM
RUNCHAIN
SCAN
SELECT
STATUS

Syntax
SUMMARY subcommand

{ SUMMARY }
{ SUMM  }

------ Report Type Parameters -------------------------------

[ KEYFIELD [REGISTERS | NOREGISTERS] ]
[ FORMAT ] [ DIALOG ]
[ EXCLUDE [GLOBAL | JPQ | LOADLIST] ]
[ TCBADDR(address-list) ]
[ TCBSUMMARY ]
[ JOBSUMMARY ]

------ Address Space Selection Parameters -----------------

[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR | ANOMALY ]
[ ASIDLIST(asidlist) ]
[ JOBLIST(joblist) | JOBNAME(joblist) ]

------ SETDEF-Defined Parameters -----------------------------

Note: You can override the following SETDEF parameters.

See “SETDEF subcommand — set defaults” on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAMES(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

Report Type Parameters

Use these parameters to select the type of report. If you omit a report type parameter, the default is KEYFIELD.

KEYFIELD

Presents the information in the ASCB, TCB, and RB key fields associated with the specified address space(s).

Information included pertains to these fields:

ASCB fields:

<table>
<thead>
<tr>
<th>AFFN</th>
<th>FLG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASID</td>
<td>FWDP</td>
</tr>
<tr>
<td>ASSB</td>
<td>LOCK</td>
</tr>
</tbody>
</table>
SUMMARY subcommand

TCB fields:
- BITS
- CMP
- DAR
- DSP
- FBYT1
- JSCB
- LMP
- NDSP
- PKF
- RTWA
- STAB
- STCB
- TSFLG

RB fields:
- WLIC
- OPSW
- LINK

CDE fields:
- NAME
- ENTPT

REGISTERS or NOREGISTERS

Specifies or suppresses display of the general purpose registers for each TCB/RB. Specify this parameter only when you specify KEYFIELD or default to KEYFIELD. If you specify FORMAT, JOBSUMMARY, or TCBSUMMARY and either REGISTERS or NOREGISTERS, IPCS processing ignores REGISTERS or NOREGISTERS.

REGISTERS specifies that registers are to be shown. The abbreviation REGS is accepted for this parameter.

NOREGISTERS suppresses the registers. The abbreviation NOREGS is accepted for this parameter.

If you omit both REGISTERS and NOREGISTERS, the default is NOREGISTERS.

FORMAT

Specifies a report containing the major control blocks associated with the specified address space or spaces. The blocks are, for example:
- ASCB
- ASSB
- ASXB
- Authorization table
- CDE
- DEB
- EED
- ENQ/DEQ suspend queue
- Extent list (XLIST)
- General CMS suspend queue
- Global service manager queue
- Job pack queue
- Linkage stack
- List of control blocks associated with open data sets
- Load list
- Local lock suspend queue
- Local service manager queue
- Local suspended SRB queue
Processor related work unit queues
RB
RSM suspended SRB deferred requests list
RSM suspended SRB I/O wait list
RSM suspended SRB cross memory deferred requests list
RSM suspended SRB cross memory I/O wait list
RTCT (only if CURRENT is specified or defaulted)
SMF CMS suspend queue
STCB
STKE
System work unit queue
TCB and TCBEXT2
TIOT
XSB

Note: For ASCBs, TCBs, CDEs, the extent list, and the load list, the bits in significant flag byte fields are explained (decoded).

After these items are formatted, IPCS invokes additional installation-supplied or other IBM-supplied exits to format control blocks.

If access registers are formatted, IPCS can identify the data space associated with the access register if the data space is accessible in the dumped environment; storage from the data space does not need to be dumped to enable the identification.

DIALOG
Directs the SUMMARY subcommand to present a data entry panel rather than accepting options in subcommand format.

EXCLUDE(GLOBAL | JPQ | LOADLIST)
Directs SUMMARY FORMAT to omit portions of the report that it normally produces.
- EXCLUDE(GLOBAL) causes global SRB formatting to be omitted.
- EXCLUDE(JPQ) causes job pack queue formatting to be omitted.
- EXCLUDE(LOADLIST) cause load list formatting to be omitted.

TCBADDR(address-list)
Directs SUMMARY FORMAT to limit its formatting related to TCBs to those whose addresses are listed. You can enter TCB addresses using decimal, hexadecimal (X'xxx'), or binary (B'bbb') format. ADDRTCB is an alias of the TCBADDR keyword.

TCBSUMMARY
Specifies a report containing a summary of the task control blocks (TCBs) for each address space processed. Each TCB summary contains:
- Job name
- ASCB name and address
- TCB name and address
- CMP field
- PKF field
- TSFLG field

If the TCBRTWA field is nonzero, the following fields are also displayed for each TCB:
- DAR field
- RTWA field
- FB YT1 field
JOBSUMMARY
Specifies a report containing a summary of the status of address spaces for a job. The report contains:
- Active CPU list
- Scheduled services
For each address space specified:
- Jobname
- ASCB location
- ASID
- Status of the address space
- Local service manager queue
- Local service priority queue
- TCB locations, completion codes, and the active indicator
- A problem list of TCBs
- Local lock suspend queue
- Local suspended SRB queue

Address Space Selection Parameters
Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs). If you omit these parameters, the default is CURRENT. For more information, see the select ASID service in z/OS MVS IPCS Customization.

You can specify several address space selection parameters. An address space might meet more than one selection criterion. The selection criterion (or criteria) that is met for each address space appears in the output. No address space is processed more than once.

ALL
Specifies processing of all address spaces in the dump.

CURRENT
Specifies the processing of each address space that was active when the dump was generated.

ERROR
Specifies processing of control blocks for any address space with an MVS error indicator or containing a task with an error indicator.

TCBERROR or ANOMALY
Specifies processing of control blocks for any address space containing a task with an error indicator. Blocks for address spaces with an error indicator are not processed.

ASIDLIST(asidlist)
Specifies a list of ASIDs for address spaces to be processed.

The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed in the notation X'nnn', F'nnn', or B'nnn'. An unqualified number is assumed to be fixed.

This subcommand does not process summary dump records (ASID X'FFFA').

JOBLIST(joblist) or JOBNAME(joblist)
Specifies a list of job names whose associated address spaces are to be
SUMMARY subcommand

processed. Use commas to separate the job names in the list; do not enclose
job names in apostrophes; and do not specify a range of job names.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the
return codes produced by the SUMMARY subcommand.

Example 1
Produce a KEYFIELD report.
  Action
  COMMAND ===> summary keyfield current
  Result
  IPCS produces the following output:

1  * * *  K E Y F I E L D S  * * *
2  SELECTED BY: CURRENT ERROR

ASCB: 00F6A000
  FWDP..... 00F6E800  ASID..... 0021  CSCB..... 02DAE530
  TSB...... 00000000  AFFN...... FFFF  ASXB..... 007FE038  DSP1...... 00
  FLG2..... 00  SRBS...... 0000  LOCK...... 00000000
  ASSB..... 01A72280

TCB: 007FE240
  CMP....... 00000000  PKF....... 00  LMP....... FF  DSP....... FF
  TSFLG..... 00  STAB...... 007FF6E0  NDSPP..... 00000000
  JSCB..... 007FFDFC  BITS...... 00000000  DAR....... 00
  RTWA..... 00000000  FBYT1..... 00
  Task non-dispatchability flags from TCBFLGS4:
    Top RB is in a wait

PRB: 007FF98
  WLIC...... 00020001  OPSSW...... 070C1000  B10234C0
  LINK...... 017FE240
  EP........ IEAVAR00  ENTPT..... 82B6C00

TCB: 007FF388
  CMP....... 00000000  PKF....... 00  LMP....... FF  DSP....... FF
  TSFLG..... 00  STAB...... 007FF680  NDSPP..... 00000000
  JSCB..... 007FFDFC  BITS...... 00000000  DAR....... 00
  RTWA..... 00000000  FBYT1..... 00
  Task non-dispatchability flags from TCBFLGS4:
    Top RB is in a wait
SUMMARY subcommand

Example 2

Produce a FORMAT report.

- **Action**
  
  COMMAND ===> summary format current

- **Result**
  
  IPCS produces the following output:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indicates the report type.</td>
</tr>
<tr>
<td>2</td>
<td>Indicates the selection criteria that were met.</td>
</tr>
</tbody>
</table>
**SUMMARY subcommand**

```
1  * * * * F O R M A T * * * *
2  GLOBAL SERVICE MANAGER QUEUE
    QUEUE IS EMPTY

    LOCAL SERVICE MANAGER QUEUE
    QUEUE IS EMPTY

    SYSTEM WORK UNIT QUEUE
    WEB QUEUE IS EMPTY

    CMS SMF LOCK SUSPEND WEB QUEUE
    WEB QUEUE IS EMPTY

    CMS ENQ/DEQ LOCK SUSPEND WEB QUEUE
    WEB QUEUE IS EMPTY

    GENERAL CMS LOCK SUSPEND WEB QUEUE
    WEB QUEUE IS EMPTY

    CPU = 01
    PROCESSOR RELATED WORK UNIT QUEUE
    WEB QUEUE IS EMPTY

RSM processing on a non-stand-alone dump may generate inconsistent
data and false validity check failures.
Data space information may be incomplete for RSM. Storage not in
dump.

    RSM SUSPENDED SRB DEFERRED REQUESTS LIST
    SSRB LIST IS EMPTY

    RSM SUSPENDED SRB I/O WAIT LIST
    SSRB LIST IS EMPTY

    RSM SUSPENDED SRB CROSS MEMORY DEFERRED REQUEST LIST
    SSRB LIST IS EMPTY

    RSM SUSPENDED SRB CROSS MEMORY I/O WAIT LIST
    SSRB LIST IS EMPTY
```
SELECTED BY: CURRENT ERROR

JOB TC

ASCB: 00F6AD00

+0000 ASCB.... ASCB  FWD.... 00F6E800  BWDP.... 00F63D00
+000C R00C..... 00000000  SVRB..... 007FF808  SYNC..... 000000F0
+0018 I0SP..... 00000000  R01C..... 0000  WQID..... 0000
+0020 SAWQ..... 00000000  ASID..... 0021  R026..... 0000
+0028 LL5..... 00  HLHI..... 01  DPH..... 0073
+002C R02C..... 00000000  LDA..... 7FF14EB0  RSMF..... 00
+0035 R035..... 000000  CSWB..... 020AE550  TSB..... 00000000
+0040 EJST..... 00000000  1250047B
+0048 EMST..... A7843AC5  96613D05  JSTL..... 0000023D
+0054 ECB..... 807FF9B  UBT..... A7843AFAF  TLCH..... 00000000
+0060 DUMP..... 007FF3BB  AFFN..... FFFF  RCTF..... 01
+0067 FLGI..... 00  TMCH..... 00000000  ASXB..... 007FE038
+0070 SWCT..... 0024  DSP1..... 00  FLG2..... 00
+0074 RSV..... 0000  SRBS..... 0000  LLWQ..... 00000000
+007C RCTP..... 007FE240  LOCK..... 00000000  LSQ..... 00000000
+0080 QECB..... 00000000  MECE..... 40000000  OUCB..... 01A720F6
+0094 OUXB..... 01C47F8  FMCT..... 0000  LEVL..... 03
+0098 FL2A..... 80  XMPQ..... 00000000  IQEA..... 00000000
+00A4 RTMC..... 00000000  MCC..... 00000000  JBNI..... 00000000
+00B0 JBNS..... 00F42AD8  SRQ1..... 00  SRQ2..... 00
+00B6 SRQ3..... 00  SRQ4..... 00  VGTT..... 00000000
+00BC PCTT..... 00000000  SSRB..... 0000  SMCT..... 00
+00C3 SRBM..... 07  SWTL..... 00000000  000C47B
+00CB SRBT..... 00000000  0033F200  ROD0..... 00000000
+00D4 00000000  TCBS..... 00000001  LSQ..... 00000000
+00E0 WPBR..... 007FFA40  NDP..... 73  TNDF..... FF
+00E6 NTSF..... FF  10DP..... FF  LOCI..... 00000000
+00EC CMLW..... 00000000  CMLC..... 00000000  SS01..... 00000000
+00F7 SS04..... 00  ASTE..... 0142EB40  LTOV..... 7FFE200
+0100 ATOV..... 7FFE3C8  ETC..... 0000  ETCT..... 0000
+0108 LXIR..... 0000  AXR..... 0000  STKH..... 007FEA50
+0110 GQEL..... 00000000  LQEL..... 02E07990  GSYN..... 00000000
+011C XTCB..... 007FFC378  CSL..... 00  CSS..... 00
+0122 R122..... 0000  GXL..... 01B90FE0
+0128 EATT..... 00000000  1038360E
+0130 INTS..... A7843AAB  4D12702  LL1..... 00
+0139 LL2..... 00  LL3..... 00  LL4..... 00
+013C RCMS..... 00000000  I0SC..... 00000006  PKML..... 0000
+0146 XCNT..... 01F4  NSQA..... 00000000  ASM..... 01A82028
+0150 ASB..... 01A72280  TCME..... 00000000  GQIR..... 00000000
+015C R15C..... 00000000  00000000  CREQ..... 00000000
+0160 RSM0..... 01B90E0  AV1..... 00  AVM2..... 00
+0172 AGEN..... 0000  ARC..... 00000000  RSMA..... 01B90D10
+017C DCTI..... 00000041

Chapter 5. IPCS subcommands 5-271
SUMMARY subcommand

LOCAL LOCK SUSPEND WEB QUEUE
QUEUE IS EMPTY

LOCAL LOCK SUSPEND SERVICE WEB QUEUE
QUEUE IS EMPTY

LOCAL SUSPENDED SRB QUEUE
QUEUE IS EMPTY

ASSB: 01A72280
+0000 ASSB..... ASSB VAFN..... 00000000
+0008 EVST..... 00000000 00000000
+0010 VFAT..... 00000000 00000000 XMF1..... 00
+0018 XMF2..... 00 XMCC..... 0000 CBTP..... 00000000
+0020 VSC..... 00000000 NVSC..... 00000000 ASRR..... 00000000
+0022 DEXP..... 00000000 STKN..... 00000840 00000002
+0028 BPSC..... 00000000 CSCT..... 00000000 BALV..... 7FFBF00
+0044 BALD..... 011D1F00 XMSE..... 00000000 TSQN..... 00000004
+0060 VCNT..... 00000000 PALV..... 7FF9E00 ASE1..... 00000000
+0068 RMA..... 02DE1000 HST..... 00000000 00000000
+0070 IJPT..... 00000000 002AA100 ANEC..... 00000000
+0078 SD0V..... 00000000 MCSO..... 00000000 DFAS..... 01A6A40
+0080 FLG0..... 00 FLG1..... 00 FLG2..... 00
+0088 FLG3..... 00 ASCB..... 00F6A000 ASRF..... 01A5680
+0090 ASRB..... 01B90E80 SSD..... 00000000 R094..... 00000000
+0098 LASB..... 037EB360 80000000 SCH..... 00000000
+00A0 FSC..... 00000000 JSAB..... 01B90C90 RCTW..... 01A6F440
+00B0 CREQ..... 00000000 CRQA..... 00000001 TLMI..... 00000000
+00C0 SDAS..... 7FF9E9C8 TPIN..... 00000000 SP1N..... 00000000
+00CB ECT1..... 00000000 ECT2..... 00000000 MT4..... 00000000
+00D0 DFP..... 00000000 R0D8..... 00000000 NTTP..... 00000000
+00E0 OECB..... 00000000 OASB..... 09F23A0 XSBA..... 7FFE9E00
+00EC R0EC..... 00000000 VAB..... 01D0B200 LMAB..... 00000000
+00F0 R0FB..... 00000000 00000000 00000000 00000000
+0104 R104..... 00000000 R108..... 00000000 TPMA..... 00000000
+0110 R0SU..... 00000000 TPMT..... 00000000 SSDT..... 00000000
+011C TAWQ..... 01A3D190 WMCL..... 01A6F700 WS3S..... 01A6F1C0
+0128 WSSS..... 00000000 CAPO..... 00000000 R130..... 00000000
+0134 00000000 00000000 00000000 00000000 00000000
+0148 00000000 00000000 JNBI..... 00000000
+0158 JBNS..... 00000000

No formatting support for JSAB
Insufficient storage for exit IAZJSABP
SUMMARY subcommand

ASXB: 007FE038
+0000  ASXB.....  ASXB     FTCB.....  007FE240   LTCB.....  007FC378
+000C  TCBS.....  0004   R00E.....  0000   MPST.....  00000000
+0014  LMA.....  00000000   VFVT.....  00000000   SAF.....  00000000
+0020  IHSA.....  007FE598   FLSA.....  000000DB  00F77500  00FD1770
+0030  812ED762  00000040  012EE761  00F77500  812ED096
+0044  00000000  00FFA488  00000000  00F77500  00F77400
+0058  7FFE44F0  81161DC2  7FFE493C  00000C60  812B0132
+006C  OMCB.....  00000000   SPSA.....  007FEA68   RSMD.....  00000000
+0078  RCTD.....  007FE480   DECW.....  00FF0A0   OUSB.....  7FFD1C0
+0084  CRWK.....  00000000   PRG.....  00000000  00000000  00000000
+0094  00000000   PSWD.....  00000000  00000000
+00A0  SIRB.....  007FE3DB   ETXK.....  007FE240   FIOE.....  00000000
+00AC  LIQE.....  00000000   FRQE.....  00000000   LRQE.....  00000000
+00BB  FSRRB.....  00000000   LSRB.....  00000000   USER..... TC
+00C7  SFLG.....  00   SENV.....  007FCF58   R0CC.....  00000000
+00D0  NSSA.....  7FFFCDE0   NSCT.....  00000000   CRB1.....  00
+00D9  CRB2.....  00   CRB3.....  00   CRB4.....  00
+00DC  PT0E.....  00000000   ROEO.....  00000000   JSVT.....  00000000
+00EB  DIVW.....  00000000   ROEC.....  00000000

1 Indicates the report type.
2 Shows the status of the various queues and SSRB lists.
3 Indicates the selection criteria that were met.

Example 3

Produce a TCBSUMMARY report.

- Action
  COMMAND ===> summary tcbsummary current
- Result
  IPCS produces the following output:
### SUMMARY subcommand

1. ******TCB SUMMARY****
2. JOB TC ASCB021 AT 00F6AD00
3. SELECTED BY: CURRENT ERROR

4. TCB: 007FE240
   CMP...... 00000000 PKF...... 00 LMP...... FF DSP...... FF
   TSFLG.... 00 STAB...... 007FF6E0 NDSP...... 00000000
   JSCB..... 007FFDFC BITS...... 00000000 FBYT1.... 00

TCB: 007FF3B8
   CMP...... 00000000 PKF...... 00 LMP...... FF DSP...... FF
   TSFLG.... 00 STAB...... 007FF6B0 NDSP...... 00000000
   JSCB..... 007FFDFC BITS...... 00000000 FBYT1.... 00

TCB: 007FF12B
   CMP...... 00000000 PKF...... 80 LMP...... FF DSP...... FF
   TSFLG.... 00 STAB...... 007FF620 NDSP...... 00000000
   JSCB..... 007FCC14 BITS...... 00000000 FBYT1.... 00

TCB: 007FC378
   CMP...... 80522000 PKF...... 80 FLGS...... 84000000 00
   LMP...... FF DSP...... FF TSFLG.... 20
   STAB...... 007FF5F0 NDSP...... 00000000 JSCB..... 007FCA0C
   BITS...... 00000000 DAR...... 00 RTWA..... 7F6FE090 ABCUR.... 00
   FBYT1.... 88

1. Indicates the report type.
2. Provides the name of the job, the address space, and its address.
3. Indicates the selection criteria that were meet.
4. Provides the address of the first TCB in the chain.

#### Example 4

Produce a JOBSUMMARY report.

- **Action**
  
  `COMMAND ==> summary jobsummary current`

- **Result**

  IPCS produces the following output:
**SUMMARY subcommand**

1. **SYSTEM SUMMARY**
   *** ACTIVITY CPU LIST ***
   CPU 0001 - SERVICE REQUEST MODE

   *** SCHEDULED SERVICES ***
   - GLOBAL SERVICE MANAGER QUEUE
     QUEUE IS EMPTY
   - LOCAL SERVICE MANAGER QUEUE
     QUEUE IS EMPTY
   - SYSTEM WORK UNIT QUEUE
     WEB QUEUE IS EMPTY
   - CMS SMF LOCK SUSPEND WEB QUEUE
     WEB QUEUE IS EMPTY
   - CMS ENQ/DEQ LOCK SUSPEND WEB QUEUE
     WEB QUEUE IS EMPTY
   - GENERAL CMS LOCK SUSPEND WEB QUEUE
     WEB QUEUE IS EMPTY
   - PROCESSOR RELATED WORK UNIT QUEUE
     WEB QUEUE IS EMPTY

   CPU 01
   RSM processing on a non-stand-alone dump may generate inconsistent data and false validity check failures. Data space information may be incomplete for RSM. Storage not in dump.

   - RSM SUSPENDED SRB DEFERRED REQUESTS LIST
     SSRB LIST IS EMPTY
   - RSM SUSPENDED SRB I/O WAIT LIST
     SSRB LIST IS EMPTY
   - RSM SUSPENDED SRB CROSS MEMORY DEFERRED REQUEST LIST
     SSRB LIST IS EMPTY
   - RSM SUSPENDED SRB CROSS MEMORY I/O WAIT LIST
     SSRB LIST IS EMPTY

2. **JOB SUMMARY**
   ---------------------------------------------------------------------
   SELECTED BY: CURRENT
   JOBNAME TC  ASCB 00F6AD00  NEXT 00F6E800  PREV 00F63D00  ASID 0021
   TCB 007FE240  NEXT 007FF3B8  PREV 00000000  COMP 00000000
   TCB 007FF3B8  NEXT 007FF128  PREV 007FE240  COMP 00000000
   TCB 007FF128  NEXT 007FC378  PREV 007FF3B8  COMP 00000000
   TCB 007FC378  NEXT 00000000  PREV 007FF128  COMP 88522000
   ---------------------------------------------------------------------

3. **PROBLEM LIST**
   - JOB TC ASID 0021 TCB 007FC378 ABEND CODE- 88522000 DAR 00
   - JOB TC ASID 0021 TCB 007FC378 SET TEMPORARY NON-DISPATCHABLE
     FLGS4 00 FLGS5 00 SCNDY 00000000 DAR 00 STPCT 00

4. **NO MACHINE CHECKS IN PROCESS**
   - NO ABENDS DETECTED FOR ASCBS
   - NO NON-DISPATCHABLE ASCBS DETECTED

---

Chapter 5. IPCS subcommands 5-275
SYMDEF subcommand — display an entry in the system symbol table

Use the SYMDEF subcommand to display an entry in the system symbol table, which contains static system symbols.

You can use IPCS-supplied traps with the SYMDEF command.

Notes:
1. SYMDEF displays the static system symbols in the system symbol table, which are specified (or the defaults accepted) in the IEASYMxx parmlib member. System symbols are different from the IPCS symbols described in "IPCS symbols" on page A-1.
2. The output that SYMDEF generates contains information for diagnostic use. The IBM Support Center might ask you to provide this information for use in problem determination.

Related subcommands
None.

Syntax

SYMDEF [ NAME(symbol) ]

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232

[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

NAME(symbol)
   Displays the symbol table entry for the specified system symbol. When specifying symbol, do not include the ampersand (&) or the period (.) that are normally part of symbol notation.
   If you do not specify this parameter, the system displays the entire symbol table.

Return Codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the SYMDEF subcommand.
SYSTRACE subcommand — format system trace entries

Use the SYSTRACE subcommand to format system trace entries for all address spaces.

Syntax

{ SYSTRACE [ TIME(HEX | GMT | LOCAL) ]

-------- Report Type Parameters -----------------------------

[PERFDATA([SHOWTRC] [DOWHERE] [SIGCPU(sss.ddddd)])]

-------- Data Selection Parameters -----------------------

[ EXCLUDE(BR) ]
[ EXCLUDE(MODE) ]
[ SORTCPU([mm/dd/yy, hh:mm:ss:dddd],N) | (mm/dd/yy, hh:mm:ss:dddd)]
[ START(mm/dd/yy, hh.mm.ss:dddd) ]
[ STOP(mm/dd/yy, hh.mm.ss:dddd) ]
[ CPU(cpu-address-range-list) ]
[ CPUMASK(cpu-hexadecimal-mask) ]
[ CPUTYPE(ZAAP|ZIIP|STANDARD) ]
[ TCB(TCB-list) ]
[ TTCH(TTCH-address | LIST) ]
[ WEB(WEB-list) ]

-------- Address Space Selection Parameters -------------

[ ALL ]
[ CURRENT ]
[ ERROR ]
[ TCBERROR ]
[ ASIDLIST(asidlist) ]
[ JOBLIST(joblist) | JOBNAME(joblist) ]
SYSTRACE subcommand

Parameters

**TIME(HEX | GMT | LOCAL)**

Specifies the type of time stamp for the system trace entries, as follows:
- HEX specifies a hexadecimal time stamp.
- GMT specifies a time stamp in Greenwich mean time.
- LOCAL specifies a time stamp in local time.

Report Type Parameters

Use these parameters to select the type of report.

**PERFDATA ([SHOWTRC] [DOWHERE] [SIGCPU(time)])**

Requests summary information for the performance data report. The intent of the PERFDATA parameter is to help identify which trace entries are using large amounts of time as derived from the output of SYSTRACE ALL TIME(LOCAL) where it is also determined that trace data is available from all processors. As SYSTRACE entries are the sole source of the PERFDATA calculations, output is not as precise as other forms of time use reporting (such as RMF reports). PERFDATA output is reported against the job running in the address space at the time of the dump.

**SHOWTRC**

Requests that a system trace table be displayed in the output. If you do not specify this parameter, the default is to exclude the system trace table.

**DOWHERE**

Requests WHERE commands to be issued for PSWs within CLKC and SRB analysis sections of PERFDATA option output, in order to display the area in a dump in which these addresses reside. This information may include the name of the load module, the name of a control block or the name of an area of storage containing the PSW address along with an offset. It is displayed in an extra field on the same row of a PERFDATA table as the PSW address.

**SIGCPU(sss.dddddd)**

Requests that CLKC analysis and WHERE analysis for SRB events are to be bypassed for events with CPU usage less than the specified time (in seconds). If you do not specify this parameter, the default is SIGCPU(0.1).

- **sss** Represents seconds. You can specify one to three decimal digits.
SYSTRACE subcommand

\texttt{dddddd}  
Represents decimal fractions of seconds. You can specify one to six decimal digits.

If a hexadecimal time stamp type is selected when specifying the \texttt{PERFDATA} parameter, the time values in the \texttt{PERFDATA} report are generated in GMT format.

Data Selection Parameters
Use these parameters to limit the scope of the data in the report. If you omit these parameters, the default is to include all trace entries.

\textbf{EXCLUDE(BR)}
Suppresses formatting of trace table entries for branch tracing if any were present in the dump. When you do not specify \texttt{EXCLUDE(BR)}, the formatted trace table shows all the types of trace table entries.

\textbf{EXCLUDE(MODE)}
Suppresses formatting of trace table entries for mode tracing if any were present in the dump. When you do not specify \texttt{EXCLUDE(MODE)}, the formatted trace table shows all the types of trace table entries.

\textbf{Note:} Specifying \texttt{EXCLUDE(BR,MODE)} suppresses formatting of trace table entries for both branch and mode tracing if any were present in the dump.

\textbf{SORTCPU}\[
\texttt{(mm/dd/yy, hh:mm:ss:dddddd, N) \mid (mm/dd/yy, hh:mm:ss:dddddd)}\]
When the \texttt{SORTCPU} option is specified, IPCS displays trace entries for each CPU separately in ascending order by CPU address. \texttt{N} indicates the number of the trace entries before and after a specified time, which are displayed for each CPU.

If \texttt{(mm/dd/yy, hh:mm:ss:dddddd, N)} is not specified, or if \texttt{N} is zero (0), or if the number of system trace entries are less than what were specified in \texttt{N}, all entries are shown. If you omit \texttt{N}, the default value is 10. Specify the date and time in the \texttt{mm/dd/yy, hh:mm:ss:dddddd} format where:

\begin{itemize}
  \item \texttt{mm} Represents month; requires two decimal digit format.
  \item \texttt{dd} Represents day; requires two decimal digit format.
  \item \texttt{yy} Represents year; requires two decimal digit format.
  \item \texttt{hh} Represents hour; requires two decimal digit format.
  \item \texttt{mm} Represents minutes; requires two decimal digit format.
  \item \texttt{ss} Represents seconds; requires two decimal digit format.
  \item \texttt{dddddd} Represents decimal fractions of seconds; you can specify one to six decimal digits.
\end{itemize}

These rules apply to the date and time specifications:
- You need to specify both a time and a date on the \texttt{SORTCPU} parameter, but you do not have to specify the time down to the milliseconds.
- If you specify \texttt{TIME(HEX)} or \texttt{TIME(GMT)} in the SYSTRACE subcommand, the specified time is in GMT format. If you specify \texttt{TIME(LOCAL)}, the time is in the local time zone. When \texttt{TIME} is not specified, a default of \texttt{TIME(HEX)} leads to time in GMT format.
- To allow for copying and pasting of time from the systrace output, use colons or periods to delimit the time field.
SYSTRACE subcommand

Examples:
1. Show all data in CPU order:
   SYSTRACE ALL SORTCPU
2. Show data in CPU order, showing a default of 10 entries around 11 am GMT:
   SYSTRACE ALL TIME(GMT) SORTCPU(12/30/09,11)
3. Show data in CPU order, showing 5 entries around 11:45:21.939233 am local:
   SYSTRACE ALL TIME(LOCAL) SORTCPU(12/30/09, 11:45:21.939233,5)

Note: When SORTCPU is specified, a default of ALL (address spaces) is assumed and any other specification for filtering by ASID is incompatible, such as the CURRENT, ERROR, TCBERROR, ASIDLIST, JOBLIST and JOBNNAME keywords. The SORTCPU parameter is compatible with the following existing SYSTRACE parameters: TCB, WEB, CPU, TIME, EXCLUDE, TTCH, START, STOP and ALL.

START(mm/dd/yy, hh.mm.ss.dddddd)
Specifies the beginning date and time for the trace entries to be formatted. When you do not specify START, IPCS starts at the beginning of the trace entries. Specify the date and time in the mm/dd/yy.hh.mm.ss.dddddd format, where:

mm represents months
dd represents days
yy represents years
hh represents hours
mm represents minutes
ss represents seconds
ddddd represents decimal fractions of seconds

These rules apply to the date and time specifications:
- You must specify a date and time on the START parameter.
- The month and day can be specified in either single or double digits.
- Separate the date from the time with a comma.
- The time can be GMT, by default or specified in a GMT parameter, or local, if specified in a LOCAL parameter.
- Hours, minutes, and seconds can be specified in single or double digits.
- The time can be truncated anywhere on the right.
- The time can be left off completely, in which case, it will default to 00:00:00.000000 (midnight).
- To allow for copying and pasting of time from the systrace output, use colons or periods to delimit the time field.

Some examples of valid date formats are:
m/dd/yy
mm/d/yy
m/d/yy
mm/dd/yy

Some examples of valid time formats are:
**SYSTRACE subcommand**

`hh.mm.ss.dddddd`

`hh.mm.ss.dd`

`hh.mm.ss`

`h.m.s`

`hh.mm`

`hh`

Use START and STOP to reduce the number of trace entries formatted.

**STOP(mm/dd/yy, hh.mm.ss.dddddd)**

Specifies the ending date and time for the trace entries to be formatted. When you do not specify STOP, IPCS stops formatting after the last trace entry.

For guidelines on how to specify the date and time, see the START parameter.

**CPU(cpu-address-range-list)**

Limits formatting to trace entries for the central processors whose addresses are specified by `cpu-address-range-list`. Use a Store CPU Address (STAP) instruction to obtain the processor address.

When specifying the processor address range list, you can use a single address, a range of addresses, or a combination of individual addresses and address ranges. The eligible processor address is 1 through 63. You can specify the addresses in decimal (nn), hexadecimal (X'h'), or binary (B'bbbb') format. And you can use mixed format when multiple addresses are involved. The following examples provide more details:

- CPU(5) or CPU(X'3d') designates a single processor. Only the trace entries captured by the processor whose address is designated are selected.
- CPU(5:7), CPU(X'3d':X'3e'), or CPU(15:X'10') designate a range of processor addresses. The first processor address in a range must be less than or equal to the second. In the case of CPU(15:X'10'), both the decimal and hexadecimal format are used to specify the range.
- CPU(5 X'3d':X'3e' 15:X'10') designates a list. In this case the individual processor addresses and the address ranges are mixed.
- If you do not specify a processor on the option, the default is to format trace entries from all central processors.

**CPUMASK(cpu-hexadecimal-mask)**

Limits formatting to only the trace entries produced on the processors specified in the CPU (cpu-hexadecimal-mask). Specify the processors using a string of hexadecimal characters. Each hexadecimal character identifies four processors, leftmost bit designates lower processor address starting from zero. The processor maximum in z/OS defines a length of this hexadecimal string. The current processor maximum is 128. Therefore, the maximum length of the hexadecimal mask string is 32. See example 4. You can combine CPUMASK, CPU, and CPUTYPE as a union of sets. If all of the parameters are omitted, all processors are included as the default.

**Examples:**

1. To show all data for processors from 0 to 11:
   
   SYSTRACE ALL CPUMASK(FFF)

2. To show all data for processors from 0 to 3 and from 8 to 11:
   
   SYSTRACE ALL CPUMASK(FOFO)

3. To show all data for processor 0 and for processors from 5 to 10:
   
   SYSTRACE ALL CPUMASK(80) CPU(5:10)
4. To show data for processors from 0 to 127:
   SYSTRACE CPUMASK(FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF)
5. To show all data for processors 2, 3, 5, 8, 9, 10, 11:
   SYSTRACE ALL CPUMASK(34F)

**CPUTYPE(ZAAP|ZIIP|STANDARD)**

Limits the entries to specific processor types. CPUTYPE(ZAAP) selects all IBM System z Application Assist Processors (zAAP) in the configuration. CPUTYPE(ZIIP) selects all IBM System z Integrated Information Processors (zIIP) in the configuration. CPUTYPE(STANDARD) selects all standard processors. You can use the following abbreviations:
- ZA for ZAAP
- ZI for ZIIP
- CP or S for STANDARD.

To view a combination of CPU types, you can combine, in any order, the ZAAP, ZIIP, and STANDARD keywords. Use a space or comma as the delimiter. You can combine CPUTYPE, CPU, and CPUMASK as a union of sets. If all of the parameters are omitted, all processors are included as the default.

**Examples:**
1. To show all data for standard processors:
   SYSTRACE ALL CPUTYPE(STANDARD)
2. To show all data for ZAAP and ZIIP processors:
   SYSTRACE ALL CPUTYPE(ZAAP ZIIP)
3. To show all data for standard and ZAAP processors:
   SYSTRACE ALL CPUTYPE(S ZAAP)
4. To show all data for ZIIP processors and for processors from 5 to 10:
   SYSTRACE ALL CPUTYPE(ZI) CPU(5:10)
5. To show all data for ZAAP and ZIIP processors, for processor 0, 2, 5, 7, and for processor from 8 to 11:
   SYSTRACE ALL CPUTYPE(ZA ZIIP) CPU(0,2,5,7) CPUMASK(00F)

**TCB(TCB-list)**

Specifies the formatting of trace entries for the listed TCB address.

**TTCH(TTCH-address | LIST)**

Specifies the formatting of the trace table snapshot designated by the specified TTCH address. The TTCH address must be designated by a positive integer. See "Positive integers" on page 2-2 for a description of the notation allowed for a positive integer. If LIST is specified, a list of available TTCHs is produced and no trace entries are formatted. Within a standalone dump, there may be older trace table snapshots containing information that may be related to the problem for which the dump was taken.

For example, SYSTRACE TTCH(LIST) produces the following list of trace table snapshots:

<table>
<thead>
<tr>
<th>TTCH</th>
<th>ASID</th>
<th>TCB</th>
<th>TIME</th>
</tr>
</thead>
</table>

In the above example:
**TTCH**

The TTCH column shows the address of the trace table.
SYSTRACE subcommand

A snapshot in the dump. The '*' in front of the TTCH address indicates that it is a mini trace table snapshot. A mini trace table snapshot only contains the most current 64K of data for each CPU. System trace data requested by RTM and ABDUMPs will receive the mini snapshot when the number of concurrent snapshots could impact system availability.

A SYSMDUMP and IEATDUMP only contains the TTCH for that dump. To see if the trace table snapshot is a mini trace, look in the output of the IPCS Status Worksheet command. The WORKSHEET shows the Trace Table Control Header (TTCH) address. The SYSTRACE TTCH (TTCH-address) command displays "MINI SYSTEM TRACE TABLE" as the title for a mini trace.

| ASID | The ASID column shows the ASID the trace table in the dump. |
| TCB  | The TCB column shows the address of the TCB associated with this ASID. |
| TIME | The TIME column shows the time that the trace table snapshot was taken. |

WEB(WEB-list)
Specifies the formatting of trace entries running on behalf of the listed WEB (work element block) addresses.

Address Space Selection Parameters
Use these parameters to obtain trace entries from particular address spaces, which you specify by their address space identifiers (ASIDs). If you omit these parameters, the default is CURRENT. For more information, see the select ASID service in z/OS MVS IPCS Customization. You can specify several address space selection parameters.

ALL
Requests formatting of system trace entries for all address spaces.

CURRENT
Requests formatting of trace entries for the current address spaces on the following, depending on the dump being formatted:
- For an SVC dump, on the processor that requested the dump.
- For a stand-alone dump, on any processor at the time of the dump.

The current address spaces include the home, primary, and secondary address spaces. CURRENT is the default when you do not specify any other parameters.

ERROR
Specifies formatting of trace entries for any address space with an error indicator or containing a task with an error indicator.

TCBERROR
Specifies formatting of trace entries for any address space containing a task with an error indicator. Entries for address spaces with an error indicator are not formatted.

ASIDLIST(asidlist)
Requests formatting of trace entries for the specified address spaces or ranges of address spaces. An address space identifier (ASID) is 1 through 65535 and is specified in decimal (nnn or F'nnn'), hexadecimal (X'hhh'), or binary (B'bbbb'). In a range, separate the first and last ASIDs by a colon (:).
SYSTRACE subcommand

In the list of ASIDs, the ranges can overlap and duplicate asids can be specified.

**JOBLIST(joblist)**
**JOBNAME(joblist)**
Requests formatting of trace entries for the address spaces associated with the specified jobs. You can specify an unlimited number of job names.

**SETDEF-Defined Parameters**

**ACTIVE or MAIN or STORAGE**
**DATASET(dsname) or DSNAME(dsname)**
**FILE(ddname) or DDNAME(ddname)**
Specifies the source of the source description containing the system trace. If one of these parameters is not specified, the source is your current source.

ACTIVE, MAIN, or STORAGE specifies central storage as the source.

DSNAME or DATASET specifies the name of a cataloged data set as the source.

FILE or DDNAME specifies the ddname of a data set as the source.

**Return Codes**
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the SYSTRACE subcommand.

**Example**
For a list of system trace entries and an example of SYSTRACE output, see [z/OS MVS Diagnosis: Tools and Service Aids](https://www.ibm.com).
The `pgmname` specifies the name of a routine.

* specifies the following IBM-supplied TCB exit routines; these exit routines are specified by parmlib members embedded in the BLSCECT parmlib member.

<table>
<thead>
<tr>
<th>Exit Routine</th>
<th>Data Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>IECDAFMT</td>
<td>Data management control blocks</td>
</tr>
<tr>
<td>IECIOFMT</td>
<td>Input/output supervisor (IOS) and execute channel program (EXCP) control blocks</td>
</tr>
<tr>
<td>IEAVTFMT</td>
<td>Recovery termination management (RTM) control blocks</td>
</tr>
<tr>
<td>IEAVSSA1</td>
<td>Vector Facility data file.IEAVSSA1 exit routine</td>
</tr>
<tr>
<td>IEAVXD01</td>
<td>Access registers</td>
</tr>
<tr>
<td>IEAVD30</td>
<td>Linkage stack</td>
</tr>
</tbody>
</table>

An installation-supplied TCB exit routine that you can specify must:

- Be named with a maximum of 8 characters. The first character must be alphabetic.
- Reside in a library available to IPCS, such as a step library, job library, or link library.

For more information about writing installation TCB exit routines, see [z/OS MVS IPCS Customization](https://www.ibm.com/systems/z/os/zos/bkserv/).

**data-descr**

Specifies the address of the TCB to be passed to the exit routine. The data description parameter consists of five parts:

- An address (required)
- Address processing parameters (optional)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

**AMASK(mask)**

Specifies an integer mask that TCBEXIT is to AND to the dump addresses passed by the exit to the storage access and format service routines. The values of the mask can be only X’00FFFFFF’ or X’7FFFFFFF’ or the corresponding decimal or binary values.

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Severe error, an error condition or user request forced early end to the subcommand processing.</td>
</tr>
<tr>
<td>16</td>
<td>Ending error, an error condition from a called service routine forced an early end to the subcommand processing.</td>
</tr>
<tr>
<td>other</td>
<td>An exit generated return code.</td>
</tr>
</tbody>
</table>

**Example 1**

Invoke an IBM-supplied TCB exit to display RTM-related control blocks.

- **Action**
  
  COMMAND ===> tcbexit ieavtfmt 21C.%
TCBEXIT subcommand

- Result
This example invokes the IBM-supplied TCB exit routine (IEAVTFMT) that processes recovery termination management (RTM) control blocks. Using the indirect addressing notation (21C.%), addressability is established to the current TCB.

The output follows.

### NOT ALL EED'S AVAILABLE COULD BE ACCESSED ###
INVALID EED TYPE ENCOUNTERED AT LOCATION 009FF750

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EED1: 009FF750</td>
<td></td>
</tr>
<tr>
<td>+0000</td>
<td>E2E3DC5 009FEA68 00000000 00000000</td>
</tr>
<tr>
<td>+0010</td>
<td>0000001B 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0020</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0030</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0040</td>
<td>009FEB00 009FAFB0 00000000 00000000</td>
</tr>
<tr>
<td>+0050</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0060</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0070</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0080</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+0090</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+00A0</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+00B0</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+00C0</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+00D0</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+00E0</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>+00F0</td>
<td>D3DC3140 40404040 40404040 40404040</td>
</tr>
<tr>
<td>+0190</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
</tbody>
</table>

Example 2
Invoke all IBM-supplied TCB exits.

- Action
COMMAND ===> tcbexit * 21C.%

- Result
This example invokes all of the IBM-supplied TCB exit routines to process TCBs and related control blocks. Using the indirect addressing notation (21C.%), addressability is established to the current TCB.

Example 3
Invoke an installation-supplied TCB exit.

- Action
COMMAND ===> tcbexit testtcb 715b0.

- Result
This example invokes an installation-supplied routine TESTTCB, passing it the TCB address X'715B0'.

TRAPLIST subcommand — list the status of IPCS traps

Use the TRAPLIST subcommand to display the status of IPCS-supplied traps.

If you write your own installation exit and use one of the exit service routines, which are described in z/OS MVS IPCS Customization, use the TRAPON, TRAPOFF, and TRAPLIST subcommands to obtain diagnostic input and output information. You can also use these subcommands to set traps when executing IPCS code that uses the exit service routines.

Related subcommands
TRAPON
TRAPLIST subcommand

TRAPOFF
GO

Syntax

TRAPLIST { ALL } { code } { (code-list) }

Parameters

ALL
code
code-list

Identifies the IPCS-supplied traps whose status is to be displayed.

ALL specifies all IPCS-supplied traps. All is the default; if you do not specify any codes, IPCS displays the status of all traps.

code specifies a code that identifies an IPCS-supplied exit service routine.

code-list specifies a list of codes. When you specify a list, separate the list members with commas and enclose the list in parentheses. Otherwise, parentheses are optional.

The codes are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Exit Service Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>Storage access service</td>
</tr>
<tr>
<td>ADS</td>
<td>Add symptom service</td>
</tr>
<tr>
<td>CBF</td>
<td>Control block formatter service</td>
</tr>
<tr>
<td>CBS</td>
<td>Control block status service</td>
</tr>
<tr>
<td>CQE</td>
<td>Contention queue element create service</td>
</tr>
<tr>
<td>CSI</td>
<td>CSVINFO macro</td>
</tr>
<tr>
<td>ECT</td>
<td>ECT exit service</td>
</tr>
<tr>
<td>EQS</td>
<td>Equate symbol service</td>
</tr>
<tr>
<td>FMT</td>
<td>Format model processor service</td>
</tr>
<tr>
<td>GTS</td>
<td>Get symbol service</td>
</tr>
<tr>
<td>MAP</td>
<td>Map service</td>
</tr>
<tr>
<td>NAM</td>
<td>Name service</td>
</tr>
<tr>
<td>NDX</td>
<td>Table of contents service</td>
</tr>
<tr>
<td>NTK</td>
<td>NAME/TOKEN lookup service</td>
</tr>
<tr>
<td>PRT</td>
<td>Standard print service</td>
</tr>
<tr>
<td>PR2</td>
<td>Expanded print service</td>
</tr>
<tr>
<td>SEL</td>
<td>Select ASID service</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol service</td>
</tr>
<tr>
<td>WHS</td>
<td>WHERE service</td>
</tr>
</tbody>
</table>

Return Codes

See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the TRAPLIST subcommand.
TRAPLIST subcommand

Example 1
List the traps and the options associated with all the exit service routines.

- Action
  COMMAND ===> traplist all

- Result
  TRAPLIST generates the following output, after the TRAPON ALL INPUT
  OUTPUT subcommand activated all the trap options for each of the exit
  service routines.

Example 2
List the trap options associated with the storage access service.

- Action
  COMMAND ===> traplist acc

- Result
  TRAPLIST generates the following output line, after the TRAPON ACC
  subcommand activated the trap options for the storage access service.

TRAPOFF subcommand — deactivate IPCS traps

Use the TRAPOFF subcommand to deactivate IPCS-supplied traps. If you write
your own installation exit and use one of the exit service routines, which are
described in z/OS MVS IPCS Customization, use the TRAPON, TRAPOFF, and
TRAPLIST subcommands to obtain diagnostic input and output information. You
can also use these subcommands to set traps when executing IPCS code that uses
the exit service routines.

Related subcommands
TRAPON
TRAPLIST
GO

Syntax

TRAPOFF { ALL }{ code }{ (code-list) }
TRAPOFF subcommand

Parameters

ALL
code
code-list

Identifies the IPCS-supplied traps to be deactivated.

ALL specifies all IPCS-supplied traps. All is the default; if you do not specify any codes, IPCS deactivates all traps.

code specifies a code that identifies an IPCS-supplied exit service routine.

code-list specifies a list of codes. When you specify a list, separate the list members with commas and enclose the list in parentheses. Otherwise, parentheses are optional.

The codes are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Exit Service Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>Storage access service</td>
</tr>
<tr>
<td>ADS</td>
<td>Add symptom service</td>
</tr>
<tr>
<td>CBF</td>
<td>Control block formatter service</td>
</tr>
<tr>
<td>CBS</td>
<td>Control block status service</td>
</tr>
<tr>
<td>CQE</td>
<td>Contention queue element create service</td>
</tr>
<tr>
<td>CSI</td>
<td>CSVINFO macro</td>
</tr>
<tr>
<td>ECT</td>
<td>ECT exit service</td>
</tr>
<tr>
<td>EQS</td>
<td>Equate symbol service</td>
</tr>
<tr>
<td>FMT</td>
<td>Format model processor service</td>
</tr>
<tr>
<td>GTS</td>
<td>Get symbol service</td>
</tr>
<tr>
<td>MAP</td>
<td>Map service</td>
</tr>
<tr>
<td>NAM</td>
<td>Name service</td>
</tr>
<tr>
<td>NDX</td>
<td>Table of contents service</td>
</tr>
<tr>
<td>NTK</td>
<td>NAME/TOKEN lookup service</td>
</tr>
<tr>
<td>PRT</td>
<td>Standard print service</td>
</tr>
<tr>
<td>PR2</td>
<td>Expanded print service</td>
</tr>
<tr>
<td>SEL</td>
<td>Select ASID service</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol service</td>
</tr>
<tr>
<td>WHS</td>
<td>WHERE service</td>
</tr>
</tbody>
</table>

Return Codes

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the TRAPOFF subcommand.

Example 1

Turn off all traps associated with the exit service routines.

- Action
  COMMAND ===> trapoff all
- Result
  All IPCS-supplied traps are deactivated.
Example 2
Turn off the traps for the storage access and control block formatter service routines.

- Action
  COMMAND ===> trapoff (acc cbf)
- Result
  Traps for the storage access and the control block formatter services are deactivated.

TRAPON subcommand — activate IPCS traps

Use the TRAPON subcommand to activate IPCS-supplied traps. If you write your own installation exit and use one of the exit service routines, which are described in [z/OS MVS IPCS Customization], you can use the TRAPON, TRAPOFF, and TRAPLIST subcommands to obtain diagnostic input and output information. You can also use these subcommands to set traps when executing IPCS code that uses the exit service routines.

If a TRAPON subcommand requests several traps, IPCS activates only supported traps. Whenever an unsupported trap is requested, IPCS issues the following message:

BLS17014I Trap of INPUT/OUTPUT(trap) is not supported for service(sss)

where sss is the name of the requested exit service routine.

Note: Activated traps are not retained between IPCS sessions.

During STOP processing, all traps are temporarily deactivated until the GO subcommand is entered to resume the stopped operation. This temporary deactivation of traps is done because some of the subcommands available during STOP processing also use exit services and therefore are also trapped.

Related subcommands
- TRAPOFF
- TRAPLIST
- GO

Syntax

```
TRAPON
   { ALL }  
   { code }   
   (code-list)

   [INPUT   { [ABDPL] } ]
   [   { [DATA] } ]
   [   { [PARMS] } ]
   [   { [STOP] } ]
   [ ]
[NOINPUT ]

   [OUTPUT  { [RETC] } ]
   [   { [DATA] } ]
   [   { [PARMS] } ]
   [   { [STOP] } ]
   [   { [ERROR] } ]
   [ ]
[NOOUTPUT ]
```

Parameters
TRAPON subcommand

ALL

code

code-list

Identifies the IPCS-supplied traps to be activated.

ALL specifies all IPCS-supplied traps. All is the default; if you do not
specify any codes, IPCS activates all traps.

code specifies a code that identifies an IPCS-supplied exit service routine.

code-list specifies a list of codes. When you specify a list, separate the list
members with commas and enclose the list in parentheses. Otherwise,
parentheses are optional.

The codes are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Exit Service Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>Storage access service</td>
</tr>
<tr>
<td>ADS</td>
<td>Add symptom service</td>
</tr>
<tr>
<td>CBF</td>
<td>Control block formatter service</td>
</tr>
<tr>
<td>CBS</td>
<td>Control block status service</td>
</tr>
<tr>
<td>CQE</td>
<td>Contention queue element create service</td>
</tr>
<tr>
<td>CSI</td>
<td>CSVINFO macro</td>
</tr>
<tr>
<td>ECT</td>
<td>ECT exit service</td>
</tr>
<tr>
<td>EQS</td>
<td>Equate symbol service</td>
</tr>
<tr>
<td>FMT</td>
<td>Format model processor service</td>
</tr>
<tr>
<td>GTS</td>
<td>Get symbol service</td>
</tr>
<tr>
<td>MAP</td>
<td>Map service</td>
</tr>
<tr>
<td>NAM</td>
<td>Name service</td>
</tr>
<tr>
<td>NDX</td>
<td>Table of contents service</td>
</tr>
<tr>
<td>NTK</td>
<td>NAME/TOKEN lookup service</td>
</tr>
<tr>
<td>PRT</td>
<td>Standard print service</td>
</tr>
<tr>
<td>PR2</td>
<td>Expanded print service</td>
</tr>
<tr>
<td>SEL</td>
<td>Select ASID service</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol service</td>
</tr>
<tr>
<td>WHS</td>
<td>WHERE service</td>
</tr>
</tbody>
</table>

INPUT

Specifies that trap processing is to be done before performing a requested
service. If the INPUT parameter is specified without any options, all
supported input trapping options are activated. The options are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABDPL</td>
<td>Displays the common exit parameter list and its extension that are passed to all services.</td>
</tr>
<tr>
<td>DATA</td>
<td>Displays data passed to a service in addition to basic parameters. The DATA option can be used only if the FMT code is specified.</td>
</tr>
</tbody>
</table>
TRAPON subcommand

PARMS Displays parameters passed to a service. The PARMS option cannot be used if the PRT and NDX codes are specified.

STOP Halts IPCS processing and prompts you for input before performing a service. If the TSO/E NOPROMPT mode is in effect when STOP processing is attempted, processing is not interrupted. During STOP processing, only the following may be entered:
- IPCS subcommands GO, HELP, NOTE, TRAPLIST, TRAPOFF, TRAPON, and TSO. Use the GO subcommand to resume processing; the END subcommand is not valid.
- CLISTs and REXX execs that contain only the previously mentioned subcommands.
- TSO/E commands that are normally accepted during an IPCS session. The use of authorized TSO/E commands requires the installation of TSO/E Release 2 or a later release.

Restriction: If you specify INPUT(STOP) or OUTPUT(STOP) when running IPCS in the background or in a full-screen dialog, it is ignored.

See 5-287 for a list of the trap options supported by the INPUT and OUTPUT parameters for each exit service routine.

NOINPUT Specifies that no trap processing is to be done before performing a requested service. NOINPUT is the default.

Note: If both NOINPUT and NOOUTPUT are specified, IPCS issues a diagnostic message, and the TRAPON subcommand ends without alteration to the status of the traps.

OUTPUT Specifies that trap processing is to be done before returning to the caller of a service. If the OUTPUT parameter is specified without any options, all supported output trapping options are activated. The options are:

Option Processing
RETC Displays the return code from the service and the service code-list.
DATA Displays the data returned by a service in addition to basic parameters. The DATA option can be used only if the ACC and SEL codes are specified.
PARMS Displays parameters returned by a service. This is the same parameter list that is displayed as input, but it will show any values changed by the service. The PARMS option cannot be used if the PRT and NDX codes are specified.
STOP Halts IPCS processing and prompts you for input before returning from a service. If the TSO/E NOPROMPT mode is in effect when STOP processing is attempted, processing is not interrupted, and no message is issued. During STOP processing only the following may be entered:
TRAPON subcommand

- IPCS subcommands GO, HELP, NOTE, TRAPLIST, TRAPOFF, TRAPON, and TSO. Use the GO subcommand to resume processing; the END subcommand is not valid.
- CLISTs and REXX execs that contain only the previously mentioned subcommands.
- TSO/E commands that are normally accepted during an IPCS session. The use of authorized TSO/E commands requires the installation of TSO/E Release 2 or a later release.

Restriction: If you specify OUTPUT(STOP) or INPUT(STOP) when running IPCS in the background or in a full-screen dialog, it is ignored.

ERROR
Specifies that the other output trap actions are to take place only when the return code from the service is not zero. This is a convenient means of reducing the output from the trap facility, but still seeing important failure-related information.

See 5-287 for a list of the trap options supported by the INPUT and OUTPUT parameters for each exit service routine.

NOOUTPUT
Specifies that no trap processing is to be done before returning to the caller of a service. NOOUTPUT is the default.

Note: If both NOINPUT and NOOUTPUT are specified, IPCS issues a diagnostic message, and the TRAPON subcommand ends without alteration to the status of the traps.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the TRAPON subcommand.

Example 1
Turn on all traps associated with the exit service routines.
- Action
  COMMAND ===> trapon all input(abdpl,parms)
- Result
  This example activates the trap for all the exit services. When a trap is hit, the ABDPL and the parameter list (if used) are displayed.

Example 2
Turn on all traps and all options associated with the storage access and the control block formatter service routines and display the return code on exit.
- Action
  COMMAND ===> trapon (acc cbf) output(retc)
- Result
  This example activates the traps for the storage access and the control block formatter service routines and displays the return code on exit from these services.

TSO subcommand — run a TSO/E command

Use the TSO subcommand to:
**TSO subcommand**

- Invoke a TSO/E command whose name is identical to an IPCS subcommand. See the description of the `tsocmd` parameter for information concerning authorized TSO/E commands.
- Invoke a CLIST or REXX exec containing TSO/E commands whose names are identical to IPCS subcommands.
- Enter TSO/E mode.

**Invoke a TSO/E Command**

Use the TSO subcommand to enter TSO/E commands whose names are identical to IPCS subcommands except when invoking ISPF.

For example, to request the display of status for all batch jobs whose job name begins with your TSO/E userid, enter:

```bash
tso status
```

If you do not precede the STATUS command with TSO, the system does not interpret the command as a TSO/E command. Note, however, that the system does not allow TSO/E commands, when invoked by IPCS, to request ISPF services. For example, using the TSO/E ALTLIB command with the QUIET option causes ALTLIB to use ISPF services, which the system does not permit.

**ISPF under IPCS**

Do not invoke the ISPF command with the TSO prefix. Instead, invoke ISPF by entering ISPF on the command line. If you enter TSO ISPF, you may obtain unpredictable results.

If TSO/E Release 2 or later is installed, you can enter installation-defined authorized commands and authorized TSO/E commands, such as TRANSMIT and RECEIVE (as determined by your installation). Otherwise, such commands end abnormally.

**Invoke a CLIST or REXX Exec Containing TSO/E Commands**

You can use the TSO subcommand to invoke a CLIST or REXX exec containing TSO/E commands. You can do this in any of the three IPCS processing modes. A CLIST or REXX exec invoked with the TSO subcommand can contain any or all of the following:

- TSO/E commands whose names are identical to IPCS subcommands. Using the TSO subcommand ensures that the TSO/E command is invoked instead of an IPCS subcommand of the same name.
- Any TSO/E command. Any TSO/E command can be included in a CLIST invoked using the TSO subcommand.
- TSO/E authorized commands in conjunction with a TSO/E function such as SYSOUTTRAP. *While in the IPCS dialog*, the SYSOUTTRAP will not trap the output from the authorized command correctly unless you use the TSO subcommand to invoke the CLIST. However, such a CLIST can be invoked successfully in batch or line mode without using the TSO subcommand.
- IPCS subcommands. To run IPCS subcommands from within a CLIST invoked using the TSO subcommand, use the BLSGSCMD dialog program to invoke the IPCS subcommands.
- ISPF commands. Invoke a CLIST containing ISPF commands from within IPCS dialog or in IPCS batch mode if ISPF is active in batch.

**Restriction:** You can define and use up to 10 global variables in CLISTs invoked through the IPCS dialog, if CLIST BLSCLIBD started the IPCS dialog. IPCS does not restrict the number of global variables you can define when the IPCS dialog is started using other approved methods. If CLIST BLSCLIBD started the IPCS
dialog, and if you require more than 10 global variables, create your own copy
of CLIST BLSCALTL and add more global variables. Modify CLIST BLSCLIBD
to point to your copy of BLSCALTL rather than to SYS1.SBLSCLUD(BLSCALTL).
For information about defining and using global variables, see: z/OS TSO/E

Enter TSO/E Mode
In line mode or batch mode IPCS, you can enter the TSO subcommand without
a command or CLIST or REXX exec invocation to suspend IPCS subcommand
processing and enter TSO/E mode. Then, commands entered in TSO/E mode
are processed as TSO/E commands until END is entered to resume IPCS
processing. When the END subcommand is entered, the highest return code
from the TSO/E command processing is returned.

Syntax

TSO [ [[%]clistnm | [%]rexnm | tsocmd ] [operands] ]

Parameters

clistnm
   Specifies the name of the CLIST to be run. If the CLIST name is the same as
   the name of a TSO/E or IPCS command, a % must precede the name.

rexnm
   Specifies the name of the REXX exec to be run. If the REXX exec name is
   the same as the name of a TSO/E or IPCS command, a % must precede the
   name.

tsocmd
   Specifies the name of a TSO/E command to be run. If TSO/E Release 2 is
   installed, tsocmd may specify the name of an installation-defined authorized
   command or an authorized TSO/E command, such as TRANSMIT or
   RECEIVE (as determined by your installation). See: z/OS TSO/E
   Customization for more information.

operands
   Specifies the operands of the TSO/E command or CLIST to be run.

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Severe, an error condition or user request forced early ending of subcommand processing.</td>
</tr>
<tr>
<td>16</td>
<td>Ending error, an error condition from a called service routine forced an early end to the processing.</td>
</tr>
<tr>
<td>any</td>
<td>The return code is generated by the TSO/E command.</td>
</tr>
</tbody>
</table>

Example 1
Display the status of all batch jobs.

   - Action
     COMMAND ===> tso status
   - Result
     This example requests the display of status for all batch jobs whose job name
     begins with your TSO/E user ID. If you do not preceded the command
     name, STATUS, with TSO, the IPCS STATUS subcommand are processed.

Example 2
Send a data set to a node and userid.
**TSO subcommand**

- **Action**

  COMMAND ===> tso transmit nodeb.user2 da('sys1.parmlib')
  or
  COMMAND ===> transmit nodeb.user2 da('sys1.parmlib')

- **Result**

  These commands request that a copy of a data set (SYS1.PARMLIB) be sent to a specified node and user (nodeb.user2). It is not necessary to precede the command name (TRANSMIT) with TSO because there is no IPCS subcommand with the name TRANSMIT. IPCS processes both of the commands in this example as TSO/E commands.

---

**VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine**

Use the VERBEXIT subcommand to run an installation-supplied or IBM-supplied verb exit routine.

**Syntax**

```
{ VERBEXIT } { pgmname }
{ VERBX } { verbname }

[ 'parameter [,parameter]...' ]
[ AMASK(mask) ]
[ SYNTAX | NOSYNTAX ]
[ TOC | NOTOC ]
```

---

**SETDEF-Defined Parameters**

Note: You can override the following SETDEF parameters. See **“SETDEF subcommand — set defaults” on page 5-232**

```
[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]
```

**Parameters**

- **pgmname**

  Specifies a verb exit routine.

  The *pgmname* can be a maximum of 8 alphanumeric characters; the first character must be alphabetic.

  An installation-supplied verb exit routine must reside in a load module library available to IPCS, such as a step library, job library, or link library. For information about writing verb exit routines, see **z/OS MVS IPCS Customization**

- **vername**

  Specifies the name of a verb exit routine.
**VERBEXIT subcommand**

For IPCS to access an installation-supplied verb exit through a verb name, your installation needs to either create or modify the BLSCUSER parmlib member.

An installation-supplied verb exit routine must reside in a load module library available to IPCS, such as a step library, job library, or link library.

For information about the BLSCUSER parmlib member and on writing verb exit routines, see [z/OS MVS IPCS Customization](#).

The table that follows lists the verb names of IBM-supplied verb exit routines. These verb exit routines are defined in SYS1.PARMLIB members. For each verb name, the table provides a cross reference telling where you can find an explanation of the verb name, its optional parameters if applicable, and information concerning the component, function, or product-specific data that these verb exit routines process.

<table>
<thead>
<tr>
<th>Verb Name</th>
<th>Component, Product, or Function</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALCWAIT</td>
<td>Allocation</td>
<td>See <a href="#">&quot;VERBEXIT ALCWAIT subcommand — list jobs waiting for devices&quot; on page 5-300</a></td>
</tr>
<tr>
<td>AOMDATA</td>
<td>Asynchronous operations manager</td>
<td>z/OS DFSMSdfp Diagnosis</td>
</tr>
<tr>
<td>ASMDATA</td>
<td>Auxiliary storage management</td>
<td>See <a href="#">&quot;VERBEXIT ASMDATA subcommand — format auxiliary storage manager data&quot; on page 5-301</a></td>
</tr>
<tr>
<td>AVMDATA</td>
<td>Availability manager</td>
<td>See <a href="#">&quot;VERBEXIT AVMDATA subcommand — format availability manager data&quot; on page 5-301</a></td>
</tr>
<tr>
<td>BLSAIPST</td>
<td>System initialization</td>
<td>See <a href="#">&quot;VERBEXIT BLSAIPST subcommand — format system initialization data&quot; on page 5-301</a></td>
</tr>
<tr>
<td>CBDATA</td>
<td>Component Broker</td>
<td>See <a href="#">&quot;VERBEXIT CBDATA subcommand — format component broker data&quot; on page 5-301</a></td>
</tr>
<tr>
<td>CICSDATA</td>
<td>Customer Information Control System</td>
<td>See <a href="#">CICS Operations and Utilities Guide</a></td>
</tr>
<tr>
<td>DAEDATA</td>
<td>Dump analysis and elimination</td>
<td>See <a href="#">&quot;VERBEXIT DAEDATA subcommand — format dump analysis and elimination data&quot; on page 5-302</a></td>
</tr>
<tr>
<td>DSNWDMP</td>
<td>DB2</td>
<td>See <a href="#">DB2 Diagnosis Guide and Reference</a></td>
</tr>
<tr>
<td>GRSTRACE</td>
<td>Global resource serialization</td>
<td>See <a href="#">&quot;VERBEXIT GRSTRACE subcommand — format Global Resource Serialization data&quot; on page 5-304</a></td>
</tr>
<tr>
<td>HASMFMTM</td>
<td>JES2</td>
<td>See <a href="#">z/OS JES2 Diagnosis</a></td>
</tr>
<tr>
<td>IEAVTSFS</td>
<td>Dumping services</td>
<td>See <a href="#">&quot;VERBEXIT IEAVTSFS subcommand — format SVC dump measurements and statistics report&quot; on page 5-306</a></td>
</tr>
<tr>
<td>IEFENFXV</td>
<td>Event notification facility (ENF)</td>
<td>See <a href="#">&quot;VERBEXIT IEFENFXV subcommand — list ENF listeners&quot; on page 5-312</a></td>
</tr>
<tr>
<td>IEFIVAWT</td>
<td>Allocation</td>
<td>See <a href="#">&quot;VERBEXIT IEFIVAWT subcommand — list pending XCF work for tape allocation&quot; on page 5-312</a></td>
</tr>
<tr>
<td>IEFIVIGD</td>
<td>Allocation</td>
<td>See <a href="#">&quot;VERBEXIT IEFIVIGD subcommand — list global tape device information&quot; on page 5-312</a></td>
</tr>
</tbody>
</table>

Note: Other products might have a dump formatter available for use with IPCS. Check the related product documentation for information.
### VERBEXIT subcommand

<table>
<thead>
<tr>
<th>Verb Name</th>
<th>Component, Product, or Function</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMSDUMP</td>
<td>Information Management System (IMS)</td>
<td><a href="#">IMS Version 9: Utilities Reference: System</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="#">IMS Version 8: Diagnosis Guide and Reference</a></td>
</tr>
<tr>
<td>IRLM</td>
<td>Information Management System (IMS)</td>
<td><a href="#">IMS Version 8: Diagnosis Guide and Reference</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="#">IMS Version 9: Utilities Reference: System</a></td>
</tr>
<tr>
<td>JESXCF</td>
<td>JES common coupling services MVS component (JES XCF)</td>
<td>See “VERBEXIT JESXCF subcommand — format data for JES XCF component” on page 5-313</td>
</tr>
<tr>
<td>JES3</td>
<td>JES3</td>
<td><a href="#">z/OS JES3 Diagnosis</a></td>
</tr>
<tr>
<td>LEDATA</td>
<td>Language Environment®</td>
<td>See “VERBEXIT LEDATA subcommand — format Language Environment data” on page 5-314</td>
</tr>
<tr>
<td>LOGDATA</td>
<td>Logrec buffer records</td>
<td>See “VERBEXIT LOGDATA subcommand — format logrec buffer records” on page 5-317</td>
</tr>
<tr>
<td>MMSDATA</td>
<td>MVS message service</td>
<td>See “VERBEXIT MMSDATA subcommand — format MVS message service data” on page 5-320</td>
</tr>
<tr>
<td>MTRACE</td>
<td>Master trace table</td>
<td>See “VERBEXIT MTRACE subcommand — format master trace entries” on page 5-320</td>
</tr>
<tr>
<td>NUCMAP</td>
<td>Modules in the nucleus</td>
<td>See “VERBEXIT NUCMAP subcommand — map modules in the nucleus” on page 5-322</td>
</tr>
<tr>
<td>SADMPMSG</td>
<td>Stand-alone dump message log</td>
<td>See “VERBEXIT SADMPMSG subcommand — format stand-alone dump message log” on page 5-326</td>
</tr>
<tr>
<td>SMSDATA</td>
<td>DFP Storage Management Subsystem</td>
<td><a href="#">z/OS DFSMSdfp Diagnosis</a></td>
</tr>
<tr>
<td>SMSXDATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRMDATA</td>
<td>System resource manager</td>
<td>See “VERBEXIT SRMDATA subcommand — format System Resource Manager data” on page 5-326</td>
</tr>
<tr>
<td>SUMDUMP</td>
<td>SVC summary dump data</td>
<td>See “VERBEXIT SUMDUMP subcommand — format SVC summary dump data” on page 5-327</td>
</tr>
<tr>
<td>SYMPTOM</td>
<td>Symptom string</td>
<td>See “VERBEXIT SYMPTOM subcommand — format symptom string” on page 5-328</td>
</tr>
<tr>
<td>TSODATA</td>
<td>Time Sharing Option</td>
<td><a href="#">z/OS TSO/E System Diagnosis: Data Areas</a></td>
</tr>
<tr>
<td>VSMDATA</td>
<td>Virtual storage management</td>
<td>See “VERBEXIT VSMDATA subcommand — format virtual storage management data” on page 5-329</td>
</tr>
<tr>
<td>VTAMMAP</td>
<td>Virtual Telecommunications Access Method (VTAM)</td>
<td><a href="#">VTAM Diagnosis</a></td>
</tr>
</tbody>
</table>

#### parameter

Specifies a parameter string to be passed to either an IBM-supplied or an installation-supplied verb exit routine.

Enclose the parameter string in apostrophes. When IPCS passes the string to the exit routine, it omits the apostrophes. If the string parameter itself includes an apostrophe, enter a pair of apostrophes; IPCS will convert them to a single apostrophe when passing the string to the exit routine.

Verb exits are responsible for parsing the string. When specifying keyword strings, be aware of the following conditions:
VERBEXIT subcommand

- Spell out the full form of the keyword strings expected by the verb exit. Not all of the verb exits recognize truncated keywords.
- Use commas to separate parameters when you specify more than one parameter and the verb exit syntax indicates comma separators are appropriate. Avoid using blanks or horizontal tabulation character to separate parameters, even if the TSO syntax rule says they are interchangeable with commas.
- Follow the special syntax rules required by verb exit routines, if any. Authors of verb exit routines are allowed to implement special syntax rules for the parameters, depending on the primary usage of the routines. For example, verb exit routines provided by DB2 might implement SQL rules rather than TSO rules.

For IBM-supplied verb exit routines, the parameter string that can be specified is described in this book under the corresponding verb name.

For installation-supplied verb exit routines, the parameter string that can be specified must have its content and meaning defined by the installation-supplied exit routine.

**AMASK(mask)**

Specifies an integer mask that VERBEXIT is to AND to the dump addresses passed by the exit to the storage access and format service routines. Only X'00FFFFFF', X'7FFFFFFF' or the corresponding decimal or binary values are accepted.

**SYNTAX or NOSYNTAX**

Specifies or suppresses a syntax check of the parameter string passed to the verb exit routine.

SYNTAX specifies the syntax check.

NOSYNTAX suppresses the syntax check and is the default.

**TOC or NOTOC**

Specifies or suppresses table or contents, print file, and terminal output.

The TOC option anticipates that the exit will write a report. IPCS writes a standard table of contents entry before giving the exit control. An error message is written if no report is written.

The NOTOC option suppresses the output.

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Severe error, an error condition or user request forced early termination of the subcommand.</td>
</tr>
<tr>
<td>16</td>
<td>Terminating error, an error condition from a called service routine forced an early termination of the subcommand.</td>
</tr>
<tr>
<td>other</td>
<td>An exit generated return code.</td>
</tr>
</tbody>
</table>

**Example**

Invoke an installation-supplied verb exit represented by the verb name HISTORY.

- Action
  
  COMMAND === verbexit history 'rb,56b34'

- Result
VERBEXIT subcommand

The installation-supplied verb exit routine HISTORY receives the parameter string RB,56B34.

VERBEXIT ALCWAIT subcommand — list jobs waiting for devices

Specify the ALCWAIT verb name on the VERBEXIT subcommand to format a list of jobs waiting for devices.

**Note:** To obtain a list of jobs holding a device group and the jobs waiting for a device group, use the ANALYZE subcommand with the RESOURCE parameter.

**Parameters**
The VERBEXIT ALCWAIT subcommand has no parameters.

**Example**
For an example of ALCWAIT output, see the allocation/unallocation component in [z/OS MVS Diagnosis: Reference](#).

VERBEXIT ASMDATA subcommand — format auxiliary storage manager data

Specify the ASMDATA verb name and optional parameters on the VERBEXIT subcommand to format diagnostic data from the auxiliary storage manager (ASM).

**Syntax**

```
VERBEXIT ASMDATA [ 'parameter [,parameter]...' ]
```

The parameters are:

- [FULL]
- [SUMMARY]
- [VIO]

**Parameters**

Use the parameters to select the type of report. If you omit the parameters, the default is FULL.

**FULL**

Produces a full report of ASM control blocks.

**SUMMARY**

Produces a summary report of the paging-related control blocks.

**VIO**

Produces a summary report of the VIO-related control blocks.

**Note:** The FULL and SUMMARY reports include SCM-related control blocks, including:

- The SCM PARTE entry
- Each IORB and its associated IOSB and EAOB
- AIAs queued to IORBs that are currently in use (I/O is active).

**Example**

For an example of ASMDATA output, see the ASM component in [z/OS MVS Diagnosis: Reference](#).
VERBEXIT AVMDATA subcommand — format availability manager data

Specify the AVMDATA verb name on the VERBEXIT subcommand to format diagnostic data from the availability manager.

**Parameters**
The VERBEXIT AVMDATA subcommand has no parameters.

VERBEXIT BLSAIPST subcommand — format system initialization data

Specify the BLSAIPST verb name on the VERBEXIT subcommand to format status data collected during IPL, NIP, and Master Scheduler Initialization (MSI) during system initialization.

**Parameters**
The VERBEXIT BLSAIPST subcommand has no parameters.

**Example**
For an example of BLSAIPST output, see the system initialization component in z/OS MVS Diagnosis: Reference.

VERBEXIT CBDATA subcommand — format component broker data

Specify the CBDATA verb name and optional parameters on the VERBEXIT subcommand to format diagnostic data for the Component Broker element in WebSphere® Application Server Enterprise Edition for z/OS. CBDATA displays the following:

- Display of the Component Broker Global control blocks
- Display of Component Broker address space control blocks
- Display of Component Broker address space control blocks with only one Component Broker TCB
- Display of ORB control block information

**Syntax**

VERBEXIT CBDATA [ 'parameter [,parameter]...' ]

The parameters are:

- [GLOBAL]
- [ASID(asid-number)]
- [ASID(asid-number BTCB(btcb-address))]
- [ASID(asid-number ORB(orb-address))]

**Parameters**

Use these parameters to format the data areas. If you omit the parameters, the default is GLOBAL.

- **GLOBAL**
  - Displays the following formatted Component Broker control blocks
    - BGV address - Component Broker Global Vector table
    - ASR Table and ASR Table entries - Active Server Repository information

- **ASID(asid-number)**
  - Displays the following formatted Component Broker control blocks
    - BACB - Component Broker address space control block
    - BTRC, TBUFSET, TBUF - Component Broker component trace control blocks
    - BOAM, BOAMX - CB BOA control blocks
VERBEXIT BLSAIPST subcommand

- ACRW queue- Application Control Region work element control blocks
- DAUE- DB2 ASR Table
- BTCB queues - Component Broker TCB

ASID(asid-number) BTCB(btcb_address)
Displays the following formatted Component Broker control blocks and the specified BTCB
- BACB - Component Broker address space control block
- BTRC,TBUFSET,TBUF - CB component trace control blocks
- BOAM,BOAMX - CB BOA control blocks
- ACRW queue- Application Control Region work element control blocks
- DAUE- DB2 ASR Table
- BTCB - Component Broker TCB
- Displays ORB information for the Component Broker TCB

ASID(asid-number) ORB(orb_address)
Displays ORB information

VERBEXIT DAEDATA subcommand — format dump analysis and elimination data

Specify the DAEDATA verb name on the VERBEXIT subcommand to format the dump analysis and elimination (DAE) data in an SVC dump or SYSMDUMP dump.

DAEDATA formats and prints the DAE data in the dump header record for the dump. If DAE data is available, DAEDATA displays the following:
- Explanation of the DAE action taken for this dump
- The number of occurrences
- The original dump identification data, including the sequence number, data, time, and the CPU serial number
- The unique identification criteria
- The MVS symptom string and symptom parameters
- The RETAIN symptom string and symptom parameters
- The symptom string verbal description
- Any additional symptoms from the SDWA

Parameters
The VERBEXIT DAEDATA subcommand has no parameters.

Example
Obtain DAE information from the dump.
- Action
  VERBEXIT DAEDATA
- Result

****** DUMP ANALYSIS AND ELIMINATION (DAE) ******

THIS DUMP WAS NOT SUPPRESSED BECAUSE
THE VRA KEY TO ALLOW SUPPRESSION OF DUPLICATE DUMPS WAS ABSENT.
VERBEXIT DAEDATA subcommand

CRITERIA FOR USE AS A UNIQUE DUMP IDENTIFIER BY DAE:

MINIMUM NUMBER OF SYMPTOMS: 05 FOUND: 09
MINIMUM TOTAL STRING LENGTH: 025 FOUND: 144

SYMPTOMS REQUIRED TO BE PRESENT:
MOD/ CSECT/

SYMPTOMS THAT ARE TO BE USED IF AVAILABLE, BUT ARE NOT REQUIRED:
PIDS/ AB/S AB/U REXN/ FI/ REGS/ HRC1/ SUB1/

MVS SYMPTOM STRING:
MOD/NUCLEUS CSECT/IARUVXCH PIDS/5752SC1CR AB/S00C4 REXN/IARRR
FI/18F4B22100EF181B8F2FD0F0 REGS/0A8D0 HRC1/00000004
SUB1/REAL#STORAGE#MANAGEMENT

RETAIN SEARCH ARGUMENT:
RIDS/NUCLEUS#L RIDS/IARUVXCH PIDS/5752SC1CR AB/S00C4 RIDS/IARRR#R
VALU/HBF2FD0F0 REGS/0A8D0 PRCS/00000004 VALU/CNAGEMENT

SYMPTOMS PRESENT FOR USE AS A UNIQUE DUMP IDENTIFIER BY DAE:

<table>
<thead>
<tr>
<th>RETAIN</th>
<th>MVS KEY</th>
<th>KEY</th>
<th>SYMPTOM DATA</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MOD/</td>
<td>RIDS/</td>
<td>NUCLEUS</td>
<td>LOAD MODULE NAME</td>
</tr>
<tr>
<td></td>
<td>CSECT/</td>
<td>RIDS/</td>
<td>IARUVXCH</td>
<td>ASSEMBLY MODULE CSECT NAME</td>
</tr>
<tr>
<td></td>
<td>PIDS/</td>
<td>PIDS/</td>
<td>5752SC1CR</td>
<td>PRODUCT/COMPONENT IDENTIFIER</td>
</tr>
<tr>
<td></td>
<td>AB/S</td>
<td>AB/S</td>
<td>S00C4</td>
<td>ABEND CODE-SYSTEM</td>
</tr>
<tr>
<td></td>
<td>REXN/</td>
<td>RIDS/</td>
<td>IARRR</td>
<td>RECOVERY ROUTINE CSECT NAME</td>
</tr>
<tr>
<td></td>
<td>FI/</td>
<td>VALU/H</td>
<td>18F4B22100EF181BBF2FD0F0</td>
<td>FAILING INSTRUCTION AREA</td>
</tr>
<tr>
<td></td>
<td>REGS/</td>
<td>REGS/</td>
<td>0A8D0</td>
<td>REG/PSW DIFFERENCE</td>
</tr>
<tr>
<td></td>
<td>HRC1/</td>
<td>PRCS/</td>
<td>00000004</td>
<td>REASON CODE</td>
</tr>
<tr>
<td></td>
<td>SUB1/</td>
<td>VALU/C</td>
<td>REAL#STORAGE#MANAGEMENT</td>
<td>COMPONENT SUBFUNCTION</td>
</tr>
</tbody>
</table>

ADDITIONAL SYMPTOM DATA NOT USED BY DAE TO IDENTIFY THIS DUMP:

<table>
<thead>
<tr>
<th>RETAIN</th>
<th>MVS KEY</th>
<th>KEY</th>
<th>SYMPTOM DATA</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VCBI2/</td>
<td>VALU/H</td>
<td>3C7800F2D00001B017C000000000011BE84401ACDC90</td>
<td>CONTROL BLOCK ID AND DATA</td>
</tr>
<tr>
<td></td>
<td>CID1/</td>
<td>VALU/C</td>
<td>SCICR</td>
<td>COMPONENT IDENTIFIER</td>
</tr>
<tr>
<td></td>
<td>AMD1/</td>
<td>VALU/C</td>
<td>04#14#B7</td>
<td>MODULE ASSEMBLY DATE</td>
</tr>
<tr>
<td></td>
<td>VRS1/</td>
<td>VALU/C</td>
<td>HBB3310</td>
<td>VERSION-PRODUCT/PTF IDENTIFIER</td>
</tr>
<tr>
<td></td>
<td>RRL1/</td>
<td>FIDS/</td>
<td>IARRRCV</td>
<td>RECOVERY ROUTINE LABEL</td>
</tr>
<tr>
<td></td>
<td>CD81/</td>
<td>VALU/C</td>
<td>5752</td>
<td>BASE COMPONENT IDENTIFIER</td>
</tr>
<tr>
<td></td>
<td>HLH1/</td>
<td>VALU/H</td>
<td>0B00C000</td>
<td>HIGHEST LOCK HELD INDICATOR</td>
</tr>
<tr>
<td></td>
<td>SUP1/</td>
<td>VALU/H</td>
<td>10000000</td>
<td>PSASUPER FLAGS</td>
</tr>
<tr>
<td></td>
<td>FRR1/</td>
<td>VALU/H</td>
<td>01ACFC90</td>
<td>FRR PARAMETER AREA</td>
</tr>
<tr>
<td></td>
<td>ASID1/</td>
<td>VALU/H</td>
<td>00DE</td>
<td>TASK RELATED ASID</td>
</tr>
<tr>
<td></td>
<td>ORCC1/</td>
<td>PRCS/</td>
<td>0C4000</td>
<td>ORIGINAL COMPLETION CODE</td>
</tr>
<tr>
<td></td>
<td>ORRC1/</td>
<td>PRCS/</td>
<td>00000004</td>
<td>ORIGINAL REASON CODE</td>
</tr>
</tbody>
</table>

**************************************************************** END OF DATA ****************************************************************
VERBEXIT GRSTRACE subcommand

VERBEXIT GRSTRACE subcommand — format Global Resource Serialization data

Specify the GRSTRACE, QCBTRACE, or Q verb name on the VERBEXIT subcommand to format diagnostic data from the major control blocks for global resource serialization.

Syntax

VERBEXIT GRSTRACE ['parameter [,parameter]...']

The parameters are:

Data Selection Parameters:

- [DETAIL]
- [SUMMARY]

Time format Parameters:

- [TIME(LOCAL|GMT|UTC)]

Additional Filter Parameters:

- [START(mm/dd/yy,hh.mm.ss.dddddd)]
- [STOP(mm/dd/yy,hh.mm.ss.dddddd)]
- [SYSNAME(sysname)]
- [QNAME(qname)]
- [RNAME(rname)]
- [STEP] [ SYSTEM] [ SYSTEMS]
- [JOBNAME(jobname)]
- [ASID(asid)]
- [TCB(tcb)]
- [RESERVE]
- [CONTENTION]

SETDEF-Defined Parameters:

Note: You can override the following SETDEF parameters.

- [DSNAME(dsname) | DATASET(dsname)]
- [FILE(ddname) | DDNAME(ddname)]
- [PATH(path-name)]
- [FLAG(severity)]
- [PRINT | NOPRINT]
- [TERMINAL | NOTERMINAL]
- [TEST | NOTEST]

Parameters

Data Selection Parameters

**DETAIL**

Provides a detailed GRSTRACE report. The detailed report contains ENQ diagnostic data in addition to all the important ENQ context information that the summary report displays.

**SUMMARY**

Provides a summary GRSTRACE report. The summary report contains all the relevant context information such as QName, RName, Sysname, Scope, Jobname, Asid, Tcb, Disposition, ownership status, wait and grant times. SUMMARY is the default.

Time format Parameters
VERBEXIT GRSTRACE subcommand

TIME(LOCAL|GMT|UTC)
  Specifies the time format to use for the GRSTRACE report.
  - LOCAL: All ENQ relevant times should be formatted in local time.
  - GMT: All ENQ relevant times should be formatted in GMT time.
  - UTC: All ENQ relevant times should be formatted in UTC time. This is
    the exact store clock timestamp.

Additional Filter Parameters
  Use these parameters to limit the scope of the data in the report. If no data
  selection parameter is selected, the default is NO FILTERING. At least one
  requestor in a resource chain must match all of the filtering options in order for
  a resource to be displayed. Wildcard values are allowed for the SYSNAME,
  JOBNAME, QNAME, and RNAME filters. Use * to match zero or more
  characters and ? for exactly one character. See 5-306 for an example.

START(mm/dd/yy,hh.mm.ss.dddddd)
  Specifies the date and time used to display ENQ resources with requests
  that occurred at or after this time. The time format must match the time
  format specified with the TIME keyword. When you do not specify START,
  IPCS starts with the oldest ENQ request. Specify the date and time in
  mm/dd/yy,hh.mm.ss.dddddd format.

  Note: The following rules apply to the date and time specifications:
  - The month and day can be specified in single or double digits.
  - Separate the date from the time with a comma.
  - The time can be local, by default or specified in a TIME(Local)
    parameter, or GMT or UTC, if specified in a Time(GMT) or
    Time(UTC) parameter.
  - Hours, minutes, and seconds can be specified in single or double
    digits.
  - The time can be truncated anywhere on the right.
  - The time can be left off completely, in which case, it defaults to
    00:00:00.000000 (midnight).

STOP(mm/dd/yy,hh.mm.ss.dddddd)
  Specifies the date and time used to display ENQ resources with requests
  that occurred up to or before this time. The time format must match the
  time format specified with the TIME keyword. When you do not specify
  STOP, IPCS ends with the newest ENQ request.

  See the START parameter for guidelines on how to specify the time and
  date.

SYSNAME(sysname)
  Displays all ENQ resources with the given specified system name. Note in
  GRS=STAR, resource requests from other systems are not maintained in
  local storage. Thus, a query specifying another system name may only
  receive data back from GRSDATA, not GRSTRACE.

QNAME(qname)
  Displays all ENQ resources with the specified QNAME (major name).

RNAME(rname)
  Displays all ENQ resources with the specified RNAME (minor name).

[STEP] [ SYSTEM] [ SYSTEMS]
  Displays all ENQ resources with a scope of STEP, SYSTEM, or SYSTEMS.
VERBEXIT GRSTRACE subcommand

**JOBNAME**(*jobname*)
Displays all ENQ resources associated with the specified job name.

**ASID**(*asid*)
Displays all ENQ resources associated with the specified address space ID.

**TCB**(*tcb*)
Displays all ENQ resources associated with the specified task

**RESERVE**
Displays only RESERVE requests that have not been converted to global ENQs.

**CONTENTION**
Displays only ENQ resources that are in ENQ contention. Device RESERVE contention is not taken into consideration.

**Example**
IP VERBX GRSTRACE ‘QNAME(SYS*R??F) RNAME(SOMESPECIFICRNAME) SYSTEMS JOBNAME(DB2*)’
Matches any resource requests that have the following:
- A QNAME starting with SYS, followed by zero or more characters until an R is found, followed by two specific characters and ending in an F (for example, SYSZRAcF)
- RNAME is SOMESPECIFICRNAME
- SCOPE=SYSTEMS
- JOBNAME starts with DB2
For an example of GRSTRACE output, see the global serialization resource component in [z/OS MVS Diagnosis: Reference](#).

VERBEXIT IEAVTSFS subcommand — format SVC dump measurements and statistics report

Specify the IEAVTSFS verb name on the VERBEXIT subcommand to format the SVC dump measurements and statistics report. The VERBEXIT IEAVTSFS output may be requested by the IBM Support Center to understand where SDUMP spent its time collecting a dump.

**Parameters**
The VERBEXIT IEAVTSFS subcommand has no parameters.

**Example**
Obtain the SVC dump measurements and statistics.
- **Action**
  VERBEXIT IEAVTSFS
- **Result**

SVC Dump Measurements and Statistics Report

Capture phase partial dump reason codes (IHASDRSN):
  00000000 00000000 00000000 00000000

Dump start 09/25/2009 08:28:27.248748
Dump end 09/25/2009 08:28:30.517536
Total dump capture time 00:00:03.268788

System nondispatchability start 09/25/2009 08:28:27.248748
System set nondispatchable 09/25/2009 08:28:27.248968
System set nondispatchable 09/25/2009 08:28:27.248978
VERBEXIT IEAVTSFS subcommand

Time to become nondispatchable 00:00:00.000010

Global storage start 09/25/2009 08:28:27.248823
Global storage capture time 00:00:01.322367
  Defers for frame availability 0
  Pages requiring input I/O 335
  Source page copied to target 5611
  Source frames re-assigned 372
  Source AUX slot IDs re-assigned 23

System was nondispatchable 00:00:01.566168

Asid 002D:
  Local storage start 09/25/2009 08:28:28.815164
  Local storage end 09/25/2009 08:28:30.305044
  Local storage capture time 00:00:01.489880
  Tasks reset dispatchable 09/25/2009 08:28:30.305066
  Tasks were nondispatchable 00:00:01.489901
  Defers for frame availability 0
  Pages requiring input I/O 0
  Source page copied to target 450
  Source frames re-assigned 20
  Source AUX slot IDs re-assigned 0

Asid 0032:
  Local storage start 09/25/2009 08:28:28.815183
  Local storage end 09/25/2009 08:28:30.482160
  Local storage capture time 00:00:01.666984
  Tasks reset dispatchable 09/25/2009 08:28:30.482202
  Tasks were nondispatchable 00:00:01.667019
  Defers for frame availability 0
  Pages requiring input I/O 25
  Source page copied to target 448
  Source frames re-assigned 23
  Source AUX slot IDs re-assigned 0

Asid 0033:
  Local storage start 09/25/2009 08:28:28.815195
  Local storage end 09/25/2009 08:28:30.343677
  Local storage capture time 00:00:02.311161
  Tasks reset dispatchable 09/25/2009 08:28:30.343714
  Tasks were nondispatchable 00:00:02.34845
  Defers for frame availability 0
  Pages requiring input I/O 2
  Source page copied to target 428
  Source frames re-assigned 24
  Source AUX slot IDs re-assigned 2

Asid 0001:
  Local storage start 09/25/2009 08:28:27.249772
  Local storage end 09/25/2009 08:28:30.181378
  Local storage capture time 00:00:03.931606
  Tasks reset dispatchable 09/25/2009 08:28:30.284617
  Tasks were nondispatchable 00:00:03.034845
  Defers for frame availability 0
  Pages requiring input I/O 20
  Source page copied to target 1706
  Source frames re-assigned 252
  Source AUX slot IDs re-assigned 16

Asid 0002:
  Local storage start 09/25/2009 08:28:27.249793
  Local storage end 09/25/2009 08:28:29.560954
  Local storage capture time 00:00:02.311161
  Tasks reset dispatchable 09/25/2009 08:28:30.248399
VERBEXIT IEAVTSFS subcommand

Tasks were nondispatchable 00:00:02.998606
Defers for frame availability 0
Pages requiring input I/O 1
Source page copied to target 295
Source frames re-assigned 52
Source AUX slot IDs re-assigned 0

Asid 002A:
Local storage start 09/25/2009 08:28:28.815210
Local storage end 09/25/2009 08:28:30.487208
Local storage capture time 00:00:01.671997
Tasks reset dispatchable 09/25/2009 08:28:30.487222
Tasks were nondispatchable 00:00:01.672011
Defers for frame availability 0
Pages requiring input I/O 8
Source page copied to target 450
Source frames re-assigned 20
Source AUX slot IDs re-assigned 0

Dump Exits
Exit address 044B76A0
Home ASID 0005
Exit start 09/25/2009 08:28:27.249729
Exit end 09/25/2009 08:28:27.249734
Exit time 00:00:00.000004
Exit attributes: Sdump, Early Global
Defers for frame availability 0
Pages requiring input I/O 10
Source page copied to target 726
Source frames re-assigned 10
Source AUX slot IDs re-assigned 0

Exit address 0441F1F8
Home ASID 0005
Exit time 00:00:00.092324
Exit attributes: Global, Sdump, SYSMDUMP
Exit address 040B2738
Home ASID 0005
Exit end 09/25/2009 08:28:28.814998
Exit time 00:00:00.000257
Exit attributes: Global, Sdump, SYSMDUMP
### VERBEXIT IEAVTSFS subcommand

<table>
<thead>
<tr>
<th>Exit address</th>
<th>Home ASID</th>
<th>Exit start</th>
<th>Exit end</th>
<th>Exit time</th>
<th>Exit attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>03DEB128</td>
<td>0002</td>
<td>09/25/2009 08:28:29.290262</td>
<td>09/25/2009 08:28:29.539836</td>
<td>00:00:00.249573</td>
<td>Local, Sdump, DFP</td>
</tr>
<tr>
<td>044CEDD8</td>
<td>0002</td>
<td>09/25/2009 08:28:29.539837</td>
<td>09/25/2009 08:28:29.539913</td>
<td>00:00:00.000060</td>
<td>Local, Sdump</td>
</tr>
<tr>
<td>0198D4B0</td>
<td>0005</td>
<td>09/25/2009 08:28:29.539913</td>
<td>09/25/2009 08:28:29.560953</td>
<td>00:00:00.021039</td>
<td>Local, Sdump, Nucleus Resident</td>
</tr>
<tr>
<td>03DEB128</td>
<td>0002</td>
<td>09/25/2009 08:28:30.111307</td>
<td>09/25/2009 08:28:30.111461</td>
<td>00:00:00.000153</td>
<td>Local, Sdump, DFP</td>
</tr>
<tr>
<td>044CEDD8</td>
<td>0002</td>
<td>09/25/2009 08:28:30.111461</td>
<td>09/25/2009 08:28:30.111505</td>
<td>00:00:00.000043</td>
<td>Local, Sdump</td>
</tr>
<tr>
<td>02B3EB68</td>
<td>0002</td>
<td>09/25/2009 08:28:30.111505</td>
<td>09/25/2009 08:28:30.111518</td>
<td>00:00:00.000013</td>
<td>Local, Sdump, SYSMDUMP</td>
</tr>
</tbody>
</table>
VERBEXIT IEAVTSFS subcommand

Exit time 00:00:00.000126
Exit attributes: Local, Sdump, DFP

Exit address 044CEDD8
Home ASID 0001
Exit start 09/25/2009 08:28:30.181273
Exit end 09/25/2009 08:28:30.181320
Exit time 00:00:00.000046
Exit attributes: Local, Sdump

Exit address 02B3EB68
Home ASID 0001
Exit start 09/25/2009 08:28:30.181320
Exit end 09/25/2009 08:28:30.181340
Exit time 00:00:00.000019
Exit attributes: Local, Sdump, SYSMDUMP

Exit address 0198D4B0
Home ASID 0001
Exit start 09/25/2009 08:28:30.181340
Exit end 09/25/2009 08:28:30.181378
Exit time 00:00:00.000038
Exit attributes: Local, Sdump, Nucleus Resident

Exit address 0198D4B0
Home ASID 002D
Exit start 09/25/2009 08:28:30.181378
Exit end 09/25/2009 08:28:30.305043
Exit time 00:00:00.193524
Exit attributes: Local, Sdump, Nucleus Resident

Exit address 03DEB128
Home ASID 0003
Exit start 09/25/2009 08:28:30.181378
Exit end 09/25/2009 08:28:30.305043
Exit time 00:00:00.193524
Exit attributes: Local, Sdump, Nucleus Resident

Exit address 044CEDD8
Home ASID 0033
Exit start 09/25/2009 08:28:30.181378
Exit end 09/25/2009 08:28:30.305043
Exit time 00:00:00.193524
Exit attributes: Local, Sdump, DFP

Exit address 02B3EB68
Home ASID 0033
Exit start 09/25/2009 08:28:30.343605
Exit end 09/25/2009 08:28:30.343656
Exit time 00:00:00.000051
Exit attributes: Local, Sdump

Exit address 0198D4B0
Home ASID 0033
Exit start 09/25/2009 08:28:30.343656
Exit end 09/25/2009 08:28:30.343672
Exit time 00:00:00.000015
Exit attributes: Local, Sdump, SYSMDUMP

Exit address 03DEB128
Home ASID 0033
Exit start 09/25/2009 08:28:30.343672
Exit end 09/25/2009 08:28:30.343677
Exit time 00:00:00.000004
Exit attributes: Local, Sdump, Nucleus Resident

Exit address 0198D4B0
Home ASID 0033
Exit start 09/25/2009 08:28:30.343677
Exit end 09/25/2009 08:28:30.481505
Exit time 00:00:00.000147
Exit attributes: Local, Sdump, DFP

Exit address 03DEB128
Home ASID 0032
Exit start 09/25/2009 08:28:30.481505
Exit end 09/25/2009 08:28:30.481653
Exit time 00:00:00.000147
Exit attributes: Local, Sdump, DFP

Exit address 044CEDD8
VERBEXIT IEAVTSFS subcommand

Home ASID 0032
Exit start 09/25/2009 08:28:30.481653
Exit end 09/25/2009 08:28:30.482146
Exit time 00:00:00.000492
Exit attributes: Local, Sdump
Exit address 02B3EB68
Home ASID 0032
Exit start 09/25/2009 08:28:30.482146
Exit end 09/25/2009 08:28:30.482162
Exit time 00:00:00.000016
Exit attributes: Local, Sdump, SYSMDUMP
Exit address 0198D4B0
Home ASID 002A
Exit start 09/25/2009 08:28:30.487094
Exit end 09/25/2009 08:28:30.487169
Exit time 00:00:00.000074
Exit attributes: Local, Sdump, DFP
Exit address 044CEDD8
Home ASID 002A
Exit start 09/25/2009 08:28:30.487169
Exit end 09/25/2009 08:28:30.487196
Exit time 00:00:00.000026
Exit attributes: Local, Sdump, Nucleus Resident
Exit address 02B3EB68
Home ASID 002A
Exit start 09/25/2009 08:28:30.487196
Exit end 09/25/2009 08:28:30.487206
Exit time 00:00:00.000010
Exit attributes: Local, Sdump, SYSMDUMP
Exit address 0198D4B0
Home ASID 002A
Exit start 09/25/2009 08:28:30.487206
Exit end 09/25/2009 08:28:30.487208
Exit time 00:00:00.000001
Exit attributes: Local, Sdump, Nucleus Resident
Exit address 0441F214
Home ASID 002A
Exit start 09/25/2009 08:28:30.487225
Exit end 09/25/2009 08:28:30.517480
Exit time 00:00:00.030254
Exit attributes: Sdump, SYSMDUMP, One Time
Exit address 0408001E
Home ASID 002A
Exit start 09/25/2009 08:28:30.517480
Exit end 09/25/2009 08:28:30.517497
Exit time 00:00:00.000016
Exit attributes: Sdump, SYSMDUMP, One Time
Exit address 02B88408
Home ASID 002A
Exit start 09/25/2009 08:28:30.517497

Chapter 5. IPCS subcommands  5-311
VERBEXIT IEFENFVX subcommand — list ENF listeners

Specify the IEFENFVX verb name on the VERBEXIT subcommand to format a list of Event Notification Facility (ENF) listeners.

Parameters
The VERBEXIT IEFENFVX subcommand has one optional parameter: an ENF event code.

Examples
• To obtain a list of ENF listeners for all the event codes:
  VERBEXIT IEFENFVX
• To obtain a list of ENF listeners for an event code 4:
  VERBEXIT IEFENFVX '4'

VERBEXIT IEFIVAWT subcommand — list pending XCF work for tape allocation

Specify the IEFIVAWT verb name on the VERBEXIT subcommand to format a list of pending XCF work for tape allocation.

Parameters
The VERBEXIT IEFIVAWT subcommand has no parameters.

Example
Obtain a list of pending XCF work for tape allocation.
– Action
  VERBEXIT IEFIVAWT
– Result

IEFHTSWT ANTR Request Queue
IEFOAWTR: 7E721540
ID.......=..... Version.. 0001 Length... 0055
Next...00000000 FuncVal.. 0004 Flags....0000
Function: Merge
SendMemT..E5010000 007FFBF8      MsgBufA.. 00000000
MsgBuf1.. 982F3CAC     MsgBufSP .20  MsgBufKy. 00
MsgBufT... 01000002
MhETOD... C7D9E2F1 F2F14040 00000000 00000001

VERBEXIT IEFIVIGD subcommand — list global tape device information

Specify the IEFIVIGD verb name on the VERBEXIT subcommand to format the global tape devices.

Parameters
The VERBEXIT IEFIVIGD subcommand has no parameters.

Example
Obtain information about global tape devices from the dump.
– Action
  VERBEXIT IEFIVIGD
VERBEXIT JESXCF subcommand — format data for JES XCF component

Specify the JESXCF verb name on the VERBEXIT subcommand to format coupling and consoles information from the JESXCF address space in the dump. This address space is for the JES common coupling services MVS component (JES XCF component).

Parameters
The VERBEXIT JESXCF subcommand has no parameters.

Example
The VERBEXIT JESXCF output may be requested by the IBM Support Center for diagnosis.

VERBEXIT LEDATA subcommand — format Language Environment data

There is one version of the LEDATA subcommand for AMODE 31/24 format, and another for AMODE 64 format. For the latest version of each IPCS LEDATA subcommand, see the following topics:

- For AMODE 31/24 format, see the topics about Formatting and analyzing system dumps and Understanding Language Environment IPCS VERBEXIT LEDATA AMODE 31/24 in z/OS Language Environment Debugging Guide.
- For AMODE 64 format, see the topics about Formatting and analyzing system dumps and Understanding Language Environment IPCS VERBEXIT LEDATA AMODE 64 in z/OS Language Environment Debugging Guide.

Specify the LEDATA verb name and optional parameters on the VERBEXIT subcommand to format diagnostic data for the Language Environment (LE) component of z/OS. LEDATA displays the following:

- A summary of the Language Environment at the time of the dump
- Runtime options
- Storage management control blocks
- Condition management control blocks
- Message handler control blocks
- C Runtime Library control blocks

Syntax

VERBEXIT LEDATA ['parameter [',parameter]'...']

The parameters are:

Report type parameters:

- SUMMARY
- HEAP | STACK | SM
- HPT(value)
- CM
- M6
- CEE DUMP
- COMP(value)
- PTBL(value)
- ALL

Data selection parameters:

- DETAIL | EXCEPTION

Control block selection parameters:

- CAA(caa-address)
- DSA(dsa-address)
- TCB(tcb-address)
- ASID(address-space-ID)
- NTHREADS(value)

Parameters

Report Type Parameters

Use these parameters to select the type of report. You can specify as many reports as you want. If you omit the parameters, the default is SUMMARY.
VERBEXIT LEDATA subcommand

**SUMmary**
Specifies a summary of the Language Environment at the time of the dump. The following information is included:
- TCB address
- Address space identifier
- Language Environment release
- Active members
- Formatted CAA, PCB, RCB, EDB, and PMCB
- Runtime options in effect.

**HEAP | STACK | SM**

**HEAP**
Specifies a report on Storage Management control blocks pertaining to HEAP storage.

**STACK**
Specifies a report on Storage Management control blocks pertaining to STACK storage.

**SM**
Specifies a report on Storage Management control blocks. This is the same as specifying both HEAP and STACK.

**HPT(** value **)**
Specifies the heappools trace (if available) be formatted.

If the value is 0 or *, the trace for every heappools poolid is formatted. If the value is a single number (1-12), the trace for the specific heappools poolid is formatted. If the HPT keyword is specified with no value, the HPT value defaults to 0.

**CM**
Specifies a report on Condition Management control blocks.

**MH**
Specifies a report on Message Handler control blocks.

**CEEDump**
Specifies a CEEDUMP-like report. Currently this includes the traceback, the Language Environment trace, and thread synchronization control blocks at process, enclave, and thread levels.

**COMP(** value **)**
Specifies component control blocks to be formatted, where value is one of the following options:

**C**
Specifies a report on C/C++ Run-Time Control Blocks.

**CIO**
Specifies a report on C/C++ I/O Control Blocks.

**COBOL**
Specifies a report on COBOL-specific Control Blocks.

**PLI**
Specifies a report on PL/I-specific Control Blocks.

**ALL**
Request a report on all the control blocks.

If the value specified in COMP is not valid, the COMP value defaults to ALL.

**Note:** When LEDATA report type ALL is specified, the COMP value defaults to ALL.
VERBEXIT LEDATA subcommand

**PTBL**(value)
Specifies the PreInit tables to be formatted, where value is one of the following options:

**CURRENT**
The PreInit table associated with the current or specified TCB is displayed.

Note that when report type ALL is specified, the PTBL value defaults to CURRENT.

**address**
The PreInit table at the specified address is displayed.

* All active and dormant Preinit tables within the current address space are displayed. This option is time consuming.

**ACTIVE**
The PreInit tables of all TCBs in the address space are displayed.

**ALL**
Specifies all above reports, in addition to a report on C Runtime Library.

**Data Selection Parameters**
Use these parameters to limit the scope of the data in the report. If no data selection parameter is selected, the default is DETAIL.

**DETAIL**
Specifies the formatting of all control blocks for the selected components. Only significant fields in each control block are formatted.

**EXCEPTION**
Specifies validating all control blocks for the selected components. The output produced names only the control block and its address for the first control block in a chain that is not valid. Validation consists of control block header verification at the very least.

**Note:** For the Summary, CEEDUMP, and C Runtime Library reports, only the DETAIL output is available.

**Control Block Selection Parameters**
Use these parameters to select the CAA and DSA control blocks used as the starting points for formatting.

**CAA**(caa-address)
Specifies the address of the CAA. If not specified, the CAA address is obtained from the TCB.

**DSA**(dsa-address)
Specifies the address of the DSA. If not specified, the DSA address is assumed to be the general purpose register (GPR) 13 value for the TCB.

**TCB**(tcb-address)
Specifies the address of the TCB. If not specified, the TCB address of the current TCB from the CVT is used.

**ASID**(address-space-ID)
Specifies the hexadecimal address space ID. If not specified, the IPCS default address space ID is used. This parameter is not needed when the dump only has one address space.

**NTHREADS**(value)
Specifies the number of TCBs for which the traceback will be displayed. If
VERBEXIT LEDATA subcommand

NTHREADS is not specified, value will default to (1). If value is specified as asterisk (*), all TCBs will be displayed.

Example
For an example of the LEDATA output, see z/OS Language Environment Debugging Guide.

VERBEXIT LOGDATA subcommand — format logrec buffer records

Specify the LOGDATA verb name on the VERBEXIT subcommand to format the logrec buffer records that were in storage when the dump was generated. LOGDATA locates the logrec records in the logrec recording buffer and invokes the EREP program to format and print the logrec records. The records are formatted as an EREP detail edit report.

Use the LOGDATA report to examine the system errors that occurred just before the error that caused the dump to be requested.

Parameters
The VERBEXIT LOGDATA subcommand has no parameters.

Example
Format the logrec buffer records in the dump.

– Action
VERBEXIT LOGREC
– Result

DUMP FOR DATSVY02 1 11:12:04 11/29/94

**LOGDATA**

DUMP FOR DATSVY02 2 11:12:05 11/29/94

TYPE: SOFTWARE RECORD
(SVC 13)
REPORT: SOFTWARE EDIT REPORT
DAY,YEAR
FORMATTED BY: IEAVTFDE HB85520
MODEL: 3090
SERIAL: 176280

JOBNAME: *MASTER*
ERRORID: SEQ=00012 CPU=0041 ASID=0001 TIME=10:38:59.6

SEARCH ARGUMENT ABSTRACT

AB/S00F4 PRC5/00000024 REGS/0E00A REGS/0C8B2

SYMPTOM DESCRIPTION

---- -------
AB/S00F4 SYSTEM ABEND CODE: 00F4
PRCS/00000024 ABEND REASON CODE: 00000024
REGS/0E00A REGISTER/PSW DIFFERENCE FOR R0E: 00A
REGS/0C8B2 REGISTER/PSW DIFFERENCE FOR R0C: 00A
SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

RECOVERY ROUTINE TYPE: FUNCTIONAL RECOVERY ROUTINE (FRR)

RECOVERY ROUTINE ACTION
THE RECOVERY ROUTINE RETRIED TO ADDRESS 8209C8E0.
AN SVC DUMP WAS NOT REQUESTED.
NO LOCKS WERE REQUESTED TO BE FREED.
THE SDWA WAS REQUESTED TO BE FREED BEFORE RETRY.
THE REGISTER VALUES TO BE USED FOR RETRY:

REGISTERS 0-7
GR: 1E050019 7F0C48C 00000000 7F0C658 7F0B250 7F0B0A8 7F0C858 7F0C830
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

REGISTERS 8-15
GR: 0065F1C 020C159B 00000001 02518A7B 82517A7C 7F0C6A8 7F0B478 8251AB48
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
VERBEXIT LOGDATA subcommand

HEXDECIMAL DUMP

HEADER
+000  40831820  00000000  0094224F  103859E9  | C....M.|
+010  FF176280  30900000  |........|

JOBNAME
+000  5CD4C1E2  E3C5D95C  |*MASTER*|

SDWA BASE
+000  00000C60  440F4000  00000000  00000000  |...-..|
+010  00000000  00000000  6204000C  440F4000  |........|
+020  00000000  7F70C65B  00FCF420  6204000C  |"..,.A|
+030  00000024  00FD1C00  00000000  7F70C4CB  |........|
+040  00FCF3EC  02518A7B  82517A7C  7F70C6A8  |.3...B.0".F|
+050  82518324  00000024  00000000  00000000  |B..C.|
+060  00000000  00000000  075C1000  8251832E  |.....*.B.C.|
+070  0002000D  03C96001  076C0000  8209DD00  |I.....B.)|
+080  0002000D  03C96001  1E050019  7F70C4BC  |I.....".D.|
+090  00000000  7F70C65B  7F70B250  7F70B0AB  |"..,.F."."Y|
+0A0  7F70C65B  7F70B330  0065F16C  020C15AB  |H..H..">..<Q|
+0B0  00000001  02518A7B  82517A7C  7F70C6A8  |......A8."0".F.Y|
+0C0  7F70B478  82518488  00000000  00000000  |"..B|
+0D0  00000000  00000000  00000000  00000000  |...............|
+0E0  00000000  00000000  040A0001  00000041  |...............|
+0F0  8209C8E0  00F93AFC  00000000  048C0000  |B.H\..9|
+100  00000000  00000000  00000000  00000000  |...............|
+110  00000000  00000000  00000000  00000000  |...............|
+120  0001000C  00000000  00000000  00000000  |...............|
+130  00000000  00000000  00000000  00F93A68  |...............|
+140  00000000  00000000  00000000  00000000  |...............|
+150  00000000  00000000  00000000  00000000  |...............|
+160  00000000  00000000  00000000  00000000  |FFFF0000|
+170  00F93D08  80000001  00010001  00000000  |.9.8|
+180  00000000  00000000  00000000  0005D9A4  |...............RU|
+190  00FF0000  |.....|

NO DATA EXISTS IN THE VARIABLE RECORDING AREA

SDWA FIRST RECORDABLE EXTENSION (SDWARC1)
+000  00000000  00000000  00000000  00000000  |...............|
+010  00000000  00000000  00000000  00000000  |...............|
+020  00000000  00000000  00000000  00000024  |...............|
**VERBEXIT MMSDATA subcommand** — format MVS message service data

Specify the MMSDATA verb name on the VERBEXIT subcommand to format diagnostic data from the MVS message service (MMS).

**Parameters**

The VERBEXIT MMSDATA subcommand has no parameters.

**Example**

For an example of the MMSDATA output, see the MMS component in z/OS MVS Diagnosis: Reference.

---

**VERBEXIT MTRACE subcommand** — format master trace entries

Specify the MTRACE verb name on the VERBEXIT subcommand to display:

- The master trace table entries for the dumped system. This table is a wraparound data area that holds the most recently issued console messages in a first-in, first-out order.
- The NIP hard-copy message buffer.
- The branch entry and NIP time messages on the delayed issue queue.
VERBEXIT MTRACE subcommand

Parameters
The VERBEXIT MTRACE subcommand has no parameters.

Example
Format master trace table entries in the dump.

- Action
  VERBEXIT MTRACE
- Result
VERBEXIT MTRACE subcommand

*** NIP MESSAGE TABLE ***
UNABLE TO ACCESS UCM - NIP MESSAGE TRACE TERMINATED

*** MASTER TRACE TABLE ***

The VERBEXIT MTRACE continues with messages like the above.

VERBEXIT NUCMAP subcommand — map modules in the nucleus

Specify the NUCMAP verb name and optional parameters on the VERBEXIT subcommand to format a map of the modules in the nucleus when the dump was loaded. The map gives for each module the name, entry point, entry point attributes, and length. When the input data set does not contain the nucleus, IPCS issues an error message.
VERBEXIT NUCMAP subcommand

Syntax

VERBEXIT NUCMAP [ 'parameter [,parameter]...' ]

The parameters are:

[ EPA ]

[ MODNAME ]

Parameters

If you omit the parameters, the output is sorted and listed twice: first, by the module entry point addresses, and second, by the module names.

EPA

Sorts the output according to module entry point addresses.

MODNAME

Sorts the output according to module names.

Example

Obtain a map of the modules in the nucleus.

- Action
  
  VERBEXIT NUCMAP

- Result
The nucleus map sorted numerically by entry point address (EPA) continues with data like the above.
This is a continuation of the nucleus map sorted alphabetically by name.

<table>
<thead>
<tr>
<th>NAME</th>
<th>LOCATION</th>
<th>ATTR</th>
<th>LENGTH</th>
<th>CSECT-NAME</th>
<th>NAME</th>
<th>LOCATION</th>
<th>ATTR</th>
<th>LENGTH</th>
<th>CSECT-NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABN00D0</td>
<td>011F012C</td>
<td>0E</td>
<td>000854</td>
<td>ISGJDI</td>
<td>AVFFS</td>
<td>01157FC0</td>
<td>1E</td>
<td>000858</td>
<td>AVFFS</td>
</tr>
<tr>
<td>ACMRSAH</td>
<td>00FF41A0</td>
<td>0B</td>
<td>000400</td>
<td>IEAVELIT</td>
<td>AVFFSFR</td>
<td>01157FE0</td>
<td>0E</td>
<td>000408</td>
<td>AVFFS</td>
</tr>
<tr>
<td>ACNRETRY</td>
<td>0118BF84</td>
<td>0E</td>
<td>00049C</td>
<td>IEAVTCR1</td>
<td>AVFFS</td>
<td>01158AB8</td>
<td>0E</td>
<td>000488</td>
<td>AVFFS</td>
</tr>
<tr>
<td>ACNSTART</td>
<td>01187E16</td>
<td>0E</td>
<td>00000A</td>
<td>IEAVCR</td>
<td>AVFSNR</td>
<td>01158AB8</td>
<td>0E</td>
<td>000488</td>
<td>AVFFS</td>
</tr>
<tr>
<td>ACNSUPER</td>
<td>0118BFAE</td>
<td>0E</td>
<td>000402</td>
<td>IEAVTCR1</td>
<td>AVFSSXIT</td>
<td>01159000</td>
<td>0E</td>
<td>000358</td>
<td>AVFFS</td>
</tr>
<tr>
<td>ADISP</td>
<td>00FF3F00</td>
<td>0B</td>
<td>000780</td>
<td>IEAVELIT</td>
<td>AVFIX</td>
<td>00FF43A4</td>
<td>0B</td>
<td>0002DC</td>
<td>IEAVELIT</td>
</tr>
<tr>
<td>AFPCN1</td>
<td>00FC6554</td>
<td>02</td>
<td>00010C</td>
<td>IECVAFP1</td>
<td>AVFMS</td>
<td>01159360</td>
<td>1E</td>
<td>000930</td>
<td>AVFMS</td>
</tr>
<tr>
<td>AFPTDC</td>
<td>00FC649C</td>
<td>02</td>
<td>00010C</td>
<td>IECVAFP1</td>
<td>AVFTMP</td>
<td>01159790</td>
<td>0E</td>
<td>000570</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AFPS</td>
<td>00FC6490</td>
<td>02</td>
<td>00010D</td>
<td>IECVAFP1</td>
<td>AVFMT</td>
<td>01159380</td>
<td>0E</td>
<td>000910</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AFTRAP</td>
<td>00FC6496</td>
<td>02</td>
<td>00010C</td>
<td>IECVAFP1</td>
<td>AVFTB</td>
<td>01157FB0</td>
<td>1E</td>
<td>000968</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHLHEAD</td>
<td>00FC9D7B</td>
<td>0E</td>
<td>000408</td>
<td>AHLMCIH</td>
<td>AVFTB</td>
<td>01158CB0</td>
<td>0E</td>
<td>000708</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHCMIH</td>
<td>00FC7C10</td>
<td>1E</td>
<td>000280</td>
<td>AVFFTBY</td>
<td>0115A542</td>
<td>1E</td>
<td>000226</td>
<td>AVFTR</td>
<td></td>
</tr>
<tr>
<td>AHCMIHB</td>
<td>00FC7C06</td>
<td>02</td>
<td>00010A</td>
<td>AHLMCIH</td>
<td>AVFTC</td>
<td>0115A678</td>
<td>1E</td>
<td>000550</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHDSTLCS</td>
<td>00FC594E</td>
<td>02</td>
<td>000072</td>
<td>AHLMCIH</td>
<td>AVFTCLPE</td>
<td>0115A7DB</td>
<td>0E</td>
<td>000350</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHPVCCRB</td>
<td>00FC594E</td>
<td>02</td>
<td>000072</td>
<td>AHLMCIH</td>
<td>AVFTCLPM</td>
<td>0115A698</td>
<td>0E</td>
<td>000530</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHPVFMR</td>
<td>0115B2C8</td>
<td>1E</td>
<td>000818</td>
<td>ACFVREL</td>
<td>AVFTR</td>
<td>0115A698</td>
<td>0E</td>
<td>000530</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHPVFMR2</td>
<td>0115B2C8</td>
<td>1E</td>
<td>000818</td>
<td>ACFVREL</td>
<td>AVFTR</td>
<td>0115A698</td>
<td>0E</td>
<td>000530</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHPVFMR3</td>
<td>0115B2C8</td>
<td>1E</td>
<td>000818</td>
<td>ACFVREL</td>
<td>AVFTR</td>
<td>0115A698</td>
<td>0E</td>
<td>000530</td>
<td>AVFTR</td>
</tr>
<tr>
<td>AHPVFMR4</td>
<td>0115B2C8</td>
<td>1E</td>
<td>000818</td>
<td>ACFVREL</td>
<td>AVFTR</td>
<td>0115A698</td>
<td>0E</td>
<td>000530</td>
<td>AVFTR</td>
</tr>
</tbody>
</table>

The nucleus map sorted alphabetically by name continues with data like the above.
VERBEXIT SADMPMSG subcommand

VERBEXIT SADMPMSG subcommand — format stand-alone dump message log

Specify the SADMPMSG verb name on the VERBEXIT subcommand to format the SADMP program run-time dump message log. These messages can help identify problems with stand-alone dump output.

Note: This log does not contain messages issued following abnormal ending errors on the SADMP output tape, or after the tape was unloaded following normal ending of SADMP.

See z/OS MVS Diagnosis: Tools and Service Aids for information about the stand-alone dump program.

Parameters
The VERBEXIT SADMPMSG subcommand has no parameters.

Example
Format the stand-alone dump program run-time dump message log.
- Action
  VERBEXIT SADMPMSG
- Result

*** STAND-ALONE DUMP VIRTUAL DUMP MESSAGE LOG ***

AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> -DUMP SP(ALL) IN ASID(ALL)
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> -DUMP DAT OF ASID(ALL)
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> -DUMP PA OF DAT
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> -END
AMD010I PROCESSING ASID=0001 ASCB=00FD5E80 JOBNAME=MASTER
AMD076I PROCESSING DATA SPACE JES2IRDS, OWNED BY ASID 0001.
AMD010I PROCESSING ASID=0002 ASCB=00F36600 JOBNAME=PCAUTH
AMD010I PROCESSING ASID=0003 ASCB=00F36400 JOBNAME=RASP
AMD010I PROCESSING ASID=0004 ASCB=00F36200 JOBNAME=TRACE
AMD010I PROCESSING ASID=0005 ASCB=00F35E00 JOBNAME=GRS
AMD057I COMPLETED SPECIFIC DUMPING FOR GRS.
AMD010I PROCESSING ASID=0006 ASCB=00F48400 JOBNAME=DUMPSRV
AMD0290 REPLY W TO WAIT AFTER NEXT FULL SCREEN, ELSE REPLY N; REPLY= -W
AMD076I PROCESSING DATA SPACE SDUMPCSA, OWNED BY ASID 0006.
AMD010I PROCESSING ASID=0008 ASCB=00F50E00 JOBNAME=CONSOLE
;
AMD056I DUMPING OF VIRTUAL STORAGE COMPLETED.

VERBEXIT SRMDATA subcommand — format System Resource Manager data

Specify the SRMDATA verb name on the VERBEXIT subcommand to format diagnostic data from the system resources manager (SRM).
VERBEXIT SRMDATA subcommand

Note: If an SVC dump generated the input data set, valid queues might appear to be incorrect because the queues can change while the SVC dump is being generated.

Syntax

```
VERBEXIT SRMDATA ['parameter [,parameter]...']
```

The parameters are:

```
[ ASQLIM ]
[ ENCQLIM ]
[ ENQQLIM ]
[ QLIM ]
```

Parameters
The parameters are provided to limit the amount of output produced.

**ASQLIM**

The maximum number of OUCB elements processed by SRMDATA per OUCB queue.

**ENCQLIM**

The maximum number of ENCB elements processed by SRMDATA per ENCB queue.

**ENQQLIM**

The maximum number of ERE/EHE elements processed by SRMDATA per queue.

**QLIM**

The maximum number of all other queue elements, not listed above, processed by SRMDATA per queue.

Example

For an example of the SRMDATA output, see the SRM component in [z/OS MVS Diagnosis: Reference](z/OS MVS Diagnosis: Reference).

---

VERBEXIT SUMDUMP subcommand — format SVC summary dump data

Specify the SUMDUMP verb name on the VERBEXIT subcommand to locate and format the summary dump data that an SVC dump or a stand-alone dump contains.

Note: For stand-alone dumps, SUMDUMP formats any summary dump records it finds in the buffers. Such records can exist in the buffers if an SVC dump is in progress when a stand-alone dump is generated.

Parameters
The VERBEXIT SUMDUMP subcommand has no parameters.

Example

Obtain the summary dump data.

- Action
  - VERBEXIT SUMDUMP
- Result
VERBEXIT SYMPTOM subcommand

Specify the SYMPTOM or SYMPTOMS verb name on the VERBEXIT subcommand to format the symptom strings contained in the header record of an SVC dump, SYSMDUMP dump, or stand-alone dump. The symptom strings are:

- The primary symptom string, consisting of:
  - Symptoms provided by dump analysis and elimination (DAE) in the dump header record when the dump is generated
  - Symptoms in a literal definition, if it exists, of the IPCS symbol SECONDARYSYMPTOMS

- The secondary symptom string, provided by IPCS as part of post-dump processing and including symptoms provided by the IPCS add symptom service

For the structure of symptom strings in a dump, see search arguments in z/OS Problem Management.

There is a restriction on the space available to secondary symptom strings in the dump header. If the display does not contain the expected information, BROWSE the dump HEADER. Truncated secondary symptom strings end with the characters '-Truncated'. The entire symptom string is only available if it has been explicitly placed into the dump, or the storage pointed to by the SYMAD pointer in the SDUMP parameter list is available.

You can use the IPCS add symptom service to add secondary symptom strings up to 256 bytes. IPCS creates the literal definition of the symbol SECONDARYSYMPTOMS from the full symptoms that fit in the first 256 bytes of the new symptom string.

Parameters
The VERBEXIT SYMPTOM subcommand has no parameters.

Example
Obtain the symptom strings from the dump.

- Action
  VERBEXIT SYMPTOM
- Result

<table>
<thead>
<tr>
<th>STORAGE TYPE</th>
<th>RANGE START</th>
<th>RANGE END</th>
<th>ASID ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMLST RNAGE --</td>
<td>023BCD70</td>
<td>023BCD7F</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>SUMLST RNAGE --</td>
<td>017E8000</td>
<td>017E8FFF</td>
<td>0001 (COMM)</td>
</tr>
<tr>
<td>SUMLST RNAGE --</td>
<td>01F98000</td>
<td>01F9CFFF</td>
<td>0001 (COMM)</td>
</tr>
<tr>
<td>SUMLST RNAGE --</td>
<td>02166000</td>
<td>02167FFF</td>
<td>0001 (COMM)</td>
</tr>
<tr>
<td>PS -----</td>
<td>00000000</td>
<td>00001FFF</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>PCCA -------</td>
<td>00F43000</td>
<td>00F4324F</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>LCCA -------</td>
<td>00F82000</td>
<td>00F82A47</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>LCCX -------</td>
<td>021C7000</td>
<td>021C771F</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>INT HANDLER DUCT</td>
<td>02232FC0</td>
<td>02232FFF</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>I.H. LINKAGE STK</td>
<td>02262000</td>
<td>0226202F</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>REGISTER AREA --</td>
<td>0000E900</td>
<td>00010FFF</td>
<td>001E</td>
</tr>
<tr>
<td>REGISTER AREA --</td>
<td>00FC4000</td>
<td>00FC6FFF</td>
<td>001E (COMM)</td>
</tr>
<tr>
<td>REGISTER AREA --</td>
<td>00000001_7F5A000</td>
<td>00000001_7F5800FF</td>
<td>001E</td>
</tr>
<tr>
<td>REGISTER AREA --</td>
<td>7FFFE000</td>
<td>7FFFEFFF</td>
<td>001E</td>
</tr>
</tbody>
</table>
VERBEXIT VSMDATA subcommand — format virtual storage management data

Specify the VSMDATA verb name and optional parameters on the VERBEXIT subcommand to format diagnostic data from virtual storage management (VSM).

Syntax

VERBEXIT VSMDATA [ 'parameter [,parameter]...' ]

The parameters are:

- [CONTROLBLOCKS] [ALL] [DETAIL]
  [SUMMARY]
- [CURRENT]
- [ERROR]
- [TCBERROR]
- [NOASIDS]
- [ASIDLIST(asidlist)]
- [JOBNAME(joblist) | JOBLIST(joblist)]
- [GLOBAL | NOGLOBAL]

- [OWNCOMM [([CSA] | [SQA])]]
  [SUMMARY]
- [DETAIL]
  [ALL]
  [ASIDLIST(asidlist)]
- [SYSTEM]
  [SORTBY(ASIDADDR | ASIDLEN | ADDRESS | TIME | LENGTH)]
  [CONTENTS(YES | NO)]

Parameters

Report Type Parameters

Use these parameters to select the type of report. If you omit a report type parameter, the default is CONTROLBLOCKS. Both the CONTROLBLOCKS and OWNCOMM parameters have two additional report type parameters — SUMMARY and DETAIL. For the CONTROLBLOCKS report, the default is DETAIL. For the OWNCOMM report, the default is SUMMARY.

CONTROLBLOCKS

Specifies a report of VSM control blocks. The blocks formatted depend on
VERBEXIT VSMDATA subcommand

the associated parameters: ALL, DETAIL, SUMMARY, CURRENT, ERROR, TCBERROR, NOASIDS, ASIDLIST, JOBNAME, GLOBAL, and NOGLOBAL.
The CONTROLBLOCKS parameter is the default; the following two commands produce the same report:

VSMDATA ALL NOGLOBAL
VSMDATA CONTROLBLOCKS ALL NOGLOBAL

OWNCOMM [([CSA] [SQA])]
Requests CSA tracker reporting from VERBEXIT VSMDATA. OWNCOMM may be entered with a CSA option, an SQA option, or both. When only one of the options is entered, it indicates that the report should only contain information pertaining to the referenced areas of common storage. Reporting regarding both may be explicitly requested or implied by omission of all qualifying options.

When you use an abbreviated form of OWNCOMM, enter OWNC at minimum.

SUMMARY
Specifies a summary CONTROLBLOCKS or OWNCOMM report. SUMMARY is the default for the OWNCOMM report but not for the CONTROLBLOCKS report. For more information about the output produced by the VERBX VSMDATA A CONTROLBLOCKS SUMMARY report, see the VSM component in z/OS MVS Diagnosis: Reference.

DETAIL
Specifies a detailed CONTROLBLOCKS or OWNCOMM report. DETAIL is the default for the CONTROLBLOCKS report but not for the OWNCOMM report.

Address Space Selection Parameters for CONTROLBLOCKS
Use these parameters to obtain data for particular address spaces, which you specify by their address space identifiers (ASIDs). If you omit these parameters with CONTROLBLOCKS, the default is CURRENT. For more information, see the select ASID service in z/OS MVS IPCS Customization.

ALL
Specifies formatting of all VSM control blocks for LSQA and the private area for all address spaces in the dump.

CURRENT
Specifies formatting of the VSM control blocks for LSQA and the private area for the address spaces that were current when the system wrote the dump.

ERROR
Specifies formatting of VSM control blocks for LSQA and the private area for any address space with an MVS error indicator or containing a task with an error indicator.

TCBERROR
Specifies processing of VSM control blocks for LSQA and the private area for any address space containing a task with an error indicator. Blocks for address spaces with an error indicator are not processed.

NOASIDS
Suppresses formatting of VSM control blocks for LSQA or the private area for any address space.
VERBEXIT VSMDATA subcommand

ASIDLIST(asidlist)
   Specifies one or more ASIDs for the address spaces for which you want to process VSM control blocks for LSQA and the private area.

   The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When specifying a range, separate the first and last ASIDs in the range with a colon. When specifying a list, separate the list members with commas or blanks.

   The ASID can be 1 through 32767 (decimal). You can specify as many ASIDs as you need; there is no system-imposed maximum.

JOBNAME(joblist) | JOBLIST(joblist)
   Specifies one or more job names whose associated address spaces are to be processed for the VSM control blocks for LSQA and the private area. Use commas to separate the job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names.

Data Selection Parameters for CONTROLBLOCKS
Use these parameters to limit the scope of the data in the report. If you omit a data selection parameter, the default is GLOBAL.

GLOBAL or NOGLOBAL
   Specifies or suppresses formatting of VSM control blocks for the SQA and CSA.

   GLOBAL specified the formatting.
   NOGLOBAL suppresses the formatting.

Address Space Selection Parameters for OWNCOMM DETAIL
Use these parameters to obtain data from particular address spaces, which you specify by their address space identifiers (ASIDs). If you omit these parameters with OWNCOMM DETAIL, the default is ALL. For more information, see the select ASID service in z/OS MVS IPCS Customization.

ALL
   Specifies formatting of data about CSA, ECSA, SQA, and ESQA storage for all address spaces in the dump.

ASIDLIST(asidlist)
   Specifies a list of ASIDs for the address spaces for which you want data about CSA, ECSA, SQA, and ESQA storage.

   The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas or blanks.

   The ASID can be 1 through 32767 (decimal). You can specify as many ASIDs as you need; there is no system-imposed maximum.

Data Selection Parameters for OWNCOMM DETAIL
Use these parameters to limit the scope of the data in the report.

SYSTEM
   Requests data about CSA, ECSA, SQA, and ESQA storage that the system uses; this storage is not "owned" by any particular address space or job.

SORTBY(ASIDADDR | ASIDLEN | ADDRESS | TIME | LENGTH)
   Indicates how IPCS is to sort the list of requests for CSA, ECSA, SQA, or ESQA storage:

   ASIDADDR   Sort the output by address space identifier, in ascending
VERBEXIT VSMDATA subcommand

order. When two or more entries have the same ASID, IPCS sorts these entries by storage address. If you omit a qualifying value with SORTBY, the default is ASIDADDR.

ASIDLEN  Sort the output by address space identifier, in ascending order. When two or more entries have the same ASID, IPCS sorts these entries by the length of the storage at the reported address.

ADDRESS  Sort the output by storage address, in ascending order.

TIME    Sort the output by the time at which the system processed the request to obtain storage, starting with the oldest request.

LENGTH  Sort the output by the length of the storage represented by each entry, starting with the smallest length value.

CONTENTS(YES | NO)  Indicates whether IPCS is to display the contents of the first 4 words of the data at the storage address. If an error occurs when the system tries to access the storage, the message Data ------] Not Available appears in this field.

If you omit CONTENTS, CONTENTS(YES) is the default.

Example 1
Format information about CSA, ECSA, SQA, and ESQA storage for address space identifiers 1, 6, 7, 8, and 9, and sort the output by storage length.
– Action
  VERBX VSMDATA 'OWNCOMM DETAIL SORTBY(LENGTH) ASIDLIST(1,6:9)'

Example 2
Format information about CSA, ECSA, SQA, and ESQA storage for all address space identifiers, and sort the output by address.
– Action
  VERBX VSMDATA 'OWNCOMM DETAIL ALL SORTBY(ADDRESS)'
  – Result
  For an example of the VSMDATA output, see the VSM component in z/OS MVS Diagnosis: Reference.

VLFDATA subcommand — format virtual lookaside facility data

Use the VLFDATA subcommand to generate diagnostic reports about virtual lookaside facility (VLF) activity in the system. Use the report type parameters to choose the kinds of VLF-related information that you want to see.

Syntax
VLFDATA subcommand

VLFDATA

-------- Report Type Parameters -----------------------------
[ CLASS(vlfclass) ] [ ALL ]
[ SHORT ]
[ MAJOR(majortype) ]
[ MINOR(minortype) ]

[ EXCEPTION ]
[ STORAGE [(vlfclass)] ]
[ SUMMARY ]
[ STATS [(vlfclass)] ]
[ USER [(vlfclass)] ]

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DSNNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]

[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

Report Type Parameters
Use these parameters to select the type of report. If you omit a report type parameter, the default is SUMMARY.

Note: In the following descriptions, vlfclass is a class name consisting of 1 to 7 alphanumeric characters or the following characters:
   $ (X'5B')
   # (X'7B')
   @ (X'7C')

CLASS(vlfclass)
Requests a report containing information about a VLF object class. Use vlfclass to specify a particular VLF class.

The CLASS reports you can request are:

Report Processing

ALL Requests all major/minor information available for the specified class.

SHORT Requests a more detailed version of the SUMMARY report for the specified class.

MAJOR(majortype) Requests a report containing details about
**VLFDATA subcommand**

all VLF objects associated with the specified major name. Specify this parameter alone or with MINOR.

The majorname can consist of up to 64 characters specified in hexadecimal, character, or binary notation.

**MINOR(minorname)**

Requests a report containing information about all VLF objects associated with the specified minor name. Specify this parameter alone or with MAJOR. The minorname can consist of up to 64 characters specified in hexadecimal, character, or binary notation.

**EXCEPTION**

Requests a report containing information about inconsistencies detected during verification of VLF dump data.

**STORAGE [(vlfclass)]**

Requests a report describing the storage management of VLF data spaces.

The vlfclass is optional, and specifies the class for which you want to see a STORAGE report. If you do not supply any class names, the report will contain storage information for all classes.

**SUMMARY**

Requests a report containing general information about each class that uses VLF services. The report includes the class type, its status at the time of the dump, related data space information, and some statistics.

**USER [(vlfclass)]**

Requests a report containing information about all identified users for the non-VLF address space that was using a VLF function at the time of error. This non-VLF address space is associated with VLF through use of a user token.

The vlfclass is optional; it limits the report to identified users for the specified class.

**STATS [(vlfclass)]**

Requests a report containing statistics.

The vlfclass is optional; it limits the report to the specified class.

**Return Codes**

See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the VLFDATA subcommand.

**Example**

See the VLF component in [z/OS MVS Diagnosis: Reference](https://www.ibm.com) for examples of VLFDATA output.

---

**WHERE subcommand — identify an area at a given address**

Use the WHERE subcommand to identify the area in a dump in which an address resides. IPCS provides a report containing:

- The address space text
- The specified address
- The name of the area in the dump at the specified address. The name can be:
WHERE subcommand

- The name of a load module. For nucleus CSECTs, the load module name is IEANUC0x, where x is obtained from field CVTNUCLS. Externally defined CSECTs within the nucleus are identified following the load module name. Externally defined CSECTs in other load modules are not displayed.

To be displayed, the module name must conform to the following naming conventions:
- The name is 1 through 8 characters.
- The first character is an uppercase EBCDIC letter or one of the following national characters:
  $ (X'5B')
  # (X'7B')
  @ (X'7C')
- The remaining characters are uppercase EBCDIC letters, national characters, or EBCDIC decimal digits.

If a module name does not conform to these conventions, IPCS displays:

MODULE(SPECIALNAME)

- The name of a control block. The parameter STRUCTURE is displayed followed by the control block name.

- The name of an area of storage that is not a module or a control block. IPCS displays AREA followed by the name of the area.

• The offset into the identified area.

• The name of the system area containing the specified address, which can be:
  - Common service area (CSA)
  - Fixed link pack area (FLPA)
  - Modified link pack area (MLPA)
  - Pageable link pack area (PLPA)
  - Private
  - Prefixed save areas (PSA)
  - Read only nucleus
  - Read/write nucleus
  - System queue area (SQA)

If after examining the return code, IPCS cannot identify the area pointed to by the specified address, IPCS issues the following message:

BLS18451I Unable to identify the area at 'addr space' address xxxxxxxx

If IPCS issues this message, enter one or more of the dump analysis subcommands, such as SUMMARY and STATUS, then reenter the WHERE subcommand. Based the dump processing for the analysis subcommands, IPCS may now be able to identify the area.

The detail in the report generated by the WHERE subcommand depends to a large extent on previous processing of the dump. For example, if after initializing a dump, you enter WHERE, IPCS generates a minimal report. But if you reenter WHERE after entering a number of subcommands, IPCS will probably produce a more detailed report.

Note: The WHERE subcommand may modify X, the current address, as follows:
1. If WHERE can associate the location identified by data-descr with a block of storage containing that location, X is set to describe the block of storage containing the location.
WHERE subcommand

2. If WHERE cannot associate the location identified by data-descr with a block of storage containing that location, X is set to describe the location identified by data-descr.

WHERE will not change X if error conditions occur, such as syntax errors or an unresolvable data-descr.

When used as a primary command, WHERE stacks a pointer to the address, but does not change the value of X. Use option 1 (BROWSE) of the IPCS dialog to find the pointer.

You can invoke WHERE as an IPCS subcommand or as an IPCS primary command. (This section refers to both the subcommand and the primary command as the “WHERE command.”) The WHERE command is useful for identifying locations of addresses found in other reports produced by IPCS subcommands.

For specified addresses in each of the system areas, the WHERE command names different areas in the dump, some more helpful than others.

Addresses in the LPA and Nucleus

The WHERE command has the greatest benefit when used on addresses in the following system areas:
- Fixed link pack area (FLPA)
- Modified link pack area (MLPA)
- Pageable link pack area (PLPA)
- Read-only nucleus
- Read-write nucleus.

For addresses in these areas, the WHERE command returns the name of a load module.

The WHERE command provides the most specific information for addresses located in the nucleus. The WHERE command provides the name of the externally defined CSECTs within the nucleus in which the address is located. They are identified following the load module name. For nucleus CSECTs, the load module name is IEANUCOx, where x is obtained from field CVTNUCLS.

Externally defined CSECTs in other load modules are not displayed.

When you invoke WHERE for an address in any of the parts of the LPA, it returns the name of a load module that contains a number of CSECTs. To find the exact CSECT you are looking for, you must do one of the following:

- If the address is in the section of dump that fits into memory, you can enter the WHERE subcommand from the Browse option of the IPCS dialog. When you press F3 to exit the WHERE output, the Browse panel will scroll to the location of that CSECT in the dump.

- If the address is not in the section of dump in memory, you can use the AMBLIST service aid to format and print the load module. The AMBLIST service aid gives you a list of the component CSECTs in the load module. See z/OS MVS Diagnosis: Tools and Service Aids for more information about using AMBLIST.

Addresses in private storage

The WHERE primary command can also be used on addresses in private and extended private area storage.

- Private area analysis may identify load modules and offsets within them.
- It may also associate the address of interest with data areas.
- z/OS R2 IPCS adds the ability to associate addresses with pages containing application subpools, AREA(SUBPOOLSKEY(key), where
WHERE subcommand

sp A three-digit decimal subpool number between 0 and 255.
key A two-digit decimal storage key between 0 and 15.

The IPCS storage map entries describe subpools in increments of 4096-byte pages associated with subpools rather than the 8-byte units of allocated storage within them.

Addresses in other areas of storage
The WHERE primary command can also be used on addresses in other areas of storage:
- Common storage area (CSA)
- Prefixed save area (PSA)
- System queue area (SQA).

For addresses in these areas, the information provided is less specific than the information provided for the LPA and nucleus addresses. When issued on an address in these areas, WHERE returns one of the following:
- The name of a control block. The parameter STRUCTURE is displayed followed by the control block name.
- The name of an area of storage (not a module or control block). The parameter AREA is typically displayed followed by the name of the area.
- The names of the load modules that are loaded by LOAD with GLOBAL=YES option from the current ASID, if the address is in the CSA or ECSA storage.

When WHERE Does Not Work
If after examining the return code IPCS determines that the area pointed to by the specified address cannot be identified, IPCS issued message BLS18451I with the address and ASID:

BLS18451I Unable to identify the area at ASID(X'0032') address 005CD478

This situation sometimes occurs when the dump directory does not contain enough information about the area of the dump. Try entering the SUMMARY or STATUS subcommand. These subcommands should fill missing information in the dump directory. Then reenter the WHERE subcommand:

ASID(X'0032') 005CD478. AREA(CURRENT)+5CC478 IN PRIVATE

The detail of the report generated by the WHERE command depends to a large extent on how much you have processed the dump. For example, if after initializing a dump, you enter WHERE, IPCS generates a minimal report. But if you enter WHERE again later in your IPCS session, after entering a number of subcommands, a more detailed report will probably be produced.

Syntax
WHERE subcommand

{ WHERE } data-descr
{ W } [SELECT([AREA][MODULE][STRUCTURE])]

------- SETDEF-Defined Parameters -----------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

data-descr
Specifies an address in a dump through the data description parameter, which consists of five parts:
- An address (required)
- Address processing parameters (see note below)
- An attribute parameter (optional)
- Array parameters (optional)
- A remark parameter (optional)

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter.

Notes:
1. An ASID may optionally be specified as part of the data description parameter. If the specified address is in private storage, and no ASID is specified, the default ASID is the only ASID searched.
2. ACTIVE, MAIN, and STORAGE cannot be specified.

SELECT([AREA][MODULE][STRUCTURE])
Specifies the data types to be returned as results of the WHERE command.

AREA
Allows WHERE to associate the location of interest with AREAs.

MODULE
Allows WHERE to associate the location of interest with MODULEs.

STRUCTURE
Allows WHERE to associate the location of interest with STRUCTUREs.

When no selection is specified or all selections are chosen, WHERE can associate the location of interest with AREAs, MODULEs, or STRUCTUREs.

Return Codes
See "Standard subcommand return codes" on page 5-2 for a description of the return codes produced by the WHERE subcommand.

Example 1
Determine where an absolute address is located.

Action
COMMAND ===> IPCS WHERE FD2834.

Result
WHERE subcommand

WHERE generates the following output line, showing that the specified address, in address space X'0058' is X'20D14' bytes into CSECT IOSUCB, which is located in load module IEANUC01 in the READ/WRITE NUCLEUS.

If the address you specified is in the portion of the dump in memory, the WHERE subcommand also takes you to that address in the dump when you press F3 to exit this screen.

If the primary command are used in this example, the item that contains the address are added to the pointer stack. If more than one item contains the address, the item with the smallest offset are added to the pointer stack.

Example 2
Use WHERE from system trace table output, which provides a history of the most recent events in the system. The WHERE command can save you from having to leave the system trace table to find the information needed. For example, if you are going through the table and you see a PSW that interests you, you can use the WHERE command to find out to what module the PSW points. Instead of having to use the VERBEXIT NUCMAP, LPAMAP, or go into the Browse panel of the IPCS dialog, you can type WHERE directly from the system trace table and find out the module name. Also, if you enter WHERE as a primary command it will put a pointer to the module on the stack.

Choose option 4 from the IPCS Primary Option Menu. Then, enter the system trace table with:

```plaintext
==> SYSTRACE
```

Now, enter WHERE on the command line of the system trace table.
WHERE subcommand

WHERE generates the following dump display reporter panel. It tells you that the address is 03D0 hexadecimal bytes into load module IGC0004B in the extended PLPA.

- Result

WHERE generates the following dump display reporter panel. It tells you that the address is 03D0 hexadecimal bytes into load module IGC0004B in the extended PLPA.

Because WHERE was invoked as a primary command, WHERE also stacks a pointer to the beginning of the load module, X'1D07000'. The pointer will appear in the Browse option of the IPCS dialog. The following shows using WHERE in the system trace table.
Example 3

Use WHERE from a logrec buffer in a dump. WHERE can help you look through this table. For example, if you are examining the error PSW in the VERBEXIT LOGDATA report and want to know where address X'120E298' in the error PSW points to, you can use WHERE directly from this screen.

- Result
First, all items that contain this address are displayed using the dump display reporter panel. The message below shows that the PSW you want to examine more closely is 200 bytes into CSECT IGVSTSKT. That CSECT is in load module IEANUC01 in the read-only nucleus.

When you press F3 to exit this screen, IPCS will stack the pointer to the beginning of the CSECT, so when you go back into Browse you can look at
WHERE subcommand

all of the detailed information in that register or PSW.

<table>
<thead>
<tr>
<th>DNAME('D46IPCS.DRVC400.SA00001') POINTERS</th>
<th>SCROLL ====&gt; CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PTR Address</strong></td>
<td><strong>Address space</strong></td>
</tr>
<tr>
<td>00001 00000000 ASID(X'0003')</td>
<td>AREA</td>
</tr>
<tr>
<td>Remarks:</td>
<td></td>
</tr>
<tr>
<td>00002 000006B0 ASID(X'0003')</td>
<td>AREA</td>
</tr>
<tr>
<td>Remarks:</td>
<td></td>
</tr>
<tr>
<td>00003 00FD7420 ASID(X'0001')</td>
<td>STRUCTURE(Cvt)</td>
</tr>
<tr>
<td>Remarks: Communications Vector Table</td>
<td></td>
</tr>
<tr>
<td>00004 0120E298 ASID(X'0058')</td>
<td>MODULE(IEANUC01)</td>
</tr>
<tr>
<td>Remarks:</td>
<td></td>
</tr>
</tbody>
</table>

END OF POINTER STACK

---

**Example 4**

Determine where an absolute address is located.

- **Action**
  
  COMMAND ====> where cda800.
  
- **Result**
  
  WHERE generates the following output line, showing that the specified address is a TCB in the PRIVATE area.

CDA800. STRUCTURE(TCB)-10 IN PRIVATE

**Example 5**

Determine the name of a module in storage.

- **Action**
  
  Given an address, enter a WHERE subcommand specifying the address.
  
  COMMAND ====> where 04a8001a
  
- **Result**
  
  IPCS identifies the address space containing the module, the module name (if the name conforms to IPCS naming conventions), the offset of the address into the module, and the storage area containing the module.

ASID(X'0179') 04A8001A. IGC006A+1A IN EXTENDED PLPA

**Example 6**

Determine the name of a module in storage when the module name does not conform to IPCS naming conventions.

- **Action**
  
  Given an address, enter a WHERE subcommand specifying the address.
  
  COMMAND ====> where 04ab001a
  
- **Result**
  
  IPCS provides the same information shown in the previous example, but instead of the name of the module, IPCS displays "SPECIALNAME". IPCS also expands the name in hexadecimal characters, and shows the module name as an eye-catcher in the output.

ASID(X'0179') 04AB001A. SPECIALNAME+A01A IN EXTENDED PLPA
WLMDATA subcommand — analyze workload manager data

Use the WLMDATA subcommand to generate reports about the workload manager (WLM) component of MVS.

Syntax

WLMDATA

-------- Report Type Parameters -----------------------------
[ POLICY ]
[ STATUS[,SYSNAME(sysname)]]
[ WORKMANAGER[,ASID(asidlist)]
  [,SUBSYSTYPE(subsstytype)]
  [,SUBSYSNAME(subsysname)]

-------- Data Selection Parameters --------------------------
[ DETAIL ]
[ EXCEPTION ]
[ SUMMARY ]

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters. See “SETDEF subcommand — set defaults” on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters

Report Type Parameters
Use these parameters to select the type of report. You can specify as many reports as you want. If you omit a report type parameter, the default is POLICY, STATUS, and WORKMANAGER.

POLICY
Requests information about the sysplex service policy.

STATUS
Requests information about WLM status for one or more systems. The parameter that can limit the scope of the STATUS report is:

SYSNAME(sysname)
Requests status information about WLM for a list of system names. If you omit the SYSNAME parameter and value, the default is status information for all systems in the sysplex.
WLMDATA subcommand

The sysname can be a single system name or a list of system names. When you specify a list, separate the names with commas. A system name has 1 to 8 characters.

WORKMANAGER
Requests information about the activity associated with work requests that are connected to WLM. The parameters that can limit the scope of the WORKMANAGER report are:

ASID(asidlist)
Specifies a list of ASIDs for the address spaces to be in the WORKMANAGER report. If you omit the ASID parameter, the default is information for all address spaces.

The asidlist can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID has 1 to 4 hexadecimal digits.

SUBSYSTYPE(subsystype)
Specifies a list of subsystem types to be in the WORKMANAGER report. If you omit the SUBSYSTYPE parameter, the default is information for all subsystem types.

The subsystype can be a single subsystem or a list of subsystems. When you specify a list, separate the list members with commas.

The subsystype has 1 to 4 characters.

SUBSYSNAME(subsysname)
Requests status information about WLM for a list of subsystem names. If you omit the SUBSYSNAME parameter and value, the default is status information for all subsystems in the sysplex.

The subsysname can be a single subsystem name or a list of subsystem names. When you specify a list, separate the names with commas. A subsystem name has 1 to 8 characters.

Data Selection Parameters
Use these parameters to limit the scope of the data in the report. If you omit a data selection parameter, the default is SUMMARY.

DETAIL
Requests a report showing detailed information for each of the selected topics.

EXCEPTION
Requests a list of exceptional or unusual conditions for each of the selected topics.

SUMMARY
Requests summary information for each of the selected topics. SUMMARY is the default.

Return Codes
See “Standard subcommand return codes” on page 5-2 for a description of the return codes produced by the WLMDATA subcommand.
XESDATA subcommand — format cross system extended services data

Use the XESDATA subcommand to request formatting of information related to cross system extended services. The information is available in three levels:

- The summary and detail levels provide diagnostic, configuration, and resource information about a particular area of cross system extended services.
- The exception level provides an automated way of detecting incorrect data areas and unusual system conditions that may be helpful in problem determination.

When an error is detected in control block chains or data content, the report contains information for the IBM Support Center.

**Note:** If the dump is not caused by an error in the cross system extended services component, the system issues the following message:

```
IXL0200I XESDATA XESSTACK report cannot be run with the current dump.
Reason: The dump did not result from an XES module failure.
```

In this case, if you know the address of the stack, and if the storage is in the dump, enter a CBFORMAT STRUCTURE(XESSTACK) subcommand.

Data selection and report type parameters limit the scope and extent of the information that appears in a report.

**Syntax**

```
XESDATA
   { DETAIL }
   { EXCEPTION }
   { SUMMARY }

-------- Report Type Parameters --------
   [ CACHE ]
   [ CONNECTION ]
   [ FACILITY ]
   [ LIST ]
   [ LOCK ]
   [ LOCKMGR ]
   [ LOCKRESOURCE ]
   [ XESSTACK ]
```
XESDATA subcommand

------- Additional Data Selection Parameters -------
[ ASID(asidlist) ]
[ CFNAME(cfname) ]
[ CONNAME(conname) ]
[ HASHVALUE(hashvalue) ]
[ JOBNAME(jobname) ]
[ LISTNUM(listnum) ]
[ LOCKMGRCONID(conid) ]
[ LTENTRY(ltentry) ]
[ REQID(reqid) ]
[ REQUESTORCONID(conid)]
[ RNAME(rname) ]
[ SOURCENAME(conname) ]
[ STRNAME(strname) ]
[ SYSNAME(sysname) ]
[ TARGETNAME(conname) ]

------- SETDEF-Defined Parameters -------------------
Note: You can override the following SETDEF parameters.
See "SETDEF subcommand — set defaults" on page 5-232
[ ACTIVE | MAIN | STORAGE ]
[ DNAME(dsnname) | DATASET(dsnname) ]
[ FILE(ddname) | DDNAME(ddname) ]
[ PATH(path-name) ]
[ FLAG(severity) ]
[ PRINT | NOPRINT ]
[ TERMINAL | NOTERMINAL ]
[ TEST | NOTEST ]

Parameters
Data Selection Parameters
Use these parameters to select the level of information in the report. If you omit these parameters, the default is SUMMARY.

DETAIL
Requests a report showing detailed information for each of the specified objects or processes.
An example is:
COMMAND ===> XESDATA DETAIL
EXCEPTION
Requests a list of exceptional or unusual conditions for the specified objects or processes.
COMMAND ====> XESDATA EXCEPTION

SUMMARY
Requests summary information for the specified objects or processes.
SUMMARY is the default.
COMMAND ====> XESDATA SUMMARY

Report Type Parameters
Use these parameters to select the type of report. If you specify more than one report type parameter, IPCS produces a report for each parameter. If you omit a report type parameter, the default is all report types.

CACHE
Requests information about outstanding cache requests for this system. Information is included for both the request as a whole, and operation-level information for the operation to each of the duplexed structure instances. The report output is:
XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
CACHE SUMMARY REPORT

The output fields for each connection are:
– Number of requests
– Requests passing filters
An example is:
COMMAND ====> XESDATA CACHE

CONNECTION
Requests information about connectors to structures in the coupling facility. The report output is:
XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
CONNECTION SUMMARY REPORT

The output fields for each connection specified are:
– Structure type
– Structure name
– Connect token
– Connect name
– Recovery status
– Diagnostic data
– Status of a pending response for an event that was delivered to the event exit.
– An indication of the user-managed or system-managed state of a rebuild process (both rebuild and duplexing rebuild).
Note that for duplexed structure instances, the report information will be split into sections that deal with the duplexed structure as a whole, and that deal with each of the allocated structure instances.
An example is:
COMMAND ====> XESDATA CONNECTION

FACILITY
Requests information about the coupling facilities and coupling facility structures known to the system. The report output is:
XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
FACILITY SUMMARY REPORT

XESDATA subcommand

Chapter 5. IPCS subcommands
The output fields for each coupling facility are:
- Name
- Node descriptor
- Facility ID
- Control unit
- Authority
- Total space
- Max structure ID
- Connected indicator
- Policy indicator
- Pathing information
  - Paths valid
  - Paths online
  - Paths miscabled
  - Paths not operational
- Remotely-connected coupling facilities, identified by their remote CF node descriptor (ND) and system identifier (SYID)

The output fields for each structure are:
- Name
- Facility
- Structure ID
- Type
- Structure version
- Relationship between duplexed structure instances

An example is:
COMMAND ====> XESDATA FACILITY

LIST
Requests information about outstanding list requests for this system.
Information is included for both the request as a whole, and operation-level information for the operation to each of the duplexed structure instances.
The report output is:
  XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
  LIST SUMMARY REPORT

The output fields for each connection are:
- Number of list headers
- Number of lock entries
- For each outstanding lock request in the serialized list:
  - Lock entry number
  - Lock ownership status
  - Lock data, if applicable
  - Queued request count
- Requests passing filters
- Number of requests

An example is:
COMMAND ====> XESDATA LIST

LOCK
Requests information about outstanding asynchronous coupling facility lock requests. Both simplex and duplex request data is included in the status information.

An example is:
COMMAND ====> XESDATA LOCK
LOCKMGR
Requests information about lock resources managed globally by the system.
The report output is:
   XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
   LOCKMGR SUMMARY REPORT

The output fields for each globally managed resource for each connection
are:
- Lock structure entry number
- Resource name
- Hash value
- Diagnostic data
- Indication of whether there is an outstanding asynchronous coupling
  facility request

An example is:
COMMAND ===> XESDATA LOCKMGR

LOCKRESOURCE
Requests information about the lock resources owned or requested by the
system. The report output is:
   XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
   LOCKRESOURCE SUMMARY REPORT

The output fields for each resource are:
- Number of lock entries
- Lock structure entry number
- Connector this entry is managed by
- Number of exclusive holders
- Number of shared holders
- Resource name
- Hash value
- Requested status
- Requested event
- Diagnostic data
- Indication of whether there is an outstanding asynchronous coupling
  facility request

An example is:
COMMAND ===> XESDATA LOCKRESOURCE

XESSTACK
Requests information about cross system extended services execution flow.
This report contains diagnostic data for the IBM Support Center. The report
output is:
   XESDATA (CROSS-SYSTEM EXTENDED SERVICES) REPORT
   XESSTACK SUMMARY REPORT

The output fields contain diagnostic data.

An example is:
COMMAND ===> XESDATA XESSTACK

Additional Data Selection Parameters
The table shows the additional data selection parameters that apply to each
report type.
XESDATA subcommand

<table>
<thead>
<tr>
<th>Data Selection Parameter</th>
<th>REPORT TYPES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CACHE</td>
</tr>
<tr>
<td>ASID</td>
<td>X</td>
</tr>
<tr>
<td>CFNAME</td>
<td>X</td>
</tr>
<tr>
<td>CONNAME</td>
<td>X</td>
</tr>
<tr>
<td>HASHVALUE</td>
<td></td>
</tr>
<tr>
<td>JOBNAME</td>
<td>X</td>
</tr>
<tr>
<td>LISTNUM</td>
<td></td>
</tr>
<tr>
<td>LOCKMGRCONID</td>
<td></td>
</tr>
<tr>
<td>LTENTRY</td>
<td></td>
</tr>
<tr>
<td>REQID</td>
<td></td>
</tr>
<tr>
<td>REQUESTORCONID</td>
<td></td>
</tr>
<tr>
<td>RNAME</td>
<td></td>
</tr>
<tr>
<td>SOURCENAME</td>
<td></td>
</tr>
<tr>
<td>STRNAME</td>
<td>X</td>
</tr>
<tr>
<td>SYSNAME</td>
<td></td>
</tr>
<tr>
<td>TARGETNAME</td>
<td></td>
</tr>
</tbody>
</table>

**ASID(asidlist)**

Requests that only information about the address spaces for the listed ASIDs be included in the report.

The *asidlist* can be a single ASID, a range of ASIDs, or a list of noncontiguous ASIDs. When you specify a range, separate the first and last ASIDs in the range with a colon. When you specify a list, separate the list members with commas.

The ASID can be 1 through 65535. An ASID can be expressed in the notation X’nnn’ or nnn for a decimal number.

An example is:

```
COMMAND ===> XESDATA ASID(X'001A') LIST DETAIL
```

**CFNAME(cfname)**

Requests that only information about the specified coupling facility be included in the report.

The *cfname* can be a single coupling facility name or a list of names. Use commas to separate names in the list; do not enclose the names in apostrophes; and do not specify a range of names. To designate coupling facility names that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:

```
COMMAND ===> XESDATA CFNAME(TESTCF)
```

**CONNAME(conname)**

Requests that only information about the connectors with the listed connector names be included in the report.

The *conname* can be a single connector name or a list of names. Use commas to separate names in the list; do not enclose the names in apostrophes; and do not specify a range of names. To designate connector
names that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:
COMMAND ==> XESDATA CONNAME(MYCONNAME1) LIST DETAIL

**HASHVALUE(hashvalue)**
Requests that only information about the listed hash values be included in the report. The hash value is derived from the resource name on the IXLLOCK macro and is used to determine what entry in the lock table is used. If you do not know the hash value, specify the resource name in the RNAME parameter.

The `hashvalue` can be a single value, a range of values, or a list of noncontiguous values. When you specify a range, separate the first and last values in the range with a colon. When you specify a list, separate the list members with commas.

An example is:
COMMAND ==> XESDATA CONNECTION HASHVALUE(00000001) DETAIL

**JOBNAME(joblist)**
Requests that only information about the address spaces associated with the listed job names be included in the report.

The `joblist` can be a single job name or a list of job names. Use commas to separate job names in the list; do not enclose job names in apostrophes; and do not specify a range of job names. To designate job names that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:
COMMAND ==> XESDATA JOBNAME(MAINASID) LIST DETAIL

**LISTNUM(listnum)**
Requests that only information about requests affecting the specified list header number or its entry be included in the report.

The `listnum` can be a single list header number or a list of numbers. Use commas to separate numbers in the list; do not enclose the numbers in apostrophes; and do not specify a range of numbers.

An example is:
COMMAND ==> XESDATA LIST LISTNUM(1) DETAIL

**LOCKMGRCONID(conid)**
Requests that only information about resources managed by the specified connection identifier be included in the report.

The `conid` can be a single connection identifier or a list of identifiers. Use commas to separate identifiers in the list; do not enclose the identifiers in apostrophes; and do not specify a range of identifiers.

An example is:
COMMAND ==> XESDATA LOCKRESOURCE LOCKMGRCONID(01)

**LTENTRY(ltentry)**
Requests that only information about the listed lock table entries be included in the report.

The `ltentry` can be a single entry or a list of entries. When you specify a list, separate the entries with commas.
XESDATA subcommand

An example is:
COMMAND ===> XESDATA LOCKMGR LTENTRY(20)

REPID(reqid)
Requests that only information about requests with the specified identifier be included in the report.

The reqid can be a single request identifier or a list of identifiers. Use commas to separate identifiers in the list; do not enclose the identifiers in apostrophes; and do not specify a range of identifiers. The identifiers can be expressed in the notation X'nnn' or nnn for decimal. To designate request identifiers that begin with the same numbers, use an asterisk (*) as a suffix. The asterisk denotes zero or more numbers, up to the maximum length of the string.

An example is:
COMMAND ===> XESDATA LIST REPID(01)

REQUESTORCONID(conid)
Requests that only information about resources requested by the specified connection identifier be included in the report.

The conid can be a single connection identifier or a list of identifiers. Use commas to separate identifiers in the list; do not enclose the identifiers in apostrophes; and do not specify a range of identifiers.

An example is:
COMMAND ===> XESDATA LIST REQUESTORCONID(01)

RNAME(rname)
Requests that only information about the resources with the listed resource names be included in the report.

The rname can be a single resource or a list of resources. When you specify a list, separate the list members with commas.

The resource can be expressed in the notation X'nnn' or nnn for a decimal number or characters. To designate resource names that begin with the same characters or numbers, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters or numbers, up to the maximum length of the string.

An example is:
COMMAND ===> XESDATA RNAME(MYLIST01) LIST DETAIL

SOURCENAME(conname)
Requests that only information about the connectors with the listed connector names from which signals are received be included in the report.

The conname can be a single connector name or a list of names. Use commas to separate names in the list; do not enclose the names in apostrophes; and do not specify a range of names. To designate connector names that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:
COMMAND ===> XESDATA CONNECTION SOURCENAME(MYCONNAME1)

STRNAME(strname)
Requests that only information about the specified coupling facility structure be included in the report.
XESDATA subcommand

The *strname* can be a single coupling facility structure or a list of structures. Use commas to separate structures in the list; do not enclose the structures in apostrophes; and do not specify a range of structures. To designate structures that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:
```
COMMAND ==> XESDATA STRNAME(LIST01)
```

SYSNAME(sysname)
Requests that only information about the specified system be included in the report.

The *sysname* can be a single system name or a list of names. Use commas to separate names in the list; do not enclose the names in apostrophes; and do not specify a range of names. To designate system names that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:
```
COMMAND ==> XESDATA CONNECTION SYSNAME(D13ID04)
```

TARGETNAME(conname)
Requests that only information about the connectors with the listed connector names to which signals are sent be included in the report.

The *conname* can be a single connector name or a list of names. Use commas to separate names in the list; do not enclose the names in apostrophes; and do not specify a range of names. To designate connector names that begin with the same characters, use an asterisk (*) as a suffix. The asterisk denotes zero or more characters, up to the maximum length of the string.

An example is:
```
COMMAND ==> XESDATA CONNECTION TARGETNAME(MYCONNAME1)
```

Example
For an example of XESDATA output, see the XES component in z/OS MVS Diagnosis: Reference.
XESDATA subcommand
Chapter 6. IPCS dialog controls

This topic describes the IPCS dialog controls. Use these controls in the IPCS full-screen problem analysis dialog (called the IPCS dialog in this information), except in TUTORIAL. The controls are:

- IPCS primary commands
- Line commands
- Program function (PF) keys
- Command codes
- Selection codes
- ISPF primary commands

z/OS MVS IPCS User’s Guide shows and describes the IPCS dialog and tells how to access and modify the dialog.

Using dialog controls

Using IPCS primary commands

Enter a primary command by typing it on the command/option line, which is the second line of a display panel, or by pressing a PF key that is defined for the specific command. When entering more than one parameter for a command, use either a blank or a comma as a separator. When entering more than one command, use a semicolon to separate the commands.

The primary commands are:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>Display ISO-8 ASCII characters</td>
</tr>
<tr>
<td>CANCEL</td>
<td>End the BROWSE option</td>
</tr>
<tr>
<td>CBFORMAT</td>
<td>Format a control block</td>
</tr>
<tr>
<td>DOWN</td>
<td>Scroll data forward</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>Display EBCDIC characters</td>
</tr>
<tr>
<td>END</td>
<td>End a subcommand or panel</td>
</tr>
<tr>
<td>EQUATE</td>
<td>Create a user_defined symbol</td>
</tr>
<tr>
<td>FIND</td>
<td>Search for a specified value</td>
</tr>
<tr>
<td>IPCS</td>
<td>Invoke an IPCS subcommand, CLIST, or REXX exec</td>
</tr>
<tr>
<td>LEFT</td>
<td>Scroll data left</td>
</tr>
<tr>
<td>LOCATE</td>
<td>Scroll the display to show specific data</td>
</tr>
<tr>
<td>MORE</td>
<td>Scroll data</td>
</tr>
<tr>
<td>OPCODE</td>
<td>Display mnemonic operation code</td>
</tr>
<tr>
<td>RENUM</td>
<td>Renumber symbol entries</td>
</tr>
<tr>
<td>RESET</td>
<td>Remove pending commands</td>
</tr>
<tr>
<td>RETURN</td>
<td>Display the IPCS Primary Option Menu</td>
</tr>
<tr>
<td>RFIND</td>
<td>Repeat the FIND command</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Scroll data right</td>
</tr>
</tbody>
</table>
SELECT
Select a pointer to display storage

SORT
Sort an IPCS-generated report

STACK
Create an IPCS-defined symbol

UP
Scroll data backward

WHERE
Identify an area at a given address

Using line commands
Enter a line command by typing the command at the beginning of a line. Enter
the first character of the command in the first column, which is blank in a
report. The second through the sixth characters of a line command, if needed,
must be typed over the next 5 columns of report text shown on the line.
Because characters in the command may match characters of report text,
exercise care to ensure that IPCS recognizes the line commands.

When entering line commands, do one of the following:
– End the line command with a delimiter character (either a blank or a special
caption) that was not displayed in the report column following the line
command.
– Type the line command and press the ENTER key, leaving the cursor under
the character following the line command.

The line commands are:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Delete screen output</td>
</tr>
<tr>
<td>E</td>
<td>Edit a pointer</td>
</tr>
<tr>
<td>F</td>
<td>Format a defined control block</td>
</tr>
<tr>
<td>I</td>
<td>Insert a pointer</td>
</tr>
<tr>
<td>R</td>
<td>Repeat a pointer</td>
</tr>
<tr>
<td>S</td>
<td>Select a pointer to display storage</td>
</tr>
<tr>
<td>S, F, or L</td>
<td>Show excluded screen output</td>
</tr>
<tr>
<td>X</td>
<td>Exclude screen output</td>
</tr>
</tbody>
</table>

Using the PF keys
Certain primary commands can be invoked through the PF keys. The PF keys
are listed in the following task tables. Note that these PF key definitions can be
modified.

Using ISPF primary commands
You can use ISPF primary commands, such as CURSOR, HELP, SPLIT, and
SWAP. See the [z/OS ISPF Dialog Tag Language Guide and Reference] for these
commands.

Commands, PF keys, and codes for panels

Through interactive panels, the IPCS dialog helps you to analyze, display, and
manage data from the source. From each panel, there is a certain set of analysis
tasks you may perform.

The following tables group together the tasks you can perform from each type of
panel. The IPCS dialog uses the following types of panels:

- Selection and data entry panels” on page 6-3
- Pointer and storage panels” on page 6-3
Selection and data entry panels

Table 6-1 summarizes the IPCS primary commands, ISPF primary commands, and PF keys that can be used on the selection and data entry panels.

- On a selection panel, select from a list of options by entering its number on the command/option line.
- On a data entry panel, supply parameters by filling in labeled fields. Many fields retain previous values.

Table 6-1. Selection and Data Entry Panels - Commands and PF Keys

<table>
<thead>
<tr>
<th>When You Want to:</th>
<th>Enter ==&gt;</th>
<th>Use PF Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get help</td>
<td>HELP command (ISPF)</td>
<td>1 or 13</td>
</tr>
<tr>
<td>Split the screen</td>
<td>SPLIT command (ISPF)</td>
<td>2 or 14</td>
</tr>
<tr>
<td>End or cancel</td>
<td>END primary command (IPCS)</td>
<td>3 or 15</td>
</tr>
<tr>
<td>Return to IPCS Primary Option Menu</td>
<td>RETURN primary command (IPCS)</td>
<td>4 or 16</td>
</tr>
<tr>
<td>Swap screens</td>
<td>SWAP command (IPCS)</td>
<td>9 or 21</td>
</tr>
<tr>
<td>Move the cursor to the command/option line</td>
<td>CURSOR command (ISPF)</td>
<td>12 or 24</td>
</tr>
<tr>
<td>Invoke an IPCS subcommand, CLIST, or REXX exec</td>
<td>IPCS primary command (IPCS)</td>
<td>—</td>
</tr>
</tbody>
</table>

Pointer and storage panels

Table 6-2 summarizes the IPCS primary commands, IPCS line commands, ISPF primary commands, and PF keys that can be used on the pointer panels and the storage panels.

Table 6-2. Pointer and Storage Panels - Commands and PF Keys

<table>
<thead>
<tr>
<th>When You Want to:</th>
<th>Enter ==&gt;</th>
<th>Use PF Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get help</td>
<td>HELP command (ISPF)</td>
<td>1 or 13</td>
</tr>
<tr>
<td>Reset entered commands</td>
<td>RESET primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Split the screen</td>
<td>SPLIT command (ISPF)</td>
<td>2 or 14</td>
</tr>
<tr>
<td>End processing</td>
<td>END primary command (IPCS)</td>
<td>3 or 15</td>
</tr>
<tr>
<td>Cancel processing</td>
<td>CANCEL primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Return to IPCS primary option menu</td>
<td>RETURN primary command (IPCS)</td>
<td>4 or 16</td>
</tr>
<tr>
<td>Search for a value</td>
<td>FIND primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Repeat the FIND command</td>
<td>RFIND primary command (IPCS)</td>
<td>5 or 17</td>
</tr>
</tbody>
</table>
### Table 6-2. Pointer and Storage Panels - Commands and PF Keys (continued)

<table>
<thead>
<tr>
<th>When You Want to:</th>
<th>Enter ====&gt;</th>
<th>Use PF Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use <strong>Symbols</strong> to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Create an IPCS defined symbol</td>
<td>STACK primary command (IPCS)</td>
<td>6 or 18</td>
</tr>
<tr>
<td>• Create a user-defined symbol</td>
<td>EQUATE primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>• Renumber stack entries</td>
<td>RENUM primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td><strong>Scroll</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Up (toward top)</td>
<td>UP primary command (IPCS)</td>
<td>7 or 19</td>
</tr>
<tr>
<td>• Down (toward bottom)</td>
<td>DOWN primary command (IPCS)</td>
<td>8 or 20</td>
</tr>
<tr>
<td><strong>Swap</strong> screens.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Display</strong> a pointer or storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Browse</strong> through a dump by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>positioning the cursor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>To a 24-bit address; the pointer is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recorded on the pointer panel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>To a 31-bit address; the pointer is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recorded on the pointer panel</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Move the cursor</strong> to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>command/option line</td>
<td>CURSOR command (ISPF)</td>
<td>12 or 24</td>
</tr>
<tr>
<td><strong>Format</strong> a control block</td>
<td>CBFORMAT primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td><strong>Identify</strong> areas of storage that</td>
<td>WHERE primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>contain an address</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Invoke</strong> an IPCS subcommand,</td>
<td>IP CS primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>CLIST, or REXX exec</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Select</strong> a pointer and display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage addressed by that selected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Delete</strong> pointers</td>
<td>D (delete) line command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td><strong>Edit</strong> a selected pointer</td>
<td>E (edit) line command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td><strong>Format</strong> a pointer with a data-type</td>
<td>F (format) line command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>of STRUCTURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Insert</strong> pointers</td>
<td>I (insert) line command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td><strong>Replicate</strong> existing pointers</td>
<td>R (repeat) line command (IPCS)</td>
<td>—</td>
</tr>
</tbody>
</table>

### Dump display reporter panels

Table 6-3 summarizes the IPCS primary commands, ISPF primary commands, and PF keys that can be used on the dump display reporter panels.

### Table 6-3. Dump display reporter panel - commands and PF keys

<table>
<thead>
<tr>
<th>When You Want to:</th>
<th>Enter ====&gt;</th>
<th>Use PF Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get help</td>
<td>HELP command (ISPF)</td>
<td>1 or 13</td>
</tr>
<tr>
<td>Reset entered commands</td>
<td>RESET primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Split the screen</td>
<td>SPLIT command (ISPF)</td>
<td>2 or 14</td>
</tr>
<tr>
<td>End processing</td>
<td>END primary command (IPCS)</td>
<td>3 or 15</td>
</tr>
</tbody>
</table>
Table 6-3. Dump display reporter panel - commands and PF keys (continued)

<table>
<thead>
<tr>
<th>When You Want to:</th>
<th>Enter ==&gt;</th>
<th>Use PF Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return to IPCS primary option menu</td>
<td>RETURN primary command (IPCS)</td>
<td>4 or 16</td>
</tr>
<tr>
<td>Search for a value</td>
<td>FIND primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Search through the IPCS output stream for text</td>
<td>EXCLUDE primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Repeat the FIND command</td>
<td>RFIND primary command (IPCS)</td>
<td>5 or 17</td>
</tr>
<tr>
<td>Scroll</td>
<td></td>
<td></td>
</tr>
<tr>
<td>To next full screen</td>
<td>MORE primary command (IPCS)</td>
<td>6 or 18</td>
</tr>
<tr>
<td>Up (toward top)</td>
<td>UP primary command (IPCS)</td>
<td>7 or 19</td>
</tr>
<tr>
<td>Down (toward bottom)</td>
<td>DOWN primary command (IPCS)</td>
<td>8 or 20</td>
</tr>
<tr>
<td>Left</td>
<td>LEFT primary command (IPCS)</td>
<td>10 or 22</td>
</tr>
<tr>
<td>Right</td>
<td>RIGHT primary command (IPCS)</td>
<td>11 or 23</td>
</tr>
<tr>
<td>To specific data</td>
<td>LOCATE primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Swap screens</td>
<td>SWAP command (ISPF)</td>
<td>9 or 21</td>
</tr>
<tr>
<td>Move the cursor to the command/option line</td>
<td>CURSOR command (ISPF)</td>
<td>12 or 24</td>
</tr>
<tr>
<td>Format a control block</td>
<td>CBFORMAT primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Identify areas of storage that contain an address</td>
<td>WHERE primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Invoke an IPCS subcommand, CLIST, or REXX exec</td>
<td>IPCS primary command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Delete screen lines permanently</td>
<td>D (delete) line command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Exclude screen lines</td>
<td>X (exclude) line command (IPCS)</td>
<td>—</td>
</tr>
<tr>
<td>Display excluded screen lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For excluded lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For the first line of excluded text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For the last line of excluded text</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPCS inventory panel

The following 2-character command codes are used to manage the inventory panel.

<table>
<thead>
<tr>
<th>Code</th>
<th>Function performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>Browse storage. This activates the BROWSE option of the IPCS dialog. You immediately see the BROWSE option pointer panel, without having to go through the BROWSE option entry panel first.</td>
</tr>
<tr>
<td>CL</td>
<td>Close the source. Resources that were obtained by dump OPEN processing are immediately released.</td>
</tr>
<tr>
<td>DD</td>
<td>Delete the source description of the indicated source from the dump directory.</td>
</tr>
<tr>
<td>DT</td>
<td>Delete translation records from the source description in the dump directory.</td>
</tr>
<tr>
<td>Code</td>
<td>Function performed</td>
</tr>
<tr>
<td>------</td>
<td>--------------------</td>
</tr>
<tr>
<td>LA</td>
<td>List the source description, with storage attributes.</td>
</tr>
<tr>
<td>LB</td>
<td>List the source description, with record locations.</td>
</tr>
<tr>
<td>LD</td>
<td>List the source description, with dumped storage summary.</td>
</tr>
<tr>
<td>LT</td>
<td>List the source description, with translation results.</td>
</tr>
<tr>
<td>LZ</td>
<td>List the source description, with all the information from the other LIST options.</td>
</tr>
<tr>
<td>OP</td>
<td>OPEN the source for processing.</td>
</tr>
<tr>
<td>SD</td>
<td>Establish a data set as the default source.</td>
</tr>
<tr>
<td>XP</td>
<td>Export dump description to RECFM = VB data set (COPYDDIR subcommand with EXPORT option)</td>
</tr>
</tbody>
</table>

**Storage panel**

The following selection codes request IPCS to:
- Interpret the word as an address in the current address space
- Place a pointer for the word in the pointer stack on the pointer panel

For use of the selection codes, see [z/OS MVS IPCS User’s Guide](#).

<table>
<thead>
<tr>
<th>Selection code</th>
<th>Actions by IPCS</th>
</tr>
</thead>
</table>
| L              | - Interpret the word as a low-precision (24-bit) address of storage in the current address space.  
- Place a pointer in the pointer stack on the pointer panel. |
| H              | - Interpret the word as a high-precision (31-bit) address of storage in the current address space.  
- Place a pointer in the pointer stack on the pointer panel. |
| %              | - Interpret the word as a low-precision (24-bit) address of storage in the current address space.  
- Place a pointer in the pointer stack on the pointer panel.  
- Display the addressed storage.  
- If more than one % is entered, use the first one (topmost and leftmost in the display) for the origin of the next display, and treat the rest as though an L had been entered. |
| ?              | - Interpret the word as a high-precision (31-bit) address of storage in the current address space.  
- Place a pointer in the pointer stack on the pointer panel.  
- Display the addressed storage.  
- If more than one ? is entered, use the first one (topmost and leftmost in the display) for the origin of the next display, and treat the rest as though an H had been entered. |

**Note:** If an incorrect selection code is entered, IPCS intensifies the error field and displays a message in the upper-right corner of the display panel.
IPCS dialog primary commands

ASCII primary command — display characters as ASCII

Use the ASCII primary command to cause the BROWSE option to display ISO-8 ASCII characters in its hexadecimal and character displays.

Syntax

ASCII

Usage notes

- ASCII can be used only from the storage panel of the BROWSE option.
- The BROWSE option begins operation displaying EBCDIC characters.
- ASCII persists until the EBCDIC primary command is used or until you exit the BROWSE option.

CANCEL primary command — end the BROWSE option

Use the CANCEL primary command to leave the IPCS BROWSE option panel and return to the previous panel. Data entered on the panel is not saved.

Syntax

{ CANCEL }
{ CAN }

Usage notes

- CANCEL can be used only in the BROWSE option.
- If you want to leave an IPCS dialog panel and save the data entered on the panel, use the END primary command.

CBFORMAT primary command — format a control block

Use the CBFORMAT primary command to format a control block.

Syntax

{ CBFORMAT } data-descr
{ CBF }

data-descr

Specifies the data description parameter, which consists of three parts:
- A symbol
- An address
- Address processing parameters

Chapter 3, “Data description parameter,” on page 3-1 has more information about the use and syntax of the data description parameter.

Note: The data-descr for the CBFORMAT primary command uses only three of the five possible parts of the data description parameter.

Usage notes

- CBFORMAT can be used from the BROWSE option pointer and storage panels, and from the dump display reporter panel.
- Descriptions of the control blocks that are formatted using the CBFORMAT primary command are added to the pointer stack.

Example

Format the CVT.
CBFORMAT primary command

- Action
  COMMAND ===> cbformat fd7bc8. str(cvt)
- Result
  The CVT is formatted and displayed, and its description is added to the pointer stack.

DOWN primary command — scroll data forward

Use the DOWN primary command to scroll forward toward the bottom of data.

Syntax

DOWN [ amount ]

Parameters

amount

Specifies one of the following scroll amounts:
- A number from 1 through 9999, representing the number of lines to be scrolled
- PAGE or P, indicating that a full screen should be scrolled
- HALF or H, indicating that a half-screen should be scrolled
- CSR or C, indicating that the screen should be scrolled to the line on which the cursor resides
- MAX or M, indicating that the screen should be scrolled to the bottom
- DATA or D, indicating that the screen should be scrolled a screen minus one line

If you do not specify an amount, IPCS uses the amount in the SCROLL amount field in the upper right corner of the screen.

Usage notes
- DOWN can be used on all IPCS dialog panels that display the SCROLL amount field.
- The scroll amount is typically displayed on the screen, following the command/option field. You can change the scroll amount by typing over the SCROLL amount field with a new amount. The new scroll amount will remain effective (except MAX or M) until you change it or until you begin a new function.
- You can temporarily override the scroll amount, without changing the SCROLL amount field, by:
  - Typing an amount as part of the scroll command and pressing the ENTER key
  - Typing a scroll amount in the command/option field, and then pressing PF8 or PF20
- The IPCS-defined PF keys 8 and 20 invoke the DOWN primary command.

Example

Scroll using the DATA value.

- Action
  COMMAND ===> down data
  or
  COMMAND ===> down d
- Result
  The screen is scrolled toward the bottom of the data by a screen minus one line.
EBCDIC primary command — display characters as EBCDIC

Use the EBCDIC primary command to cause the BROWSE option to display EBCDIC characters in its hexadecimal and character displays.

Syntax

\[
\text{EBCDIC}
\]

Usage notes
- EBCDIC can be used only from the storage panel of the BROWSE option.
- The BROWSE option begins operation displaying EBCDIC characters.
- EBCDIC persists until the ASCII primary command is used or until you exit the BROWSE option.

END primary command — end a subcommand or panel

Use the END primary command to leave an IPCS dialog panel and return to the previous panel. All data entered on the panel is saved.

Syntax

\[
\text{END}
\]

Usage notes
- END can be used in all IPCS dialog options.
- The IPCS-defined PF keys 3 and 15 invoke the END primary command.

EQUATE primary command — create a user-defined symbol

Use the EQUATE primary command to create a user-defined symbol in the symbol table and to associate an address and address processing parameters with the symbol.

If the specified symbol already exists in the symbol table, the new address and address processing parameters overlay the previous information.

Syntax

\[
\{ \text{EQUATE} \} \ symbol
\{ \text{EQU} \} \\
\{ \text{EQ} \}
\]

\[ \text{[ data-descr | X ]} \]

Parameters

symbol
- Specifies the symbol being defined. When specifying symbol, do not include the ampersand (&) or the period (.) that are normally part of symbolic notation. The symbol is 1 through 31 alphanumeric characters; the first character must be a letter or one of the following characters:
  - $ (X'5B')
  - # (X'7B')
  - @ (X'7C')

data-descr or X
- Specifies the data description parameter, which consists of two parts:
  - An address
  - Address processing parameters

\[ \text{Chapter 3, “Data description parameter,” on page 3-1} \] has more information about the syntax and use of the data description parameter.
EQUATE primary command

If you omit the data description parameter, the default is X, the current address.

Usage notes
- EQUATE can be used only in the BROWSE option.
- There are two special symbols, CURSOR and X, that are accepted in the BROWSE option on the storage panel. These symbols associate a location in a dump and are used in the same manner as other symbols, such as the CVT and TCB symbols.
  - CURSOR indicates the word of storage at which you position the cursor.
    By placing the cursor in the selection field preceding a word of storage or by placing the cursor under a word of storage, you can reference the word of storage. CURSOR is not in effect if the position of the cursor does not identify a word of storage or if you leave the storage panel.
  - X indicates the starting address of the data displayed on the storage panel.
    X remains in effect even if you leave the storage panel.
- To add your user-defined symbol to the address pointer stack on the pointer panel of the BROWSE option, use the STACK primary command.

Example 1
Set X to a specific address.
- Action
  COMMAND ===> equate X 522836
- Result
  X, the current address, becomes X’522836’.

Example 2
Equate a specific address to a user-defined symbol.
- Action
  COMMAND ===> equate failingtcb 51368.
- Result
  A symbol table entry is created for FAILINGTCB and is identified at address X’51368’.

EXCLUDE primary command — exclude lines from display
Use the EXCLUDE primary command to search through visible (not excluded already) IPCS output stream text for a specified value. When that value is found, mark the line(s) containing the value as excluded.

All options of the EXCLUDE primary command are similar to those supported by the FIND primary command — and very similar to the EXCLUDE primary command supported by ISPF EDIT and VIEW. No option is supported to search already excluded lines of a report.

Syntax

```
EXCLUDE { relational-operator }
EX value
X { column { column } }
  { ALL }
  { FIRST }
  { LAST }
  { NEXT }
  { PREVIOUS }
```

Usage notes
When EXCLUDE processing is successful, the following actions take place:
EXCLUDE primary command

- The line immediately preceding the first one excluded is displayed. The "Top of Data" line may be shown if the line was the first in the report. This behavior is similar to that exhibited by the EXCLUDE primary command of ISPF EDIT and VIEW.
- An ISPF "n lines excluded" message will be shown.

FIND primary command — search for a specified value

Use the FIND primary command to search through all dump output for a single occurrence of a specified value.

Syntax

The syntax of the FIND primary command varies depending on whether you are in the BROWSE option or any other option except TUTORIAL.

- Syntax for the BROWSE Option

```
{ FIND } [ relational-operator ]
{ F } value
   [ BOUNDARY(bdy [,index]) ]
   [ BREAK | NOBREAK ]
   [ MASK(mask) | NOMASK ]
   [ FIRST ]
   [ LAST ]
   [ NEXT ]
   [ PREVIOUS ]
```

- Syntax for searching the IPCS output stream

```
{ FIND } [ relational-operator ]
{ F } value
   [ col [ col ]]
   [ ALL ]
   [ FIRST ]
   [ LAST ]
   [ NEXT ]
   [ PREVIOUS ]

   [ X ]
   [ NX ]
```

Parameters

relational-operator

Specifies one of the following symbolic or programming operators to be used with the value operand:

>[LT]<|LE|>=|NE|EF|>=|NL|GT|=<NE]

Note: If a programming relational-operator is entered alone, such as FIND EQ, IPCS interprets EQ not as a search value but as an operator and does not perform a search. Enter the command with a relational-operator and a value. For example, FIND EQ EQ causes IPCS to search for an occurrence of EQ.
FIND primary command

value
Specifies the general value that IPCS is to search for. See "General values" on page 2-3 for more information, the syntax, and examples.

col [col ]
Specifies that FIND is to limit the search to specified columns. When entering a single column number, the value must start in the specified column. When entering a pair of column numbers, indicating the first and last columns to be searched, the string is found if it is completely contained within the designated columns. The column range is 1 through 250. The default is 1.

BOUNDARY(bdy[,index])
Specifies that IPCS is to divide storage into strings bdy bytes in length. The address of each string is divisible by bdy. FIND performs only one comparison with data whose first byte lies within any string. The abbreviation BDY is accepted for this parameter.

The index value designates which byte in the string FIND is to select. The index can be a single value or a range, with the first and last values separated by a colon. For example:

BDY(1) or BDY(1,1) or BDY(1,1:1)
  FIND examines each byte.

BDY(2) or BDY(2,1) or BDY(2,1:1)
  FIND performs comparisons with strings originating at even-numbered addresses.

BDY(2,2) or BDY(2,2:2)
  FIND performs comparisons with strings originating at odd-numbered addresses.

BDY(5,5) or BDY(5,5:5)
  FIND performs comparisons only with strings originating at addresses 4 bytes past an address divisible by 5.

BDY(7,6:7)
  FIND performs comparisons only with strings originating at addresses 5 or 6 bytes past an address divisible by 7.

BDY(8) or BDY(8,1) or BDY(8,1:1)
  FIND performs comparisons only with strings aligned on doubleword boundaries.

Both bdy and index can be 1 through 2 raised to the thirty-first power 2^{31} and can be expressed in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...') notation.

When you specify this option, it remains in effect until you specify a new search argument or you override this option. If you enter a new search argument and omit BDY, the default is BDY(1,1).

BREAK
NOBREAK
BREAK specifies that FIND is to stop processing if it cannot retrieve storage from the dump to continue the search. This happens if the required storage was not acquired through the GETMAIN macro or the required storage is not contained in the dump.
FIND primary command

NOCASE specifies that FIND is to continue processing if it cannot retrieve storage from the dump. FIND continues the search with the next available address in the dump.

When you specify this option, it remains in effect until you specify a new search argument or you override this option. If you enter a new search argument and omit NOCASE, the default is CASE.

MASK(mask)
NOMASK

MASK defines a value that is logically ANDed with both operands before performing the comparison. The mask must be the same size as the data items being compared. The mask is specified using the same value notation used for either operand.

See Chapter 2, “Literal values,” on page 2-1 for more information.

NOMASK suppresses masking.

ALL
FIRST
LAST
NEXT
PREVIOUS

ALL specifies that a search for all occurrences is to be done. A message “n matches found” will display the number of matches found. Enter the HELP primary command immediately to see a longer message showing both the search argument and the number of matches to be shown.

FIRST specifies that a search for the first occurrence of the value is to be done. The search starts at the beginning of the displayed report or address space; the search finishes at the end of the report or address space.

LAST specifies that a search for the last occurrence of the value is to be done. The search starts at the end of the displayed report or address space; the search finishes at the beginning of the report or address space.

NEXT specifies that a search for the next occurrence of the value is to be done. The search starts at the beginning of the line being displayed (if the cursor is on the command/option line), or at the cursor location (if the cursor is within the data display area). The search finishes at the end of the displayed report or address space.

PREVIOUS specifies that a search for the previous occurrence of the value is to be done. The search starts at the end of the line preceding the first line being displayed (if the cursor is on the command/option line), or at the cursor location (if the cursor is within the data display area). The search finishes at the beginning of the displayed report or address space. The abbreviation PREV is accepted for this parameter.

Usage notes
– FIND can be used in all options except TUTORIAL. Note that the syntax varies depending on which option you are using.
– The starting point for the search initiated by the FIND primary command depends on the command parameters that control the direction of the search (FIRST, LAST, NEXT, PREVIOUS) and on the position of the cursor.
– Use the RFIND primary command (PF key 5 or 17) to continue the search for the specified argument.

Example 1
Search for a value in columns 1 through 9.
**FIND primary command**

- **Action**
  
  COMMAND ===> find abc 1 9

- **Result**
  
  FIND searches for the value abc only in columns 1 through 9. When found, the value is intensified.

**Example 2**

Find a search argument repeatedly.

- **Action**
  
  The following screens depict use of the FIND and RFIND primary commands. Figure 6-1 shows the FIND command entered on the COMMAND line to search through the display and find the first occurrence of the search argument “dsp”.

- **Result**
  
  Figure 6-2 on page 6-15 shows the results of the FIND command. IPCS highlights the line that contains the search argument, positions the cursor at the beginning of the search argument, and displays a message in the upper right corner of the display indicating in which line and column the argument was found.

  Figure 6-3 on page 6-15 is a result of pressing PF5 to invoke the RFIND command. This screen displays the next occurrence of the search argument following the position of the cursor. Notice that the display message is changed, reflecting a newly found search argument.

### IPCS OUTPUT STREAM

<table>
<thead>
<tr>
<th>COMMAND ===&gt; f dsp_</th>
<th>SCROLL ===&gt; CSR</th>
</tr>
</thead>
</table>

*********** KEYFIELDS ***********

* * * * K E Y F I E L D S * * * *

**JOBNAME** RASP

SELECTED BY: CURRENT

**ASCB**: 00F09E00

<table>
<thead>
<tr>
<th>FNOP...</th>
<th>00F09C00</th>
<th>ASID...</th>
<th>0003</th>
<th>TRQP...</th>
<th>80F09301</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5CB.....</td>
<td>00F103C8</td>
<td>TSB......</td>
<td>00000000</td>
<td>AFN.....</td>
<td>FFFF</td>
</tr>
<tr>
<td>ASXB.....</td>
<td>00AFD9F00</td>
<td>DSP1......</td>
<td>00</td>
<td>FLG2......</td>
<td>C4</td>
</tr>
<tr>
<td>SRBS.....</td>
<td>0000</td>
<td>LOCX......</td>
<td>00000000</td>
<td>ASSB.....</td>
<td>01AB6D00</td>
</tr>
<tr>
<td>TCB: 00AFE178</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMP......</td>
<td>00000000</td>
<td>PKF......</td>
<td>00</td>
<td>LMP......</td>
<td>FF</td>
</tr>
<tr>
<td>TSFLG.....</td>
<td>00</td>
<td>STAB.....</td>
<td>00AFD9300</td>
<td>NOSP.....</td>
<td>00000000</td>
</tr>
<tr>
<td>JSCB.....</td>
<td>00AFD0D4</td>
<td>BITS......</td>
<td>00000000</td>
<td>DAR......</td>
<td>00</td>
</tr>
<tr>
<td>RTWA.....</td>
<td>00000000</td>
<td>FBYT1....</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task non-dispatchability flags from TCBFLGS4:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top RB is in a wait</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PRB**: 00AFD060

| WLIC..... | 00020001 | FLCE.... | 00C12630 | OPSW..... | 070C1000 | 810D7C20 |

*Figure 6-1. Using FIND on the Dump Display Reporter Panel*
FIND primary command

Use the IPCS primary command to invoke an IPCS subcommand, CLIST, or REXX exec from any of the panels of the IPCS dialog. The subcommand, CLIST, or REXX exec is entered exactly as though it was being invoked under IPCS in line mode. If the subcommand, CLIST, or REXX exec sends a report to the terminal, you view the report using the dump display reporter panel.
**IPCS primary command**

**Note:** Do not use the IPCS primary command to invoke a CLIST that contains a combination of a TSO/E CLIST function, such as SYSOUTTRAP, and an authorized TSO/E command, such as LISTD. Such a CLIST should be invoked only in IPCS line or batch mode or in a TSO/E environment.

**Syntax**

```
IPCS { subcommand }
IP { clist }
{ rexx-exec }
```

**Parameters**

- `subcommand` Specifies the IPCS subcommand to be run.
- `clist` Specifies the CLIST to be run.
- `rexx-exec` Specifies the REXX exec to be run.

**Usage notes**

- The IPCS primary command can be used in all options except TUTORIAL.
- There are two special symbols, CURSOR and X, that are accepted in the BROWSE option on the storage panel. These symbols are associated with a location in a dump and are used in the same manner as other symbols, such as the CVT and TCB symbols. These symbols affect how the subcommand, CLIST, or REXX exec processes.
  - CURSOR indicates the word of storage at which you position the cursor. By placing the cursor in the selection field preceding a word of storage or by placing the cursor under a word of storage, you can reference the word of storage. CURSOR is not in effect if the position of the cursor does not identify a word of storage or if you leave the storage panel.
  - X indicates the starting address of the data displayed on the storage panel. X remains in effect even if you leave the storage panel.
- If before entering this command you were processing the overriding dump source (as noted on the entry panel of the Browse option), IPCS will not process that dump source but will instead process the current default dump source.

**Example 1**

Change the SETDEF default parameters.

- Action
  
  COMMAND ====> ipcs setdef print

- Result
  
  While in the BROWSE option, this command invokes the SETDEF subcommand to override the existing message routing default parameters.

**Example 2**

Locate a module and display its storage.

- Action
  
  COMMAND ====> ipcs findmod iefbr14 noverify

- Result
  
  While in the BROWSE option on the storage panel, FINDMOD locates module IEFBR14, modifies X (the current address), and scrolls the storage containing the module into view.
Example 3
Display an array.

– Action
  COMMAND ===> ipcs list x unsigned dim(5)

– Result
  While in the BROWSE option on the storage panel, LIST displays an array of 5 unsigned numbers whose first entry occupies the current address, X. The unsigned operand translates the numbers to decimal and displays the numbers on the dump display reporter panel.

LEFT primary command — scroll data left
Use the LEFT primary command to scroll toward the first, or left-most, column of the data.

Syntax

LEFT [ amount ]

Parameter

amount
  Specifies one of the following scroll amounts:
  – A number from 1 through 9999, representing the number of columns to be scrolled
  – PAGE or P, indicating that a full screen should be scrolled
  – HALF or H, indicating that a half-screen should be scrolled
  – CSR or C, indicating that the screen should be scrolled to the position on which the cursor resides
  – MAX or M, indicating that the screen should be scrolled to the left margin
  – DATA or D, indicating that the screen should be scrolled a page minus one column

If you do not specify an amount, IPCS uses the amount in the SCROLL amount field in the upper right corner of the screen.

Usage notes

– LEFT can be used on all IPCS dialog panels that display the SCROLL amount field.
– The scroll amount is typically displayed on the screen, following the command/option field. You can change the scroll amount by typing over the SCROLL amount field with the new amount. The new scroll amount will remain effective (except MAX or M) until you change it or until you begin a new function.
– You can temporarily override the scroll amount, without changing the SCROLL amount field, by:
  - Typing an amount as part of the scroll command and pressing the ENTER key
  - Typing a scroll amount in the command/option field and then pressing PF10 or PF22
– The IPCS-defined PF keys 10 and 22 invoke the LEFT primary command.

Example
Scroll using the cursor value.

– Action
LEFT primary command

One of the following:
COMMAND ===> left csr
COMMAND ===> left c

- Result
The panel is scrolled to the position of the cursor within the data.

LOCATE primary command — scroll the display to show specific data

Use the LOCATE primary command to:

- Scroll to a particular line in the report while on the dump display reporter panel.
- Locate a particular pointer while in the BROWSE option on the pointer panel.
- View a storage location while in the BROWSE option on the storage panel.

Syntax

```
{ LOCATE } relative-line-number
{ LIST } pointer-number
{ LOC } data-descr
{ L }
```

Parameters

relative-line-number
Indicates which line in the dump display reporter panel should be scrolled to the top of the screen. The relative-line-number is a decimal number.

Use relative-line-number only on a dump display reporter panel.

pointer-number
Causes the indicated pointer to be scrolled to the top of the pointer stack on the pointer panel. The pointer-number is a symbol entry and can be entered without leading zeros.

Use pointer-number only on the pointer panel of the BROWSE option.

data-descr
Specifies the data description parameter, which consists of two parts:
- An address
- Address processing parameters

LOCATE an address can only be used in a BROWSE option storage panel.

Chapter 3, “Data description parameter,” on page 3-1 explains the use and syntax of the data description parameter. However, the following exceptions apply to the LOCATE primary command only:

- There are two special symbols, CURSOR and X, that are accepted in the BROWSE option on the storage panel. These symbols associate a location in a dump and are used in the same manner as other symbols, such as the CVT and TCB symbols.

  - CURSOR indicates the word of storage at which you position the cursor. By placing the cursor in the selection field preceding a word of storage or by placing the cursor under a word of storage, you can reference the word of storage. CURSOR is not in effect if the position of the cursor does not identify a word of storage or if you leave the storage panel.
LOCATE primary command

- X indicates the starting address of the data displayed on the storage panel. X remains in effect even if you leave the storage panel.

While browsing through a dump, use the IPCS-defined PF keys:
- 10 or 22 to invoke the primary command chain, STACK X; LOCATE CURSOR%
  The % selection code indicates a 24-bit address of storage.
- 11 or 23 to invoke the primary command chain, STACK X; LOCATE CURSOR?
  The ? selection code indicates a 31-bit address of storage.

STACK X requests that an entry to the address pointer stack on the pointer panel be added with the address contained in the word of storage indicated by the cursor's current position.

LOCATE CURSOR requests that IPCS locate and display the data found at the address contained in the word of storage indicated by the cursor's current position.

Example 1
Display a specific line number on a dump display reporter panel.
- Action
  COMMAND ===> locate 14
- Result
  After pressing the ENTER key, line 14 is scrolled to the top of the screen.

Example 2
Display a specific pointer on the pointer panel of the BROWSE option.
- Action
  COMMAND ===> locate 33
- Result
  After pressing the ENTER key, IPCS displays pointer 33 in the address pointer stack.

Example 3
Display a literal address on a BROWSE option storage panel.
- Action
  COMMAND ===> locate 0.
- Result
  IPCS displays the literal request for location X'0'.

Example 4
Display a symbolic address on a BROWSE option storage panel.
- Action
  COMMAND ===> list cvt
- Result
  IPCS displays the symbolic request for the storage described by the symbol CVT. Note that:
  - Symbol CVT and numerous other IPCS symbols describe blocks of storage including a prefix, storage preceding the nominal address of the communications vector table. IPCS shows the prefix when such a block is requested.
  - Symbol ASVT and other IPCS symbols describe blocks of storage whose nominal address precedes the first byte of storage occupied by the block. IPCS begins the display at the physical origin of the block.
LOCATE primary command

In all situations involving a symbolic description, IPCS attempts to begin the display at the physical origin of the block described by the symbol.

Example 5
Display a general purpose register on a BROWSE option storage panel.
- Action
  COMMAND ===> locate 1r
- Result
  IPCS displays general purpose register 1.

Example 6
Display an indirect address on a BROWSE option storage panel.
- Action
  COMMAND ===> locate 10.%?
- Result
  IPCS displays the storage accessed by both:
  - The 24-bit pointer at location X'10'
  - The 31-bit pointer addressed by the first pointer

Example 7
Display an indirect address on a BROWSE option storage panel.
- Action
  COMMAND ===> loc cvt+24n%
- Result
  IPCS displays the storage accessed by the 24-bit pointer at decimal offset 8 in the storage described by the symbol CVT.

Example 8
Display a symbolic address and an ASID on a BROWSE option storage panel.
- Action
  COMMAND ===> loc private asid(57)
- Result
  IPCS displays the storage in the private area for address space 57.

MORE primary command — scroll data
Use the MORE primary command to scroll to the next full screen of data or the end of data.

Syntax
MORE

Usage notes
- MORE can be used on all IPCS dialog panels that display the scroll amount field in the upper right corner of the screen.

OPCODE primary command — display operation code
Use the OPCODE primary command to display one of the following mnemonic operation codes:
- An instruction explicitly entered as a search-argument on the OPCODE primary command.
- The operation code of the instruction identified by the cursor position when the cursor is placed over the specific halfword where the instruction of interest originates.
OPCODE primary command

- The operation code beginning in the first halfword shown on the screen when
  the previous means to identify the instruction of interest have not been used.

Syntax

OPCODE [search-argument]

Parameter

search-argument
  The hexadecimal digits representing the instruction of interest. If less digits
  are entered than needed to complete an instruction, trailing zero digits are
  supplied.

Usage notes
  - OPCODE can be entered while viewing the storage panel of the IPCS dialog
    browse option.

RENUM primary command — renumber symbol entries

Use the RENUM primary command to renumber all address pointer entries on the
pointer panel of the BROWSE option in ascending order from 00001 through 99999.
RENUM processing automatically renumbers the address pointer entries in the
symbol table in your user dump directory in ascending order from Z1 through
Z99999.

If there are any unused numbers after renumbering the symbols, RENUM
eliminates these numbers and permits the STACK primary command to add more
entries to the address pointer stack of the pointer panel in the BROWSE option and
to the address pointer stack in the symbol table.

Syntax

{ RENUM }
{ REN }

Usage notes
  - RENUM can be used only in the BROWSE option.

REPORT primary command — process IPCS output streams

Use the REPORT primary command when viewing an IPCS output stream to
initiate processing of report text. REPORT initiates a line mode session similar to
that initiated by the IPCS primary command except that the list of subcommand
accepted differs.

Syntax

VERB OPERANDS
REPORT { subcommand }
RPT { clist }
{ rexx-exec }

Usage notes
  - This session is run with ISPF application ID ISR in effect. This activates any
    personalized program function key definitions and other defaults that you
    have defined during normal use of BROWSE and VIEW services.
  - IPCS adds lines of output to an output stream incrementally, based on the
    last line that you have viewed. When the REPORT primary command is
    used, IPCS makes the current output stream available to it. In the following
REPORT primary command

discussion of the REPORT primary command, the term *entire report* refers to all lines in the output stream at the time the primary command is requested. If you want to have the primary command run against a completed report, you must first use primary command DOWN MAX or its equivalent.

- The following subcommands are available during a REPORT session:
  - BROWSE (alias B) — Use the BROWSE subcommand of REPORT to display some or all lines of a report using ISPF BROWSE. BROWSE processing will be performed with the application ISR command table and program function key definitions in effect.

**Syntax of BROWSE**

```
VERB OPERANDS
BROWSE   [ line-number[:line-number] ]
B        [ relative-report-number | 1 ]
```

------ SETDEF-Defined Parameter --------------

**Note:** You can override the following SETDEF parameter. See the SETDEF Subcommand in [z/OS MVS IPCS Commands](#).

- **line-number[:line-number]**
  This option explicitly specifies the range of lines to be browsed. The default is the entire report being referenced. The end of the range may be overstated to request all lines beginning with the first to be browsed.

  The initial line in a report is always line 1.

- **relative-report-number**
  This operand specifies the report number. Report 0 is reserved for terminal output produced by the REPORT command itself. Report 1, the default, is the report being viewed at the time that the REPORT primary command was entered. Reports nested, if any, under the current ISPF logical screen are numbered from 2 onward.

- CLOSE
- END
- EVALRPT

Use EVALRPT to copy information about one line in a report to a command procedure variable. The intended use for EVALRPT is where the common actions anticipated by IPCS are not appropriate or require embellishment. **Example:** For example, if you combine NOTE with some command procedure logic, a report copied to IPCSPRNT can have one or more IPCSTOC entries added to identify pages where significant data starts.

EVALRPT will be rejected if an attempt is made to invoke it directly using the REPORT primary command.

**Syntax of EVALRPT**

```
VERB OPERANDS
EVALRPT   [ line-number | 1 ]
            [ relative-report-number | 1 ]
```
REPORT primary command

VERB OPERANDS

- CLIST(variable-list)
- DIALOG(variable-list)
- REXX(variable-list)

**line-number**
This operand specifies the line being referenced. Lines are numbered sequentially beginning with 1.

**relative-report-number**
This operand specifies the report number. Report 0 is reserved for terminal output produced by the REPORT command itself. Report 1, the default, is the report being viewed at the time that the REPORT primary command was entered. Reports nested, if any, under the current ISPF logical screen are numbered from 2 onward.

**CLIST(variable-list)**
**DIALOG(variable-list)**
**REXX(variable-list)**
This operand specifies the data to be accessed and used to update command procedure variables.

**Syntax of EVALRPT variable-list**

- LINEMAX(variable-name)
- REPORTMAX(variable-name)
- TEXT(variable-name)
- VISIBILITY(variable-name)

**LINEMAX(variable-name)**
This option returns the number of lines in the referenced report. Partially-viewed reports may not be extended during processing of the REPORT primary command. Only those lines already written are accessible.

**REPORTMAX(variable-name)**
This option returns the number of reports nested under the logical screen when the REPORT primary command was entered.

**TEXT(variable-name)**
This option returns the text of the referenced line in the report.
Note: CLIST(variable-name) is supported but not recommended for processing of a REPORT primary command. Processing free-form text in a CLIST is feasible but requires considerable expertise.

**VISIBILITY(variable-name)**
This option returns VISIBLE or EXCLUDED.

- HELP (alias H)
- IPCSPRNT– Use the IPCSPRNT subcommand of REPORT to copy some or all lines of a report to the IPCS print file. If any lines are longer than the print file line size, they are truncated.

**Syntax of IPCSPRNT**

**VERB OPERANDS**

- IPCSPRNT

  [ line-number[:line-number] ]

  [ relative-report-number | 1 ]
## REPORT primary command

<table>
<thead>
<tr>
<th>VERB</th>
<th>OPERANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIEW</td>
<td>[ line-number[:line-number] ]</td>
</tr>
</tbody>
</table>

### line-number[:line-number]
This option explicitly specifies the range of lines to be browsed. The default is the entire report being referenced. The end of the range may be overstated to request for all lines beginning with the first to be browsed.

The initial line in a report is always line 1.

### relative-report-number
This operand specifies the report number. Report 0 is reserved for terminal output produced by the REPORT command itself. Report 1, the default, is the report being viewed at the time that the REPORT primary command was entered. Reports nested, if any, under the current ISPF logical screen are numbered from 2 onward.

### EXCLUDE(SUMMARIZE)
EXCLUDE(DISPLAY)
EXCLUDE(OMIT)
The EXCLUDE option specifies the treatment of lines within the selected range that have been excluded from display on the screen.

- EXCLUDE(SUMMARIZE), the default, places one line into the print file for each group of excluded lines encountered. The line indicates the number of excluded lines within the selected range of lines that were in exclude.
- EXCLUDE(DISPLAY) prints the excluded lines/
- EXCLUDE(OMIT) prints only those lines visible.

Visible lines within the selected range are always printed as shown.

- ISPEXC
- NOTE (alias N)
- OPEN
- VIEW (alias V) — Use the VIEW subcommand of REPORT to display some or all lines of a report using ISPF VIEW. VIEW processing will be performed with the application ISR command table and program function key definitions in effect. Both visible and excluded lines within the selected range are initially made visible in VIEW.

### Syntax of VIEW

```
  VERB OPERANDS
  VIEW [ line-number[:line-number] ]
  V [ relative-report-number | 1 ]
```

### Note:
You can override the following SETDEF parameter. See the SETDEF Subcommand in [z/OS MVS IPCS Commands](http://www.ibm.com).
REPORT primary command

**line-number[:line-number]**
This option explicitly specifies the range of lines to be browsed. The default is the entire report being referenced. The end of the range may be overstated to request for all lines beginning with the first to be browsed.

The initial line in a report is always line 1.

**relative-report-number**
This operand specifies the report number. Report 0 is reserved for terminal output produced by the REPORT command itself. Report 1, the default, is the report being viewed at the time that the REPORT primary command was entered. Reports nested, if any, under the current ISPF logical screen are numbered from 2 onward.

END, ISPEXEC, and NOTE subcommands act the same way they do in a line mode IPCS session. You should rarely need to enter END.

**EXAMPLE**
- For example, REPORT VIEW will display the entire current report using ISPF VIEW and return to the original context when that viewing has been completed.

### RESET primary command — remove pending commands
Use the RESET primary command to remove all pending primary and line commands. After pressing the ENTER key, you can start to enter commands again.

**Syntax**

```
RESET
```

**Usage notes**
- RESET can be used in all IPCS dialog options (on selected panels) except TUTORIAL.

### RETURN primary command — display the IPCS primary option menu
Use the RETURN primary command to return directly to the IPCS primary option menu, bypassing all intermediate panels.

**Syntax**

```
RETURN
```

**Usage notes**
- RETURN can be used in all IPCS dialog options.
- The IPCS-defined PF keys 4 and 16 invoke the RETURN primary command.

### RFIND primary command — repeat the FIND command
Use the RFIND primary command to repeat a search at the location following the position of the cursor. The search is for a single occurrence of a value that was previously entered with the FIND command.

**Syntax**

```
RFIND
```

**Usage notes**
RFIND primary command

- RFIND can be used in all IPCS dialog options (on selected panels) except TUTORIAL.
- The IPCS-defined PF keys 5 and 17 invoke the RFIND primary command. See the FIND primary command for an example.

RIGHT primary command — scroll data right

Use the RIGHT primary command to scroll toward the last, or right-most, column of the data.

Syntax

```
RIGHT [ amount ]
```

**amount**

Specifies one of the following scroll amounts:
- A number from 1 through 9999, representing the number of columns to be scrolled
- PAGE or P, indicating that a full screen should be scrolled
- HALF or H, indicating that a half-screen should be scrolled
- CSR or C, indicating that the screen should be scrolled to the position on which the cursor resides
- MAX or M, indicating that the screen should be scrolled to the right margin
- DATA or D, indicating that the screen should be scrolled a page minus one column

If you do not specify an amount, IPCS uses the amount in the SCROLL amount field in the upper right corner of the screen.

Usage notes

- RIGHT can be used on all IPCS dialog panels that display the SCROLL amount field.
- The scroll amount is typically displayed on the screen, following the command/option field. You can change the scroll amount by typing over the SCROLL amount field with a new amount. The new scroll amount will remain effective (except MAX or M) until you change it or until you begin a new function.
- You can temporarily override the scroll amount, without changing the SCROLL amount field, by:
  - Typing an amount as part of the scroll command and pressing the ENTER key
  - Typing a scroll amount in the command/option field and then pressing PF11 or PF23
- The IPCS-defined PF keys 11 and 23 invoke the RIGHT primary command.

**Example**

Scroll using a numeric amount.

- Action
  COMMAND ===> right 9
- Result
  The panel is scrolled to the right by nine columns.
SELECT primary command — select a pointer to display storage

Use the SELECT primary command to choose a pointer from the address pointer stack on the pointer panel. IPCS then uses the pointer to display storage that is addressed by that pointer.

Note that you can also use the S (select) line command.

**Syntax**

```
{ SELECT } pointer-number
{ SEL }  
{ S }  
```

**Parameter**

`pointer-number`

Identifies the pointer being selected.

The `pointer-number` is the number of the pointer being selected. Leading zeros can be omitted. The `pointer-number` can be used only on the pointer panel of the BROWSE option.

**Usage notes:**

- SELECT can only be used in the BROWSE option.

**Example**

Select the third pointer from the pointer stack to view the storage location at X'00000210'.

The screen shows the SELECT primary command.

**SORT primary command — sort an IPCS report**

Use the SORT primary command to sort an IPCS report based on columns of interest within the report.

Sorting is done as though the report were produced using ISO-8 ASCII characters. This causes columns of equal-length hexadecimal numbers to sort in numeric sequence since uppercase ISO-8 ASCII letters collate after decimal digits.

**Syntax**
SORT primary command

SORT [col1 [col2] [A | D] [...]]
  [X | NX]

Parameter

col1
Specifications the first column of a group of columns to be used as the sort key. The column number must be entered as a decimal number between 1 and 250. If col1 is entered alone, 250 is used as the final column.

If no groups of columns are specified, the entire report line is used as a sort key.

Up to five groups of non-overlapping columns may be designated. If two or more groups are designated, each group other than the last must include either a col2 designation, an indication of sort order, or both.

col2
Specifies the final column of a group of columns to be used as a sort key. The column number must be entered as a decimal number between col1 and 250.

A  Indicates whether the columns are to be sorted in the default, ascending sequence (A) or in the descending sequence (D). The letters may be entered in either upper or lower case.

D

X  NX  Restricts sort activity to excluded (X) or visible (NX) lines of the report. The default is to sort all lines in the report.

Usage notes:

Example

STACK primary command — create an IPCS-defined symbol

Use the STACK primary command to create, in the next available entry, an IPCS-defined symbol for the address pointer stack. IPCS places the symbol in two locations:

- On the pointer panel of the BROWSE option in ascending order from 00001 through 99999

  and

- In the symbol table in your user dump directory in ascending order from Z1 through Z99999

If symbol entry 99999 or Z99999 is reached, IPCS suspends the stack updates. You should use the RENUM primary command to renumber all entries.

Syntax

STACK  [ data-descr | X ]

Parameters

data-descr or X
  Specifies the data description parameter, which consists of two parts:
  - An address
  - Address processing parameters
Chapter 3, “Data description parameter,” on page 3-1 has more information about the syntax and use of the data description parameter.

If you omit the data description parameter, the default is X, the current address.

Usage notes
- STACK can only be used in the BROWSE option.
- The IPCS-defined PF keys 6 and 18 invoke the STACK primary command.
- There are two special symbols, CURSOR and X, that are accepted in the BROWSE option on the storage panel. These symbols associate a location in a dump and are used in the same manner as other symbols, such as the CVT and TCB symbols.
  - CURSOR indicates the word of storage at which you position the cursor. By placing the cursor in the selection field preceding a word of storage or by placing the cursor under a word of storage, you can reference the word of storage. CURSOR is not in effect if the position of the cursor does not identify a word of storage or if you leave the storage panel.
  - X indicates the starting address of the data displayed on the storage panel. X remains in effect even if you leave the storage panel.

While browsing through a dump, use the IPCS-defined PF keys:
- 10 or 22 to invoke the primary command chain, STACK X; LOCATE CURSOR%
  The % selection code indicates a 24 bit address of storage.
- 11 or 23 to invoke the primary command chain, STACK X; LOCATE CURSOR?
  The ? selection code indicates a 31 bit address of storage.

STACK X requests that an entry to the address pointer stack on the pointer panel be added with the address contained in the word of storage indicated by the cursor’s current position.

LOCATE CURSOR requests that IPCS locate and display the data found at the address contained in the word of storage indicated by the cursor’s current position.

Example
Add an address pointer to the stack.
- Action
  COMMAND ===> stack cvt asid(x'0001')
- Result
  This command adds a pointer entry to the pointer panel. It specifies address space 1 and indicates that this is the communications vector table (CVT) under the remarks column. The processing of this command updates both the pointer panel of the BROWSE option and the symbol table.

UP primary command — scroll data backward
Use the UP primary command to scroll backward toward the top of data.

Syntax

```
UP [ amount ]
```

Parameter

amount
  Specifies one of the following scroll amounts:
UP primary command

- A number from 1 through 9999, representing the number of lines to be scrolled
- PAGE or P, indicating that a full screen should be scrolled
- HALF or H, indicating that a half-screen should be scrolled
- CSR or C, indicating that the screen should be scrolled to the line on which the cursor resides
- MAX or M, indicating that the screen should be scrolled to the top
- DATA or D, indicating that the screen should be scrolled a page minus one line

If you do not specify an amount, IPCS uses the amount in the SCROLL amount field in the upper right corner of the screen.

Usage notes
- UP can be used on all IPCS dialog panels that display the SCROLL amount field.
- The scroll amount is typically displayed on the screen, following the command/option field. You can change the scroll amount by typing over the SCROLL amount field with a new amount. The new scroll amount will remain effective (except MAX or M) until you change it or until you begin a new function.
- You can temporarily override the scroll amount, without changing the SCROLL amount field, by:
  - Typing an amount as part of the scroll command and pressing the ENTER key
  - Typing a scroll amount in the command/option field, and then pressing PF7 or PF19
- The IPCS-defined PF keys 7 and 19 invoke the UP primary command.

Example
Scroll using the MAX operand.

Action:
COMMAND ===> up max
or
COMMAND ===> up m

- Result
The panel is scrolled to the top of the data.

WHERE primary command — identify an area at a given address

Use the WHERE primary command to identify an area at a given address. See the WHERE subcommand for more examples of the primary command.

Syntax

{ WHERE } data-descr
{ W } data-descr

Parameter
data-descr
  Specifies the data description parameter, which consists of two parts:
  - An address
  - Address processing parameters
WHERE primary command

Chapter 3, “Data description parameter,” on page 3-1 has more information about the syntax and use of the data description parameter.

Note: The WHERE primary command uses only two of the five possible parts of a data description parameter.

Usage notes
- WHERE can be used from the BROWSE option pointer panel and storage panel, and from the dump display reporter panel.
- WHERE produces a brief report describing all areas, structures, and modules that contain the address of interest.
- The area, structure, or module with the closest address to the address of interest is the one that will be added to the pointer stack. (More than one area may satisfy the search criteria.)

Example
Identify an area at a given address.

Action
The following screen shows the WHERE primary command being issued from the BROWSE option pointer panel. The same results occur if the command are issued from a dump display reporter panel.

```
DSNAME('D46IPCS.DRVC400.SA00001') POINTERS  --------------------------
[COMMAND ===> w 6b0 ]   SCROLL ===> CSR
                        --------------------------
PTR  Address  Address space  Data type
00001 00000000 ASID(X'0003') AREA
Remarks:
00002 00000680 ASID(X'0003') AREA
Remarks:
00003 00FD7420 ASID(X'0001') STRUCTURE(Cvt)
Remarks: Communications Vector Table
*****************************************************************************
END OF POINTER STACK*****************************************************************************
```

Result
First, all items that contain this address are displayed using the dump display reporter panel.

```
IPCS OUTPUT STREAM  --------------------------  LINE 0 COLS 1 78
COMMAND ===> _   SCROLL ===> CSR
*****************************************************************************
END OF DATA*****************************************************************************
```

Then the item with the smallest offset that contains the address ‘6b0’ – in this case, the PSA – is added to the pointer stack. The following screen shows the updated pointer stack.
IPCS dialog line commands

D line command — delete screen output

Use the D line command to permanently omit specific lines from the screen.

Syntax

\{ D \}
\{ Dn \}
\{ DD-DD \}

Parameters

n Represent a decimal number in the range of 1 through 9999.
- Represents an inclusive number of lines.

Usage notes

- D can be entered on the dump display reporter panel and on the pointer panel of the BROWSE option.
- When entering line commands, remember to do one of the following:
  - End the line command with a delimiter character (either a blank or a special character) that was not displayed in the report column following the line command.
  - Type the line command and press the ENTER key, leaving the cursor under the character following your line command.
- If you request a report that is too large to be held in virtual storage all at once, use D to omit sections of the report.
- More than one line command can be entered at a time. For example, before pressing the ENTER key the D, X, and S, F, or L line commands can be entered on the same screen.

Example

The following screens depict use of the D line command and the resulting display output after pressing the ENTER key.

The first screen shows using D on the dump display reporter panel.
This screen shows the result of using the D line command.

**E line command — edit a pointer**

Use the E line command on the pointer panel of the BROWSE option to edit a selected pointer.

**Syntax**

Chapter 6. IPCS dialog controls 6-33
E line command

E

Usage notes
– E can be used only on the BROWSE option pointer panel.
– After entering an E next to any pointer, the editing panel appears, as shown in Figure 6-5.

Use the editing panel to edit, add, or delete information in the selected pointer’s definition by typing the requested information in the appropriate fields.
– While the complete value of each field is displayed from the editing panel, certain fields may be truncated when you return to the pointer stack in the BROWSE option after editing.

Example
Edit a pointer on the pointer panel.

Figure 6-4. Using E on the Pointer Panel

Figure 6-5. Pointer Editing Panel
F line command — format a defined control block

Use the F line command to request the formatting of a pointer whose data type is defined as STRUCTURE on the pointer panel of the BROWSE option.

**Syntax**

```
F
```

**Usage notes**

- F can only be used from the BROWSE option pointer panel.
- The pointer on the pointer panel must be defined as a control block with the data type STRUCTURE.

**Example**

Format a control block on the pointer panel.

- **Action**
  
  The following screen shows where to enter the F line command.

- **Result**
  
  IPCS formats the CVT.

---

I line command — insert a pointer

Use the I line command to insert a pointer in the address pointer stack on the pointer panel of the BROWSE option. The inserted pointer describes the default address space after the selected pointer.
I line command

Syntax

\{ I \}
\{ In \}

Parameter

n  Represents a decimal number of 1 through 9999. If you omit n, the default is 1 pointer.

Usage notes

– The I line command can be used only while in the BROWSE option on the pointer panel.
– When inserting a pointer, IPCS supplies an address of 00000000.
– Entering the I line command causes IPCS to renumber the following existing pointers.

Example

Insert a pointer on the pointer panel.

– Action

The following screen shows use of the I line command and the resulting display output after pressing the ENTER key.

```

DSNAME('D83DUMP.DUMPC.PB00465') POINTERS

COMMAND ===> _ SCROLL ===> CSR

ASID(X'0014') is the default address space

PTR Address Address space Data type
00001 00000000 HEADER AREA
Remarks: Comment 1

<p>|i0002 00FD7BC8 ASID(X'0014') AREA |</p>
<table>
<thead>
<tr>
<th>Remarks: Comment 2</th>
</tr>
</thead>
</table>

00003 0000210 ASID(X'0014') AREA
Remarks: Comment 2
00004 00FD7B40 ASID(X'0001') STRUCTURE(Cvt)
Remarks: Communications Vector Table
Remarks: Communications Vector Table

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

– Result

The following screen shows the results of using an I line command.

---

6-36  z/OS V1R13.0 MVS IPCS Commands
R line command — repeat a pointer

Use the R line command to duplicate (or repeat) a selected pointer on the pointer panel of the BROWSE option.

Syntax

{ R }
{ Rh }

Parameter

n Represents the number of times the pointer should be repeated. The n is a decimal number from 1 through 9999.

Usage notes

– R can be used only while in the BROWSE option on the pointer panel.
– Entering R causes the existing pointers to be renumbered.

Example

Repeat an existing pointer twice on the pointer panel.

– Action

The following screen depicts use of the R line command.

---

Result
R line command

The following screen shows the resulting display output after pressing the ENTER key.

S line command — select a pointer to display storage

Use the S line command to choose a pointer from the address pointer stack on the pointer panel. IPCS then uses the pointer to display storage that is addressed by that pointer.

Note that you can also use the SELECT primary command.

Syntax

S

Example

Select the third pointer from the pointer stack to view the storage location at X'00000210'.

The screen shows the S line command.

S, F, and L line commands — show excluded screen output

Use the S, F, or L line command to request that specific lines be displayed from excluded lines in full screen. The lines to be shown are chosen by using the
indentation of the data. The lines that are indented closest to the left margin are displayed. If several lines are indented equally, the first lines are shown.

**Syntax**

- `{ S }`  
- `{ Sn }`  
- `{ F }`  
- `{ Fn }`  
- `{ L }`  
- `{ Ln }`

**Operations**

- **S** Shows a selected line from a block of excluded lines.
- **F** Shows the first line of excluded text.
- **L** Shows the last line of excluded text.

**Parameter**

- **n** Specified the number of excluded lines to be shown. The `n` is a decimal number of 1 through 9999.

**Usage notes**

- S, F, or L can be entered only on the dump display reporter panel.
- When entering line commands, do one of the following:
  - End the line command with a delimiter character, which can be either a blank or a special character, that was not displayed in the report column following the line command.
  - Type the line command and press the ENTER key, leaving the cursor under the character following your line command.
- More than one line command can be entered at a time. For example, before pressing the ENTER key the D, X, and S, F, or L line commands can be entered on the same screen.

**Example**

Use the F line command to show 2 excluded lines of text.

**Action**

The following screen shows the F line command on the dump display report panel.
### S, F, and L line commands

The following screen shows the resulting display output after pressing the ENTER key.
X line command — exclude screen output

Use the X line command to request that specific lines be suppressed from screen output. IPCS displays a statement that indicates the number of lines not being shown.

Syntax

{ Xn }
{ XX-XX }

Parameters

n Represents a decimal number in the range of 1 through 9999.

Usage notes

– The X line command can only be entered on the dump display reporter panel.
– When entering line commands, remember to do one of the following:
  - End the line command with a delimiter character, which can be either a blank or a special character, that was not displayed in the report column following the line command.
  - Type the line command and press the ENTER key, leaving the cursor under the character following your line command.
– More than one line command can be entered at a time. For example, before you press the ENTER key, enter the D, X, and S, F, or L line commands on the same screen.

Example

The following screens depict use of the X line command and the resulting display output after pressing the ENTER key.

The first screen shows using X on the dump display reporter panel.
X line command

This screen shows the result of using the X line command.

---

This screen shows the result of using the X line command.
Chapter 7. IPCS CLISTs and REXX EXECs

This topic describes some of the CLISTs and REXX execs that IPCS supplies. These CLISTs and REXX execs do the following:

- Print system storage areas
- Create problem screening reports
- Create a user dump directory or a sysplex dump directory
- Run a chain of save areas

CLISTs that are used to customize IPCS are described in z/OS MVS IPCS Customization.

System library SYS1.SBLSCL10 holds machine-readable copies of each CLIST and REXX EXEC. The names of the CLISTs begin with the letters BLSC, REXX EXECs with BLSX. This topic describes those CLISTs and REXX execs that IPCS users may invoke directly to perform tasks. See the z/OS MVS IPCS User’s Guide for more information about invoking CLISTs and REXX execs and running them in batch mode.

Task Directory for IPCS CLISTs and REXX EXECs

The following table summarizes the CLISTs and REXX EXECs supplied with IPCS:

### Analyze a dump

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obtain a stand-alone dump screening report</td>
<td>“BLSCSCAN CLIST — obtain a stand-alone dump screening report” on page 7-12</td>
</tr>
<tr>
<td>Obtain an SVC dump screening report</td>
<td>“BLSCBSVB CLIST — obtain an SVC dump screening report” on page 7-5</td>
</tr>
<tr>
<td>Obtain a SYSMDUMP dump screening report</td>
<td>“BLSCBSYB CLIST — obtain a SYSMDUMP dump screening report” on page 7-6</td>
</tr>
<tr>
<td>Format save area chain</td>
<td>“BLSCEPTR CLIST — run a save area chain” on page 7-9</td>
</tr>
<tr>
<td>List entry points with the same name</td>
<td>“BLSXWHERE REXX EXEC — find All modules with the same entry point name” on page 7-14</td>
</tr>
</tbody>
</table>

### Customize an IPCS session

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create or allocate a user dump directory or a sysplex dump directory</td>
<td>“BLSCDDIR CLIST — create a dump directory” on page 7-8</td>
</tr>
<tr>
<td>Remove uncataloged dump directory entries</td>
<td>“BLSCDROP CLIST — issue IPCS DROPDUMP for uncataloged DSNAMES entries” on page 7-9</td>
</tr>
<tr>
<td>Define IPCS dialog libraries to ISPF</td>
<td>See BLSCLIBD CLIST in the topic about BLSCLIBD CLIST - Activate IPCS Dialog Services in z/OS MVS IPCS Customization</td>
</tr>
</tbody>
</table>
### Print dump analysis reports

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print a stand-alone dump screening report</td>
<td>&quot;BLSCBSAA CLIST — print a stand-alone dump screening report&quot; on page 7-3</td>
</tr>
<tr>
<td>Print an SVC dump screening report</td>
<td>&quot;BLSCBSVA CLIST — print an SVC dump screening report&quot; on page 7-4</td>
</tr>
<tr>
<td>Print a SYSMDUMP dump screening report</td>
<td>&quot;BLSCBSYA CLIST — print a SYSMDUMP dump screening report&quot; on page 7-6</td>
</tr>
<tr>
<td>Print a stand-alone dump detailed report</td>
<td>&quot;BLSCBSAP CLIST — print a stand-alone dump detailed report&quot; on page 7-3</td>
</tr>
<tr>
<td>Print an SVC dump detailed report</td>
<td>&quot;BLSCBSVP CLIST — print an SVC dump detailed report&quot; on page 7-5</td>
</tr>
<tr>
<td>Print a SYSMDUMP dump detailed report</td>
<td>&quot;BLSCBSYP CLIST — print a SYSMDUMP dump detailed report&quot; on page 7-7</td>
</tr>
</tbody>
</table>

### Print storage data

<table>
<thead>
<tr>
<th>When you want to</th>
<th>Use the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print common storage areas</td>
<td>&quot;BLSCPCSA CLIST — print common storage areas” on page 7-10.</td>
</tr>
<tr>
<td>Print nucleus storage areas</td>
<td>&quot;BLSCPNUC CLIST — print nucleus storage areas” on page 7-10.</td>
</tr>
<tr>
<td>Print one or more storage areas</td>
<td>&quot;BLSCPRINT CLIST — print a dump” on page 7-11.</td>
</tr>
<tr>
<td>Print private storage areas</td>
<td>&quot;BLSCPRIV CLIST — print private storage areas” on page 7-11.</td>
</tr>
<tr>
<td>Print global system queue areas</td>
<td>&quot;BLSCPSQA CLIST — print global system queue areas” on page 7-12</td>
</tr>
</tbody>
</table>

### Sample CLISTs and REXX EXECs

<table>
<thead>
<tr>
<th>For This Subcommand</th>
<th>See the Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPARE</td>
<td>5-46</td>
</tr>
<tr>
<td>EVALDEF</td>
<td>5-97</td>
</tr>
<tr>
<td>EVALDUMP</td>
<td>5-99</td>
</tr>
<tr>
<td>EVALMAP</td>
<td>5-103 &quot;BLSXWHERE REXX EXEC — find All modules with the same entry point name” on page 7-14.</td>
</tr>
<tr>
<td>EVALSYM</td>
<td>5-109</td>
</tr>
<tr>
<td>RUNCHAIN</td>
<td>5-222</td>
</tr>
</tbody>
</table>

---

1. Print dump analysis reports
2. Print storage data
3. Sample CLISTs and REXX EXECs
BLSCBSAA CLIST — print a stand-alone dump screening report

Use the BLSCBSAA CLIST to print an initial screening report for a stand-alone dump. BLSCBSAA copies the stand-alone dump from tape to DASD. The stand-alone dump tape must be allocated to file IEFNDER. BLSCBSAA routes the output dump report to the IPCSPRNT data set.

The IBM-supplied cataloged procedure BLSJIPCS is designed to invoke this CLIST. You can run BLSJIPCS from JCL or from an operator console.

BLSCBSAA produces the same dump report as does the BLSCSCAN CLIST. See "BLSCSCAN CLIST — obtain a stand-alone dump screening report" on page 7-12 and z/OS MVS IPCS User’s Guide for other ways to obtain an initial screening report for a stand-alone dump.

The following examples show how to run BLSCBSAA with the BLSJIPCS cataloged procedure.

Syntax for JCL invocation

```
//stepname EXEC PROC=IPCS,
//CLIST=BLSCBSAA,
//DUMP=sadump.dsname
//*
//* The following DD statement is required for CLIST=BLSCBSAA
//*
//IEFPROC.IEFRDER DD .... Input dump for copy
//*
//* The following DD statement is optional. If omitted, the
dump directory is dynamically allocated.
//*
//IEFPROC.IPCSDDIR DD .... IPCS dump directory
```

Syntax for operator console invocation

```
ALLOCATE INFILE(IEFRDER) and OUTFILE(IPCSDUMP)
START BLSJIPCS,CLIST=BLSCBSAA,DUMP='sadump.dsname'
```

CLIST listing

See the BLSCBSAA member of SYS1.SBLSCLI0.

BLSCBSAP CLIST — print a stand-alone dump detailed report

Use the BLSCBSAP CLIST to print detailed storage information for a stand-alone dump. Because this CLIST prints the storage, it should only be used in exceptional circumstances, for example, when debugging an application that does not provide IPCS support.

BLSCBSAP copies the stand-alone dump from tape to DASD. The stand-alone dump tape must be allocated to file IEFNDER. BLSCBSAP routes the output dump report to the IPCSPRNT data set.

The IBM-supplied cataloged procedure BLSJIPCS is designed to invoke this CLIST. You can run BLSJIPCS from JCL or from an operator console.

See "BLSCSCAN CLIST — obtain a stand-alone dump screening report" on page 7-12 and z/OS MVS IPCS User’s Guide for other ways to obtain information from a stand-alone dump.
The following examples show how to run BLSCBSAP with the BLSJIPCS cataloged procedure.

**Syntax for JCL invocation**

```
//stepname EXEC PROC=IPCS,
//CLIST=BLSCBSAP,
//DUMP=sadump.dsname
/*
// The following DD statement is required for CLIST=BLSCBSAP
/**
//IEFPROC.IEFRDER DD .... Input dump for copy
/**
// The following DD statement is optional. If omitted, the
// dump directory is dynamically allocated.
/**
//IEFPROC.IPCSDDIR DD .... IPCS dump directory
```  

**Syntax for operator console invocation**

```
ALLOCATE INFILE(IEFRDER) and OUTFILE(IPCSDUMP)
```

```
START BLSJIPCS,CLIST=BLSCBSAP,DUMP='sadump.dsname'
```

**CLIST listing**

See the BLSCBSAP member of SYS1.SBLSCLI0.

---

**BLSCBSVA CLIST — print an SVC dump screening report**

Use the BLSCBSVA CLIST to print an initial screening report for an SVC dump. BLSCBSVA routes the output dump report to the IPCSPRNT data set.

The IBM-supplied cataloged procedure BLSJIPCS is designed to invoke this CLIST. You can run BLSJIPCS from JCL or from an operator console.

This CLIST produces the same dump report as does the BLSCBSVB CLIST. See “BLSCBSVB CLIST — obtain an SVC dump screening report” on page 7-5 and z/OS MVS IPCS User’s Guide for other ways to obtain an initial screening report for an SVC dump.

The following examples show how to run BLSCBSVA with the BLSJIPCS cataloged procedure.

**Syntax for JCL invocation**

```
//stepname EXEC PROC=IPCS,
//DUMP=svcdump.dsname
/*
// The following DD statement is optional. If omitted, the
// dump directory is dynamically allocated.
/**
//IEFPROC.IPCSDDIR DD .... IPCS dump directory
```  

**Syntax for operator console invocation**

```
START BLSJIPCS,DUMP='svcdump.dsname'
```

**CLIST listing**

See the BLSCBSVA member of SYS1.SBLSCLI0.
BLSCBSVB CLIST — obtain an SVC dump screening report

Use the BLSCBSVB CLIST to create an initial screening report for an SVC dump. Using the IPCS dialog, invoke BLSCBSVB through the SUBMIT option, then the Prepare SVC Dump for Analysis option. IPCS submits a batch job for the CLIST that routes the output dump report to a SYSOUT data set.

You can invoke BLSCBSVB directly from an IPCS session, but the CLIST takes a long time to complete processing.

**IPCS batch invocation**

You must supply the data set name, dump directory name, and sysout class.

```
-------------------- Prepare SVC Dump for IPCS Analysis ----------------------
COMMAND ===]  
Enter/verify parameters for the job.  
Use ENTER to submit the job, END to terminate without job submission.  
DATA SET NAME ===]  
DUMP DIRECTORY ===]  
SYSOUT CLASS ===]  
```

**IPCS dialog invocation**

BLSCBSVB uses the current dump data set and dump directory.

```
-------------------- IPCS Subcommand Entry ------------------------------
Enter a free-form IPCS subcommand, CLIST, or REXX EXEC invocation below:
===] \BLSCBSVB  
```

**CLIST listing**

See the BLSCBSVB member of SYS1.SBLSCLI0.

BLSCBSVP CLIST — print an SVC dump detailed report

Use the BLSCBSVP CLIST to print detailed storage information for an SVC dump. Because this CLIST prints the storage, it should only be used in exceptional circumstances, for example, when debugging an application that does not provide IPCS support.

The IBM-supplied cataloged procedure BLSJIPCS is designed to invoke this CLIST. You can run BLSJIPCS from JCL or from an operator console.


The following examples show how to run BLSCBSVP with the BLSJIPCS cataloged procedure.

**Syntax for JCL invocation**
BLSCBSVP CLIST

//stepname EXEC PROC=IPCS,
//CLIST=BLSCBSVP,
//DUMP=svcdump.dsnname
/*** The following DD statement is optional. If omitted, the
**/ dump directory is dynamically allocated.
***/
//IEFPROC.IPCSDIR DD .... IPCS dump directory

Syntax for operator console invocation

START BLSJIPCS,CLIST=BLSCBSVP,DUMP='svcdump.dsnname'

CLIST listing
See the BLSCBSVP member of SYS1.SBLSCLI0.

BLSCBSYA CLIST — print a SYSMDUMP dump screening report

Use the BLSCBSYA CLIST to print an initial screening report for an SVC dump.
BLSCBSYA routes the output dump report to the IPCSPRNT data set.

The IBM-supplied cataloged procedure BLSJIPCS is designed to invoke this CLIST.
You can run BLSJIPCS from JCL or from an operator console.

This CLIST produces the same dump report as does the BLSCBSYB CLIST. See
"BLSCBSYB CLIST — obtain a SYSMDUMP dump screening report" on page 7-5 and
z/OS MVS IPCS User’s Guide for other ways to obtain an initial screening report for
an SVC dump.

The following examples show how to run BLSCBSYA with the BLSJIPCS cataloged
procedure.

Syntax for JCL invocation

This JCL runs BLSCBSYA with cataloged procedure IPCS.

//stepname EXEC PROC=IPCS,
//CLIST=BLSCBSYA,
//DUMP=sysmdump.dsnname
/*** The following DD statement is optional. If omitted, the
**/ dump directory is dynamically allocated.
***/
//IEFPROC.IPCSDIR DD .... IPCS dump directory

Syntax for operator console invocation

START BLSJIPCS,CLIST=BLSCBSYA,DUMP='sysmdump.dsnname'

CLIST listing
See the BLSCBSYA member of SYS1.SBLSCLI0.

BLSCBSYB CLIST — obtain a SYSMDUMP dump screening report

Use the BLSCBSYB CLIST to create an initial screening report for a SYSMDUMP
dump. Using the IPCS dialog, invoke BLSCBSYB through the SUBMIT option, then
the Prepare SYSMDUMP Dump for Analysis option. IPCS submits a batch job for
the CLIST that routes the output dump report to a SYSOUT data set.
BLSCBSYB CLIST

You can invoke BLSCBSYB directly from an IPCS session, but the CLIST takes a long time to complete processing.

IPCS batch invocation
You must supply the data set name, dump directory, and sysout class.

```
------------------------ Prepare SYSMDUMP for IPCS Analysis ------------------------
COMMAND =

Enter/verify parameters for the job.
Use ENTER to submit the job, END to terminate without job submission.

DATA SET NAME =
DUMP DIRECTORY =
SYSOUT CLASS =
```

IPCS dialog invocation
BLSCBSYB uses the current dump data set and dump directory.

```
------------------------ IPCS Subcommand Entry ------------------------

Enter a free-form IPCS subcommand, CLIST, or REXX EXEC invocation below:

==] %BLSCBSYB
```

CLIST listing
See the BLSCBSYB member of SYS1.SBLSCLI0.

BLSCBSYP CLIST — print a SYSMDUMP dump detailed report

Use the BLSCBSYP CLIST to print detailed storage information for a SYSMDUMP dump. Because this CLIST prints the storage, it should only be used in exceptional circumstances, for example, when debugging an application that does not provide IPCS support. BLSCBSYP routes the output dump report to the IPCSPRNT data set.

The IBM-supplied cataloged procedure BLSJIPCS is designed to invoke this CLIST. You can run BLSJIPCS from JCL or from an operator console.

See "BLSCSCAN CLIST — obtain a stand-alone dump screening report" on page 7-12 and z/OS MVS IPCS User’s Guide for other ways to obtain information from a stand-alone dump.

The following examples show how to run BLSCBSYP with the BLSJIPCS cataloged procedure.

Syntax for JCL invocation

```
//stepname EXEC PROC=IPCS,
//CLIST=BLSCBSYP,
//DUMP=sysmdump.dsn
/*
/* The following DD statement is optional. If omitted, the
/* dump directory is dynamically allocated.
/*
//IEFPROC.IPCSDDIR DD .... IPCS dump directory
```

Syntax for operator console invocation

```
START BLSJIPCS,CLIST=BLSCBSYP,DUMP=sysmdump.dsn
```
BLSCDDIR CLIST — create a dump directory

The IBM-supplied BLSCDDIR CLIST can be used to do the following:
- Create a sysplex dump directory
- Create a user dump directory when accessing IPCS
- Create user dump directories that satisfy special needs
- Create multiple user dump directories so that, for example, you can do simultaneous interactive and batch processing

BLSCDDIR uses IBM-defined defaults that can be reset by your installation. For a user dump directory, the installation determines the size and volume default values that best suit your installation's needs using information found in z/OS MVS IPCS Customization.

For more information about the use of BLSCDDIR, see z/OS MVS IPCS User's Guide.

Syntax

```%
BLSCDDIR [ DATACLAS(data-class) ]
[ DSNAME(dsname) ]
[ FILE(filename) ]
[ MGMTCLAS(management-class) ]
[ NDXCISZ(index-class) ]
[ NOENQ ]
[ RECORDS(records) ]
[ STORCLAS(storage-class) ]
[ VOLUME(volume) ]
```

Parameters

**DATACLAS(data-class)**
Specifies the data class for the new directory. If you omit this parameter, there is no data class specified for the new directory.

**DSNAME(dsname)**
Specifies the fully-qualified name you want to assign to the directory. If you omit this parameter, the IBM-supplied defaults are:
- If you have a userid prefix, prefix.DDIR
- Otherwise, SYS1.DDIR

**FILE(filename)**
Specifies the name of the file with which the ALLOCATE command associates the DSNAME. The IBM-supplied default is IPCSDDIR.

**MGMTCLAS(management-class)**
Specifies the management class for the new directory. If you omit this parameter, there is no management class specified for the new directory.
NDXCISZ(index-cisz)
Specifies the control interval size for the index portion of the new directory.
If you omit this parameter, the IBM-supplied default is 4096 bytes.

NOENQ
Suppresses ENQ processing that is intended to block other instances of
IPCS from using the directory being prepared for use by IPCSDDIR. IPCS
itself uses this option when it has already established the needed
serialization. Manual use of this option is not recommended.

RECORDS(records)
Specifies the number of records you want the directory to accommodate. If
you omit this parameter, the IBM-supplied default is 5000; your
installation’s default might vary.

STORCLAS(storage-class)
Specifies the storage class for the new directory. If you omit this parameter,
there is no storage class specified for the new directory.

VOLUME(volume)
Specifies the VSAM volume on which the directory should reside. If you
omit DATACLAS, MGMTCLAS, STORCLAS, and VOLUME, the
IBM-supplied default is VSAM01. Otherwise, there is no IBM-supplied
default.

CLIST listing
See the BLSCDDIR member of SYS1.SBLSCLI0.

BLSCDROP CLIST — issue IPCS DROPDUMP for uncataloged
DSNAME entries
Use the BLSCDROP CLIST to issue DROPDUMP against data sets that are
described through DSNAME in the currently allocated dump directory, yet are not
catalogued. This cleans out entries that are no longer associated with a cataloged
dump data set.

Note: If the data set was renamed, use the IPCS ALTER subcommand to change
the name of the dump or trace data set in the IPCS dump directory, before
using BLSCDROP (or issuing the IPCS DROPDUMP CLIST).

Syntax

%BLSCDROP

CLIST listing
See the BLSCDDIR member of SYS1.SBLSCLI0.

BLSCEPTR CLIST — run a save area chain
BLSCEPTR follows the forward chain of save areas. Beginning with the failing
TCB, it finds the first problem program’s save area. BLSCEPTR locates the entry
point address in the save area, then goes to that address to check the entry point
identifier.

You should supply the address of the failing TCB when you invoke BLSCEPTR.
Otherwise BLSCEPTR uses the default address found in field PSATOLD
(PSA+X’21C’).
BLSCCEPTR CLIST

The subcommands in this CLIST create the following symbols in the IPCS symbol table:

- **EPnnn**  
  Entry points saved in the save area chain. For example, the symbol EP001 represents the entry point saved in the first save area on the chain.

- **EPIDnnn**  
  The entry point identifier string for the entry point represented by EPnnn.

- **SAnnn**  
  The save area holding the entry point address represented by EPnnn.

**Syntax**

```bash
%BLSCCEPTR [TCB(address)]
```

**Parameter**

- **TCB(address)**  
  The address of the TCB that BLSCCEPTR uses to start chaining the save areas. If you do not specify a TCB address, BLSCCEPTR uses the address found in PSATOLD (PSA+X'21C').

**CLIST listing**

See the BLSCCEPTR member of SYS1.SBLSCLI0.

---

BLSCPCSA CLIST — print common storage areas

Use the BLSCPCSA CLIST to print the common storage area (CSA) and extended common storage area (ECSA) from the current dump. See [z/OS MVS IPCS Customization](https://www.ibm.com/support/knowledgecenter/SSDFA0_2.2.0/com.ibm.zos.mvs.doc/zos_mvs_ipcs.html) for more information about writing a CLIST that uses BLSCPCSA to create a custom dump report.

**Syntax**

```bash
%BLSCPCSA
```

**CLIST listing**

See the BLSCPCSA member of SYS1.SBLSCLI0.

---

BLSCPNUC CLIST — print nucleus storage areas

Use the BLSCPNUC CLIST to print the following nucleus storage areas from a dump:

- Read-write nucleus
- Extended read-write nucleus
- Read-only nucleus
- Dynamic address translation (DAT) off nucleus

See [z/OS MVS IPCS Customization](https://www.ibm.com/support/knowledgecenter/SSDFA0_2.2.0/com.ibm.zos.mvs.doc/zos_mvs_ipcs.html) for more information about writing a CLIST that uses BLSCPNUC to create a custom dump report.

**Syntax**

```bash
%BLSCPNUC
```

**CLIST listing**

See the BLSCPNUC member of SYS1.SBLSCLI0.
BLSCPRIV CLIST — print private storage areas

BLSCPRIV prints the private and extended private storage areas for an address space. See [z/OS MVS IPCS Customization] for more information about writing a CLIST that uses BLSCPRIV to create a custom dump report.

Syntax

```%BLSCPRIV asid```

Parameter

`asid`

The address space identifier (ASID) for the address space to be printed.

CLIST listing

See the BLSCPRIV member of SYS1.SBLSCLI0.

BLSCPRNT CLIST — print a dump

Use the BLSCPRNT CLIST to print one or more of the following storage areas from a dump:

- Common storage areas
- Nucleus storage areas
- Global system queue areas
- Control block summary information and the private area for one or more of the following:
  - Each active address space at the time of the dump
  - An address space specified by job name.

Syntax

```%BLSCPRNT [ CSA ] [ NUCLEUS ] [ SQA ] [ CURRENT ] [ JOBNAME(jobname) ]```

Parameters

Separate parameters with a comma.

**CSA**

Specifies BLSCPRNT is to print the common storage area (CSA) and extended CSA (ECSA).

**NUCLEUS**

Specifies BLSCPRNT is to print the following areas:

- Read-write nucleus
- Extended read-write nucleus
- Read-only nucleus
- Dynamic address translation (DAT) off nucleus

**SQA**

Specifies BLSCPRNT is to print the global system queue area (SQA) and extended SQA (ESQA).
**BLSCPRNT CLIST**

**CURRENT**
Specifies BLSCPRNT is to print control block summary information and the private area for each active address space at the time of the dump.

**JOBNAME(jobname)**
Specifies BLSCPRNT is to print control block summary information and the private area for the address space specified by JOBNAME(jobname).

**Example of IPCS dialog invocation**
Enter the following five commands in succession.

```plaintext
ALLOCATE DDNAME(IPCSTOC) SYSOUT(x)
ALLOCATE DDNAME(IPCSPRNT) SYSOUT(x)
SETDEF DSNAME('dump.dname')
%BLSCPRNT NUCLEUS,SQA,CSA,CURRENT,JOBNAME(jobname)
CLOSE PRINT
```

**Example of IPCS batch invocation**

```plaintext
//jobname JOB (acct#),'name',MSGCLASS=A,REGION=4M
//PRTDUMP EXEC PGM=IKJEFT01
//SYSPROC DD DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSTOC DD SYSOUT=* 
//IPCSPRNT DD SYSOUT=* 
//SYSTSIN DD *
%BLSCDDIR DSNNAME(userid.ddir) VOLUME(volid)... (optional)
IPCS
SETDEF DSN('dump.dname') PRINT
%BLSCPRNT NUCLEUS,SQA,CSA,CURRENT,JOBNAME(jobname)
/*

**CLIST listing**
See the BLSCPRNT member of SYS1.SBLSCLI0.

**BLSCPSQA CLIST — print global system queue areas**

Use the BLSCPSQA CLIST to print the global system queue area (SQA) and the extended SQA (ESQA) from a dump. See [z/OS MVS IPCS Customization](https://www.ibm.com/support/knowledgecenter/S5T7TN_2.4.6/sect12_24430014.html) for more information about writing a CLIST that uses BLSCPSQA to create a custom dump report.

**Syntax**

```plaintext
%BLSCPSQA
```

**CLIST listing**
See the BLSCPSQA member of SYS1.SBLSCLI0.

**BLSCSCAN CLIST — obtain a stand-alone dump screening report**

Use the BLSCSCAN CLIST to create an initial screening report for a stand-alone dump. The IPCS dialog option used to run BLSCSCAN depends on the location of the stand-alone dump:

- If it is on tape, use the IPCS dialog SUBMIT option, then the Prepare Stand-Alone Dump for Analysis option. IPCS submits a batch job for the CLIST that copies the dump to DASD and routes the output dump report to a SYSOUT data set.
• If it is already on DASD, use the IPCS dialog SUBMIT option, then the Perform Supplementary Dump Analysis option. IPCS submits a batch job for the CLIST that routes the output dump report to a SYSOUT data set.

You can invoke BLSCSCAN directly from an IPCS session, but the CLIST takes a long time to complete processing.

**IPCS batch invocation for Tape**

Use this option if the stand-alone dump is on tape.

```
------------------ Prepare Stand Alone Dump for Analysis ------------------
COMMAND ==>

Enter/verify parameters for the job.
Use ENTER to submit the job, END to terminate without job submission.

INPUT DUMP TAPES:
  GENERIC UNIT ==>[3480] UNIT COUNT ==>[1]  
  VOLUME SERIAL (Enter at least one, if more, separate with a comma.)
    ==> TAPIN1 
  LABEL (Separate subparameters with a comma.)
    ==> 1,NL 

OUTPUT DASD DUMP DATA SET:
  DATA SET NAME ==>[DUMMY] 
  GENERIC UNIT ==>[3380] 
  VOLUME SERIAL (Enter at least one, if more, separate with a comma.)
    ==> SCRO06 

SPACE FOR OUTPUT DASD DUMP DATA SET (Number of blocks)
  PRIMARY ==>[62000] SECONDARY ==>[1000] 

DUMP DIRECTORY ==>[NHAN.IPCS410.DDIR] 
SYSOUT CLASS ==>[H] 
```

**IPCS batch invocation for DASD**

Use this option if the stand-alone dump is on DASD. You must specify BLSCSCAN as the CLIST to be invoked.

```
------------------ Perform Supplementary IPCS Dump Analysis ------------------
COMMAND ==>

Enter/verify parameters for the job.
Use ENTER to submit the job, END to terminate without job submission.

DATA SET NAME ==>
DUMP DIRECTORY ==>
SYSOUT CLASS ==>

IPCS SUBCOMMAND, CLIST or REXX EXEC:
  ==> BLSCSCAN 

ADDITIONAL CLIST or REXX EXEC LIBRARIES: (optional)
  ==>
  ==>
```

**IPCS dialog invocation**

BLSCSCAN uses the current dump data set and dump directory.
BLSXWHER REXX EXEC — find All modules with the same entry point name

Use the BLSXWHER EXEC to find all modules in dump storage associated with the same entry point name. BLSXWHER searches for modules with the same entry point in private area storage. For ASID(1), BLSXWHER also searches modules in the link pack area (LPA). BLSXWHER displays the storage map entry for each module, identifying the starting address and other attributes for the module.

Before searching for the modules, BLSXWHER maps the modules in the private area and, for ASID(1), the LPA.

Syntax

```
%BLSXWHER {epname} [ASID(asid)]
```

Parameters

**epname**
- Specifies the name of an entry point. BLSXWHER finds all modules with this entry point.

**ASID(asid)**
- Specifies the address space that BLSXWHER will search. If no ASID is specified, BLSXWHER uses the default address space for the dump. See “Address processing parameters” on page 3-7 for information about specifying asid.

IPCS dialog invocation

BLSXWHER finds the storage map entries for load module ILRPGEXP in the default address space, if any exist.

---

REXX EXEC listing

See the BLSXWHER member of SYS1.SBLSCLI0.
Chapter 8. IPCS batch mode

IPCS can be used in batch mode in a TSO/E environment. Consider using a batch job when you:
- Use IPCS subcommands to print selected portions of a dump
- Load system dump data sets from tape or mass storage
- Unload system dump data sets to tape or mass storage
- Perform time-consuming dump analysis

Note that there are some subcommand restrictions for using IPCS in batch mode. These restrictions are indicated under the applicable subcommand.

JCL needed to run IPCS in batch mode

The following figure shows the JCL needed to run IPCS in batch mode, and it shows how to invoke the BLSCSCAN CLIST to format a problem screening report for a stand-alone dump. The control information is saved in a dump directory data set that can be used for later formatting sessions in batch mode or at a terminal. This example assumes that you have an existing dump directory data set. For more information, see the z/OS MVS IPCS User’s Guide.

Note: If you plan to use the IPCS output at a terminal after the batch job has completed, you may want to specify message and SYSOUT classes for held output rather than the MSGCLASS=A and SYSOUT=A on the DD statements in the example.
IPCS cataloged procedure

The IPCS cataloged procedure is found in member BLSJIPCS of SYS1.PROCLIB.
The procedure:
- Invokes program IKJEFT01
- Allocates the dump data set, IPCS parmlib members CLIST library, and output data sets.

BLSJIPCS has the following syntax:

```
//IPCS PROC CLIST=BLSCBSVA,DUMP=
//IEFPROC EXEC PGM=IKJEFT01,REGION=4M,DYNAMNBR=10,
// PARM=('%&CLIST.', '&DUMP.')
/*
/**
/**
//IPCDUMP DD DSN=afone,DISP=SHR DUMP OR TRACE DATA SET
//SYSPROC DD DSN=SYS1.SBLSCLI0,DISP=SHR CLIST PROCEDURES
//SYSTSPRT DD SYSOUT=A BATCH TSO/E COMMANDS
/*
/**
/**
//SYSTOC DD SYSOUT=A PRINT FILE TABLE OF CONTENTS
//IPCSPPRT DD SYSOUT=A PRINT FILE
```

Running CLISTs with BLSJIPCS

BLSJIPCS is designed to run with the following CLISTs:
- "BLSCBSAA CLIST — print a stand-alone dump screening report” on page 7-3
- "BLSCBSAP CLIST — print a stand-alone dump detailed report” on page 7-3
- "BLSCBSVA CLIST — print an SVC dump screening report” on page 7-4
- "BLSCBSVP CLIST — print an SVC dump detailed report” on page 7-5
- "BLSCBSYA CLIST — print a SYSMDUMP dump screening report” on page 7-6
- "BLSCBSYP CLIST — print a SYSMDUMP dump detailed report” on page 7-7
IPCS symbols

This appendix lists the definitions of all symbols that IPCS may automatically define. IBM recommends that installation-defined CLISTs and other dump analysis procedures do not use symbols that might conflict with these names.

Defining symbols

If a dump analysis subcommand needs a control block, it automatically locates the control block, validates it, and creates a definition for it in the symbol table and storage map of your current user dump directory.

When a subcommand creates a definition, it uses the symbol name in the following table. All numbers, n, are decimal numbers, except where specified differently.

Notes:
1. Most symbols are defined by IPCS only for SVC dumps.
2. To provide acceptable performance, IPCS places definitions in the symbol table for a dump only upon demand. The `z/OS MVS IPCS User’s Guide` describes how a data description (data-descr) parameter on a subcommand can cause dynamic definition of a symbol, if it did not exist in the symbol table.
3. A function that accesses data for which an IPCS name exists (for example, an ASCB) does not always associate an IPCS symbol with that data.
4. The symbol table is used only by IPCS. Note that many functions can be performed in a non-IPCS environment where the symbol table is not available.

Creating symbols

If you explicitly create or modify one of the symbols, rather than let IPCS create or modify it, you might bypass IPCS's validity checking process. For example, if you create the symbol UCB000E with the following subcommand:

```
equate ucb000e 4140.
```

and later use the FINDUCB subcommand to locate the UCB for device 000E, the FINDUCB subcommand finds the symbol in the symbol table and displays the storage at the address associated with that symbol. Because your EQUATE subcommand did not specify STRUCTURE(UCB), the storage at X'4140' was not validity checked to ensure that it is a UCB.

IPCS symbol definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Associated data</th>
<th>Data type definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABENDCODE</td>
<td>ABEND code</td>
<td>STRUCTURE(SDWAABCC)</td>
</tr>
<tr>
<td>AFT</td>
<td>The ASN-first-table control block</td>
<td>STRUCTURE(AFTE)</td>
</tr>
<tr>
<td>ASCBnnnnn</td>
<td>The address space control block for address space nnnnn</td>
<td>STRUCTURE(ASCB)</td>
</tr>
<tr>
<td>ASMVT</td>
<td>The system auxiliary storage management vector table</td>
<td>STRUCTURE(ASMVT)</td>
</tr>
<tr>
<td>ASTnnnn</td>
<td>The ASN-second-table control block for address space group nnnn</td>
<td>STRUCTURE(ASTE)</td>
</tr>
</tbody>
</table>
## IPCS Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Associated data</th>
<th>Data type definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTEnnnnn</td>
<td>The ASN-second-table control block entry for address space nnnnn</td>
<td>STRUCTURE(ASTE)</td>
</tr>
<tr>
<td>ASVT</td>
<td>The system address space vector table</td>
<td>STRUCTURE(ASVT)</td>
</tr>
<tr>
<td>ASXBnnnnn</td>
<td>The address space extension block for address space nnnnn</td>
<td>STRUCTURE(ASXB)</td>
</tr>
<tr>
<td>BLSQXBT</td>
<td>Table of system materials built from parmlib members BLSCECT, BLSCUSER, ...</td>
<td>STRUCTURE(BLSQXBT)</td>
</tr>
<tr>
<td>BLSQXBTnnnnn</td>
<td>Table of materials used by IPCS in ASID nnnnn for processing of dumps and traces generated by an ESA-mode system.</td>
<td>STRUCTURE(BLSQXBT)</td>
</tr>
<tr>
<td>BLSQXBTG</td>
<td>Table of materials used by IPCS in ASID nnnnn for processing of dumps and traces generated by a system supporting z/Architecture.</td>
<td>STRUCTURE(BLSQXBT)</td>
</tr>
<tr>
<td>CDEpgmname</td>
<td>A contents directory entry for entry point pgmname</td>
<td>STRUCTURE(CDE)</td>
</tr>
<tr>
<td>COMMON</td>
<td>The system common area</td>
<td>AREA(COMMON)</td>
</tr>
<tr>
<td>COMPONENTID</td>
<td>Component ID</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>CPUD</td>
<td>CPU Dependent Block</td>
<td>STRUCTURE(CPUD)</td>
</tr>
<tr>
<td>CSA</td>
<td>The common system area</td>
<td>AREA(CSA)</td>
</tr>
<tr>
<td>CSD</td>
<td>The common system data area</td>
<td>STRUCTURE(CSD)</td>
</tr>
<tr>
<td>CSECT</td>
<td>Control section</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>CURSOR</td>
<td>A fullword pointer identified by the position of the cursor on the display terminal</td>
<td></td>
</tr>
<tr>
<td>CVT</td>
<td>The system communications vector table</td>
<td>STRUCTURE(CVT)</td>
</tr>
<tr>
<td>CVTVSTGX</td>
<td>The virtual storage address extension to the system communications vector table</td>
<td>STRUCTURE(CVTVSTGX)</td>
</tr>
<tr>
<td>CVTXTNT2</td>
<td>The system communications vector table extension</td>
<td>STRUCTURE(CVTXTNT2)</td>
</tr>
<tr>
<td>DAESYMPTOMS</td>
<td>The symptoms provided by the program that requested the dump and, possibly, by the program that produced the dump. These are MVS symptoms, which are used by dump analysis and elimination (DAE) to identify duplicate dumps. If the primary symptom string is longer than 256 bytes, this symbol contains the first 256 bytes of the symptom string.</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>DATOFFNUCLEUS</td>
<td>The portion of the system nucleus that is used with dynamic address translation turned off</td>
<td>AREA(DATOFFNUCLEUS)</td>
</tr>
<tr>
<td>DIB</td>
<td>A control block maintained to support the data-in-virtual function</td>
<td>STRUCTURE(DIB)</td>
</tr>
<tr>
<td>DIBX</td>
<td>A control block maintained to support the data-in-virtual function</td>
<td>STRUCTURE(DIBX)</td>
</tr>
<tr>
<td>DUMPINGPROGRAM</td>
<td>The name of the program that produced the dump</td>
<td>CHARACTER</td>
</tr>
</tbody>
</table>
### IPCS Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Associated data</th>
<th>Data type definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMPINGSYSTEM</td>
<td>The system that wrote and was represented by the dump</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>DUMPPRIMARYDSNAME</td>
<td>The name of the original data set to which the dump was written</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>DUMPREQUESTOR</td>
<td>The name of the program that requested the dump</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>DUMPTIMESTAMP</td>
<td>The time from the time-of-day (TOD) clock presented in the following format: mm/dd/yyyy hh:mm:ss.ffffff</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>DUMPTOD</td>
<td>The time from the time-of-day (TOD) clock in a bit string</td>
<td>STRUCTURE(TODCLOCK)</td>
</tr>
<tr>
<td>ECSA</td>
<td>The extended common system area</td>
<td>AREA(ECSA)</td>
</tr>
<tr>
<td>EFLPA</td>
<td>The extended fixed link pack area</td>
<td>AREA(EFLPA)</td>
</tr>
<tr>
<td>EMLPA</td>
<td>The extended modified link pack area</td>
<td>AREA(EMLPA)</td>
</tr>
<tr>
<td>ENUCLEUS</td>
<td>The extended nucleus</td>
<td>AREA(ENUCLEUS)</td>
</tr>
<tr>
<td>EPnnnnnn</td>
<td>Entry point nnnnn in an entry point trace</td>
<td>MODULE</td>
</tr>
<tr>
<td>EPIIDnnnnn</td>
<td>Entry point identifier nnnnn in an entry point trace</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>EPLPA</td>
<td>The extended pageable link pack area</td>
<td>AREA(EPLPA)</td>
</tr>
<tr>
<td>ERRORID</td>
<td>The error identifier used in logrec software records associated with this dump on the same system. If multiple dumps were requested, the same ERRORID appears on these dumps.</td>
<td>STRUCTURE(ERRORID)</td>
</tr>
<tr>
<td>ESQA</td>
<td>The extended system queue area</td>
<td>AREA(ESQA)</td>
</tr>
<tr>
<td>FINDAREA</td>
<td>The area currently being searched by the FIND subcommand. This area may be explicitly changed with the EQUATE subcommand and implicitly changed with the FIND subcommand. FINDAREA is defined by the FIND subcommand for all types of dump data sets; it is not limited to SVC dumps.</td>
<td>AREA(FINDAREA)</td>
</tr>
<tr>
<td>FLPA</td>
<td>The fixed link pack area</td>
<td>AREA(FLPA)</td>
</tr>
<tr>
<td>GDA</td>
<td>The global data area</td>
<td>STRUCTURE(GDA)</td>
</tr>
<tr>
<td>IARHVCOM</td>
<td>The high virtual common area</td>
<td>AREA(IARHVCOM)</td>
</tr>
<tr>
<td>IARHVSHR</td>
<td>The high virtual shared area</td>
<td>AREA(IARHVSHR)</td>
</tr>
<tr>
<td>IEAVESLA</td>
<td>The system lock area</td>
<td>STRUCTURE(IEAVESLA)</td>
</tr>
<tr>
<td>IEFJESCTPX</td>
<td>Pageable JESCT extension</td>
<td>STRUCTURE(IEFJESCTPX)</td>
</tr>
<tr>
<td>IEFZB445</td>
<td>Device allocation default table</td>
<td>STRUCTURE(IEFZB445)</td>
</tr>
<tr>
<td>IHSAnnnnn</td>
<td>The interrupt handler save area for address space nnnnn</td>
<td>STRUCTURE(IHSA)</td>
</tr>
<tr>
<td>INCIDENTTOKEN</td>
<td>The incident token for all dumps initiated by a single dump request</td>
<td>STRUCTURE(IEAINTKN)</td>
</tr>
<tr>
<td>ISGGVT</td>
<td>The global resource serialization vector table</td>
<td>STRUCTURE(ISGGVT)</td>
</tr>
</tbody>
</table>
## IPCS Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Associated data</th>
<th>Data type definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISGGVTX</td>
<td>The global resource serialization vector table extension</td>
<td>STRUCTURE(ISGGVTX)</td>
</tr>
<tr>
<td>ISGQHTG</td>
<td>The global resource serialization queue hash table for global resources</td>
<td>STRUCTURE(ISGQHT)</td>
</tr>
<tr>
<td>ISGQHTL</td>
<td>The global resource serialization queue hash table for local (system) resources</td>
<td>STRUCTURE(ISGQHT)</td>
</tr>
<tr>
<td>ISGQHTS</td>
<td>The global resource serialization queue hash table for step resources</td>
<td>STRUCTURE(ISGQHT)</td>
</tr>
<tr>
<td>ISGRSV</td>
<td>The global resource serialization ring status vector</td>
<td>STRUCTURE(ISGRSV)</td>
</tr>
<tr>
<td>ITTCTAB</td>
<td>Component trace anchor block</td>
<td>STRUCTURE(ITTCTAB)</td>
</tr>
<tr>
<td>ITTCTQE name</td>
<td>Component name CTRACE queue entry</td>
<td>STRUCTURE(ITTCTQE)</td>
</tr>
<tr>
<td>LCCAnn</td>
<td>The logical configuration communication area for processor nn</td>
<td>STRUCTURE(LCCA)</td>
</tr>
<tr>
<td>LCCAVT</td>
<td>The LCCA vector table</td>
<td>STRUCTURE(LCCAVT)</td>
</tr>
<tr>
<td>LCCXnn</td>
<td>LCCA extension for cpu nn</td>
<td>STRUCTURE(LCCX)</td>
</tr>
<tr>
<td>LDAnnnnn</td>
<td>LDA for ASID nnnnn</td>
<td>STRUCTURE(LDA)</td>
</tr>
<tr>
<td>LOADMODULE</td>
<td>Load module</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>LPDEpgmname</td>
<td>The link pack directory entry for pgmname</td>
<td>STRUCTURE(LPDE)</td>
</tr>
<tr>
<td>MLPA</td>
<td>The modified link pack area</td>
<td>AREA(MLPA)</td>
</tr>
<tr>
<td>NUCLEUS</td>
<td>The nucleus</td>
<td>AREA(NUCLEUS)</td>
</tr>
<tr>
<td>NVT</td>
<td>The nucleus initialization program (NIP) vector table</td>
<td>STRUCTURE(NVT)</td>
</tr>
<tr>
<td>OSRELEASE</td>
<td>Version, release, and modification level</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>PART</td>
<td>The page address resolution table. This symbol is defined only by the ASMCHECK subcommand.</td>
<td>STRUCTURE(PART)</td>
</tr>
<tr>
<td>PCCAnn</td>
<td>The physical configuration communication area for processor nn</td>
<td>STRUCTURE(PCCA)</td>
</tr>
<tr>
<td>PCCAVT</td>
<td>The PCCA vector table</td>
<td>STRUCTURE(PCCAVT)</td>
</tr>
<tr>
<td>PFT</td>
<td>The system page frame table</td>
<td>STRUCTURE(PFT)</td>
</tr>
<tr>
<td>pgmname</td>
<td>A load module or portion of a load module originating at entry point pgmname</td>
<td>MODULE(pgmname)</td>
</tr>
<tr>
<td>PGTnnnnnaaaaa</td>
<td>The page table for address space nnnnn, segment aaaaa</td>
<td>STRUCTURE(PGTE)</td>
</tr>
<tr>
<td>PLPA</td>
<td>The pageable link pack area</td>
<td>AREA(PLPA)</td>
</tr>
<tr>
<td>PMRNUMBER</td>
<td>Program Management Record (PMR) number</td>
<td>CHARACTER</td>
</tr>
</tbody>
</table>
### IPCS Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Associated data</th>
<th>Data type definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARYSYMPTOMS</td>
<td>The symptoms provided by the program that requested the dump and, possibly, by the program that produced the dump. These are RETAIN symptoms, which are used to search the RETAIN database. If the primary symptom string is longer than 256 bytes, this symbol contains the first 256 bytes of the symptom string.</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>PRIVATE</td>
<td>The private area</td>
<td>AREA(PRIVATE)</td>
</tr>
<tr>
<td>PRIVATEX</td>
<td>The extended private area</td>
<td>AREA(PRIVATEX)</td>
</tr>
<tr>
<td>PSAnn</td>
<td>The prefixed storage area for processor nn</td>
<td>STRUCTURE(PSA)</td>
</tr>
<tr>
<td>PSAVALID</td>
<td>A usable PSA represented in the dump. PSAVALID is obtained by accessing the PSA for the processor on which a stand-alone dump was IPLed and by accessing the PSA at location 0 for other types of dumps.</td>
<td>STRUCTURE(PSA)</td>
</tr>
<tr>
<td>PSW</td>
<td>The program status word at or near the error point in a virtual dump</td>
<td>STRUCTURE(PSW)</td>
</tr>
<tr>
<td>PSWnn</td>
<td>The program status word for CPU nn in a stand-alone dump</td>
<td>STRUCTURE(PSW)</td>
</tr>
<tr>
<td>PVT</td>
<td>The system paging vector table</td>
<td>STRUCTURE(PVT)</td>
</tr>
<tr>
<td>RCE</td>
<td>RSM Control and Enumeration Area</td>
<td>STRUCTURE(RCE)</td>
</tr>
<tr>
<td>REASONCODE</td>
<td>Reason code</td>
<td>STRUCTURE(SDWACRC)</td>
</tr>
<tr>
<td>REGACC</td>
<td>Access registers at or near the error point in a virtual dump</td>
<td>STRUCTURE(REGACC)</td>
</tr>
<tr>
<td>REGACCnn</td>
<td>Access registers for CPU nn in a stand-alone dump</td>
<td>STRUCTURE(REGACC)</td>
</tr>
<tr>
<td>REGCTL</td>
<td>Control registers at or near the error point in a virtual dump</td>
<td>STRUCTURE(REGCTL)</td>
</tr>
<tr>
<td>REGCTLnn</td>
<td>Control registers for CPU nn in a stand-alone dump</td>
<td>STRUCTURE(REGCTL)</td>
</tr>
<tr>
<td>REGFLT</td>
<td>Floating point registers at or near the error point in a virtual dump</td>
<td>STRUCTURE(REGFLT)</td>
</tr>
<tr>
<td>REGFLTnn</td>
<td>Floating point registers for CPU nn in a stand-alone dump</td>
<td>STRUCTURE(REGFLT)</td>
</tr>
<tr>
<td>REGFPC</td>
<td>Floating point control register at or near the error point in an unformatted dump</td>
<td>STRUCTURE(REGFPC)</td>
</tr>
<tr>
<td>REGFPCnn</td>
<td>Floating point control register for CPU nn in a stand-alone dump</td>
<td>STRUCTURE(REGFPC)</td>
</tr>
<tr>
<td>REGGEN</td>
<td>General purpose registers at or near the error point in a virtual dump</td>
<td>STRUCTURE(REGGEN)</td>
</tr>
<tr>
<td>REGGENnn</td>
<td>General purpose registers for CPU nn in a stand-alone dump</td>
<td>STRUCTURE(REGGEN)</td>
</tr>
<tr>
<td>REGG64H</td>
<td>High-order halves (bits 0-31) of 64-bit general registers</td>
<td>STRUCTURE(REGG64H)</td>
</tr>
<tr>
<td>REGG64Hnn</td>
<td>High-order halves (bits 0-31) of 64-bit general registers for cpu nn</td>
<td>STRUCTURE(REGG64H)</td>
</tr>
</tbody>
</table>
### IPCS Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Associated data</th>
<th>Data type definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>REG32CTL*</td>
<td>32-bit control registers at or near the error point in a virtual dump.</td>
<td>STRUCTURE(REGCTL32)</td>
</tr>
<tr>
<td>REG32CTLnn*</td>
<td>32-bit control registers for CPU nn in a stand-alone dump.</td>
<td>STRUCTURE(REGCTL32)</td>
</tr>
<tr>
<td>REG32GEN*</td>
<td>32-bit general purpose registers at or near the error point in a virtual dump.</td>
<td>STRUCTURE(REGGEN32)</td>
</tr>
<tr>
<td>REG32GENnn*</td>
<td>32-bit general purpose registers for CPU nn in a stand-alone dump.</td>
<td>STRUCTURE(REGGEN32)</td>
</tr>
<tr>
<td>REG64CTL*</td>
<td>64-bit control registers at or near the error point in a virtual dump.</td>
<td>STRUCTURE(REGCTL64)</td>
</tr>
<tr>
<td>REG64CTLnn*</td>
<td>64-bit control registers control registers for CPU nn in a stand-alone dump.</td>
<td>STRUCTURE(REGCTL64)</td>
</tr>
<tr>
<td>REG64GEN*</td>
<td>64-bit general purpose registers at or near the error point in a virtual dump.</td>
<td>STRUCTURE(REGGEN64)</td>
</tr>
<tr>
<td>REG64GENnn*</td>
<td>64-bit general purpose registers for CPU nn in a stand-alone dump.</td>
<td>STRUCTURE(REGGEN64)</td>
</tr>
<tr>
<td>REMOTEDUMP</td>
<td>Indicator that dumps on other systems in the sysplex were requested:</td>
<td>CHARACTER</td>
</tr>
<tr>
<td></td>
<td>• The request for this dump also requested dumps on other systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This is a dump requested by another system</td>
<td></td>
</tr>
<tr>
<td>RONUCLEUS</td>
<td>The read-only portion of the nucleus</td>
<td>AREA(RONUCLEUS)</td>
</tr>
<tr>
<td>RTCT</td>
<td>The recovery termination control table</td>
<td>STRUCTURE(RTCT)</td>
</tr>
<tr>
<td>SAnnnnn</td>
<td>Save area nnnnn in an entry point or 72-byte save area trace</td>
<td>STRUCTURE(REGSAVE)</td>
</tr>
<tr>
<td>SCCB</td>
<td>The service call control block</td>
<td>STRUCTURE(SCCB)</td>
</tr>
<tr>
<td>SCVT</td>
<td>The secondary CVT</td>
<td>STRUCTURE(SCVT)</td>
</tr>
<tr>
<td>SDWAHDR</td>
<td>The SDWA saved in a dump header record</td>
<td>STRUCTURE(SDWAHDR)</td>
</tr>
<tr>
<td>SECONDARYSYMPTOMS</td>
<td>The symptoms provided by IPCS subcommands used to analyze the dump. These are RETAIN symptoms, which are used to search the RETAIN database. If the secondary symptom string is longer than 256 bytes, this symbol contains the first 256 bytes of the symptom string.</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>SGTnnnnn</td>
<td>The segment table for address space nnnnn</td>
<td>STRUCTURE(SGTE)</td>
</tr>
<tr>
<td>SLIPTRAP</td>
<td>The SLIP command that requested the dump. If the actual command is longer than 256 bytes, it is truncated.</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>SRBPT</td>
<td>SRB Promotion Table</td>
<td>STRUCTURE(SRBPT)</td>
</tr>
<tr>
<td>SVT</td>
<td>Supervisor Vector Table</td>
<td>STRUCTURE(SVT)</td>
</tr>
<tr>
<td>SVTX</td>
<td>SVT Extension</td>
<td>STRUCTURE(SVTX)</td>
</tr>
<tr>
<td>Symbol</td>
<td>Associated data</td>
<td>Data type definition</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>TCBCURRENT</td>
<td>The current TCB. TCBCURRENT is only meaningful in context of a system-detected problem that results in a SYSMDUMP or system dump being recorded. The concept doesn't work when the system operator causes a SADUMP to be written or uses the DUMP command nor does it work with dumps requested by programs that are not running under a TCB.</td>
<td>STRUCTURE(TCB)</td>
</tr>
<tr>
<td>TCBnnnnnaaaa</td>
<td>The task control block for address space nnnnn, in position aaaaa in the priority queue&lt;br&gt;The highest priority TCB in address space 1 is TCB00001AAAAA; the next TCB on the queue is TCB00001AAAAB, ...&lt;br&gt;The last 2 characters in this name are alphabetic and range from AAAAA through AZZZZ, BAAAA, ... BZZZZ, ...</td>
<td>STRUCTURE(TCB)</td>
</tr>
<tr>
<td>TITLE</td>
<td>The dump title, which is contained in the dump header. TITLE is defined only during dump initialization for SVC dumps. IPCS does not support dynamic location of the title if the symbol is DROPPED from the symbol table.</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>UCBddddd</td>
<td>The unit control block for device dddd. The dddd designates the device number in hexadecimal.</td>
<td>STRUCTURE(UCB)</td>
</tr>
<tr>
<td>UCM</td>
<td>The unit control module</td>
<td>STRUCTURE(UCM)</td>
</tr>
<tr>
<td>X</td>
<td>The “current address” in a dump. This symbol is defined by most IPCS subcommands in all types of dumps supported by IPCS.</td>
<td>STRUCTURE(UCM)</td>
</tr>
<tr>
<td>XLpgmname</td>
<td>An extent list for entry point pgmname</td>
<td>STRUCTURE(XTLST)</td>
</tr>
<tr>
<td>Znnnnn</td>
<td>A dump location that is added to the pointer stack as nnnnn, whenever executing the STACK subcommand, the STACK primary command, or the IPCS dialog. The suffix nnnnn designates a sequenced number.</td>
<td>STRUCTURE(XTLST)</td>
</tr>
</tbody>
</table>

* Provided to support migration from 32-bit to 64-bit values.
• The REG32 symbols describe 64 bytes of data. For dumps of z/Architecture mode systems, bits 0-31 of 64-bit registers are eliminated.
• The REG64 symbols describe 128 bytes of data. For dumps of ESA mode systems, the 32-bit registers are extended with leading zeros.
IPCS Symbols
IPCS special symbols for system control blocks

IPCS special symbols are summarized in the following chart. The variables in the chart are:

- **a**: Represents 1 uppercase letter, A through Z
- **n**: Represents 1 decimal digit
- **x or d**: Represents 1 EBCDIC-hexadecimal digit, 1 decimal digit from 0 through 9, or 1 uppercase letter, A through F

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Symbol description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCBnnnnn</td>
<td>ASCB1</td>
<td>ASCB99999</td>
<td>The address space control block for address space nnnnn.</td>
</tr>
<tr>
<td>ASTnnnn</td>
<td>AST0</td>
<td>AST9999</td>
<td>The address space second table corresponding to ENTRY(nnnn) in the address space first table. (An equivalent definition is that this is the address space second table for system address spaces from nnnn<em>16 through nnnn</em>16+15.)</td>
</tr>
<tr>
<td>ASTEnnnnn</td>
<td>ASTE1</td>
<td>ASTE9999</td>
<td>The address space second table entry for address space nnnnn.</td>
</tr>
<tr>
<td>ASXBnnnnn</td>
<td>ASXB1</td>
<td>ASCB99999</td>
<td>The address space extension block for address space nnnnn.</td>
</tr>
<tr>
<td>IHSAnnnnn</td>
<td>IHSA1</td>
<td>IHSA99999</td>
<td>The interrupt handler save area for address space nnnnn.</td>
</tr>
<tr>
<td>LCCAnn</td>
<td>LCCA0</td>
<td>LCCA99</td>
<td>The logical configuration communication area for processor nn.</td>
</tr>
<tr>
<td>PCCAnn</td>
<td>PCCA0</td>
<td>PCCA99</td>
<td>The physical configuration communication area for processor nn.</td>
</tr>
<tr>
<td>PGTnnnnnnaaaaa</td>
<td>PGT1A</td>
<td>PGT99999ZZZZZ</td>
<td>The page table for segment aaaaa (base 26 number) in address space nnnnn.</td>
</tr>
<tr>
<td>PSAnn</td>
<td>PSA0</td>
<td>PSA99</td>
<td>The prefixed storage area for processor nn.</td>
</tr>
<tr>
<td>PSWnnn</td>
<td>PSW0</td>
<td>PSW99</td>
<td>The program status word for processor nn.</td>
</tr>
<tr>
<td>REGACnnn</td>
<td>REGACC0</td>
<td>REGACC99</td>
<td>The access registers for processor nn.</td>
</tr>
<tr>
<td>REGCTLnn</td>
<td>REGCTL0</td>
<td>REGCTL99</td>
<td>The control registers for processor nn.</td>
</tr>
<tr>
<td>REGFLTnn</td>
<td>REGFLT0</td>
<td>REGFLT99</td>
<td>The floating point registers for processor nn.</td>
</tr>
<tr>
<td>REGFPnnn</td>
<td>REGFPC0</td>
<td>REGFPC99</td>
<td>The floating point control register for processor nn.</td>
</tr>
<tr>
<td>REGGENnnn</td>
<td>REGGEN0</td>
<td>REGGEN99</td>
<td>The general purpose registers for processor nn.</td>
</tr>
<tr>
<td>REG32CTLnn</td>
<td>REG32CTL0</td>
<td>REG32CTL99</td>
<td>The 32-bit control registers for processor nn.</td>
</tr>
<tr>
<td>REG32GENnn</td>
<td>REG32GEN0</td>
<td>REG32GEN99</td>
<td>The 32-bit general purpose registers for processor nn.</td>
</tr>
<tr>
<td>REG64CTLnn</td>
<td>REG64CTL0</td>
<td>REG64CTL99</td>
<td>The 64-bit control registers for processor nn.</td>
</tr>
<tr>
<td>REG64GENnn</td>
<td>REG64GEN0</td>
<td>REG64GEN99</td>
<td>The 64-bit general purpose registers for processor nn.</td>
</tr>
<tr>
<td>SGTnnnnn</td>
<td>SGT1</td>
<td>SGT99999</td>
<td>The segment table for address space nnnnn.</td>
</tr>
<tr>
<td>TCBnnnnnnaaaaa</td>
<td>TCB1A</td>
<td>TCB99999ZZZZZ</td>
<td>The task control block in position aaaaa (base 26 number) on the priority chain in address space nnnnn.</td>
</tr>
<tr>
<td>UCBddd</td>
<td>UCB0</td>
<td>UCBFFFF</td>
<td>The unit control block for the device number dddd.</td>
</tr>
</tbody>
</table>
### Special symbols

**Table B-1. IPCS special symbols (continued)**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Symbol description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Znnnnn</td>
<td>Z1</td>
<td>999999</td>
<td>A dump location that is added to the pointer stack as nnnnn, whenever executing the STACK primary command, the STACK subcommand, or the IPCS dialog. The suffix nnnnn designates a sequenced number.</td>
</tr>
</tbody>
</table>
Control blocks and data areas scanned, mapped, and formatted

This appendix lists the control blocks and data areas in system dumps that the CBFORMAT subcommand can scan, create a storage map entry for, or format.

For some control blocks or data areas, IPCS creates a storage map entry but does not scan the block or area.

Notes referenced in the right column are at the end of the chart.

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>AFT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>AFTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>AIA</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ALE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>AMDCPMAP</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ASCB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ASEI</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ASMHD</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ASMVT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ASPCT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ASSB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>AST</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ASTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ASVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ASXB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>CACHE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CDE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CDEMAJOR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CDEMINOR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CLTE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CQE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>CSD</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CSRC4POL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CSRCPOOL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CVT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CVTVSTGX</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>CVTXTNT2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1988, 2012
## Control Blocks and Data Areas

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCB</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>DEIB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DEIE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DIB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DIBX</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DOA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DOM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DSAB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DSNT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>EED</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>FRRS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>GDA</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>GEPL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>HED</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>HTBL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IATYDAT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYDMC</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYDSS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYFCT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYIOP</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYMEME</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYMEMH</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYMPC</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYMPC</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYOSD</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYOSED</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYOSEF</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYOSEV</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYRQCI</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYRQCM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYRQFX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYRQGM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYRQMD</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYRQOS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYSEL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYSPB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYSVT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IATYTVT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>10</td>
</tr>
<tr>
<td>IEFJESCT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFJESLA</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
## Control Blocks and Data Areas

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEFJESCTPX</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>IEFJSCVT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFJSSVT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFOWTR</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFZAGT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFZB41C</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFZB445</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFZIGDE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IEFZTSRA</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IHSA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>IORB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IOSB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>IRB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>6</td>
</tr>
<tr>
<td>ISGGVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>9</td>
</tr>
<tr>
<td>ISGGVTX</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>9</td>
</tr>
<tr>
<td>ISGQCB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>9</td>
</tr>
<tr>
<td>ISGQEL</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>9</td>
</tr>
<tr>
<td>ISGQHT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>9</td>
</tr>
<tr>
<td>ISGRPT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ISGRSV</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>9</td>
</tr>
<tr>
<td>ISGSAHT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>DXCYERE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXYEVE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXYWRE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DGAARTE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXDIRCT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXGINV</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXIPSTK</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>14</td>
</tr>
<tr>
<td>DXLBCB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXLCB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXLCBV</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXLCBC</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXLSAB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXPCNTL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXQUE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DXSTRCB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>JCT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>JCTX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>JESCT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
## Control Blocks and Data Areas

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFCB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>JFCBE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>JFCBX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>JQE</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>JSCB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LCCA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>LCCAVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LGE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LGVT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LGVTE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LLE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LPDE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>3</td>
</tr>
<tr>
<td>LPDEFINAL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>3</td>
</tr>
<tr>
<td>LPDEMAJOR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>3</td>
</tr>
<tr>
<td>LPDEMINOR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>3</td>
</tr>
<tr>
<td>LPDENULL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>3</td>
</tr>
<tr>
<td>LS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LSE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LSEH</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LSET</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LSSD</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>LSSG</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>MEPL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>MGCRE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>NVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ORE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>OUCB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OUSB</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>OUXB</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PART</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>PARTE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>PAT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>PCB</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PCCA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>PCCAVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PFT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>4</td>
</tr>
<tr>
<td>PFTE</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PGT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>5, 12</td>
</tr>
</tbody>
</table>
### Control Blocks and Data Areas

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PPD</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PRB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PSA</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>PSW</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>PVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PXT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>RB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>RCE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>RDCM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>REGACC</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>REGCTRL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>REGFLT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>REGGEN</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>REGS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>REGSAVIM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>RMCT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>RQE</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>RSMHD</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>RTCT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>RTM2WA</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>RT1W</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>SART</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SARTE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SAT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SBC</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SCB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SCCB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SCCW</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SCFS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SCT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>SCTX</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>SCVT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>SDCT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SDUMP</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SDWA</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SDWAHDR</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SEPL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SGT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>2, 12</td>
</tr>
</tbody>
</table>
Control Blocks and Data Areas

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>SIOT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>15</td>
</tr>
<tr>
<td>SIRB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>6</td>
</tr>
<tr>
<td>SPD</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>SPQE</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>SRB</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SSRB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>STCB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>STKE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SUPVT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SVRB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>6</td>
</tr>
<tr>
<td>SVT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SVTX</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>SXT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>TCB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>12</td>
</tr>
<tr>
<td>TDCM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>TIA</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>TIO</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>11</td>
</tr>
<tr>
<td>TIOE</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>11</td>
</tr>
<tr>
<td>TIRB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>6</td>
</tr>
<tr>
<td>TODCLOCK</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>17</td>
</tr>
<tr>
<td>TODECNULL</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>18</td>
</tr>
<tr>
<td>TODEC4</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>19</td>
</tr>
<tr>
<td>TQE</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>TSB</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>UCB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>12</td>
</tr>
<tr>
<td>UCBCTC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCBDA</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCBEXT</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>UCBGFX</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCBTAPE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCBTP</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCBUR</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCB3270</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>UCM</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>UCM3270</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>8</td>
</tr>
<tr>
<td>VCOM</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>VF</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>VSWK</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
## Control Blocks and Data Areas

<table>
<thead>
<tr>
<th>Control block or data area</th>
<th>Scanned</th>
<th>Storage map entry</th>
<th>Formatted</th>
<th>Notes (following the table)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>WEB</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>WEE</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>WQE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>WSAVTC</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>WSAVTG</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>WSMAP</td>
<td></td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>XCFSTACK</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>13</td>
</tr>
<tr>
<td>XESSTACK</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>13</td>
</tr>
<tr>
<td>XSB</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>XTLST</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Notes:

1. AFTE is validated as if it was specified as AFT. AFT is stored in the symbol table and storage map.
2. ASTE is validated as if it were specified as AST. AST is stored in the symbol table and storage map.
3. CDEMAJOR, CDEMINOR, LPDE, LPDEMAJOR, and LPDEMINOR are validated as if they were specified as CDE. The correct structure type is stored in the symbol table and storage map.
4. PFT is validated as if it were specified as PFTE. PFTE is stored in the symbol table and storage map.
5. PGT is validated as if it were specified as PGTE. PGTE is stored in the symbol table and storage map.
6. These control blocks are validated as if they were specified as RB. The correct structure type is stored in the symbol table and storage map. IRB, PRB, SIRB, and TIRB are validated as if they were specified as LPDE. The correct structure type is stored in the symbol table and storage map.
7. SGT is validated as if it were specified as SGTE. SGTE is stored in the symbol table and storage map.
8. UCBCTC, UCBDA, UCBGFX, UCBTAPE, UCBTP, UCBUR, and UCB3270 are validated as if they were specified as UCB. The correct structure type is stored in the symbol table and storage map.
9. ISGGVT, ISGGVTX, ISGQCB, ISGQEL, ISGQHT, ISRPT, ISGRSV, and ISGSAHT are referenced, without the prefix ISG, in z/OS MVS Data Areas in z/OS Internet Library at http://www.ibm.com/systems/z/os/zos/bkserv.
   For example, ISGGVT is listed under GVT.
10. These JES3 control blocks can be formatted by issuing the CBFORMAT subcommand with the address of the requested control block. For example, using the IPCS dialog BROWSE option or a CLIST to determine the address of the control block, enter CBFORMAT 9FD308 STRUCTURE(IATYSEL). Only the IATYSVT allows you to use the symbol name in the subcommand, CBFORMAT IATYSVT STRUCTURE(IATYSVT).
11. TIOT formats the entire task input output table (TIOT). TIOTE formats a single TIOT entry. If your system has DFP Version 3.2 with APARs OY29785
Control Blocks and Data Areas

...and OY29786 installed, and DB2 Version 2.2 with APAR PL59415 installed, you must use TIOTE to format TIOT entries. TIOT will not find all TIOT entries. Otherwise, you can use either TIOT or TIOTE.

12. These symbols have a special naming convention in IPCS. See “IPCS special symbols for system control blocks” on page B-1.

13. XCFSTACK and XESSTACK are dynamic area stack structures that contain information that is internal to XES and XCF.

14. IXGIPSTK is a dynamic area stack structure that contains information internal to system logger. For example, using the IPCS dialog BROWSE option or a CLIST enter CBFORMAT nnnnnnn FORMAT(IXGIPSTK), where nnnnnnn is the address of a system logger dynamic stack.

15. These scheduler work area (SWA) control blocks can be formatted using the CBFORMAT command or subcommand. Specify the address of the X'10' byte SWA prefix that precedes the control block rather than the address of the actual SWA block itself.


17. See Principles of Operation, topic TOD-clock. This is IPCS support for the 8-byte value store by the STCK instruction.

18. Same as TODCLOCK except that zero values are treated as a special case that implies the absence of a valid TOD-clock value in a data area field.

19. TODCLOCK LENGTH(4), bits 0-31 of a TOD-clock value are saved in some data areas.
Print dump to IPCS conversion summary

The following table describes the control statements or functions formerly available through the print dump (AMDPRDMP) service aid, and points to the equivalent IPCS subcommand or function.

Table D-1. AMDPRDMP - IPCS conversion summary

<table>
<thead>
<tr>
<th>Print dump control statement or function</th>
<th>IPCS equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMDATA control statement</td>
<td>ASMDATA verb exit</td>
</tr>
<tr>
<td></td>
<td>Use VERBEXIT ASMDATA to format certain ASM control blocks.</td>
</tr>
<tr>
<td></td>
<td>Use VERBEXIT ASMDATA subcommand — format auxiliary storage manager data” on page 5-300</td>
</tr>
<tr>
<td></td>
<td>See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT ASMDATA subcommand — format auxiliary storage manager data” on page 5-300.</td>
</tr>
<tr>
<td>AVMDATA control statement</td>
<td>AVMDATA verb exit</td>
</tr>
<tr>
<td></td>
<td>Use VERBEXIT AVMDATA to format the contents of accessible availability manager control blocks.</td>
</tr>
<tr>
<td></td>
<td>See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT AVMDATA subcommand — format availability manager data” on page 5-301.</td>
</tr>
<tr>
<td>Copy and clear a source SYS1.DUMP data set</td>
<td>COPYDUMP</td>
</tr>
<tr>
<td></td>
<td>Use COPYDUMP CLEAR to clear a SYS1.DUMP data set after copying.</td>
</tr>
<tr>
<td></td>
<td>See “COPYDUMP subcommand — copy dump data” on page 5-52.</td>
</tr>
<tr>
<td></td>
<td>For sample JCL to print, offload, and clear a dump, see the z/OS MVS IPCS User’s Guide.</td>
</tr>
<tr>
<td>CPUDATA control statement</td>
<td>STATUS DATA subcommand</td>
</tr>
<tr>
<td></td>
<td>Use the STATUS subcommand to gather processor-related debugging information.</td>
</tr>
<tr>
<td></td>
<td>See “STATUS subcommand — describe system status” on page 5-243.</td>
</tr>
<tr>
<td>CVT control statement</td>
<td>EQUATE subcommand</td>
</tr>
<tr>
<td></td>
<td>Use EQUATE CVT address when you want to associate the address of the CVT control block with a symbol.</td>
</tr>
<tr>
<td></td>
<td>See “EQUATE subcommand — create a symbol” on page 5-92.</td>
</tr>
<tr>
<td>CVTMAP control statement</td>
<td>CBFORMAT subcommand</td>
</tr>
<tr>
<td></td>
<td>Use the CBFORMAT subcommand to display the contents of the CVT control block.</td>
</tr>
<tr>
<td></td>
<td>See “CBFORMAT subcommand — format a control block” on page 5-29.</td>
</tr>
<tr>
<td>DAEDATA control statement</td>
<td>DAEDATA verb exit</td>
</tr>
<tr>
<td></td>
<td>Use VERBEXIT DAEDATA to format DAE dump data.</td>
</tr>
<tr>
<td></td>
<td>See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT DAEDATA subcommand — format dump analysis and elimination data” on page 5-302.</td>
</tr>
<tr>
<td>Dumped storage summary</td>
<td>LISTDUMP subcommand</td>
</tr>
<tr>
<td></td>
<td>Use the LISTDUMP subcommand to provide a summary of the storage in one or more dumps.</td>
</tr>
<tr>
<td></td>
<td>LISTDUMP is described under “LISTDUMP subcommand — list dumps in dump directory” on page 5-153.</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1988, 2012

D-1
### Print Dump to IPCS Conversion

Table D-1. AMDPRDMP - IPCS conversion summary (continued)

<table>
<thead>
<tr>
<th>Print dump control statement or function</th>
<th>IPCS equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDIT control statement</td>
<td>GTFTRACE subcommand</td>
</tr>
</tbody>
</table>
|                                          | • Use the GTFTRACE subcommand to format GTF trace records in a dump or in a separate GTF trace file. These incompatibilities are a result of the conversion:  
|                                          |   • Equal signs in print dump are replaced by parentheses in IPCS.  
|                                          |   • Standard IPCS data set and routing capabilities are available.  
|                                          |   • START and STOP times will now also apply to blocks of records in dumps, and can be specified in GMT or LOCAL time.  
|                                          | • See “GTFTRACE subcommand — format GTF trace records” on page 5-126 |
| END control statement                    | END subcommand   |
|                                          | • Use the END subcommand to end IPCS sessions, subcommand processing, and CLIST processing.  
|                                          | • See “END subcommand — end an IPCS session” on page 5-90 |
| FORMAT control statement                 | SUMMARY subcommand |
|                                          | • Use the SUMMARY subcommand with the FORMAT parameter to format major control blocks. This report will include the RTM2 work area(s) in the dump.  
|                                          | • See “SUMMARY subcommand — summarize control block fields” on page 5-263 |
| Format the SDWA                          | STATUS FAILDATA subcommand |
|                                          | • Use the STATUS FAILDATA subcommand to format the SDWA in the dump header. |
| GO control statement                     | Run a CLIST of IPCS subcommands |
|                                          | • Use a CLIST to run a series of predefined IPCS subcommands against a source data set. See the subcommand descriptions to help you determine which subcommands you want to run.  
|                                          | • See Chapter 8, “IPCS batch mode,” on page 8-1 |
| GRSTRACE control statement (also QCBTRACE or Q) | VERBEXIT GRSTRACE subcommand |
|                                          | • Use the GRSTRACE or QCBTRACE or Q verb names on the VERBEXIT subcommand to format the address of the major control blocks associated with global resource serialization, the contents of control blocks on the global resources queue, and latch statistics.  
|                                          | • See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT GRSTRACE subcommand — format Global Resource Serialization data” on page 5-304 |
| IMSDUMP control statement                | VERBEXIT IMSDUMP subcommand |
|                                          | • Use the IMSDUMP verb name on the VERBEXIT subcommand to format the contents of Information Management System (IMS) control blocks in the dump.  
|                                          | • For more information about IMS DUMP formatting, see the following topics in the IMS Version 11 information center: [http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp)  
|                                          |   – Invoking the IMS Offline Dump Formatter  
|                                          |   – IMS Dump Formatter menus |
| INDEX DD statement                       | IPCSTOC data set |
|                                          | • Allocate an IPCSTOC data set to capture the entries made by the IPCS TOC service. The service makes entries to this data set whenever a subcommand is issued with the PRINT parameter.  
|                                          | • See the topic “Print and table of contents data sets” in z/OS MVS IPCS User’s Guide |

D-2 z/OS V1R13.0 MVS IPCS Commands
<table>
<thead>
<tr>
<th>Print dump control statement or function</th>
<th>IPCS equivalent</th>
</tr>
</thead>
</table>
| IOSDATA control statement                | IOSCHECK subcommand  
  • Use the IOSCHECK subcommand to format the contents of specific I/O supervisor (IOS) control blocks and related diagnostic information.  
  • See “IOSCHECK subcommand — format I/O supervisor data” on page 5-139. |
| IRLM control statement                   | VERBEXIT IRLM subcommand  
  • Use the IRLM verb name on the VERBEXIT subcommand to format IMS resource lock manager (IRLM) control blocks in a dump.  
  • Use the IRLM SDUMP system services described in the IMS Version 11 information center: [http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp) |
| JES2 control statement                   | VERBEXIT HASMFMTM subcommand  
  • Use the VERBEXIT HASMFMTM subcommand to format control blocks associated with JES2.  
  • JES2 dump formatting is described in [z/OS JES2 Diagnosis](http://publib.boulder.ibm.com/infocenter/dzichelp/). |
| JES3 control statement                   | VERBEXIT JES3 subcommand  
  • Use the VERBEXIT JES3 subcommand to format control blocks associated with JES3.  
  • JES3 dump formatting is described in [z/OS JES3 Diagnosis](http://publib.boulder.ibm.com/infocenter/dzichelp/). |
| LOGDATA control statement                | VERBEXIT LOGDATA subcommand  
  • Use the VERBEXIT LOGDATA subcommand to format the in-storage LOGREC buffer records.  
  • See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT LOGDATA subcommand — format logrec buffer records” on page 5-317. |
| LPAMAP control statement                 | LPAMAP subcommand  
  • Use the LPAMAP subcommand to format information about the pageable link pack area (PLPA) and active LPA.  
  • See “LPAMAP subcommand — list link pack area entry points” on page 5-174. |
| MTRACE control statement                 | VERBEXIT MTRACE subcommand  
  • Use the MTRACE verb name on the VERBEXIT subcommand to format the master trace table.  
  • See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT MTRACE subcommand — format master trace entries” on page 5-320. |
| NEWDUMP control statement                | SETDEF subcommand  
  • Use the SETDEF parameters for data set source specification to alter the source you want to use for dump processing.  
  • See “SETDEF subcommand — set defaults” on page 5-232. |
| NEWTAPE control statement                | VERBEXIT NUCMAP subcommand  
  • Use the NUCMAP verb name on the VERBEXIT subcommand to format the modules in the nucleus at the time of the dump.  
  • See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT NUCMAP subcommand — map modules in the nucleus” on page 5-322. |
| ONGO control statement                   | Create a CLIST of IPCS subcommands  
  • Create a CLIST to process a predefined series of IPCS subcommands.  
  • See Chapter 8, “IPCS batch mode,” on page 8-1. Refer also to the subcommand descriptions to help you determine which subcommands you want to process. |
### Print Dump to IPCS Conversion

**Table D-1. AMDPRDMP - IPCS conversion summary (continued)**

<table>
<thead>
<tr>
<th>Print dump control statement or function</th>
<th>IPCS equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PRINT CSA, SQA, NUCLEUS control statements</strong></td>
<td>LIST subcommand and CLISTS</td>
</tr>
<tr>
<td>- Use these symbols on the LIST subcommand to format and display information for the CSA, SQA, and NUCLEUS:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CSA, ECSA</td>
</tr>
<tr>
<td></td>
<td>CSA storage above and below 16 megabytes.</td>
</tr>
<tr>
<td></td>
<td>SQA, ESQA</td>
</tr>
<tr>
<td></td>
<td>SQA storage above and below 16 megabytes.</td>
</tr>
<tr>
<td></td>
<td>NUCLEUS, ENUCLEUS, RONUCLEUS, DATOFFNUCLEUS</td>
</tr>
<tr>
<td></td>
<td>Read/write nucleus storage below and above 16 megabytes; read-only nucleus storage; the DAT-OFF portion of the nucleus.</td>
</tr>
<tr>
<td></td>
<td>PRIVATE, PRIVATEX</td>
</tr>
<tr>
<td></td>
<td>Private area below and above 16 megabytes.</td>
</tr>
<tr>
<td></td>
<td>- Use the BLSCPCSA, BLSCPNUC, BLSCPRIV, and BLSCPSQA CLISTs to print information from these system areas.</td>
</tr>
<tr>
<td></td>
<td>- See “LIST subcommand — display storage” on page 5-151 for information about the LIST subcommand.</td>
</tr>
<tr>
<td></td>
<td>- See Chapter 8, “IPCS batch mode,” on page 8-1 for a description of these CLISTs.</td>
</tr>
<tr>
<td><strong>PRINT STORAGE, REAL control statements</strong></td>
<td>LIST subcommand</td>
</tr>
<tr>
<td>- Use the LIST subcommand to display storage contents.</td>
<td></td>
</tr>
<tr>
<td>- See “LIST subcommand — display storage” on page 5-151.</td>
<td></td>
</tr>
<tr>
<td><strong>PRINT CURRENTJOBNAME control statements</strong></td>
<td>BLSCPRNT CLIST</td>
</tr>
<tr>
<td>- Use the BLSCPRNT CLIST to gather address space selection information and generate storage map entries defining the address spaces in a dump. To do this, BLSCPRNT runs several IPCS subcommands. Among them are: EVALMAP, LIST, LISTMAP, SELECT, and SUMMARY.</td>
<td></td>
</tr>
<tr>
<td>- See “BLSCPRNT CLIST — print a dump” on page 7-11 for a description of BLSCPRNT and its operands.</td>
<td></td>
</tr>
<tr>
<td><strong>Q or QCBTRACE control statements</strong></td>
<td>See GRSTRACE control statement.</td>
</tr>
<tr>
<td><strong>RSMDATA control statement</strong></td>
<td>RSMDATA subcommand</td>
</tr>
<tr>
<td>- Use the RSMDATA subcommand to format information about the real storage management component.</td>
<td></td>
</tr>
<tr>
<td>- See “RSMDATA subcommand — analyze real storage manager data” on page 5-202.</td>
<td></td>
</tr>
<tr>
<td><strong>SADMPMSG control statement</strong></td>
<td>VERBEXIT SADMPMSG subcommand</td>
</tr>
<tr>
<td>- Use the SADMPMSG verb name of the VERBEXIT subcommand to format the SADMP execution-time virtual storage dump message log.</td>
<td></td>
</tr>
<tr>
<td>- See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-290 and “VERBEXIT SADMPMSG subcommand — format stand-alone dump message log” on page 5-320.</td>
<td></td>
</tr>
<tr>
<td><strong>SEGTAB control statement</strong></td>
<td>EQUATE subcommand</td>
</tr>
<tr>
<td>- Use the SGT symbol with an address on the EQUATE subcommand to associate the segment table with its address and storage attributes.</td>
<td></td>
</tr>
<tr>
<td>- See “EQUATE subcommand — create a symbol” on page 5-92.</td>
<td></td>
</tr>
</tbody>
</table>
Table D-1. AMDPRDMP - IPCS conversion summary (continued)

<table>
<thead>
<tr>
<th>Print dump control statement or function</th>
<th>IPCS equivalent</th>
</tr>
</thead>
</table>
| SMSDATA control statement | VERBEXIT SMSDATA subcommand  
• Use the SMSDATA verb name on the VERBEXIT subcommand to format storage management subsystem (SMS) control blocks in a dump.  
• SMSDATA is described in *MVS/DFP Diagnosis Reference*. |
| SRMDATA control statement | VERBEXIT SRMDATA subcommand  
• Use the SRMDATA verb name on the VERBEXIT subcommand to format certain control blocks associated with the system resources manager component.  
• See "VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine" on page 5-296 and "VERBEXIT SRMDATA subcommand — format System Resource Manager data" on page 5-326 |
| SUMDUMP control statement | VERBEXIT SUMDUMP subcommand  
• Use the SUMDUMP verb name on the VERBEXIT subcommand to format the summary dump data provided by SVC dumps.  
• See “VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine” on page 5-296 and “VERBEXIT SUMMARY subcommand — format SVC summary dump data” on page 5-327 |
| SUMMARY control statement | SUMMARY subcommand  
• Use the SUMMARY subcommand to display or print dump data associated with an address space.  
• Note: SUMMARY will not produce the dumped storage summary that the SUMMARY JOB SUMMARY control statement produces. If you want to do this, use the LISTDUMP subcommand; see “LISTDUMP subcommand — list dumps in dump directory” on page 5-153.  
• See “SUMMARY subcommand — summarize control block fields” on page 5-263 |
| TITLE control statement | OPEN subcommand  
• Use the TITLE parameter on the OPEN subcommand to specify a title you want to appear on each page of the IPCS print file.  
• See “OPEN subcommand — prepare resources for use by IPCS” on page 5-190 |
| TRACE control statement | SYSTRACE subcommand  
• Use the SYSTRACE subcommand to format trace entries for all address spaces.  
• See “SYSTRACE subcommand — format system trace entries” on page 5-277 |
| TSODATA control statement | VERBEXIT TSODATA subcommand  
• Use the TSODATA verb name on the VERBEXIT subcommand to format information about selected TSO/E address spaces.  
• TSODATA is described in *TSO/E V2 Diagnosis: Guide and Index*. |
| VSMCDATA control statement | VERBEXIT VSMCDATA subcommand  
• Use the VSMCDATA verb name on the VERBEXIT subcommand to format and print the contents of certain VSM control blocks.  
• See "VERBEXIT subcommand — run an installation-supplied or an IBM-supplied verb exit routine" on page 5-296 and "VERBEXIT VSMCDATA subcommand — format virtual storage management data" on page 5-325 |
| VTAMMAP control statement | VERBEXIT VTAMMAP subcommand  
• Use the VTAMMAP verb name on the VERBEXIT subcommand to format VTAM control blocks helpful to VTAM problem determination.  
• See *VTAM Diagnosis*. |
Print Dump to IPCS Conversion
Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you may view the information through the z/OS Internet Library web site or the z/OS Information Center. If you continue to experience problems, send an email to mhvrcfs@us.ibm.com or write to:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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 Primer](#), [z/OS TSO/E User’s Guide](#), and [z/OS ISPF User’s Guide Vol I](#) for information 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 Library Server versions of z/OS books in the Internet library at:

Notices

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

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.
1623-14, Shimotsuruma, Yawata-shi
Kanagawa 242-8502 Japan

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.

© Copyright IBM Corp. 1988, 2012
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:

Site Counsel
IBM Corporation
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.

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.
Programming interface information

This book documents information NOT intended to be used as Programming Interfaces of z/OS.

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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.
Index

Special characters

* sign
  literal value notation 2-6

A

ABEND command
to cancel IPCS processing 1-3
ABENDCODE symbol
for IPCS A-1
ABSOLUTE parameter
in IPCS data description parameter 3-8
access register
formatting related dump data
IEAVD30 exit routine 5-285
IEAVXD01 exit routine 5-285
access register data
analyzing dumps 5-22
accessibility E-1
contact IBM E-1
features E-1
screen readers F-1
ACTIVE parameter
of SETDEF IPCS subcommand 5-235
ADDDUMP IPCS subcommand
description 5-7
examples 5-8
address
identifying where an address resides in a dump 5-334
address expression
in IPCS data description parameter 3-3
ADDRESS parameter
in IPCS data description parameter 3-3
address pointer entry
renumbering 5-201
address positional parameter
in IPCS data description parameter 3-3
address processing parameter
description 3-7
address range
in IPCS data description parameter 3-3
address space
displaying ASID, job name, and ASCB
address in a dump 5-229
identifying in dump through a STOKEN 5-179
address type
floating-point in IPCS 3-5
general-purpose in IPCS 3-5
indirect in IPCS 3-5
literal in IPCS 3-5
relative in IPCS data description parameter 3-4
address type (continued)
symbolic in IPCS data description parameter 3-4
address-processing-parameter
of SETDEF IPCS subcommand 5-235
AFF symbol
for IPCS A-1
ALCWAIT IPCS verb name
description 5-300
allocatable device
analyzing dumps 5-9
allocate queue 5-20
allocation
obtaining dump output 5-312
alphabetic character symbol used in picture strings 2-4
ALTER IPCS subcommand
description 5-9
ALTLIB command of TSO/E special considerations for an IPCS session 4-1
ANALYZE IPCS subcommand
description 5-9
AOM (asynchronous operations manager) formatting dump data 5-297
AOMDATA IPCS verb name 5-297
ASID (address space identifier)
displaying for each address space in a dump 5-229
ASID parameter
in IPCS data description parameter 3-8
ASM (auxiliary storage manager)
analyzing in dump 5-28
formatting data in dump 5-300
ASMCHECK IPCS subcommand
description 5-28
ASMDATA IPCS verb name
description 5-300
ASMTVT symbol
for IPCS A-1
assistive technologies E-1
ASTEEnnn symbol
for IPCS A-2
ASTnnnnn symbol
for IPCS A-1
ASVT symbol
for IPCS A-2
ASXbnnnn symbol
for IPCS A-2
attention processing
for IPCS CLISTs 1-3
for IPCS REXX execs 1-4
for IPCS subcommands 1-3
attribute parameter
description 3-13
availability manager
formatting data in dump 5-301
AVMDATA IPCS verb name
description 5-301

B

batch job
creating dump directory for IPCS processing 7-8
directing IPCS output 1-2
batch mode processing 8-1
binary fullword
notation 2-4
binary halfword
notation 2-4
binary number
notation on subcommands 2-2, 2-3
BIT parameter
in IPCS data description parameter 3-14
BLOCK parameter
in IPCS data description parameter 3-8
CONFIRM parameter of SETDEF IPCS subcommand 5-235
console management formatting dump data using COMCHECK IPCS subcommand 5-40
contention data analyzing dumps 5-9
control block adding entry to symbol table 5-218 format 5-29
formatting and displaying using CBFORMAT IPCS subcommand 5-29
formatting with CBFORMAT IPCS primary command 6-7
IPCS symbol B-1
key fields from dump using SUMMARY IPCS subcommand 5-263
list of control blocks scanned, mapped and formatted by IPCS C-1 obtain status 5-35
search a chain 5-218
search an array 5-216
status using CBSTAT IPCS subcommand 5-35
validating using the SCAN IPCS subcommand 5-226
control register formatting in a dump 5-245

D
D IPCS line command description 6-32
DAE (dump analysis and elimination) formatting data in dump 5-302
DAEDATA IPCS verb name description 5-302
DAESYMTOMS symbol for IPCS A-2
DAT (communication vector table) formatting with CBFORMAT IPCS subcommand 5-32
DAT-off nucleus print storage from a dump 7-10

D data area describing in a dump 3-1
displaying 3-10
identifying in dump through an STOKEN 5-179
data-in-virtual formatting dump data 5-79
DATASET parameter of SETDEF IPCS subcommand 5-236
DATOFFNUCLEUS symbol for IPCS A-2
DB2 formatting dump data 5-297
DDNAME parameter of SETDEF IPCS subcommand 5-236
decimal number converting to hexadecimal 5-133
notation on subcommands 2-2
default value displaying IPCS default values using SETDEF IPCS subcommand 5-232
PROFILE-defined 5-197
setting IPCS default values using SETDEF IPCS subcommand 5-232
delayed issue queue branch entry and NIP time messages 5-320
deletion storage map records 5-87
device allocation data obtaining formatted output 5-300
DFP (Data Facility Product) formatting dump data 5-297
diagnostic worksheet displaying from dump 5-243
DIB symbol for IPCS A-2
DIBX symbol for IPCS A-2
DIMENSION parameter in IPCS data description parameter 3-18
displaying operation code using OPCODE IPCS primary command 6-20
DIVDATA IPCS subcommand description 5-79
DLF (data lookaside facility) obtaining diagnosis data 5-82
DLFDATA IPCS subcommand description 5-82
DOCPU IPCS subcommand description 5-77
stand-alone dump 5-77
DOMAIN parameter in IPCS data description parameter 3-9
DOWN IPCS primary command description 6-8
DROPDUMP clear uncataloged DSNAM 7-9
DROPDUMP IPCS subcommand description 5-84
examples 5-86
DROPMAP IPCS subcommand description 5-87

Index X-3
DROPSYM IPCS subcommand
  description 5-88
  examples 5-90
dsnme entries
  BLSCDROP CLIST 7-9
dsnwmp IPCS verb name 5-297
dump
  analyzing
    access register data 5-22
    data-in-virtual data 5-79
    ENQ contention 5-9
    for allocatable devices 5-9
    for contention data 5-9
    for control block status 5-35
    for real frames 5-9
    for resource contention data 5-9
    for suspend locks 5-9
    I/O contention 5-9
  analyzing component trace entries 5-66
  analyzing system status 5-243
    close to IPCS processing 5-38
    copying 5-46, 5-52
  data comparison 5-43
    displaying storage 5-151
    displaying symbol definitions 5-165
    extracting a single dump from a string of dumps 5-52
    multiple dumps in a single data set 5-52
    printing 7-11
    reducing the size of a dump 5-52
    retrieving data into variables 5-109
    retrieving dump directory information 5-97
    suppressing output 5-240
dump analysis
  using IPCS commands 6-1
    using IPCS line commands 6-2
    using IPCS primary commands 6-1
    using ISPF primary commands 6-2
dump data set
  displaying dump titles 4-7
  dump directory
  adding source description 5-7
  copying source description 5-49
  creating with BLSCDDIR CLIST 7-8
  deleting record 5-84
  displaying list of sources 5-153
  example of creating and initializing 1-2
  freeing space 5-88
  initialize using IPCSDDIR command 4-6
  multiple processors 5-77
  opening for IPCS processing 5-190
  retrieving information 5-97
  dump display reporter panel
    IPCS and ISPF primary commands and PF keys 6-4
dump header
  formatting title 5-152
dump title
  displaying using LIST IPCS subcommand 5-151
  displaying using SYSDSCAN command 4-7
  formatting 5-152
  obtaining a summary of dump titles 5-52
  dumping services
    obtaining dump output 5-306
  DUMPINGPROGRAM symbol
    for IPCS A-2
  DUMPINGSYSTEM symbol
    for IPCS A-3
  DUMPORIGINALDSNAME symbol
    for IPCS A-3
  DUMPREQUESTOR symbol
    for IPCS A-3
  DUMPTIMESTAMP symbol
    for IPCS A-3
  dynamic configuration change
    displaying EDT data 5-160
E IPCS line command
  description 6-33
  EBCDIC IPCS primary command description 6-9
  ECSA (extended common storage area)
    print storage from a dump 7-10
  ECSA symbol
    for IPCS A-3
  EDT (eligible device table)
    displaying data 5-160
    primary EDT 5-160
    secondary EDT 5-160
  EFLPA symbol
    for IPCS A-3
  EMLPA symbol
    for IPCS A-3
  END IPCS primary command
    description 6-9
  END IPCS subcommand
    description 5-90
    return codes 5-90
  ENF
    list listeners 5-312
  ENQ contention
    analyzing dumps 5-9
  ENTRY parameter
    in IPCS data description parameter 3-17
  ENUCLEUS symbol
    for IPCS A-3
  EPLPA symbol
    for IPCS A-3
  EPTRACE IPCS subcommand
    description 5-91
  EVALDEP IPCS subcommand
    description 5-104
  EVALDEF IPCS subcommand
    description 5-105
  EVALDUMP IPCS subcommand
    description 5-97
  example used in a CLIST 5-99
  EVALMAP IPCS subcommand
    description 5-100
  example used in a CLIST 5-103
  EVALPROF IPCS subcommand
    description 5-104
  example used in a CLIST 5-109
  EVALUATE IPCS subcommand
    description 5-109
    return codes
      CLIST, REXX, or DIALOG return codes 5-113
  error report
    FULL form report for RSM trace entries 5-74
  QUERY FULL report
    for the
      COMP1.ASID(0200).FUNC2.SVC3 trace 5-72
  QUERY report
    COMP1 multiple-trace component trace 5-71
    COMP1.ASID(0200) HEAD level 5-72
    COMP1.ASID(0200).FUNC2.SVC3 trace 5-72
    Request a list of traces defined in a dump 5-71
    SHORT form report for RSM trace entries 5-73
    SUMMARY form report for RSM trace entries 5-73
    TALLY form report 5-76
  EXCLUDE IPCS primary command
    description 6-10
  exit routine
    for analyzing ASCBs in a dump 5-25
    invoking
      using ASCBEXIT IPCS
        subcommand 5-25
      invoking TCB exit routine
        using TCBEXIT IPCS
        subcommand 5-284
      invoking using VERBEXIT IPCS
        subcommands 5-284
    testing installation-supplied exits 5-240
  exit service
    activating traps 5-290
    deactivating traps 5-288
exit service (continued)  5-286
listing status of traps  5-286
extended MCS console
formatting dump data using
COMMCHECK IPCS
subcommand  5-40
extended private storage area
printing storage from a dump  7-11
extended read-write nucleus
print storage from a dump  7-10
external interrupt
formatting GTF trace records  5-130

F
F IPCS line command
description  6-35, 6-38
F parameter
in IPCS data description
parameter  3-16
FILE parameter
of SETDEF IPCS subcommand  5-236
FIND IPCS primary command
description  6-11
example using quoted-string
notation  2-5
FIND IPCS subcommand
description  5-114
example using picture strings  2-4
FINDAREA symbol  5-114
for IPCS A-3
FINDMOD IPCS subcommand
description  5-118
FINDSWA IPCS subcommand
description  5-120
FLAG parameter
of SETDEF IPCS subcommand  5-238
FLOAT parameter
in IPCS data description
parameter  3-14
floating-point address type
in IPCS data description
parameter  3-5
FLPA symbol
for IPCS A-3
full-screen display  6-1
using IPCS commands  6-1
fullword pointer
notation  2-3

G
GDA symbol
for IPCS A-3
general purpose register
formatting in a dump  5-245
general value
ASCII character string notation  2-5
binary fullword notation  2-4
binary halfword notation  2-4
character string notation  2-3, 2-5
fullword pointer notation  2-3
hexadecimal string notation  2-6
notation types  2-3

I
I IPCS line command
description  6-35
I/O contention
analyzing dumps  5-9
IARHVC0M symbol
for IPCS A-3
IARHVSHR symbol
for IPCS A-3
IEAV30 exit routine
running  5-285
IEAVESLA symbol
for IPCS A-3
IEAVFTPMT exit routine
running  5-285
IEAVTSPS IPCS verb name
description  5-306
IEAVXID01 exit routine
running  5-285
IECDAFMT exit routine
running  5-285
IECIOFMT exit routine
running  5-285
IEFENFVX IPCS verb name
description  5-312
IEFIVAWT IPCS verb name
description  5-312
IEFIVGID IPCS verb name
description  5-312
IEFJESCOTPX symbol
for IPCS A-3
IEFRDR file  7-3
IEFZB445 symbol
for IPCS A-3
IHSAnnnnn symbol
for IPCS A-3
IKJEFT01 program
invoked by BLSJIPCS cataloged
procedure  8-2
IMS (Information Management System)
formatting dump data  5-297
IMSDKUMP IPCS verb name  5-297
INCIDENTTOKEN symbol
for IPCS A-3
indirect address
in IPCS data description
parameter  3-5
information
online help  5-133
INFORMATIONAL parameter
of SETDEF IPCS subcommand FLAG
parameter  5-239
INSTRUCTION parameter
in IPCS data description
parameter  3-14
integer
converting decimal to
hexadecimal  5-133
converting hexadecimal to
decimal  5-133
positive value notations  2-2
signed value notations  2-2
INVENTORY IPCS subcommand
description  5-133
inventory panel
IPCS command codes  6-5
IOS (input/output supervisor)
formatting related dump data
IECIOFMT exit routine  5-285
IOSCHECK IPCS subcommand  5-135
IOSCHECK IPCS subcommand
description  5-135
IPCS (interactive problem control system)
72-Byte save areas  5-91
access line mode  1-2
batch mode processing  8-1
description of subcommands  5-1
directing output  1-2
end a session  5-90
interrupt processing  1-3
introduction  1-1
IPCS primary and line
commands  6-1
messages  1-4
setting session parameters  4-5
sources for processing  1-1
start a session  4-5
syntax conventions  1-6
IPCS (interactive problem control system) (continued)
  task directory for subcommands 5-2
  TSO/E commands 4-1
  IPCS command of TSO/E description 4-5
  PARM parameter 4-5
  TASKLIB parameter 4-5

IPCS dialog
  ISPF primary commands 6-1
  line commands 6-1
  task directory 6-2
  primary commands 6-1
  task directory 6-2
  returning to previous panel 6-9
  stand-alone dump analysis from DASD 7-12
  from tape 7-12
  SVC dump analysis 7-12
  from tape 7-12
  from DASD 7-12
  SysMDump dump analysis 7-6
  SVC dump analysis 7-5
  SYSMDUMP dump analysis 7-6
  TASKLIB parameter of TSO/E 4-3

IPCS primary command
  description 6-15

IPCS subcommand
  invoking from IPCS dialog 6-15
  IPCSDATA IPCS subcommand description 5-141
  IPCSDDIR command of TSO/E CONFIRM parameter 4-7
  description 4-6
  return codes 4-7
  REUSE parameter 4-6
  IPCSDDIR data set
    opening for IPCS processing 5-190
    output for COPYDDIR IPCS subcommand 5-49
  IPCSINDD dname 5-52
  IPCSPRnn parmlib member specified on IPCS command 4-5
  IPCSPRNT subcommand 6-23
  IPCSPRNT data set
    add notes to output 5-184
    opening for IPCS processing 5-190
    output from BLSCBSAA CLIST 7-3
    output from BLSCBSAP CLIST 7-3
    output from BLSCBSVP CLIST 7-5
    output from BLSCBSYA CLIST 7-6
    output from BLSCBSYP CLIST 7-7
    routing subcommand listing 5-239
  IPCSTOC data set
    opening for IPCS processing 5-190
  IPLDATA IPCS subcommand description 5-149
  IRLM IPCS verb name 5-297
  ISGVT symbol
    for IPCS A-3
  ISGVTX symbol
    for IPCS A-4
  ISQHTG symbol
    for IPCS A-4
  ISQHTL symbol
    for IPCS A-4
  ISQHTS symbol
    for IPCS A-4
  ISGRSV symbol
    for IPCS A-4

ISPEEXEC IPCS subcommand
  description 5-150

ISPF (Interactive System Productivity Facility)
  function pool dialog variables 5-100
  retrieving dump directory information 5-97
  retrieving session default values into variables 5-94
  retrieving storage map information into variables 5-100
  retrieving symbol table information into variables 5-105, 5-109
  invoking under IPCS 5-294
  request a dialog service 5-150
  request IPL reports 5-149
  ITTCTAB symbol
    for IPCS A-4

IPCSDDIR IPCS subcommand
  description 5-141
  CONFIRM parameter 4-7
  description 4-6
  return codes 4-7
  REUSE parameter 4-6
  IPCSDDIR data set
    opening for IPCS processing 5-190
    output for COPYDDIR IPCS subcommand 5-49
  IPCSINDD dname 5-52
  IPCSPRnn parmlib member specified on IPCS command 4-5
  IPCSPRNT subcommand 6-23
  IPCSPRNT data set
    add notes to output 5-184
    opening for IPCS processing 5-190
    output from BLSCBSAA CLIST 7-3
    output from BLSCBSAP CLIST 7-3
    output from BLSCBSVP CLIST 7-5
    output from BLSCBSYA CLIST 7-6
    output from BLSCBSYP CLIST 7-7
    routing subcommand listing 5-239
  IPCSTOC data set
    opening for IPCS processing 5-190
  IPLDATA IPCS subcommand description 5-149
  IRLM IPCS verb name 5-297
  LEDATA IPCS verb name description 5-314
  LEFT IPCS primary command description 6-17
  LENGTH parameter
    in IPCS data description parameter 3-2, 3-6
    of SETDEF IPCS subcommand 5-239
  LIBRARY parameter of BLS9CALL command 4-3
  line command
    IPCS dialog 6-1
    task directory 6-2
    removal using RESET primary command 6-25
    viewing IPCS output stream 6-21
  line size
    adjusting 5-197
  list
    dump title 5-151
  LIST IPCS subcommand description 5-151
  LISTDUMP IPCS subcommand description 5-153
  LISTEDT IPCS subcommand description 5-160
  example 5-163
  listeners
    ENF code 5-312
  LISTMAP IPCS subcommand description 5-163
  example 5-165
  LISTSYS IPCS subcommand description 5-165
  examples 5-167
  LISTTOD IPCS subcommand description 5-168
  example 5-169
  LISTUCB IPCS subcommand description 5-170
  literal
    assign value 5-172
    literal address
      in IPCS data description parameter 3-5
    LITERAL IPCS subcommand description 5-172
    literal value
      finding in a dump 5-114
      general values 2-3
      positive integer 2-2
      signed integer 2-2
      symbolic literal 2-7
      types 2-1
    load module
      information from WHERE IPCS subcommand or primary command 5-336
    load module library
      specify search order for IPCS analysis parameter 3-5
    LOADMODULE symbol
      for IPCS A-4
    LOCATE IPCS primary command description 6-18
parameter string
format when passed to verb exit
routine 5-298
parm parameter of BLS9CALL
command 4-3
PARM parameter of IPCS command 4-5
PART symbol
for IPCS A-4
PATCH subcommand 5-193, 5-197
PCCAnn symbol
for IPCS A-4
PCCAVT symbol
for IPCS A-4
PDS parameter
of SETDEF IPCS subcommand 5-239
PF key
used in IPCS dialog 6-2
PFT symbol
for IPCS A-4
pgmname symbol
for IPCS A-4
PGTnnnnnaaaaa symbol
for IPCS A-4
picture string
classes of characters 2-4
notation 2-4
PLPA (pageable link pack area)
list entry points in a dump 5-174
PLPA symbol
for IPCS A-4
PMRNUMBER symbol
for IPCS A-4
pointer
displaying using LOCATE IPCS
primary command 6-18
duplicating using R IPCS line
command 6-37
displaying under the IPCS BROWSE
option 6-33
inserting using I IPCS line
command 6-35
selecting using S IPCS line
command 6-38
selecting using SELECT IPCS primary
command 6-27
sorting using SORT IPCS primary
command 6-27
stack symbol, Znnnnn A-7
pointer panel
IPCS and ISPF primary commands,
IPCS line commands, and PF
keys 6-3
POINTER parameter
in IPCS data description
parameter 3-16
pointer stack
renumbering entries 5-201
renumbering symbols in IPCS
BROWSE 6-21
positional parameter
data description 3-1
description for IPCS 1-5
POSITIONS parameter
in IPCS data description
parameter 3-2, 3-7
positive integer
binary notation 2-2
decimal notation 2-2
hexadecimal notation 2-2
notation types 2-2
signed notation 2-2
PRDMP service aid
equivalent functions in IPCS D-1
primary command
invoking for IPCS through PF
keys 6-2
IPCS 6-1
IPCS dialog 6-1
task directory 6-2
process IPCS output streams 6-21
removal using RESET primary
command 6-25
PRIMARYSYMPTOMS symbol
for IPCS A-5
print data set
opening for IPCS processing 5-190
routing subcommand listing 5-239
print dump
clean-up subcommand 5-239
conversion to IPCS D-1
PRINT parameter 1-2
of SETDEF IPCS subcommand 5-239
printed report
setting line and page size 5-197
printing
storage areas from a dump 7-11
private storage area
printing 7-11
PRIVATE symbol
for IPCS A-5
PRIVATEX symbol
for IPCS A-5
problem determination
displaying system status from
dump 5-243
PROFILE IPCS subcommand
description 5-197
example 5-200
program interrupt
formatting GTF trace records 5-131
PSAnn symbol
for IPCS A-5
PSAVALID symbol
for IPCS A-5
PSW symbol
for IPCS A-5
PSWnn symbol
for IPCS A-5
PVT symbol
for IPCS A-5
Q IPCS verb name
description 5-304
QCBTRACE IPCS verb name
description 5-304
quoted string
notation 2-5
QCBTRACE IPCS verb name
description 5-304
quoted string
notation 2-5
R IPCS line command
description 6-37
range
of addresses in IPCS data description
parameter 3-3
RANGE parameter
in IPCS data description
parameter 3-3
RB (request block)
displaying using SUMMARY IPCS
subcommand 5-263
RBA parameter
in IPCS data description
parameter 3-10
RCE symbol
for IPCS A-5
RDCM control block
formatting dump data using
COMCHECK IPCS
subcommand 5-40
read-only nucleus
print storage from a dump 7-10
read-write nucleus
print storage from a dump 7-10
real frame
analyzing dumps 5-9
REAL parameter
in IPCS data description
parameter 3-10
REASONCODE symbol
for IPCS A-5
REG32CTL symbol
for IPCS A-5
REG32CTLn symbol
for IPCS A-6
REG32GEN symbol
for IPCS A-6
REG32GENnn symbol
for IPCS A-6
REG64CTLn symbol
for IPCS A-6
REG64GEN symbol
for IPCS A-6
REG64GENnn symbol
for IPCS A-6
REGFPC symbol
for IPCS A-5
REGFPCnn symbol
for IPCS A-5
REGGEN symbol
for IPCS A-5
REGSTACK symbol
for IPCS A-5
REGSTACKnn symbol
for IPCS A-5
REGTOP symbol
for IPCS A-5
REGTOPnn symbol
for IPCS A-5
REGGEN symbol
for IPCS A-5
REGGENnn symbol for IPCS A-5
relative address in IPCS data description release IPCS resource 5-38
REMARK parameter description 3-18 in IPCS data description parameter 3-18 of SETDEF IPCS subcommand DISPLAY parameter 5-237
REMOTEDUMP symbol for IPCS A-6
RENUM IPCS primary command description 6-21
RENUM IPCS subcommand description 5-201
REPORT subcommand 6-22 REPORT IPCS primary command description 6-21 REQUEST parameter of SETDEF IPCS subcommand DISPLAY parameter 5-237
RESET IPCS primary command description 6-25
resource contention data analyzing dumps 5-9 return code for most IPCS subcommands 5-2 from the EVALUATE subcommand CHECK option 5-113 default option 5-113 RETURN IPCS primary command description 6-25
REXX exec description 7-1 format a value in a variable 5-133 identifying libraries 4-1 invoked from IPCS dialog 6-15 invoking with TSO subcommand of IPCS 5-294 retrieving dump directory information 5-97 retrieving information into variables 5-94 retrieving storage map information into variables 5-100 retrieving symbol table information into variables 5-105 variables 5-97, 5-100, 5-133
RFIND IPCS primary command description 6-25
RIGHT IPCS primary command description 6-26
RIM (resource initialization manager) analyzing with CBSTAT IPCS subcommand 5-37
RONUCLEUS symbol for IPCS A-6
RSM (real storage manager) displaying diagnostic information 5-202
RSMDATA IPCS subcommand description 5-202 report/parameter matrix 5-206
RTCT symbol for IPCS A-6
RTM (recovery termination manager) formatting related dump data IEAVTMFT exit routine 5-285 running 5-285
RUNARRAY IPCS subcommand description 5-216 RUNCHAIN IPCS subcommand description 5-218 example used in a CLIST 5-222 RUNCPool IPCS subcommand description 5-223
S
S IPCS line command (select) description 6-38
S IPCS line command (show) description 6-38
SADMP message log obtaining formatted output 5-326 SADMP/MSG IPCS verb name description 5-326 SAM/PAM/DAM formatting GTF trace records 5-131 save area CLIST to follow the forward chain 7-9 SCALAR parameter in IPCS data description parameter 3-18 SCAN IPCS subcommand description 5-226 SCCB symbol for IPCS A-6 Scheduler Work Area locate 5-120 screen output displaying excluded lines using S, F, or L IPCS line command 6-38 for IPCS using D IPCS line command 6-32 suppressing lines using X IPCS line command 6-41 screen readers accessibility F-1 scrolling dump data by changing scroll amount field 6-8 using DOWN IPCS primary command 6-8 using LEFT IPCS primary command 6-17 using LOCATE IPCS primary command 6-18 using MORE IPCS primary command 6-20 using RIGHT IPCS primary command 6-26 using UP IPCS primary command 6-29 SCVT symbol for IPCS A-6 SDWAHDR symbol for IPCS A-6 search for a module in a dump 5-118 for a SWA block 5-120 for a value using FIND IPCS primary command 6-11 repeat a search using RFIND IPCS primary command 6-25 search through output stream 6-10 search value repeat 2-6 SECONDARYSYMPTOMS symbol for IPCS A-6 SELECT IPCS primary command description 6-27 SELECT IPCS subcommand description 5-229 selection code for IPCS storage panel 6-6 selection panel IPCS and ISPF primary commands and PF keys 6-3 SERIOUS parameter of SETDEF IPCS subcommand FLAG parameter 5-239 SERVERDATA report 5-20 session default value displaying IPCS default values using SETDEF IPCS subcommand 5-232 retrieving information into variables 5-94 setting IPCS default values using SETDEF IPCS subcommand 5-232 SETDEF IPCS subcommand description 5-232 SEVERE parameter of SETDEF IPCS subcommand FLAG parameter 5-239 SGTnnnnn symbol for IPCS A-6 SHORT IPCS subcommand description 5-209 shortcut keys E-1 SHRDATA IPCS subcommand description 5-206 signed integer binary notation 2-3 hexadecimal notation 2-2 notation types 2-2 SIGNED parameter in IPCS data description parameter 3-18 SLIP trace records formatting GTF trace records 5-131 SLIPTRAP symbol for IPCS A-6 SMCS console formatting dump data using COMCHECK IPCS subcommand 5-40 SMF (system management facilities) displaying diagnostic information 5-241 SMFDATA IPCS subcommand description 5-241

Index X-9
SMS (Storage Management Subsystem) formatting dump data 5-297
SMS DATA IPCS verb name 5-297
SMSX DATA verb name 5-297
SORT IPCS primary command description 6-27
source
  types for IPCS processing 1-1
SQA (global system queue area) print 7-12
SRM (system resources manager) formatting GTF trace records 5-131
obtaining control blocks from dump 5-326
SRMDATA IPCS verb name description 5-326
SSIDATA IPCS subcommand description 5-242
stack entry
  controlling duplication 5-197
  creating 5-242
STACK IPCS primary command description 6-28
STACK IPCS subcommand description 5-242
stand-alone dump
  CLIST for initial analysis description 7-12
  printed 7-3
  CLIST to print storage 7-3
  DOCPU 5-77
  obtain data for multiple processors 5-77
  source for IPCS processing 1-1
  stand-alone dump formatting message log from dump 5-326
STATUS IPCS subcommand description 5-243
STATUS parameter in IPCS data description parameter 3-10
STATUS parameter of BLS9CALL command 4-4
STOKEN (space token) decipher using NAME subcommand 5-179
storage
  describe in a dump 3-1
  displaying from dump 5-151
  obtaining contents summary report 5-153
  printing from a dump 7-11
  storage area
  IPS symbol B-1
storage display
  using LOCATE IPCS primary command 6-18
storage key
  displaying current default values 5-222
  setting IPCS default values using SETDEF IPCS subcommand 5-232
storage map
  creating entries for address spaces 5-229
storage map (continued)
  deleting records using DROPMAP
  IPCS subcommand 5-87
  displaying entries 5-163
  list of entries created for control blocks or data areas by IPCS C-1
  retrieving information into variables 5-100
storage panel
  IPCS and ISPF primary commands, IPCS line commands, and PF keys 6-3
  IPCS selection codes 6-6
STORAGE parameter
  of SETDEF IPCS subcommand 5-235
  of SETDEF IPCS subcommand DISPLAY parameter 5-238
STRDATA IPCS subcommand description 5-253
string comparison
  by COMPARE IPCS subcommand 5-44
STRUCTURE parameter in IPCS data description parameter 3-16
  information from WHERE IPCS subcommand or primary command 5-337
subcommand
  IPCS online help 5-133
  literal values 2-1
  of IPCS 5-1
subcommand listing
  routing to print data set 5-239
subcommand of REPORT
  BROWSE 6-22
  EVALRPT 6-22
  IPCSPRNT 6-23
  VIEW 6-24
subspace
  identifying in dump through an STOKEN 5-179
subsystem console formatting dump data using COMCHECK IPCS
subcommand 5-40
subsystem information
  displaying through IPCS 5-242
SUMDUMP IPCS verb name description 5-327
SUMDUMP parameter in IPCS data description parameter 3-11
SUMMARY IPCS subcommand description 5-263
suppress
  IPCS screen output
  using X line command 6-41
suspend lock contention
  analyzing dumps 5-9
SVC dump
  CLIST for initial analysis 7-5
  printed form 7-4
  CLIST to print storage 7-5
  source for IPCS processing 1-1
SVC summary dump
  obtaining formatted output 5-327
symbol
  creating IPCS-defined
  using STACK IPCS primary command 6-28
  using STACK IPCS subcommand 5-242
  creating user-defined
  using EQUATE IPCS primary command 6-9
  using EQUATE IPCS subcommand 5-92
deleting from the symbol table 5-88
  displaying definitions for a dump 5-165
  displaying using SYMDEF IPCS subcommand 5-276
  for dump storage areas B-1
  IPCS naming conventions A-1
  IPCS special symbols B-1
  renumbering pointers in IPCS
  BROWSE 6-21
  X (current address) in IPCS 3-4
SYMBOL parameter
  of SETDEF IPCS subcommand
  DISPLAY parameter 5-238
symbol table
  add entries for a cell chain 5-223
  add entries for a control block chain 5-218
  adding symbol 5-242
  deleting entries 5-88
  displaying for IPCS 5-276
  IPCS naming conventions A-1
  renumbering stack symbol entries 5-201
  retrieving entry information into variables 5-105
  retrieving information through EVALSYM IPCS
  subcommand 5-105
  symbols established by BLSCEPTR
  CLIST 7-9
symbolic address
  in IPCS data description parameter 3-4
SYMDumpn data set
  clear D-1
SYSDSCAN command of TSO/E compatibility for different versions of MVS/SP 4-7
description 4-7
return codes 4-7
SYSin parameter of BLS9CALL command 4-4
SYSLib parameter of BLS9CALL command 4-4