z/OS

MVS Interactive Problem Control System (IPCS) Commands

SA22-7594-03

z/OS

MVS Interactive Problem Control System (IPCS) Commands

SA22-7594-03

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page G-1.

Fourth Edition, September 2002

This is a major revision of SA22-7594-02.

This edition applies to Version 1 Release 4 of z/OS™ (5694-A01), Version 1 Release 4 of z/OS.e™ (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order documents through your IBM® representative or the IBM branch office serving your locality. Documents are not stocked at the address below.

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

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

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

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

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

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

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

© Copyright International Business Machines Corporation 1988, 2002. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Tables xi

About this document xiii
Who should use this document xiii
Where to find more information xiii
Information updates on the web xiii
Accessing z/OS licensed documents on the Internet xiv
Using LookAt to look up message explanations xiv

Summary of changes xv

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 1-5

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 3-15
Remark Parameters 3-16

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

© Copyright IBM Corp. 1988, 2002

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
ARCHECK Subcommand — Format Access Register Data 5-23
ASCBEXIT Subcommand — Run an ASCB Exit Routine 5-26
ASCHDATA Subcommand — Analyze APPC/MVS Transaction Scheduler Data 5-27
ASMCHECK Subcommand — Analyze Auxiliary Storage Manager Data 5-29
CBFORMAT Subcommand — Format a Control Block. 5-30
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. 5-51
COPYTRC Subcommand — Copy Trace Entries or Records 5-56
COUPLE Subcommand — Analyze Cross-System Coupling Data 5-60
CPUTRACE Subcommand — Format Instruction Address Trace Entries 5-64
CTRACE Subcommand — Format Component Trace Entries 5-65
DIVDATA Subcommand — Analyze Data-in-Virtual Data 5-76
DLFDATA Subcommand — Format Data Lookaside Facility Data. 5-80
DROPDUMP Subcommand — Delete Source Description Data 5-82
DROPMAP Subcommand — Delete Storage Map Records 5-85
DROPSYM Subcommand — Delete Symbols 5-86
END Subcommand — End an IPCS Session 5-88
EQUATE Subcommand — Create a Symbol 5-89
EVALDEF Subcommand — Format Defaults 5-91
EVALDUMP Subcommand — Format Dump Attributes 5-94
EVALMAP Subcommand — Format a Storage Map Entry 5-97
EVALSYM Subcommand — Format the Definition of a Symbol 5-100
EVALUATE Subcommand — Retrieve Dump Data for a Variable 5-105
CLIST, REXX, or DIALOG Option 5-107
Default Option 5-108
CHECK Option. 5-109
FIND Subcommand — Locate Data in a Dump 5-109
FINDMOD Subcommand — Locate a Module Name 5-114
FINDUCB Subcommand — Locate a UCB 5-115
GO Subcommand — Resume IPCS Trap Processing 5-117
GRSDATA Subcommand — Format Global Resource Serialization Data 5-117
GTFTRACE Subcommand — Format GTF Trace Records. 5-120
HELP Subcommand — Get Information About Subcommands 5-126
INTEGER Subcommand — Format or List a Number 5-127

IOSCHECK Subcommand — Format I/O Supervisor Data 5-129
IPCSDATA Subcommand — Request a Report about IPCS Activity 5-135
IPLDATA Subcommand — Request IPL Reports 5-143
ISPEXEC Subcommand — Request an ISPF Dialog Service 5-144
LIST Subcommand — Display Storage 5-145
LISTDUMP Subcommand — List Dumps in Dump Directory 5-147
LISTEDT Subcommand — Format the Eligible Device Table (EDT) 5-154
LISTMAP Subcommand — List Storage Map Entries 5-157
LISTSYM Subcommand — List Symbol Table Entries 5-159
LISTTOD Subcommand — List TOD Clock Image 5-162
LISTUCB Subcommand — List UCBs 5-163
LITERAL Subcommand — Assign a Value to a Literal 5-165
LOGGER Subcommand — Format System Logger Address Space Data 5-167
LPAMAP Subcommand — List Link Pack Area Entry Points 5-167
MERGE and MERGEEND Subcommands — Merge Multiple Traces 5-170
NAME Subcommand — Translate an STOKEN 5-172
NAMETOKN Subcommand — Display the Token from a Name/Token Pair 5-174
NOTE Subcommand — Generate a Message 5-177
OMVSDATA Subcommand — Format z/OS UNIX Data 5-180
OPCODE Subcommand — Retrieve Operation Code 5-182
OPEN Subcommand — Prepare Resources for Use by IPCS 5-183
PATCH Subcommand 5-186
Adding or Replacing a Patch 5-188
Deleting Patches 5-188
Listing Patches 5-189
PROFILE Subcommand — Set Preferred Line and Page Size Defaults 5-190
RENUM Subcommand — Renumber Symbol Table Entries 5-193
RSMDATA Subcommand — Analyze Real Storage Manager Data 5-195
RUNARRAY Subcommand — Process an Array of Control Blocks 5-208
RUNCHAIN Subcommand — Process a Chain of Control Blocks 5-209
RUNCPOOL Subcommand — Process a CPOOL 5-214
Examples. 5-216
SCAN Subcommand — Validate System Data Areas 5-218
SELECT Subcommand — Generate Address Space Storage Map Entries 5-220
SETDEF Subcommand — Set Defaults. 5-223
SMFDATA Subcommand — Obtain System Management Facilities Records 5-232
SSIDATA Subcommand — Display Subsystem Information 5-233
STACK Subcommand — Create a Symbol in the Stack 5-233
STATUS Subcommand — Describe System Status 5-234
STRDATA Subcommand — Format Coupling Facility Structure Data 5-245
SUMMARY Subcommand — Summarize Control Block Fields 5-255
SYMDEF Subcommand — Display an Entry in the System Symbol Table 5-267
SYSTRACE Subcommand — Format System Trace Entries 5-268
TCBEXIT Subcommand — Run a TCB Exit Routine 5-272
TRAPLIST Subcommand — List the Status of IPCS Traps 5-274
TRAPOFF Subcommand — Deactivate IPCS Traps 5-276
TRAPON Subcommand — Activate IPCS Traps 5-278
TSO Subcommand — Run a TSO/E Command. 5-281
VERBEXIT Subcommand — Run an Installation-Supplied or an
IBM-Supplied Verb Exit Routine. 5-284
VERBEXIT ALCWAIT Subcommand — List Jobs Waiting for Devices 5-287
VERBEXIT ASMDATA Subcommand — Format Auxiliary Storage Manager
Data. 5-288
VERBEXIT AVMDATA Subcommand — Format Availability Manager Data 5-288
VERBEXIT BLSAIPST Subcommand — Format System Initialization Data 5-288
VERBEXIT CBDATA Subcommand — Format Component Broker Data 5-288

VERBEXIT DAEDATA Subcommand — Format Dump Analysis and Elimination Data 5-289
VERBEXIT GRSTRACE Subcommand — Format Global Resource Serialization Data 5-291
VERBEXIT IEFIVAWT Subcommand — List Pending XCF Work for Tape Allocation 5-292
VERBEXIT IEFIVIGD Subcommand — List Global Tape Device Information 5-292
VERBEXIT JESXCF Subcommand — Format Data for JES XCF Component 5-293
VERBEXIT LEDATA Subcommand — Format Language Environment Data 5-294
VERBEXIT LOGDATA Subcommand — Format Logrec Buffer Records 5-296
VERBEXIT MMSDATA Subcommand — Format MVS Message Service Data 5-299
VERBEXIT MTRACE Subcommand — Format Master Trace Entries 5-299
VERBEXIT NUCMAP Subcommand — Map Modules in the Nucleus 5-302
VERBEXIT SADMPMSG Subcommand — Format Stand-Alone Dump Message Log 5-305
VERBEXIT SRMDATA Subcommand — Format System Resource Manager Data. 5-305
VERBEXIT SUMDUMP Subcommand — Format SVC Summary Dump Data 5-306
VERBEXIT SYMPTOM Subcommand — Format Symptom String 5-307
VERBEXIT VSMDATA Subcommand — Format Virtual Storage Management Data. 5-308
VLFDATA Subcommand — Format Virtual Lookaside Facility Data. 5-311
WHERE Subcommand — Identify an Area at a Given Address 5-313
WLMDATA Subcommand — Analyze Workload Manager Data 5-321
XESDATA Subcommand — Format Cross System Extended Services® Data 5-323

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 6-6
ASCII Primary Command — Display Characters as ASCII. 6-6
CANCEL Primary Command — End the BROWSE Option 6-7
CBFORMAT Primary Command — Format a Control Block 6-7
DOWN Primary Command — Scroll Data Forward 6-7
EBCDIC Primary Command — Display Characters as EBCDIC. 6-8
END Primary Command — End a Subcommand or Panel. 6-9
EQUATE Primary Command — Create a User-Defined Symbol. 6-9
FIND Primary Command — Search for a Specified Value 6-10
IPCS Primary Command — Invoke an IPCS Subcommand, CLIST, or REXX Exec 6-15
LEFT Primary Command — Scroll Data Left 6-16
LOCATE Primary Command — Scroll the Display to Show Specific Data 6-17
MORE Primary Command — Scroll Data 6-20
OPCODE Primary Command — Display Operation Code 6-20
RENUM Primary Command — Renumber Symbol Entries 6-20
RESET Primary Command — Remove Pending Commands 6-21
RETURN Primary Command — Display the IPCS Primary Option Menu 6-21
RFIND Primary Command — Repeat the FIND Command 6-21
RIGHT Primary Command — Scroll Data Right 6-21
SELECT Primary Command — Select a Pointer to Display Storage. . . . 6-22
SORT Primary Command — Sort an IPCS Report 6-23

STACK Primary Command — Create an IPCS-Defined Symbol 6-24
UP Primary Command — Scroll Data Backward 6-25
WHERE Primary Command — Identify an Area at a Given Address. . . . 6-26
IPCS Dialog Line Commands 6-28
D Line Command — Delete Screen Output. 6-28
E Line Command — Edit a Pointer. 6-29
F Line Command — Format a Defined Control Block 6-31
I Line Command — Insert a Pointer 6-31
R Line Command — Repeat a Pointer 6-33
S Line Command — Select a Pointer to Display Storage. 6-34
S, F, and L Line Commands — Show Excluded Screen Output 6-34
X Line Command — Exclude Screen Output 6-37

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 7-2
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
BLSCEPTR CLIST — Run a Save Area Chain 7-9
BLSCPCSA CLIST — Print Common Storage Areas 7-9
BLSCPNUC CLIST — Print Nucleus Storage Areas 7-10
BLSCPRIV CLIST — Print Private Storage Areas 7-10
BLSCPRNT CLIST — Print a Dump 7-10
BLSCPSQA CLIST — Print Global System Queue Areas. 7-12
BLSCSCAN CLIST — Obtain a Stand-Alone Dump Screening Report 7-12
BLSXWHER REXX EXEC — Find All Modules with the Same Entry Point Name. 7-13

Chapter 8. IPCS Batch Mode 8-1
JCL Needed to Run IPCS in Batch Mode 8-1
IPCS Cataloged Procedure 8-1
Running CLISTs with BLSJIPCS 8-2

Appendix A. Problem Management Subcommands A-1
ADDDSN Subcommand — Add a Data Set Name to a Problem A-1
ADDPROB Subcommand — Add a Problem to IPCS A-4
DELDSN Subcommand — Delete a Data Set from a Problem. A-11
DELPROB Subcommand — Delete a Problem from IPCS A-13
LISTDSN Subcommand — List Data Set Attributes. A-15
LISTPROB Subcommand — List Problems. A-16
MODDSN Subcommand — Modify Data Set Attributes A-25
MODPROB Subcommand — Modify Problem Attributes A-28

Appendix B. IPCS Symbols B-1
Defining Symbols B-1

Creating Symbols B-1
IPCS Symbol Definitions B-1

Appendix C. IPCS Special Symbols for System Control Blocks C-1

Appendix D. Control Blocks and Data Areas Scanned, Mapped, and Formatted D-1

Appendix E. Print Dump to IPCS Conversion Summary E-1

Appendix F. Accessibility F-1
Using assistive technologies. F-1
Keyboard navigation of the user interface F-1

Notices G-1
Programming Interface Information G-2
Trademarks. G-2

Index X-1

Figures

5-1. PROFILE-Defined Defaults 5-191
6-1. Using FIND on the Dump Display Reporter Panel 6-14
6-2. Result of Using FIND 6-14
6-3. Result of Using PF5/RFIND 6-15
6-4. Using E on the Pointer Panel 6-30
6-5. Pointer Editing Panel 6-30
6-6. Result of Using Edit 6-31

© Copyright IBM Corp. 1988, 2002

Tables

1-1. Destination of IPCS Output 1-3
5-1. Standard Subcommand Return Codes 5-2
5-2. GRS Resource Status Values 5-119
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
C-1. IPCS Special Symbols C-1
E-1. AMDPRDMP - IPCS Conversion Summary. E-1

© Copyright IBM Corp. 1988, 2002

About this document

This document supports z/OS (5694) and z/OS.e (5655-G52).

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 alphabetical 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.

Short title used in this document	Title	Order number
Advanced Communications Function for TCAM	*Advanced Communications Function (ACF) for TCAM, Diagnosis Guide*	SC30-3137
CICS® Operations Guide	*CICS/ESA® 3.1 Operations Guide*	SC33-0668
IMS/MVS Diagnosis Guide and Reference	*IMS/MVS Version 2 Diagnosis Guide and Reference*	LY27-9526
Information/Management Library: Problem, Change, and Configuration Management User's Guide	*Information/Management Library: Problem, Change, and Configuration Management User's Guide*	SC34-4328

Information updates on the web

For the latest information updates that have been provided in PTF cover letters and Documentation APARs for z/OS and z/OS.e, see the online document at:

`http://www.s390.ibm.com:80/bookmgr-cgi/bookmgr.cmd/BOOKS/ZIDOCMST/CCONTENTS`

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

© Copyright IBM Corp. 1988, 2002

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

`http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/`

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

1. z/OS.e customers received a Memo to Licensees, (GI10-0684) that includes this key code.

Summary of changes

Summary of changes
for SA22-7594-03
z/OS Version 1 Release 4

The document contains information previously presented in *z/OS MVS IPCS Commands*, SA22-7594-02, which supports z/OS Version 1 Release 3.

New information

- The COPYCAPD subcommand is added to generate a report showing all captured dumps present in a standalone dump and then copy the captured dumps data to an output data set.
- The LISTTOD subcommand is added to convert TOD clock values to legible time stamps.
- The OPCODE subcommand enhances the display of multi-byte operation codes associated with z/Architecture™.
- The VERBX SRMDATA subcommand has optional parameters that can be used to limit the amount of output produced.
- Information is added to indicate this document supports z/OS.e.

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

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes
for SA22-7594-02
z/OS Version 1 Release 3

The document contains information previously presented in *z/OS MVS IPCS Commands*, SA22-7594-01, which supports z/OS Version 1 Release 2.

New information

- New VERBEXIT subcommands are added:
  - VERBEXIT IEFIVAWT allows you to format a list of pending XCF work for tape allocation.
  - VERBEXIT IEFIVIGD allows you to format the global tape devices.
- The IPLDATA subcommand allows you to request reports about the IPL process and options.
- The HIGHVIRTUAL parameter is added to the RSMDATA subcommand to request the high virtual page report.
- The SORTBY parameter is added to the RUNCHAIN subcommand to control the order of processing for chain elements.
- An appendix with z/OS product accessibility information has been added.

© Copyright IBM Corp. 1988, 2002

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

Summary of changes
for SA22-7594-01
z/OS Version 1 Release 2

The document contains information previously presented in *z/OS MVS IPCS Commands*, SA22-7594-00, which supports z/OS Version 1 Release 1.

New information

- Many IPCS subcommands have additional support to allow users to access hierarchical file system (HFS) paths through the new PATH keyword and the existing FILE keyword.
- The WHERE service is updated to indicate that the service will provide information about addresses that fall in private or extended private area storage.
- The EVALDUMP, EVALSYM, and LISTDUMP subcommands allow processing of a directory other than the IPCS user's session directory.

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

Summary of changes
for SA22-7594-00
z/OS Version 1 Release 1

The document contains information also presented in *OS/390® MVS IPCS Commands*.

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:

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

- **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, as well as 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.

© Copyright IBM Corp. 1988, 2002

– Line mode from any terminal supported by TSO/E. See “Starting IPCS”.
– Batch mode using TSO/E commands, IPCS subcommands, CLISTs, and REXX execs. See Chapter 8, “IPCS Batch Mode” on page 8-1.

Recommendations

The information in this section is intended as a ‘quick path’ into IPCS. The *z/OS MVS IPCS User’s Guide* provides more detailed usage information. It is recommended as an introduction and refresher to using IPCS as your installation’s dump analysis tool.

Starting IPCS

The procedure you follow to start IPCS depends on the specific customization, if any, that you or your installation have provided. *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:
      ```
      ALTLIB 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, 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

Message Routing Parameters	Using IPCS in Line or Full-Screen Mode, the Output is Directed to:	Using IPCS in Batch Mode, the Output is Directed to:
PRINT and TERMINAL	IPCSPRNT data set and Terminal	SYSTSPRT and IPCSPRNT data sets
PRINT and NOTERMINAL	IPCSPRNT data set	IPCSPRNT data set
NOPRINT and TERMINAL	Terminal	SYSTSPRT data set
NOPRINT and NOTERMINAL	Terminal	SYSTSPRT data set

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 SYSMDUMP data set allocated to your session.
- Enter the TSO/E END command. IPCS ends the interrupted 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 may 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:

Message	Book
IPCS	*z/OS MVS Dump Output Messages*
TSO/E	*z/OS TSO/E Messages*
MVS BCP	*z/OS MVS System Messages, Vol 1 (ABA-AOM)* *z/OS MVS System Messages, Vol 2 (ARC-ASA)* *z/OS MVS System Messages, Vol 3 (ASB-BPX)* *z/OS MVS System Messages, Vol 4 (CBD-DMO)* *z/OS MVS System Messages, Vol 5 (EDG-GFS)* *z/OS MVS System Messages, Vol 6 (GOS-IEA)* *z/OS MVS System Messages, Vol 7 (IEB-IEE)* *z/OS MVS System Messages, Vol 8 (IEF-IGD)* *z/OS MVS System Messages, Vol 9 (IGF-IWM)* *z/OS MVS System Messages, Vol 10 (IXC-IZP)*

User completion codes indicate a problem with IPCS processing. See the IPCS chapter in *z/OS MVS Diagnosis: Reference* 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)
```

Many parameters are unique to a 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 5-223

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 5-223 for an explanation of each SETDEF-defined parameter.

Notation	Meaning	Book Syntax Example	Sample Entry Example
UPPERCASE	Uppercase indicates the item must be entered using the characters shown. Enter the item in either uppercase or lowercase.	`SUMMARY KEYFIELD`	`summary keyfield`
lowercase	Lowercase indicates a variable item. Substitute your own value for the item.	`LENGTH(length)`	`length(24)`
‘ ’	Apostrophes indicate a parameter string. Enter the apostrophes as shown.	`VERBX VSMDATA 'parameter,parameter'`	`verbx vsmdata 'error,global'`

Syntax Conventions

Notation	Meaning	Book Syntax Example	Sample Entry Example
( )	Parentheses must be entered as shown.	`FLAG(severity)`	`flag(info)`
{ }	Single braces represent group-related items that are alternatives. You must enter exactly one of the items.	`{ COMCHECK | COMK }`	`comcheck`
[ ]	Single brackets represent single or group-related items that are optional. Enter one or none of the items.	`GTFTRACE [DEBUG]`	`gtftrace`
{ } { } { }	Stacked braces represent group-related items that are alternatives. You must enter exactly one of the items.	`{ ASCBEXIT }  { pgmname }` `{ ASCBX  }  { *  }`	`ascbx *`
[ ] [ ] [ ]	Stacked brackets represent group-related items that are optional. Enter one or none of the items.	`[ TERMINAL  ]` `[ NOTERMINAL ]`	`terminal`
____	Underscore indicates a default option. If you select an underscored alternative, you need not specify it when you enter the command.	`SCAN [ SUMMARY  ]` `[ NOSUMMARY ]`	`scan`
|	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.	`RDCM[(ALL | LIST | address)]`	`rdcm(all)`
...	Ellipsis indicates that the preceding item or group of items can be repeated one or more times.	`SUB((subname[.subname]...))`	`sub((sub1.` `func2.svc3))`

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.

To Describe	Specify
Positive binary numbers	B‘bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb’
Positive decimal numbers	nnnnnnnnnn
Positive hexadecimal numbers	X‘xxxxxxxx’ or X‘xxxxxxxx_xxxxxxxx’ Note that in OS/390 Release 10, an underscore (_) can be used between hexadecimal digits to improve legibility for values greater than 32 bits.

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

To Describe	Specify
Signed binary numbers	B‘[+|-]bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb’
Signed decimal numbers	[+|-]nnnnnnnnnn or F‘[+|-]nnnnnnnnnn’
Signed hexadecimal numbers	X‘[+|-]xxxxxxxx’ or X‘[+|-]xxxxxxxx_xxxxxxxx’ Note that in OS/390 Release 10, an underscore (_) can be used between hexadecimal digits to improve legibility for values greater than 32 bits.

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

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

© Copyright IBM Corp. 1988, 2002

- 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 2147843647 (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.

Note that in OS/390 Release 10, an underscore (_) can be used between hexadecimal digits to improve legibility for values greater than 32 bits.

B‘bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb’
This notation describes a binary number. The value, bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb, 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
−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’
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‘[+]xxxxxxxx’
X‘−xxxxxxxx’
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, bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb, 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 as well as 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 as well as 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"
  ```

With OS/390 Release 10, 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, xxxxxxxx_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.

Example:

```
find C'aBc'
```

Result: IPCS finds the first occurrence of aBc.

Literal Values

F‘[(Ln)][+]nnnnnnnnnn’
F‘[(Ln)]−nnnnnnnnnn’

This notation describes a signed binary fullword. The value, [+|-]nnnnnnnnnn, 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’
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	Description of Class
=	Any character
@	Alphabetic characters
#	Numeric characters
$	Special characters
¬	Non-blank characters
.	Invalid characters
-	Non-numeric characters
<	Lowercase alphabetics
>	Uppercase alphabetics

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
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 employedd 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 via use of 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'
equate 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.

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)

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

- An attribute parameter (optional)

To Describe an Address in	Specify the Parameter
An area	AREA
A bit string	BIT or HEXADECIMAL
A character string	CHARACTER
A signed number	SIGNED
An unsigned number	UNSIGNED
A pointer	POINTER
A module	MODULE
A control block	STRUCTURE

© Copyright IBM Corp. 1988, 2002

- Array parameters (optional)

To Provide	Specify the Parameter
An array	ENTRIES(xx [:yy])
An array of dimension	DIMENSION(nnn) [ENTRY(xx)]
A single entity	SCALAR

- A remark parameter (optional)

To Provide	Specify the Parameter
A comment about an address	REMARK('text')
No comment	NOREMARK

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:

  ```
  list 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.

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.

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 may be defined as well as used by the same IPCS subcommand if the following conditions are met:
   - The symbolic address conforms to IPCS naming conventions. See Appendix B, “IPCS Symbols” on page B-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

Prior to OS/390 Release 10, a relative address is a maximum of eight hexadecimal digits preceded by a plus sign (+). You may not precede a relative address with a minus sign. A relative address is relative to the current address, X.

With OS/390 Release 10 and higher, a relative address value may designate a maximum of 16 hexadecimal digits. Underscores (_) may be used as separators when the value is entered.

- Literal address

Prior to 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.

With OS/390 Release 10 and higher, underscores (_) may 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 and/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 (?). With OS/390 Release 10, 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^{64}-1.
- 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.

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‘[+ | -]nnn’), signed hexadecimal (X‘[+ | -]xxx’), signed binary (B‘[+ | -]bbb’).

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.

Data Description Parameter

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

```
[ 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          ]
[ DSNAME(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.

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 (that is, the address space in which IPCS is running), IPCS accesses only the private storage of its own address space, plus common storage.
- 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...’).

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 15 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.

DOMAIN(identifier)
The dump storage that supplements the storage pages that record system status. The valid *domain-ids* are:

DOMAIN(CPUTRACE)
The CPU loop trace information that can be formatted using the IPCS CPUTRACE subcommand. Dumps might contain CPU loop trace information that is related to each processor in the configuration. If you do not use the CPU parameter to indicate the address of the CPU containing the trace records you want, IPCS uses a default CPU address.

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 requestor 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.

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.

HEADER
The storage at the address or address range is in the header record for a system dump. 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: In OS/390 Release 10 and higher, 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^{24}-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.

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

Note: Do not use these parameters for:
- Volatile common or private storage
- Prefixed storage

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

    ```
    DSNAME(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

```
DSNAME(vsam.cluster.dsname(index))
```

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

HFS files may be referenced via this notation.

- Those HFS 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 HFS files.
- RBA access is supported for any HFS 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 desired portion with the DSNAME parameter instead avoids allocating the portions.

PATH(hfspath)
Specifies a valid path name.

- Path names can be directly referenced. There is no need to associate a path with a *ddname* prior to asking IPCS to process an HFS path.
- Partially-qualified path names may be used. IPCS will determine the fully-qualified names.

Generally, PATH can be entered as follows:

PATH(‘/pathname’)
PATH(“/pathname used in IPCS dialog”)
Path names can always be entered within apostrophes. Quotation marks (double quotes) may 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 quotes 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, or horizontal tabulation characters (EBCDIC X'05'), apostrophes (single quotes), or quotation marks (double quotes).

PATH(/x/y/z)
Quoted string notation is not always required. Most path names may be entered without enclosing punctuation.

PATH(partially-qualified-name)
Existing HFS paths that can be qualified when they are entered are accepted by IPCS as though 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             ]
[ SIGNED | F                ]
[ UNSIGNED                  ]
[ POINTER | PTR             ]
[ MODULE[(name)]            ]
[ STRUCTURE[(cbname)]       ]
```

AREA[(name)]

The storage indicated by the address or in the range is an area of storage (a subpool, a buffer, etc.) 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 alphameric 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.

SIGNED or F

The storage indicated by the address or in the address range is a signed 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.

UNSIGNED
The storage indicated by the address or in the address range is an unsigned 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.

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.

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 alphameric characters and the first character must be alphabetic.

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 the “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. The name can be a maximum of 31 alphameric 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.

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, etc. In the process, these control blocks may also be validated and entered in the storage map but not in the symbol table.

Array Parameters

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

```
[ ENTRIES(xx[:yy])                                ]
[ [DIMENSION(nnn) | MULTIPLE(nnn)] [ENTRY(xx)]  ]
[ SCALAR                                          ]
```

ENTRY(xx[:yy]) or ENTRIES(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 xx and yy. The value xx must be less than or equal to yy. These values can range from -2^{31} to $2^{31}-1$ and can be specified using signed decimal ([+|-]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.

In OS/390 Release 10, 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:

```
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:

```
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:

```
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.

```
[ 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:

```
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.

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 5-281). 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:

When You Want To	Use the
Begin an IPCS session	"IPCS Command — Start an IPCS Session" on page 4-5
Identify libraries of CLISTs and REXX EXECs	"ALTLIB Command — Identify Libraries of CLISTs and REXX EXECs"
Initialize a dump directory	"IPCSDDIR Command — Initialize a User or Sysplex Dump Directory" on page 4-6
View dump titles	"SYSDSCAN Command — Display Titles in Dump Data Sets" on page 4-7
Pass control to TSO command processors	"BLS9 Command — Session of TSO Commands" on page 4-2
Pass control for a System/370™ interface	"BLS9CALL Command — Call a Program" on page 4-3

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

© Copyright IBM Corp. 1988, 2002

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 prior to 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 via 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 than 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, SYSMDUMP, 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 ]
      [ SYSIN(sysin) ]
      [ SYSLIB(syslib) ]
      [ SYSLIN(syslin) ]
      [ SYSLMOD(syslmod) ]
      [ SYSPRINT(sysprint) ]
      [ SYSPUNCH(syspunch) ]
      [ SYSTERM(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 steblib and 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 usually 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.)

SYSIN(sysin)
Specifies the file name to be passed to the processing program and used instead of SYSIN.

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 progarm and used instead of SYSPRINT.

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

SYSTERM(systerm)
Specifies the file name to be passed to the processing program and used instead of SYSTERM.

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. Prior to 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 ]
```

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.

CONFIRM
NOCONFIRM

CONFIRM causes the IPCS user to be prompted before IPCS runs a IPCSDDIR REUSE command.

NOCONFIRM authorizes immediate processing of an IPCSDDIR REUSE command.

Return Codes

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

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

Code	Explanation
00	Successful completion.
other	Either a nonzero return code from IKJPARS or a nonzero return code from dynamic allocation.

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.
2. The MVS/SP Version 4 SYSDSCAN works when invoked on MVS/SP Version 4, Version 3, and Version 2 systems.

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.
4. SYSDSCAN of MVS/SP Version 3 works when invoked on MVS/SP Version 4, Version 3, and Version 2 systems.
5. SYSDSCAN of MVS/SP Version 2 does not correctly process MVS/SP Version 3 or Version 4 dumps, although it does process MVS/SP Version 2 dumps when invoked on MVS/SP Version 4, Version 3, and Version 2 systems.

Chapter 5. IPCS Subcommands

This chapter 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 1-5 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**

  Once the batch job has established an IPCS session, you can enter subcommands just as you would 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 5-223.

© Copyright IBM Corp. 1988, 2002

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

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

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

When You Want to	Use the
Check resource contention	“ANALYZE Subcommand — Perform Contention Analysis” on page 5-9
Display access register data	“ARCHECK Subcommand — Format Access Register Data” on page 5-23
Display ASCB-related data areas	“ASCBEXIT Subcommand — Run an ASCB Exit Routine” on page 5-26
Display z/OS UNIX® System Services (z/OS UNIX) address spaces and tasks	“OMVSDATA Subcommand — Format z/OS UNIX Data” on page 5-180

When You Want to	Use the
Format selected control blocks	“CBFORMAT Subcommand — Format a Control Block” on page 5-30
Check the status of a control block or unit of work	“CBSTAT Subcommand — Obtain Control Block Status” on page 5-35
Search for a module by name	“FINDMOD Subcommand — Locate a Module Name” on page 5-114
Search for a UCB	“FINDUCB Subcommand — Locate a UCB” on page 5-115
Display a map of the link pack area	“LPAMAP Subcommand — List Link Pack Area Entry Points” on page 5-167
Translate an STOKEN	“NAME Subcommand — Translate an STOKEN” on page 5-172
Display the token from a name/token pair.	“NAMETOKN Subcommand — Display the Token from a Name/Token Pair” on page 5-174
Repair data residing in a dump or manage the list of patches in effect for a dump.	“PATCH Subcommand” on page 5-186
Identify address spaces satisfying specified selection criteria.	“SELECT Subcommand — Generate Address Space Storage Map Entries” on page 5-220
Display system status at the time of the dump	“STATUS Subcommand — Describe System Status” on page 5-234
Display formatted control blocks	“SUMMARY Subcommand — Summarize Control Block Fields” on page 5-255
Display TCB-related data areas	“TCBEXIT Subcommand — Run a TCB Exit Routine” on page 5-272
Identify area(s) containing a given address	“WHERE Subcommand — Identify an Area at a Given Address” on page 5-313

View Dump Storage

When You Want to	Use the
Locate data in a dump	“FIND Subcommand — Locate Data in a Dump” on page 5-109
Display storage	“LIST Subcommand — Display Storage” on page 5-145
Display the eligible device table (EDT)	“LISTEDT Subcommand — Format the Eligible Device Table (EDT)” on page 5-154
Display one or more UCBs	“LISTUCB Subcommand — List UCBs” on page 5-163
Search through a chain of control blocks	“RUNCHAIN Subcommand — Process a Chain of Control Blocks” on page 5-209

View Trace Information

When You Want to	Use the
Display console-initiated loop trace data	“CPUTRACE Subcommand — Format Instruction Address Trace Entries” on page 5-64
Display component trace data	“CTRACE Subcommand — Format Component Trace Entries” on page 5-65

When You Want to	Use the
Display data-in-virtual trace data	“DIVDATA Subcommand — Analyze Data-in-Virtual Data” on page 5-76
Display GTF trace data	“GTFTRACE Subcommand — Format GTF Trace Records” on page 5-120
Merge several trace data reports	“MERGE and MERGEEND Subcommands — Merge Multiple Traces” on page 5-170
Display trace data in the master trace table	“VERBEXIT MTRACE Subcommand — Format Master Trace Entries” on page 5-299
Display system trace entries	“SYSTRACE Subcommand — Format System Trace Entries” on page 5-268

Check System Components and Key System Areas

To Obtain a Diagnostic Report for	Use the
Advanced Program-to-Program Communication (APPC) component	“APPCDATA Subcommand — Analyze APPC/MVS Component Data” on page 5-18
APPC/MVS transaction scheduler	“ASCHDATA Subcommand — Analyze APPC/MVS Transaction Scheduler Data” on page 5-27
Auxiliary storage manager (ASM) component	“ASMCHECK Subcommand — Analyze Auxiliary Storage Manager Data” on page 5-29 “VERBEXIT ASMDATA Subcommand — Format Auxiliary Storage Manager Data” on page 5-288
Availability management component	“VERBEXIT AVMDATA Subcommand — Format Availability Manager Data” on page 5-288
Communications task component	“COMCHECK Subcommand — Analyze Communications Task Data” on page 5-40
Cross-system coupling facility (XCF)	“COUPLE Subcommand — Analyze Cross-System Coupling Data” on page 5-60
Cross system extended services (XES)	“XESDATA Subcommand — Format Cross System Extended Services® Data” on page 5-323
Data-in-virtual component	“DIVDATA Subcommand — Analyze Data-in-Virtual Data” on page 5-76
Data lookaside facility (DLF) component	“DLFDATA Subcommand — Format Data Lookaside Facility Data” on page 5-80
Dump analysis and elimination (DAE) component	“VERBEXIT DAEDATA Subcommand — Format Dump Analysis and Elimination Data” on page 5-289
Global resource serialization component	“VERBEXIT GRSTRACE Subcommand — Format Global Resource Serialization Data” on page 5-291
Information Management System (IMS™) product	See *IMS/ESA® Utilities Reference*
IMS resource lock manager (IRLM) product	See *IMS/ESA Utilities Reference* or
Input/output supervisor (IOS) component	“IOSCHECK Subcommand — Format I/O Supervisor Data” on page 5-129
Job entry subsystem 2 (JES2) component	See *z/OS JES2 Diagnosis*
Job entry subsystem 3 (JES3) component	See *z/OS JES3 Diagnosis*
LOGREC buffer records	“VERBEXIT LOGDATA Subcommand — Format Logrec Buffer Records” on page 5-296

To Obtain a Diagnostic Report for	Use the
MVS message service (MMS) component	"VERBEXIT MMSDATA Subcommand — Format MVS Message Service Data" on page 5-299
Modules in the nucleus	"VERBEXIT NUCMAP Subcommand — Map Modules in the Nucleus" on page 5-302
Real storage manager (RSM) component	"RSMDATA Subcommand — Analyze Real Storage Manager Data" on page 5-195
Stand-alone dump message log	"VERBEXIT SADMPMSG Subcommand — Format Stand-Alone Dump Message Log" on page 5-305
Storage management subsystem (SMS) component	See *z/OS DFSMSdfp Diagnosis Reference*
System logger component	"LOGGER Subcommand — Format System Logger Address Space Data" on page 5-167
System resource manager (SRM) component	"VERBEXIT SRMDATA Subcommand — Format System Resource Manager Data" on page 5-305
System symbol table (which is different from the IPCS symbol table - it contains system symbols for general system use)	"SYMDEF Subcommand — Display an Entry in the System Symbol Table" on page 5-267
Subsystem Interface (SSI) component	"SSIDATA Subcommand — Display Subsystem Information" on page 5-233
Structures of the coupling facility	"STRDATA Subcommand — Format Coupling Facility Structure Data" on page 5-245
SVC summary dump data	"VERBEXIT SUMDUMP Subcommand — Format SVC Summary Dump Data" on page 5-306
Symptom string	"VERBEXIT SYMPTOM Subcommand — Format Symptom String" on page 5-307
Telecommunication Access Method (TCAM) product	See *Advanced Communications Function for TCAM*
Time Sharing Option Extensions (TSO/E) product	See *TSO/E V2 Diagnosis: Guide and Index*
Virtual lookaside facility (VLF) component	"VLFDATA Subcommand — Format Virtual Lookaside Facility Data" on page 5-311
Virtual storage manager (VSM) component	"VERBEXIT VSMDATA Subcommand — Format Virtual Storage Management Data" on page 5-308
Virtual Telecommunication Access Method (VTAM®) product	See *VTAM Diagnosis*
Workload manager (WLM)	"WLMDATA Subcommand — Analyze Workload Manager Data" on page 5-321

Retrieve Information in Variables

When You Want to	Use the
Format IPCS default values	"EVALDEF Subcommand — Format Defaults" on page 5-91
Format a dump data set name or information regarding a dump data set	"EVALDUMP Subcommand — Format Dump Attributes" on page 5-94
Format information regarding an entry in the storage map for a dump data set	"EVALMAP Subcommand — Format a Storage Map Entry" on page 5-97
Format information regarding an entry in the symbol table for a dump data set	"EVALSYM Subcommand — Format the Definition of a Symbol" on page 5-100

When You Want to	Use the
Format dump storage or protection keys	“EVALUATE Subcommand — Retrieve Dump Data for a Variable” on page 5-105

Maintain the User Dump Directory or Sysplex Dump Directory

When You Want to	Use the
Add a source description to a dump directory	“ADDDUMP Subcommand — Add a Source Description to a Dump Directory” on page 5-7
Delete records in a source description from a dump directory	“DROPDUMP Subcommand — Delete Source Description Data” on page 5-82
Delete records of control blocks that have been located in a dump	“DROPMAP Subcommand — Delete Storage Map Records” on page 5-85
Delete IPCS symbols from the IPCS symbol table	“DROPSYM Subcommand — Delete Symbols” on page 5-86
Create an IPCS symbol with a user-defined name	“EQUATE Subcommand — Create a Symbol” on page 5-89
List dumps represented in a dump directory	“LISTDUMP Subcommand — List Dumps in Dump Directory” on page 5-147
List storage map entries	“LISTMAP Subcommand — List Storage Map Entries” on page 5-157
List attributes of symbols in the IPCS symbol table	“LISTSYM Subcommand — List Symbol Table Entries” on page 5-159
Assign a value to an IPCS symbol in the symbol table	“LITERAL Subcommand — Assign a Value to a Literal” on page 5-165
Renumber all stack symbols in the IPCS symbol table	“RENUM Subcommand — Renumber Symbol Table Entries” on page 5-193
Validate control blocks	“SCAN Subcommand — Validate System Data Areas” on page 5-218
Create storage map entries for address spaces satisfying specified selection criteria	“SELECT Subcommand — Generate Address Space Storage Map Entries” on page 5-220
Add an IPCS symbol (Znnnnn) to the IPCS pointer stack	“STACK Subcommand — Create a Symbol in the Stack” on page 5-233

Perform Utility Functions

When You Want to	Use the
End the use of resources by IPCS	“CLOSE Subcommand — Release Resources in Use by IPCS” on page 5-38
Perform logical data comparisons	“COMPARE Subcommand — Compare Dump Data” on page 5-43
Copy records describing a dump data set from one dump directory to another	“COPYDDIR Subcommand — Copy Source Description from Dump Directory” on page 5-49
Copy dump data from one data set to another	“COPYDUMP Subcommand — Copy Dump Data” on page 5-51
Copy trace data to a data set from one or more dump or trace data sets	“COPYTRC Subcommand — Copy Trace Entries or Records” on page 5-56

When You Want to	Use the
End an IPCS session	“END Subcommand — End an IPCS Session” on page 5-88
Obtain descriptive information about the IPCS command and its subcommands	“HELP Subcommand — Get Information About Subcommands” on page 5-126
Format an integer using decimal digits, hexadecimal digits, or four EBCDIC characters	“INTEGER Subcommand — Format or List a Number” on page 5-127
Request ISPF dialog services	“ISPEXEC Subcommand — Request an ISPF Dialog Service” on page 5-144
Produce messages and control spacing and pagination	“NOTE Subcommand — Generate a Message” on page 5-177
Prepare resources for use by IPCS	“OPEN Subcommand — Prepare Resources for Use by IPCS” on page 5-183
Control session output format	“PROFILE Subcommand — Set Preferred Line and Page Size Defaults” on page 5-190
Set, change, and display IPCS session defaults	“SETDEF Subcommand — Set Defaults” on page 5-223
Transfer system management facility (SMF) records to a preallocated SMF (VSAM) data set	“SMFDATA Subcommand — Obtain System Management Facilities Records” on page 5-232
Invoke a non-IPCS TSO/E command or subcommand function	“TSO Subcommand — Run a TSO/E Command” on page 5-281

Debug a Dump Exit Program

When You Want to	Use the
Resume trap processing from a STOP trap	“GO Subcommand — Resume IPCS Trap Processing” on page 5-117
Display the status of currently active traps	“TRAPLIST Subcommand — List the Status of IPCS Traps” on page 5-274
Selectively disable traps	“TRAPOFF Subcommand — Deactivate IPCS Traps” on page 5-276
Selectively enable traps	“TRAPON Subcommand — Activate IPCS Traps” on page 5-278

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 SYSMDUMP 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 Appendix B, “IPCS Symbols” on page B-1.

DUMPINGPROGRAM
DUMPORIGINALDSNAME
DUMPREQUESTOR
DUMPTIMESTAMP
DUMPTOD
ERRORID
INCIDENTTOKEN
PRIMARYSYMPTOMS
REMOTEDUMP
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 5-223.

            [ 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 an HFS.

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

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 5-223.

          [ TEST | NOTEST ]
```

Parameters

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, **NEWNM** may be used 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.

  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 5-16 for an example of an EXCEPTION report and 5-17 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 PARMLIB 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

```
ANALYZE

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

                [ EXCEPTION ]

                [ RESOURCE ]

                [ ASID ]

                [ ALL ]

                [ XREF | NOXREF ]

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.

                [ 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.

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 may 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 WAITED 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 WAITED ON BY:
     JOBNAME=S1302     ASID=0012  TCB=009FA950
```

1 Names the contention report type, ASID. The report is organized by ASID.

2 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.

3 and 4 Lists other units of work experiencing contention.

Example 2

Produce a RESOURCE contention report.

– Action

```
COMMAND ===> analyze resource
```

– Result

The following report is produced.

```
1        CONTENTION REPORT BY RESOURCE NAME

2    RESOURCE #0001:
   NAME=DB1.XMITDAT.LATCH.SET  ASID=0024  Latch#=7

 RESOURCE #0001 IS HELD BY:

   JOBNAME=S1400        ASID=0056       TCB=009FA490
   DATA=EXCLUSIVE  RETADDR=80105788  REQID=01023A9D923B2020

 RESOURCE #0001 IS REQUIRED BY:

   JOBNAME=S360         ASID=0065        TCB=009FA380
   DATA=EXCLUSIVE  RETADDR=80105788  REQID=00000A9D923B2020

   JOBNAME=S370         ASID=0076        TCB=009FB260
   DATA=EXCLUSIVE  RETADDR=80105788  REQID=11000A9D923B2020


3    RESOURCE #0002:
   NAME=DB2.TRANDAT.LATCH.SET  ASID=0024  Latch#=5

 RESOURCE #0002 IS HELD BY:

   JOBNAME=S920         ASID=0112        TCB=009E4790
   DATA=SHARED     RETADDR=80105788  REQID=11000A9D92300000

   JOBNAME=S921         ASID=0147        TCB=009F7440
   DATA=SHARED     RETADDR=80105788  REQID=11000A9AAA300000

 RESOURCE #0002 IS REQUIRED BY:

   JOBNAME=S922         ASID=0083        TCB=009E8228
   DATA=EXCLUSIVE  RETADDR=80105788  REQID=1BBB0A9AAA300000


4    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


5    RESOURCE #0004:
  NAME=LOCAL LOCK FOR ASID 001A

 RESOURCE #0002 IS HELD BY:

   JOBNAME=DATJINT   ASID=001A  TCB=009FE240
   DATA=INTERRUPTED AND NOW DISPATCHABLE
```

1 Names the contention report type, RESOURCE. The report is organized by resource name.

2 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.
- Information about each job that either owns or is waiting to obtain the latch:
  - JOBNAME - The job name
  - ASID - The associated home ASID
  - TCB - The TCB address of the requestor, if the requestor is a task; the value '00000000', if the requestor 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 requestor called the Latch_Obtain service
  - REQID - The requestor ID (an 8-byte field that identifies the latch requestor).

3 Identifies a resource experiencing contention. Because the resource shown in the example report is associated with a latch, the report shows the same information that is listed for resource #0001 above.

4 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.

5 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.

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.

```
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,SYSALLDA

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,SYSALLDA

  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.

```
 BLS01003I No units of work meet the exception criteria

1 A RESOURCE LOCKOUT WAS DETECTED FOR THE FOLLOWING JOBS

2 JOBNAME=S1301     ASID=0011  TCB=009FA950

JOBNAME=S1301    HOLDS:

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

AND IS WAITING FOR:

 RESOURCE #0002:
    NAME=Device Group 001C
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR,SONORA

3 JOBNAME=S1400     ASID=0013  TCB=009FA490

JOBNAME=S1400    HOLDS:

 RESOURCE #0002:
    NAME=Device Group 001C
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR,SONORA

AND IS WAITING FOR:

 RESOURCE #0001:
    NAME=Device Group 001B
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR

4 JOBNAME=S1401     ASID=0014  TCB=009FA490

JOBNAME=S1400    HOLDS:

 RESOURCE #0001:
    NAME=Device Group 001B
    DATA=(ALC) ASSOCIATED WITH 3800,SYSPR

AND IS WAITING FOR:

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

1 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.

2	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.
3 **and** **4**	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 — 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 5-223.
             [ 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 wish. 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)

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
  - 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

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 ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
“SETDEF Subcommand — Set Defaults” on page 5-223.

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

              [ FLAG(severity) ]

              [ PRINT | NOPRINT ]

              [ TERMINAL | NOTERMINAL ]

              [ TEST | NOTEST ]
```

Parameters

data-descr

Specifies the data description parameter, which supplies the location of the desired control block or access list. 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  00010007  0101000B  FFFFFFFF
      8-11 00000000  00010006  DDDDDDDD  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.... 00D26140  ASTSN... 00000001
  ASTE REAL ADDRESS: 00D26140
     +0000  ATO..... 00C0F0B0  AX...... 0001      ATL..... 0030
     +0008  STD..... 0040C07F  LTD..... 80412000  PALD.... 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 **3** through **5**).

3 Shows the output message indicating an untranslatable ALET. An ALET is usually 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

```
Command:
 ===> 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
    +0000  OPTB1... 01        SN...... 01        EAX..... 0001
    +0008  ASTE.... 00D26080  ASTSN... 00000001

4  ASTE REAL ADDRESS: 00D26080
    +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')
```

1 Identifies the ALET value used for translation.

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

3 and 4 The formatted ALE and ASTE control blocks used for translation.

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

5 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

```
{ASCBEXIT } { pgmname | * }
{ASCBX    }

             asid

            [ AMASK(mask) ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
            [ 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 on 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

Code	Explanation
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

```
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.

ASCHDATA Subcommand

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 5-223.
             [ 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. 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.

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

```
{ASMCHECK | ASMK }

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF Subcommand — Set Defaults” on page 5-223.

            [ 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 — 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. Appendix D, “Control Blocks and Data Areas Scanned, Mapped, and Formatted” on page D-1 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 would be 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

```
{ CBFORMAT | CBF }

              data-descr

              [ EXIT | NOEXIT ]

              [ FORMAT(name) ]

              [ MODEL(name) ]

              [ VIEW(fieldlist) ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters.
See
“SETDEF Subcommand — Set Defaults” on page 5-223.

```
              [ 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)
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.

Note: FORMAT 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.

Note: MODEL does not require use of the STRUCTURE(cbname) attribute parameter.

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

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

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

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
   +000D  SCFL2.... 00        SCFL3.... 00        SCFL4.... 00
   +0010  AXPAL.... 0000      AXPOL.... 0000      MF1CP.... 0000
   +0016  ACR...... 00        FLAGS.... 80        MAFF..... 00000000  0000000
   +0020            00000000  00000000  00000000  00000000  00000000  0000000
   +0038            00000000  00000000  00000000  00000000  00000000  0000000
   +0050            00000000  00000000  00000000  00000000  00000000  0000000
   +0068  RV044.... 0000      DDRCT.... 0000      GDCC..... 00000001
   +0070  GDINT.... 00000001  GDTOD.... 00000001  TCNT..... 00000000
   +007C  UCNT..... 00000000  MASK..... 80004000  20001000  08000400  0200010
   +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.

```
UCBPRFIX: 006F8020
   -0008  LOCK..... 00000000  IOQ...... 00000000

UCBOB: 006F8028
   +0000  JBNR..... 00        FL5...... 00        ID....... FF
   +0003  STAT..... 00        CHAN..... 8000      FL1...... 00
   +0007  FLB...... 20        NXUCB.... 00000000  WGT...... 06
   +000D  NAME..... UCB       TBYT1.... 00        TBYT2.... 00
   +0012  DVCLS.... 41        UNTYP.... 01        FLC...... 00
   +0015  EXTP..... D0E001    CTCAL.... 00000000  CTCF1.... 00
   +001D  RV042.... 000000    CTCWA.... 00000000

UCBCMXT: 006F8000
   +0000  ETI...... 19        STI...... 00        FL6...... 00
   +0003  ATI...... 2C        SNSCT.... 02        FLP1..... 00
   +0006  STLI..... 00        FL7...... 00        IEXT..... 01D55B68
   +000C  CHPRM.... 00        SATI..... 00        ASID..... 0000
   +0011  WTOID.... 000000    DDT...... 00FCCCC0  CLEXT.... 00000000
   +001C  DCTOF.... 0000

UCBXPX: 01D55B68
   +0000  RSTEM.... 00        MIHKY.... 07        MIHTI.... 00
   +0003  HOTIO.... 00        IOQF..... 00000000  IOQL..... 00000000
   +000C  SIDA..... 0000      SCHNO.... 0000      PMCW1.... 0000
   +0012  MBI...... 0000      LPM...... 00        LPUM..... 00
   +0017  PIM...... 00        CHPID.... 00000000  00000000
   +0020  LEVEL.... 01        IOSF1.... 40        IOTKY.... 00
   +0023  MIHFG.... 00        LVMSK.... 00000001

 ACTUAL UCB COMMON SEGMENT ADDRESS 01D0E028

 DEVICE IS DYNAMIC
```

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.

```
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..... 00010100  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  STLI..... 00        FL7...... 00        IEXT..... 01D54D38
   +000C  CHPRM.... 00        SATI..... 00        ASID..... 0000
   +0011  WTOID.... 000000    DDT...... 00FCA728  CLEXT.... 00F0B7B0
   +001C  DCTOF.... 0000

UCBXPX: 01D54D38
   +0000  RSTEM.... 00        MIHKY.... 04        MIHTI.... 00
   +0003  HOTIO.... 40        IOQF..... 00000000  IOQL..... 00000000
   +000C  SIDA..... 0001      SCHNO.... 0029      PMCW1.... 289C
   +0012  MBI...... 0049      LPM...... C0        LPUM..... 40
   +0017  PIM...... C0        CHPID.... 60700000  00000000
   +0020  LEVEL.... 01        IOSF1.... 08        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.

```
UCBPRFIX: 01D422E0
   -0008  LOCK..... 00000000  IOQ...... 00FC1980

UCBOB: 01D422E8
   +0000  JBNR..... 00        FL5...... 88        ID....... FF
   +0003  STAT..... 04        CHAN..... 01B8      FL1...... 00
   +0007  FLB...... 00        NXUCB.... 00000000  WGT...... 08
   +000D  NAME..... 1B0       TBYT1.... 30        TBYT2.... 30
   +0012  DVCLS.... 20        UNTYP.... 0E        FLC...... 00
   +0015  EXTP..... D422C1    VTOC..... 00000000  VOLI..... 00000000
   +0022  STAB..... 00        DMCT..... 00        SQC...... 00
   +0025  FL4...... 00        USER..... 0000      BASE..... 00F0B608
   +002C  NEXP..... 01D42248

UCBCMXT: 01D422C0
   +0000  ETI...... 00        STI...... 00        FL6...... 09
   +0003  ATI...... 40        SNSCT.... 18        FLP1..... 22
   +0006  STLI..... 00        FL7...... 00        IEXT..... 01D550B8
   +000C  CHPRM.... 00        SATI..... 00        ASID..... 0000
   +0011  WTOID.... 000000    DDT...... 00FCA728  CLEXT.... 00F0B7B0
   +001C  DCTOF.... 0000

UCBXPX: 01D550B8
   +0000  RSTEM.... 00        MIHKY.... 00        MIHTI.... 00
   +0003  HOTIO.... 40        IOQF..... 00000000  IOQL..... 00000000
   +000C  SIDA..... 0001      SCHNO.... 0034      PMCW1.... 289C
   +0012  MBI...... 0051      LPM...... C0        LPUM..... 40
   +0017  PIM...... C0        CHPID.... 60700000  00000000
   +0020  LEVEL.... 01        IOSF1.... 08        IOTKY.... 00
   +0023  MIHFG.... 00        LVMSK.... 00000001

 Actual UCB Common segment address 01D422E8

 Device is dynamic

 Usable alias UCB of a parallel access volume

 Base UCB 01B0 is at address 00F0B808
```

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)
  ```

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
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ 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)

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

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

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.

```
STATUS FOR STRUCTURE(ASCB) AT 00FC8B00. NOCPU ASID(X'0001')
   IRA10102I This address space is on the SRM IN queue.
```

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.

```
STATUS FOR STRUCTURE(TCB) AT 007FA030. ASID(X'07D1')
  IEA21005I Task is in recovery processing, LIFO summary of active
              recovery environments follows:
  IEA21007I In ESTAE at IGC0006C+010D70 for S003F at IGC0006C+01045A,
              SDWA at 007B0B40
```

Example 3

Analyze an ASCB at a specified address.

– Action

```
COMMAND ===> cbstat f62180. structure(ascb)
```

– Result

CBSTAT generates the following output for the ASCB.

```
STATUS FOR STRUCTURE(ASCB) AT 00F62180. CPU(X'00') ASID(X'0001')
   IRA10102I This address space is on the SRM WAIT queue.
   IRA10104I The reason for swap-out is long wait (3).
```

Example 4

View data about failing NIP RIMs.

– Action

```
COMMAND ===> cbstat structure(storestatus)
```

– Result

CBSTAT generates the following output for the ASCB.

```
STATUS FOR STRUCTURE(STORESTATUS) AT 00FD7100 NOCPU ASID(X'0001')
IEA41001I NIP RIM IEAVNP11 has failed
IEA41002I ABEND=0C4 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 5-223.

```
           [ TEST | NOTEST ]
```

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.

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 an HFS.

CLOSE FILE(IPCSDDIR) 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

Determines 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 could 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 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
- 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

```
{ COMCHECK | COMK }

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

            [ MCSINFO                                ]
            [ DATABLKS[( LIST | address )]           ]
            [ LISTNAMES(keyname)                     ]
            [ NAME(nnnnnnnn) | ID(iiiiiiii)          ]
            [ RDCM[( ALL | LIST | address )]         ]
            [ SBC                                    ]
            [ SYSCONS                                ]
            [ SYSPLEX[( CNTRLMEM | SYSMEM )]         ]
            [ TDCM[( ALL | LIST | address )]         ]
            [ UCM                                    ]
            [ UCME[( ALL | LIST | address )]         ]
            [ UPDATES[( ALL | LIST | address )]      ]

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

Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
            [ 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. 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 an MCS or extended MCS 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.

NAME(nnnnnnnn)
Requests summary information for an MCS or extended MCS console. Specify the console's 2- to 8-character symbolic name.

RDCM[(ALL | LIST | address)]
Requests summary control block information for RDCMs.
ALL
Gives the status of all 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 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 console at the time of the dump. It formats the unit control module individual device entries (UCMEs).

ALL
Gives the status of all MCS consoles. It formats all UCMEs.

LIST
Lists the address of each UCME in the dump.

address
Gives the status of one MCS 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:

```
                 COMMUNICATION TASK ANALYSIS

                     ADDRESS OF ALL UCME

ADDRESS OF CNID#       1 : 00FD63D0
ADDRESS OF CNID#       2 : 00FD6420
ADDRESS OF CNID#       3 : 00FD6470
ADDRESS OF CNID#       4 : 00FD64C0
ADDRESS OF CNID#       5 : 00FD6510
ADDRESS OF CNID#       6 : 00FD6560
.
.
.
ADDRESS OF CNID#      54 : 00FD7460
```

– 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.

- 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 prior to comparison (TRUNCATE parameter).
- The shorter string may be padded to the length of the longer prior to 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**

```
{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 ----------
Note: You can override the following SETDEF parameters.
See
“SETDEF Subcommand — Set Defaults” on page 5-223.

```
            [ FLAG(severity) ]

            [ PRINT | NOPRINT ]

            [ TERMINAL | NOTERMINAL ]

            [ TEST | NOTEST ]
```

• **Parameters**

data-descr
ADDRESS(X)
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 prior to 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.

PAD[(value)] or TRUNCATE

PAD authorizes numeric comparison and comparison of operands of differing lengths by padding the shorter compare operand prior to comparison.

PAD(value) specifies a 1-byte value to be used to pad data prior to 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 prior to comparison.

- **Return Codes**

Code	Explanation
00	The operands are equal.
04	The first operand is low.
08	The first operand is high.
12	The comparison is incomplete.

- **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
          { 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 5-223.

```
          [ ACTIVE|MAIN|STORAGE     ]
          [ DSNAME(dslist)|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 unallocates 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 unallocate 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.

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

Code	Explanation
00	End of file reached. The input data set has been closed and a dump has been copied to the output data set.
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.

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 be evaluated, the user can request another COPYCAPD subcommand to select a specific dump.

– Action

```
COMMAND ===>  COPYCAPD
```

– Result

```
copycapd


Number  Time stamp           Title
------  ------------------  ---------------------------------------------


     1  09/19/2001 22:27:42  COMPON=GRS,COMPID=5752SCSDS,ISSUER=ISGREC,
        MODULE=ISGWFP  ,EP=ISGWFPR ,ABEND=S0602,REASON=00000000

1 captured dump processed
```

When title text will not fit on the first line, it is broken at a blank or comma and continued under the time stamp.

Example 2

Request to report and copy the captured dump.

– Action

```
COMMAND ===>  COPYCAPD 1 SPACE(5000) OUTDSN(my.captured.dump)
```

– Result

```
copycapd 1 space(5000) outdsn(my.captured.dump)


Number  Time stamp           Title
------  -------------------  ---------------------------------------------


     1  09/19/2001 22:27:42  COMPON=GRS,COMPID=5752SCSDS,ISSUER=ISGREC,
        MODULE=ISGWFP  ,EP=ISGWFPR ,ABEND=S0602,REASON=00000000

1 captured dump processed


BLS18169I Dump 1 is being copied
DATA SPACE ASID(X'0005') DSPNAME(00003SDU) STOKEN(X'800027000000002E')
DATA SPACE ASID(X'0005') DSPNAME(00000SDU) STOKEN(X'800023000000002A')
DATA SPACE ASID(X'0005') DSPNAME(00001SDU) STOKEN(X'800025000000002B')
DATA SPACE ASID(X'0005') DSPNAME(00002SDU) STOKEN(X'800026000000002C')
BLS18100I ASID(X'0005') DSPNAME(00000SDU) 7EE73008 not available for PRDDATA

IEA11004I DAT-off nucleus could not be accessed.
IEA11005I 23 SUMDUMP pages were not accessible.
IEA11005I 1 first reference page was not accessible.
BLS18170I 57,669 records 239,903,040 bytes, copied
```

COPYDDIR Subcommand — Copy Source Description from Dump Directory

Use the COPYDDIR subcommand to copy one or more source descriptions from one dump directory to another. The 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. IPCS copies the source description:

- From an input directory: This directory is specified in the COPYDDIR subcommand. It can be a user dump directory or the sysplex dump directory.
- To an output directory: This 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.

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.

See *z/OS MVS IPCS User's Guide* for information about dump directories.

Syntax

```
COPYDDIR   { INDATASET(dsname)|INDSNAME(dsname) }
           { INFILE(ddname)|INDDNAME(ddname)    }

           [ SUMMARY | NOSUMMARY ]

           [ DSNAME(dslist)|DATASET(data-set-list)             ]
           [ FILE(ddname-range-list)|DDNAME(ddname-range-list) ]
           [ PATH(path-name ...)                               ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
           [ 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 directory, from which you are copying the source description. One of these parameters is required.

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.

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) would specify 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 an HFS.

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

COPYDDIR 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 — 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
- 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.

COPYDUMP can reduce the amount of DASD space for a stand-alone dump. If you identify the address space that contains the problem or is associated with the problem, COPYDUMP can extract those problem address spaces that you want to analyze from the dump. COPYDUMP always includes address spaces 1 through 4 in the new dump data set, as well as any data spaces associated with the address spaces. Use the LISTDUMP subcommand to see the address spaces available in a dump data set.

The 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
          { OUTDSNAME(dsname)|OUTDATASET(dsname)|ODS(dsname) }
          { OUTFILE(ddname)|OFILE(ddname)                    }

          [ INDSNAME(dsname)|INDATASET(dsname)|IDS(dsname)   ]
          [ INFILE(ddname|IPCSINDD) | IFILE(ddname|IPCSINDD) ]


          [ DEFAULT ]

          [ SPACE(nnnn[,mmmm]|1500,1500) ]

          [ CLOSE | LEAVE ]

          [ ASIDLIST(ddddd[:ddddd]) ]

          [ NOSKIP | SKIP[(nnn|1|EOF)] ]

          [ NOCLEAR | CLEAR ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
          [ CONFIRM | NOCONFIRM ]

          [ PRINT | NOPRINT ]

          [ TERMINAL | NOTERMINAL ]

          [ TEST | NOTEST ]
```

Parameters

OUTDSNAME(dsname) or OUTDATASET(dsname) or ODS(dsname)
OUTFILE(ddname) or OFILE(ddname)

Specifies the output data set into which the dump is to be copied. An output data set must be specified, unless SKIP(EOF) is specified; if SKIP(EOF) and an output data set are both specified, IPCS ignores the output data set.

OUTDSNAME, OUTDATASET, or ODS specifies the name of the output data set. After copying, IPCS closes and unallocates 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

indicate the amount of space to be allocated. If the SPACE parameter is omitted, COPYDUMP uses default amounts.

OUTFILE or OFILE 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.

After copying, COPYDUMP closes the data set, but does not directly unallocate it. You must use the JCL option FREE=CLOSE to release the data set at the earliest possible moment.

INDSNAME(dsname) or INDATASET(dsname) or IDS(dsname)
INFILE(ddname|IPCSINDD) or IFILE(ddname|IPCSINDD)

Specifies the input data set from which the dump is copied. If one of these parameters is not specified, IPCS does following:

– 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. The designated data set must exist and must be cataloged. After copying, COPYDUMP closes and unallocates the input data set.

If a prior COPYDUMP subcommand left this data set open, processing of the data set is resumed where it left off. Note that INDSNAME or INDATASET may not be used to resume processing of a data set initially opened using INFILE or IFILE.

If the data set is not open, it is dynamically allocated, opened, and processed beginning with the first record,

If the data set resides on a volume that is not mounted as RESIDENT or RESERVED, MVS MOUNT authorization is required.

INFILE or IFILE specifies the ddname of the input data set. This data set must be allocated by JCL or the TSO/E ALLOCATE command before COPYDUMP is entered. After copying, COPYDUMP closes the input data set. but does not directly unallocate it. You must use the JCL option FREE=CLOSE to release the data set at the earliest possible moment.

If a prior COPYDUMP subcommand left this data set open, processing of the data set is resumed where it left off. Note that INFILE or IFILE may not be used to resume processing of a data set initially opened using INDSNAME or INDATASET.

If the data set is not open, it is dynamically allocated, opened, and processed beginning with the first record.

PATH(path-name ...)

Specifies a path-name or list of path-names of a file or directory on an HFS 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, 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 prior to 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 a range of ASIDs. When you specify a range, separate the first and last ASIDs 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 ASIDs you specify on this parameter, COPYDUMP always copies the dump records for address spaces 1 through 4 to the output data set.

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.

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.
– 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 via 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

Code	Explanation
00	End of file reached. The input data set has been closed and a dump has been copied to the output data set.
04	End of dump reached. The input data set has been left open, positioned immediately after the dump copied by this subcommand.
08	End of file reached prior to reaching the dump to be copied. (This return code is always produced if SKIP(EOF) is specified and COPYDUMP reaches end of file.)
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 prior to 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.

Code	Explanation
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.
28	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.

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 [ 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 5-223.

        [ 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.

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 unallocates each data set. When the SETDEF-defined default source is specified as an input data set, IPCS does not close or unallocate 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 unallocate 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 unallocate it.

OUTDATASET(dsname) or OUTDSNAME(dsname) or ODS(dsname)
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 unallocates 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 unallocate 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 would 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 would 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
dddddd	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.ddddddd)
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.

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

Code	Explanation
00	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.
04	No valid trace entries or records meeting the selection criteria were found. No trace data was copied to the output data set.
08	A processing error occurred. Some, but not all trace entries or records were copied to the output data set.
12	An error occurred in COPYTRC processing. No trace entries or records were copied to the output data set.
16	Dynamic allocation of the output data set failed. No trace entries or records were copied to the output data set.
20	The COPYTRC subcommand has a syntax error.
24	The COPYTRC subcommand has a semantic error.

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 signalling 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

--------- 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) ]
```

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

Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ 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 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.

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.

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.

CPUTRACE Subcommand — Format Instruction Address Trace Entries

Use the CPUTRACE subcommand to display the instruction address trace entries, which might indicate where potential loops were occurring in the system.

IPCS formats the trace for each central processor (CP). There is one trace record per central processor and up to 1,000 entries per trace record. The display of trace entries includes numbered entries and an indication of repeated addresses. The time stamps on the trace entries are provided by the processor controller, whose clock is not synchronized with clocks used by the MVS system.

If the dump contains no instruction address trace records, IPCS issues message BLS01080I. If the PSW was enabled for external interrupts, IPCS issues message BLS01081I.

Note: IPCS discards any address that is 2, 4, or 6 bytes greater than the previous address. As a result, some instructions might not be shown in the trace. For example, a branch instruction around the load register instruction might cause the instruction at LABEL1 to be discarded:

```
offsets
+10            BH    LABEL1
+14            LR    1,2
+16    LABEL1  LR    9,10
```

Similarly, a single MC instruction would cause an address to be discarded.

Syntax

```
CPUTRACE
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ 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 CPUTRACE 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   ]


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

          [ GMT|LOCAL ]

          [ START(mm/dd/yy,hh.mm.ss.dddddd) ]

          [ STOP(mm/dd/yy,hh.mm.ss.dddddd) ]

          [ EXCEPTION ]

          [ LIMIT(nnnnnnnn) ]

          [ ENTIDLIST(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 5-223.

```
[ 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 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/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.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 (nnnnnnnn) 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

     COMPONENT  SUB NAME

0001 COMP1      ASID(0010).FUNC2.SVC2
0002 COMP1      ASID(0020).FUNC1.SVC3
0003 COMP1      ASID(0200).FUNC2.SVC3
0004 COMP1      ASID(0200).FUNC1.SVC3
0005 COMP1      ASID(0012).FUNC1.SVC1
0006 COMP2      FUNCA
0007 COMP2      FUNCB
0008 COMP3
0009 COMP4
.
.
.
```

Example 2

Produce a QUERY report for the COMP1 multiple-trace component trace in Example 1.

– Action

```
COMMAND ===>  ctrace query(COMP1)
```

– Result

CTRACE produces the following output. The report is similar to the general query report, listing only the traces from the COMP1 component name.

```
        COMPONENT TRACE QUERY SUMMARY

     COMPONENT  SUB NAME

0001 COMP1      ASID(0010).FUNC2.SVC2
0002 COMP1      ASID(0020).FUNC1.SVC3
0003 COMP1      ASID(0200).FUNC2.SVC3
0004 COMP1      ASID(0200).FUNC1.SVC3
0005 COMP1      ASID(0012).FUNC1.SVC1
```

Example 3

Produce a QUERY report for the COMP1.ASID(0200) HEAD level.

– Action

```
COMMAND ===>  ctrace query(COMP1) sub((ASID(0200)))
```

– Result

CTRACE produces the following output.

```
        COMPONENT TRACE QUERY SUMMARY

     COMPONENT  SUB NAME

0001 COMP1      ASID(0200).FUNC2.SVC3
0002 COMP1      ASID(0200).FUNC1.SVC3
```

Example 4

Produce a QUERY report for the COMP1.ASID(0200).FUNC2.SVC3 trace.

– Action

```
COMMAND ===>  ctrace query(COMP1) sub((ASID(0200).func2.svc3))
```

– Result

CTRACE produces the following output.

```
COMPONENT TRACE QUERY SUMMARY

 COMP(COMP1)   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(COMP1) sub((ASID(0200).func2.svc3)) full
```

– Result

CTRACE produces the following output.

```
COMPONENT TRACE QUERY SUMMARY

 COMP(COMP1)   SUBNAME((ASID(0200).FUNC2.SVC.))

               START = 01/05/90 14:37:48.963576  GMT
               STOP  = 01/05/90 14:39:21.354861  GMT

             OPTIONS: COMASID,DMPREC,BUFF=(7,50)
```

Example 6

Produce a SHORT form report for RSM trace entries.

– Action

```
COMMAND ===>  ctrace comp(sysrsm) lim(10)
```

– Result

CTRACE produces the following output.

```
COMPONENT TRACE SHORT FORMAT
COMP(SYSRSM)
**** 01/05/90

MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
--------  --------  ---------------  -----------

RSGSNG    00000006  14:37:48.926973  Get Single Frame
RSEPAG    00000008  14:37:48.927078  Enqueue Pageable Frame
XEPEXIT   00000002  14:37:48.927177  External Entry Point Exit
XEPENTRY  00000001  14:37:48.927734  External Entry Point Entry
RSGSNG    00000006  14:37:48.927853  Get Single Frame
RSEPAG    00000008  14:37:48.927953  Enqueue Pageable Frame
XEPEXIT   00000002  14:37:48.928052  External Entry Point Exit
XEPENTRY  00000001  14:37:48.928554  External Entry Point Entry
RSGSNG    00000006  14:37:48.928668  Get Single Frame
RSEPAG    00000008  14:37:48.928772  Enqueue Pageable Frame
```

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.

```
COMPONENT TRACE SUMMARY 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
RSEPAG    00000008  14:37:48.927078  Enqueue Pageable Frame
     FUNC1... VSMGTMN           VSM Getmain Service
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 88004001 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 88004000
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
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
RSGSNG    00000006  14:37:48.927853  Get Single Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08084003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08084000
RSEPAG    00000008  14:37:48.927953  Enqueue Pageable Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08004003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08004000
XEPEXIT   00000002  14:37:48.928052  External Entry Point Exit
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 00000003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 00000000
XEPENTRY  00000001  14:37:48.928554  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
RSGSNG    00000006  14:37:48.928668  Get Single Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08084003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08084000
RSEPAG    00000008  14:37:48.928772  Enqueue Pageable Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08004003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08004000
```

Example 8

Produce a FULL form report for RSM trace entries.

– Action

```
COMMAND ===>  ctrace comp(sysrsm) lim(10) full
```

– Result

CTRACE produces the following output.

```
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 0125FBEC FFC00000 03000000 00000000 7FFE4000 01B77F00 00000000
RSEPAG    00000008  14:37:48.927078  Enqueue Pageable Frame
     FUNC1... VSMGTMN           VSM Getmain Service
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 88004001 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 88004000
     KEY..... 0036     ADDR.... 01B32DC0 ALET.... 00000000
     1900
     KEY..... 0001     ADDR.... 012A6000 ALET.... 00000000
     01A12AAC 0129A7E0 81C00000 03000000 0000000A 00989000 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.... 01B32DC0 ALET.... 00000000
     1900
     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 81ED81AE


RSGSNG    00000006  14:37:48.927853  Get Single Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08084003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08084000
     KEY..... 0036     ADDR.... 01B2FDC0 ALET.... 00000000
     04001200
     KEY..... 0001     ADDR.... 012A26A0 ALET.... 00000000
     0129E7E0 0125FBEC 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.. 08004003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08004000
     KEY..... 0036     ADDR.... 01B2FDC0 ALET.... 00000000
     0400
     KEY..... 0001     ADDR.... 012A26A0 ALET.... 00000000
     01A12AAC 012A6000 81C00000 03000000 0000000A 0098A000 01B14E80 00000000
```

```
XEPEXIT   00000002  14:37:48.928052  External Entry Point Exit
     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..... 0016     ADDR.... 00000004 ALET.... 00000000
     KEY..... 0017     ADDR.... 04000E00 ALET.... 00000000
     KEY..... 0027     ADDR.... 7FF14228 ALET.... 00000000
     KEY..... 0002     ADDR.... 012A26A0 ALET.... 00000000


XEPENTRY  00000001  14:37:48.928554  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.... 0098B000 ALET.... 00000000
     KEY..... 0032     ADDR.... 00F2B088 ALET.... 00000000
     070C2000 8243D124


RSGSNG    00000006  14:37:48.928668  Get Single Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08084003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08084000
     KEY..... 0036     ADDR.... 01B2FDC0 ALET.... 00000000
     04001200
     KEY..... 0001     ADDR.... 0129E7E0 ALET.... 00000000
     01292A80 0125FBEC FFC00000 03000000 00000000 02F1C000 01B77700 00000000


RSEPAG    00000008  14:37:48.928772  Enqueue Pageable Frame
     FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
     JOBN1... CONSOLE  ASID1... 000A     PLOCKS.. 08004003 CPU..... 0001
     JOBN2... CONSOLE  ASID2... 000A     RLOCKS.. 08004000
     KEY..... 0036     ADDR.... 01B2FDC0 ALET.... 00000000
     0400
     KEY..... 0001     ADDR.... 0129E7E0 ALET.... 00000000
     01A12AAC 012A26A0 81C00000 03000000 0000000A 0098B000 01B77700 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.

```
COMPONENT TRACE TALLY REPORT
COMP(SYSRSM)
    TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)

FMTID    COUNT  INTERVAL   MNEMONIC  DESCRIBE
-------- ----- ----------  --------  ----------------------------
00000001     4        855  XEPENTRY  External Entry Point Entry
00000002     4        944  XEPEXIT   External Entry Point Exit
00000003     0             FIX       Page Being Fixed
00000004     0             FREE      Page Being Freed
00000005     0             RSGDBL    Get Double Frame
00000006     3        847  RSGSNG    Get Single Frame
00000007     0             RSEFIX    Enqueue Fixed Frame
00000008     3        847  RSEPAG    Enqueue Pageable Frame
00000009     0             RSESQA    Enqueue SQA Frame
0000000A     0             RSESBUF   Enqueue Storage Buffer Frame
0000000B     0             RSEDEFER  Enqueue Deferred Frame
0000000C     0             RSEVRW    Enqueue V=R Waiting Frame
0000000D     0             RSDFIX    Dequeue Fixed Frame
0000000E     3        170  RSDPAG    Dequeue Pageable Frame
0000000F     0             RSDSQA    Dequeue SQA Frame
00000010     0             RSDSBUF   Dequeue Storage Buffer Frame
00000011     0             RSDDEFER  Dequeue Deferred Frame
00000012     0             RSDVRW    Dequeue V=R Waiting Frame
00000013     0             RSFDBL    Free Double Frame
00000014     3        162  RSFSNG    Free Single Frame
00000015     0             ESGET     Get Expanded Storage
00000016     0             ESENQ     Enqueue Expanded Storage
00000017     0             ESDEQ     Dequeue Expanded Storage
00000018     0             ESFREE    Free Expanded Storage
00000019     0             PAGER2A   Page Request Real to Auxiliary
0000001A     0             PAGER2P   Page Request Real to Permanent
0000001B     0             PAGER2E   Page Request Real to Expanded
0000001C     0             PAGER2R   Page Request Real to Real
0000001D     0             PAGEA2R   Page Request Auxiliary to Real
0000001E     0             PAGEP2R   Page Request Permanent to Real
0000001F     0             PAGEE2R   Page Request Expanded to Real
00000020     0             PAGEREL   Page Request Related
00000021     0             PAGEDEF   Page Request Deferred
00000022     0             FUNCREQ   Function Request
00000023     2 ] 16 min.     TRACEB    Trace Buffer

Total trace entries:       22
```

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

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

              [ DETAIL ]

              [ SUMMARY ]

              [ EXCEPTION ]

              [ TRACE      {OLDEST(n) } ]
              [ FULLTRACE  {NEWEST(n) } ]


-------- 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 5-223.

```
[ 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 their address spaces identifiers (ASIDs). If you omit these parameters, the defaults are CURRENT and ERROR. 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.

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) }


-------- Address Selection Parameters ----------------------
                [ ASIDLIST(list) ]

                [ CURRENT ]

                [ ERROR ]

                [ TCBERROR ]

                [ JOBLIST(list)|JOBNAME(list) ]


-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
“SETDEF Subcommand — Set Defaults” on page 5-223.
              [ 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 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.

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*.

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* 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 SYSMDUMP 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
- Delete source descriptions to free space in the directory
- Delete translation records from one or more source descriptions

Related Subcommands
ADDDUMP
LISTDUMP

Syntax

```
{DROPDUMP } [RECORDS  {( ALL | ANALYSIS | TRANSLATION  } ]
{DROPD
            [ SUMMARY | NOSUMMARY ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
“SETDEF Subcommand — Set Defaults” on page 5-223.

```
            [ ACTIVE|MAIN|STORAGE            ]
            [ DSNAME(dslist)|DATASET(dslist) ]
            [ FILE(ddlist)|DDNAME(ddlist)    ]

            [ TEST | NOTEST ]
```

Parameters

RECORDS(ALL)
RECORDS(ANALYSIS)

RECORDS(TRANSLATION)

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 DROPDUMP 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 SGTnnnnn symbol in the symbol table. Then, you can run DROPDUMP 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.

– 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 would be incorrect if the symbols for the control blocks are incorrect or if the ANALYZE exit routines specified by parmlib members embedded in the BLSCECT parmlib member have been redefined. If you determine that the contention records are incorrect, enter DROPDUMP RECORDS(TRANSLATION) to delete all contention records. Then you can run ANALYZE or STATUS CPU CONTENTION to have IPCS gather the contention records again.

DROPDUMP RECORDS(TRANSLATION) does not edit the symbol table or the storage map. For editing, use DROPMAP, 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 DROPDUMP 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.

```
BLS18206I All records for 1 dump dropped
```

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.

```
BLS18206I Translation records for 1 dump dropped
```

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 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

LISTMAP
SCAN

Syntax

```
{DROPMAP } [RANGE (address:address)] [data-descr]
{DROPM   }

           [ SUMMARY | NOSUMMARY ]
```

-------- SETDEF-Defined Parameter ----------
Note: You can override the following SETDEF parameter.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
           [ TEST | NOTEST ]
```

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 record s 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)
- 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 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.

You can delete symbols when you want to free space in the dump directory.

Related Subcommands

EQUATE
LISTSYM
RENUM
STACK

Syntax

```
{ 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 5-223.

```
            [ 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 Appendix B, "IPCS Symbols" on page B-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 Appendix B, "IPCS Symbols" on page B-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)
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 unallocates the data set directory, problem directory, and any dumps allocated to the user. The subcommand closes but does not unallocate 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.

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
LISTSYM
RENUM
STACK

Syntax

```
{ EQUATE }    [ symbol | X ]    [ data-descr | X ]
{ EQU    }
{ EQ     }    [ DROP | NODROP ]
```

-------- SETDEF-Defined Parameter ----------
Note: You can override the following SETDEF parameter.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ TEST | NOTEST ]
```

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.

- 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.

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 Subcommand

```
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 5-223.

```
           [ 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:

```
[ 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.

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

EVALDEF Subcommand

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.SBLSCLI0 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 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.

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.

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 5-223.

```
           [ 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 ]
[ ¬>  | NG ]
[ ¬<  | NL ]
[  =  | EQ ]
```

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.

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 | 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 >= ACTIVE CLIST(SOURCE(SRC) BLOCKS(JL) BYTES(JY))
```

See the BLSCEDUM member of SYS1.SBLSCLI0 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 SYSMDUMP 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

           [ 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 5-223.

```
           [ 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.

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:

```
[ 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:

Parameter	Default	Specifying DECIMAL changes the default	Specifying HEXADECIMAL changes the default
ADDRESS	HEXADECIMAL	yes	
DIMENSION	DECIMAL		yes
ENTRY	DECIMAL		yes
LENGTH	DECIMAL		yes
POSITION	HEXADECIMAL	yes	

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*.

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 >= 0. ABSOLUTE STRUCTURE CLIST(QUALIFICATION(Q) +
    ADDRESS(A) DATATYPE(T))
```

See the BLSCEMAP member of SYS1.SBLSCLI0 for the complete listing.

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 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
EVALMAP

Syntax

```
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 5-223.

```
           [ ACTIVE | MAIN | STORAGE            ]
           [ DSNAME(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 ]
[  =  | 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 to identify which symbol's definition is to be retrieved.

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.

INDATASET(dsname)
INDSNAME(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.

The syntax for *var-list* is as follows:

```
[ DECIMAL | F     ]
[ HEXADECIMAL | 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.

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 upper case 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.
- 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

DSNAME(dsname) or DATASET(dsname)
FILE(ddname) or DDNAME(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 BLSCESYM 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 BLSCESYM 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.

“Default Option” on page 5-108 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 5-107.

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*.

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 5-223.

```
             [ 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 or not 1 to 4 bytes of storage can be accessed in a dump. “CHECK Option” on page 5-109 below discusses this option further.

CLIST(var-list)
DIALOG(var-list)
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”, 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.

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 upper case 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.
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.

Return Codes for the CLIST, REXX, or DIALOG Option

Code	Explanation
00	Successful completion
04	Description of data was edited.
12	Data not available or not defined. The variables are not changed.

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 prior to 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 could be interpreted as data.

Return Codes for the Default Option

Code	Explanation
12	Severe, requested storage cannot be retrieved.
16	Terminating, an error condition from a called service routine forced an early termination.
other	Successful completion, uses the requested data as a return code.

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

Code	Explanation
00	Successful completion
04	Description of data was edited – If the length is more than 4 bytes, LENGTH(4) is substituted. – If an array containing multiple entries is described, DIMENSION(1) is substituted. – Only the UNSIGNED data type is supported. If another data type is described, UNSIGNED is substituted.
08	Four bytes of data were retrieved but the initial byte does not contain X'00'. Significance is lost if the first byte of a fullword is removed. That byte does not contain X'00'.
12	Data not available or not defined.

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, etc.

– 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.

The 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 could be outside the current search range only if you 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 5-223.

```
            [ 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|¬>|NG|=|EQ|>=|GE|¬<|NL|>|GT|¬|NE]

value
Specifies a general value. See Chapter 2, "Literal Values" on page 2-1 for 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 prior to 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

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:

Offset	Description
00	Name of NUCLEUS CSECT in EBCDIC
08	Address of NUCLEUS CSECT
0C	Length of NUCLEUS CSECT

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
2. Searches the active link pack area (LPA) queue in the dump for the module in the MLPA/EMLPA 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 5-223.

```
            [ 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

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.

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.

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 UCBdddd, where *dddd* is the device number.

FINDUCB processes the specified device number as follows:

1. Searches the symbol table for the symbol UCBdddd. 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.

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 5-223.

```
[ 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 5-278 for more information.

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, RESERVE, and latch service interfaces.

Related Subcommands

ANALYZE
STATUS

Syntax

```
GRSDATA
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
“SETDEF Subcommand — Set Defaults” on page 5-223.

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

    [ FLAG(severity)   ]

    [ PRINT | NOPRINT ]

    [ TERMINAL | NOTERMINAL ]

    [ TEST | NOTEST ]
```

Return Codes

GRSDATA Subcommand

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.

```
Global system resources  1

  Major.. CL8'RESERVEQ'  2

    Minor.. CL008'RESERVEM'  3

      SCOPE. SYSTEMS  SYSNAME. SY1         JOBNAME. GRSTOOL  4
      ASID.. 001B        TCB..... 008F1B90         STATUS.. EXCLUSIVE  5

      SCOPE. SYSTEMS  SYSNAME. SY1          JOBNAME. GRSTOOL    RESERVE. 0273
      ASID.. 001C         TCB..... 008F1B90         STATUS.. WAITEXC     ECB..... 05004614

  Major.. CL8'RESERVEZ'  >

    Minor.. CL009'RESERVEZ4'

      SCOPE. SYSTEMS  SYSNAME. SY2         JOBNAME. RTARGET4
      ASID.. 0020        TCB..... 008F1B90         STATUS.. SHARED

      SCOPE. SYSTEMS  SYSNAME. SY2          JOBNAME. GRSTOOL
      ASID.. 0021         TCB..... 008F1B90         STATUS.. WAITEXC     SVRB..... 008FF738

      SCOPE. SYSTEMS  SYSNAME. SY2          JOBNAME. MRGUY        RESERVE
      ASID.. 0022         TCB..... 008F1B90         STATUS.. WAITEXC     ECB..... 05004614


 Major.. CL8'SYSZWLM'

    Minor.. CL0019'WLM_SYSTEM_SY1'
              XL019'E6D3D46D E2E8E2E3 C5D46DE2 E8F14040 404040'

      SCOPE. SYSTEMS  SYSNAME. SY1         JOBNAME. WLM
      ASID.. 000B        TCB..... 008FD7C0        STATUS.. EXCLUSIVE


    Minor.. CL0019'WLM_SYSTEM_SY2'
              XL019'E6D3D46D E2E8E2E3 C5D46DE2 E8F24040 404040'

      SCOPE. SYSTEMS  SYSNAME. SY2         JOBNAME. WLM
      ASID.. 000B        TCB..... 008FD7C0        STATUS.. EXCLUSIVE
```

1 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 via the SDATA=GRSQ option of SDUMP.

2 Major resource names are presented using notation similar to that used by assembler language coders. GRSDATA expects that upper case 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,

```
XL8'D9C5E2C5 D9E5C5D4'
```

3 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.

4 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.

5 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 5-2 may appear:

Table 5-2. GRS Resource Status Values

Value	Meaning
EXCLUSIVE	Exclusive status held
MCEXC	Exclusive must-complete status held
MCSHR	Shared must-complete status held
SHARED	Shared status held
WAITEXC	Awaiting exclusive status
WAITMCE	Awaiting exclusive must-complete status
WAITMCS	Awaiting shared must-complete status
WAITSHR	Awaiting shared status

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 requestor 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 } [ ASCB(ascb-address-list) ]
{GTF      } [ ASID(asidlist) ]
            [ JOBNAME(joblist) | JOBLIST(joblist) ]

            [ BEGINFIRST ]
            [ BEGINOLD   ]

            [ CICS((text)) ]

            [ CPU(cpu-id) ]

            [ 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 5-223.

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

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ TEST | NOTEST ]

[ FLAG(severity) ]
```

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, just as you would any TSO/E command.

```
Command ===>GTFTRACE DD(SYSTRACE) IO(D34,D0C,ED8,+
          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.

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-id)
Specifies that events occurring on the central processor whose physical identifier is *cpu-id* be formatted. *cpu-id* is a hexadecimal number from 0 to F.

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 could 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 zeroes.

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 and/or IBM subsystem event records from these subsystems:

OPEN/CLOSE/EOV
SAM/PAM/DAM
VTAM
VSAM

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:

Symbol	ID	Subsystem
AM01	FF5	VSAM
APTH	FE2	TSO/VTAM TGET/TPUT trace
APTR	FE3	VTAM reserved
CL01	FF1	VTAM buffer contents trace (USER)
CL02	FF0	VTAM SMS (buffer use) trace
DB2V	F5F	DB2/VSAM transparency
DMA1	FFF	OPEN/CLOSE/EOV
FSI4	F54	FSI trace
FSI5	F55	FSI trace
FSI6	F56	FSI trace
FSI7	F57	FSI trace
FSI8	F58	FSI trace
FSI9	F59	FSI trace
FSIA	F5A	FSI trace
FSIB	F5B	FSI trace
FSIC	F5C	FSI trace
FSID	F5D	FSI trace
INT1	FE1	VTAM internal table
OSIC	F53	OSI Communication Subsystem
SPD1	FF3	SAM/PAM/DAM
SPD2	FF4	SAM/PAM/DAM
SPD3	FF6	SAM/PAM/DAM
SPD4	FF7	SAM/PAM/DAM
SPD5	FF8	SAM/PAM/DAM
SPD6	FF9	SAM/PAM/DAM
SPD7	FFA	SAM/PAM/DAM
SPD8	FFB	SAM/PAM/DAM
SPD9	FFC	SAM/PAM/DAM
SPDA	FFD	SAM/PAM/DAM
SPDB	FFE	SAM/PAM/DAM
TCA0	FA0	TCAM
TCA1	FA1	TCAM
TCA2	FA2	TCAM
TCA3	FA3	TCAM
TCA4	FA4	TCAM
TCA5	FA5	TCAM
TCA6	FA6	TCAM
TCA7	FA7	TCAM
TCA8	FA8	TCAM
TCA9	FA9	TCAM
TPIO	FEF	VTAM buffer contents trace

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:

ID	Issued by
000-3FF	GTF user program
400-5F0	Reserved for IBM Use
5F1	PVM

ID	Issued by
5F2-5F3	Reserved for IBM Use
5F4-5F5	NetView® System Monitor
5F6-F47	Reserved for IBM Use
F48	IOS
F49	BDT
F4F	OSAM
F50-F52	Reserved for IBM Use
F53	OSI Communications Subsystem
F54-F5D	FSI
F5E	Reserved for IBM Use
F5F	DB2®
F60	JES3
F61	VSAM Buffer Manager
F62	Dynamic output SVC installation exit
F63	Converter/Interpreter installation exit
F64	APPC/VM VTAM Support (AVS)
F66-F6A	VTAM
F6C	CICS
FA0-FA9	TCAM
FAA	VTAM VM/SNA Console Services (VSCS)
FAB-FAE	

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.

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.
- 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 Subcommand

```
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 5-223.

```
             [ 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.

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 and ERROR are the defaults. Address space selection parameters only apply with the CAPTURE parameter.

Syntax

IOSCHECK Subcommand

```
{ IOSCHECK } [ ACTVUCBS ]
{ 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 5-223.

```
[ 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 and formats active UCBs, as well as these associated control blocks:

IOQ
IOSB
SRB
EWA

ALLUCBS

Validates the I/O control blocks and formats all UCBs, along with these associated control blocks:

IOQ
IOSB
SRB
EWA

CAPTURE

Formats the captured UCB pages in an address space (based on the address space selection parameters) along with these associated control blocks:

IOQ
IOSB
SRB

EWA

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 could 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.

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.

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.

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 yyyyyyyy
```

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 defaults are CURRENT and ERROR. 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

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 or contain an error.

– Action

```
COMMAND ===> IOSCHECK CAPTURE
```

– Result

This example formats the captured UCB information for any address space that is active or contains an error indicator. The output would look similar to the following for each address space:

```
                    * * * ADDRESS SPACE CAPTURE DATA * * *

ASID 000F

                    ACTUAL        CAPTURE       CAPTURE
                    PAGE ADDRESS  PAGE ADDRESS  UCB COUNT
                    ------------  ------------  ------------

                    01D0E000      006F8000      00000005

                    01D0F000      006F7000      00000003
```

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')
  ```

  By default IPCSDATA attempts to process the CURRENT and ERROR ASIDs, formatting IPCS session data, if any, from them.
- 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
            ------- 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 5-223.

```
[ ACTIVE | MAIN | STORAGE           ]
[ DSNAME(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 5-220.

- SETDEF-Defined Parameters

  Overrides defaults established via the SETDEF subcommand or the Defaults option of the IPCS dialog. See “SETDEF Subcommand — Set Defaults” on page 5-223.

- Diagnosis — Sample IPCSDATA Reports
  - **Example 1.** Sample IPCSDATA Common Storage Report

    The following sample includes parmlib information. Use of the NOPARMLIB option would eliminate all lines of the report following the one beginning “BLSQXBT”.

1. Statistics are acquired via the VSAM SHOWCB ACB programming interface, and no equivalent interface is supported for ACB images retrieved from a dump.

```
Common storage report

  BLSQXBT at 0D35CCC0 LENGTH(4927)

    SYSDDIR 'MVSSPT.SYSPLEX.DMPDIR'

    DATA STRUCTURE(ALE) MODEL(IEAALEP)
    DATA STRUCTURE(ASCB) FIND(BLSSASCB) MODEL(IEAASCBP) SCAN(BLSVASCB)
    DATA STRUCTURE(ASSB) MODEL(IEAASSBP) SCAN(BLSVASSB)
    DATA STRUCTURE(AST) FIND(BLSSASTE) GROUP(ASTE) MODEL(IEAASTEP) SCAN(+
         BLSVASTE)
    DATA STRUCTURE(ASTE) FIND(BLSSASTE) MODEL(IEAASTEP) SCAN(BLSVASTE)
    DATA STRUCTURE(ASXB) FIND(BLSSASXB) MODEL(IEAASXBP) SCAN(BLSVASXB)

    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) FIND(BLSSCDE) GROUP(CDE) 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(IEALSSDP) 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(RB) FORMAT(IEARBF,JBB2125) SCAN(BLSVRB)
    DATA STRUCTURE(REGACC) FIND(BLSSREGA) MODEL(BLSBREGA)
    DATA STRUCTURE(REGCTL) FIND(BLSSREGC) MODEL(BLSBREGC)
    DATA STRUCTURE(REGFLT) FIND(BLSSREGF) MODEL(BLSBREGF)
    DATA STRUCTURE(REGGEN) FIND(BLSSREGG) MODEL(BLSBREGG)
    DATA STRUCTURE(REGS) MODEL(BLSBREGS)
    DATA STRUCTURE(REGSAVIM) MODEL(BLSBREGI)
    DATA STRUCTURE(RTM2WA) MODEL(IEAVTRP2) SCAN(IEAVTRV2)
```

```
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(SVRB) FORMAT(IEARBF,JBB2125) GROUP(RB) SCAN(BLSVRB)

DATA STRUCTURE(TCB) FIND(BLSSTCB) MODEL(IEATCBP) SCAN(BLSVTCB)
DATA STRUCTURE(TIOT) FORMAT(BLSQTIOF)
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(UCBDA) 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(UCBTP) FIND(IOSVUCBS) FORMAT(IOSVFMTU,JBB2125) GROUP(UCB) +
     SCAN(IOSVUCBV)
DATA STRUCTURE(UCBUR) 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(XTLST) FIND(BLSSXTLS) MODEL(CSVFMXTL) SCAN(BLSVXTLS)
```

– **Example 2.** Sample IPCSDATA Private Storage Report

  The following sample includes parmlib, open data set and task information.

  - Use of the NOPARMLIB option would eliminate 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 would eliminate 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 via LISTCAT. SHRPOOL, BFRFND and BUFRDS are accumulated within a single session and can only be obtained via IPCSDATA against ACTIVE storage.
  - Use of the NOTASK option would eliminate the report lines starting with the line beginning “Master BLSUZZ2”.

```
ASID(X'0305'), JOBNAME(RLW)

  BLSUZZ1 at 000388B0

  Dump directory BLSUZZ4 at 00050E00
    FILE(IPCSDDIR) DSNAME('RLW.DDIR')
    NLOGR(6135) NRETR(52452) NINSR(13209) NUPDR(253) NDELR(19792)
    CINV(22528) NCIS(208) NSSS(6) SHRPOOL(15)
    BFRFND(39103) BUFRDS(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(IEAALEP)
    DATA STRUCTURE(AMDCPMAP) MODEL(BLSBCPST)
    DATA STRUCTURE(AR) FORMAT(IEAVXD02)
    DATA STRUCTURE(ASCB) FIND(BLSSASCB) MODEL(IEAASCBP) SCAN(BLSVASCB)

      EXIT CBSTAT(ASCB) EP(BLSAFLG)
      EXIT CBSTAT(ASCB) EP(IEAVTRCA)
      EXIT CBSTAT(ASCB) EP(IRARMCBS)
      EXIT CBSTAT(ASCB) EP(BPXGMCBS)

      EXIT FORMAT(ASCB) EP(IEASRBQ2)

    DATA STRUCTURE(ASEI) MODEL(ASEASEIP)
    DATA STRUCTURE(ASMHD) MODEL(ILRMASMH)
    DATA STRUCTURE(ASMVT) FIND(ILRFASMV) MODEL(ILRMASMV)
    DATA STRUCTURE(ASPCT) FORMAT(ILRPASPC)
    DATA STRUCTURE(ASSB) FIND(BLSSASSB) MODEL(IEAASSBP) SCAN(BLSVASSB)

      EXIT FORMAT(ASSB) EP(CSVPDLCB)
      EXIT FORMAT(ASSB) EP(IAZJSABP)

    DATA STRUCTURE(AST) FIND(BLSSASTE) GROUP(ASTE) MODEL(IEAASTEP) SCAN(+
         BLSVASTE)
    DATA STRUCTURE(ASTE) FIND(BLSSASTE) MODEL(IEAASTEP) SCAN(BLSVASTE)
    DATA STRUCTURE(ASVT) FIND(BLSSASVT) SCAN(BLSVASVT)
    DATA STRUCTURE(ASXB) FIND(BLSSASXB) MODEL(IEAASXBP) SCAN(BLSVASXB)

    DATA STRUCTURE(BLSLNTRC) SCAN(BLSVNTRC)
    DATA STRUCTURE(BLSQXBT) FIND(BLSSXBT) SCAN(BLSVXBT)
    DATA STRUCTURE(BLSRARQ) SCAN(BLSVARQ)
    .
```

```
DATA STRUCTURE(CACHE) MODEL(ILRMCACH)
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(CIBAL) FIND(IATIFBAL) MODEL(IATIPBAL)
.
.
.
DATA STRUCTURE(STORESTATUS)

  EXIT CBSTAT(STORESTATUS) EP(IEAVNIPW)
  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(BLSAFLG)
  EXIT CBSTAT(TCB) EP(IEAVTRCA)
  EXIT CBSTAT(TCB) EP(IEAVG701)
  EXIT CBSTAT(TCB) EP(BPXGMCBS)

  EXIT FORMAT(TCB) EP(IECDAFMT)
  EXIT FORMAT(TCB) EP(IECIOFMT)
  EXIT FORMAT(TCB) EP(IEAVTFMT)
  EXIT FORMAT(TCB) EP(IEAVD30)
  EXIT FORMAT(TCB) EP(IEAVXD01)
  EXIT FORMAT(TCB) EP(IEAVSSA1)

DATA STRUCTURE(TDCM) 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(BLSSECSA)
DATA AREA(EFLPA) FIND(BLSSEFLP)
DATA AREA(EMLPA) FIND(BLSSEMLP)
DATA AREA(ENUCLEUS) FIND(BLSSENUC)
DATA AREA(EPLPA) FIND(BLSSEPLP)
DATA AREA(ESQA) FIND(BLSSESQA)

DATA AREA(FLPA) FIND(BLSSFLPA)
```

```
DATA AREA(MLPA) FIND(BLSSMLPA)

DATA AREA(NUCLEUS) FIND(BLSSNUC)

DATA AREA(PLPA) FIND(BLSSPLPA)
DATA AREA(PRIVATE) FIND(BLSSPRIV)
DATA AREA(PRIVATEX) FIND(BLSSPRIX)

DATA AREA(RONUCLEUS) FIND(BLSSRONU)

DATA AREA(SQA) FIND(BLSSSQA)

EXIT ANALYZE EP(IARZANAL)
EXIT ANALYZE EP(IEAVESLX)
EXIT ANALYZE EP(IEFAB4WX)
EXIT ANALYZE EP(IOSVFMTH)
EXIT ANALYZE EP(ISGDCONT)
EXIT ANALYZE EP(IXCFMLAN)

EXIT VERB(ALCWAIT) EP(IEFAB4WX) HELP(IEFAB4WP) ABSTRACT('Allocation wait +
    summary')
EXIT VERB(AOMDATA) EP(AOMIPCS) 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(ASMCHECK) HELP(ILRASMCH) ABSTRACT('Auxiliary storage paging +
    activity') PARM('PGM(BLSGSCMD) PARM(ASMCHECK TERMINAL NOPRINT)')
EXIT VERB(ASMDATA) EP(ILRFTMAN) HELP(ILRASMDH) ABSTRACT('ASM control +
    block analysis')
EXIT VERB(AVMDATA) EP(AVFRDFMT) HELP(AVFHELP) ABSTRACT('AVM control +
    block analysis')

EXIT VERB(CICSDATA) EP(DFHPDX) ABSTRACT('CICS analysis')
EXIT VERB(CICS212) EP(DFHPD212) ABSTRACT('CICS Version 2 Release 1.2 +
    analysis')
EXIT VERB(CICS321) EP(DFHPD321) ABSTRACT('CICS Version 3 Release 2.1 +
    analysis')
EXIT VERB(CICS330) EP(DFHPD330) ABSTRACT('CICS Version 3 Release 3 +
    analysis')
EXIT VERB(CICS410) EP(DFHPD410) ABSTRACT('CICS Version 4 Release 1 +
    analysis')
.
.
.
EXIT VERB(VTAMMAP) EP(ISTRAFD1) ABSTRACT('VTAM control block analysis')

DIALOG NAME(XESDATA) HELP(IXLHDIA) ABSTRACT('XES analysis') PARM('PANEL(+
    IXLF1PMN)')
```

```
  PANDEF SUBCOMD(CTRACE) COMPONENT(SYSXCF) INPUT(IXCF1PTI) HELP(IXCHTDI)
  PANDEF SUBCOMD(CTRACE) COMPONENT(SYSXES) INPUT(IXLF1PTI) HELP(IXLHTDI)
  PANDEF SUBCOMD(CTRACE) COMPONENT(SYSOMVS) INPUT(BPXCTPAN) HELP(BPXHCTP1)

  SYMBOL NAME(AFT) STRUCTURE(AFT)
  SYMBOL PREFIX(ASCB) SUFFIX(COUNT1) STRUCTURE(ASCB)
  SYMBOL NAME(ASMVT) STRUCTURE(ASMVT)
  SYMBOL PREFIX(ASSB) SUFFIX(COUNT1) STRUCTURE(ASSB)
  SYMBOL PREFIX(AST) SUFFIX(COUNT0) STRUCTURE(ASTE)
  SYMBOL PREFIX(ASTE) SUFFIX(COUNT1) STRUCTURE(ASTE)
  SYMBOL NAME(ASVT) STRUCTURE(ASVT)
  SYMBOL PREFIX(ASXB) SUFFIX(COUNT1) STRUCTURE(ASXB)

  SYMBOL NAME(BLSQXBT) STRUCTURE(BLSQXBT)
  SYMBOL PREFIX(BLSQXBT) SUFFIX(COUNT1) STRUCTURE(BLSQXBT)

  SYMBOL PREFIX(BLSUZZ1) SUFFIX(COUNT1) STRUCTURE(BLSUZZ1)

  SYMBOL PREFIX(CDE) SUFFIX(NAME) STRUCTURE(CDE)
  .
  .
  .
  SYMBOL PREFIX(TCB) SUFFIX(DUALCOUNT) STRUCTURE(TCB)
  SYMBOL NAME(TITLE)

  SYMBOL PREFIX(UCB) SUFFIX(UNIT) STRUCTURE(UCB)
  SYMBOL NAME(UCM) STRUCTURE(UCM)

  SYMBOL PREFIX(WTRFSCB) SUFFIX(COUNT1NAME) STRUCTURE(WTRFSCB)

  SYMBOL PREFIX(XL) SUFFIX(NAME) STRUCTURE(XTLST)

  SYMBOL PREFIX(Z) SUFFIX(COUNT1)

Master BLSUZZ2 at 00038AB0 and BLSUZZ7 at 000399F0 for TCB at 008CC988

Processing BLSUZZ2 at 00040000 and BLSUZZ7 at 0DAB8000 for TCB at 008CC6F0



  BLSRZZ6 at 0005C000 for ACTIVE

    BUFPOOL(0DB08C80) BUFL(4160) BUFNO(2)

  BLSRZZ6 at 00072000 for DSNAME('H44IPCS.W4BG.PMR0X980.B519.MD520')
    BUFPOOL(0DD2A100) BUFL(24960) BUFNO(42)
    INUSE(42) REQUESTS(2225) READS(117)
```

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 ]
```

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

Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ ACTIVE | MAIN | STORAGE            ]
[ DSNAME(dsname) | DATASET(dsname) ]
[ 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 01:06:03.318 on 05/22/2001
Release z/OS 01.01.00
Used LOADT1 in SYS1.PARMLIB on 021D
IEASYM LIST=(TS,L)
IEASYS LIST=TS (OP)
IODF device 021D
IPL device 021D volume PRIPK2
```

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

The syntax of the IPCS ISPEXEC subcommand is the same as the syntax of the ISPF ISPEXEC command. The ISPEXEC command is documented in *ISPF Examples*.

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 5-223.

```
[ 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 entered. 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.

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.

```
TITLE
LIST 00000000. HEADER POSITION(X'+0020') LENGTH(19) CHARACTER
00000020. | HANG AFTER HOTSTART                            |
```

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

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:

To Get This Result	Make This Request
Absolute PSA	list 0:0fff absolute
Real PSAs for each central processor	list 0:0fff cpu(n) real
Absolute storage above the PSA	list 1000:7fffffff absolute

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) | INFILE(ddname) ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
            [ 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:

Attribute	Explanation
ABSOLUTE	Represents a storage frame that was in processor storage when a stand-alone dump was requested.
COMMON	Represents common virtual storage.

MISSING Represents storage not available in the dump.

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:

```
00F0C000-00F0EFFF
```

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.

INDATASET(dsname)
INDSNAME(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.

DSNAME or DATASET 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
```

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.

```
Source of Dump                                Blocks            Bytes
ACTIVE   . . . . . . . . . . . . . . . . . .    2  . . . . . 8,208
FILE(CLIC)   . . . . . . . . . . . . . . . . .  10  . . . . . 2,640
FILE(MENUS)  . . . . . . . . . . . . . . . . .  10  . . . . . 2,640
DSNAME('D46RLW1.LOG.MISC')   . . . . . . . . . .  79  . . . . 187,809
DSNAME('D46RLW1.RLW.CLIST')  . . . . . . . . . .  25  . . . . . 6,600
DSNAME('D46RLW1.SYSMDUMP')   . . . . . . . . .   1,218  . . . 4,998,672
DSNAME('D46RLW1.XMIT.NAMES')   . . . . . . . . .   2  . . . . . 3,520
DSNAME('D46RLW1.THIS.IS.A.LONG.DSNAME.FOR.TESTING(TESTMEMB)')
                                             1,040  . . . .  31,200
DSNAME('D83DUMP.DUMPC.PB06511')  . . . . . .   1,346  . . . 5,523,984
DSNAME('D83DUMP.DUMPC.PB07251')  . . . . . .  24,142  . .  99,078,768
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.

```
Source of Dump                              Blocks              Bytes
DSNAME('RLW.HBB5520.SAMPLE.SVCDUMP')    . .   994  . . . . 4,135,040
  ABSOLUTE
    00A95000:00A95FFF ABSOLUTE
    00CD3000:00CD3FFF ABSOLUTE
    010A7000:010A7FFF ABSOLUTE
    012A8000:012A8FFF 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
    00F4A000:00F4DFFF COMMON TRANSLATED
    00F8B000:00F90FFF COMMON TRANSLATED
    00F9C000:00FA0FFF COMMON TRANSLATED
    00FA6000:00FA6FFF COMMON TRANSLATED
    00FBF000:00FC9FFF COMMON TRANSLATED
    00FCF000:00FD2FFF COMMON TRANSLATED
    00FD3000-00FD5FFF COMMON MISSING TRANSLATED
    00FD6000:00FD6FFF COMMON TRANSLATED
        .
        .
        .
    020F6000:02106FFF COMMON TRANSLATED
  1,589,248, X'00184000', bytes described in ASID(X'0001')

  ASID(X'0004')
    7F735000:7F745FFF
  69,632, X'00011000', bytes described in ASID(X'0004')
```

```
ASID(X'0015')
  00000000:00000FFF COMMON PREFIXED
  006D4000:006D5FFF
  006DC000:006DCFFF
  006E2000:006E3FFF
  006EA000:006FFFFF
  00AF2000:00B0CFFF COMMON
  00F0C000:00F4DFFF COMMON
      .
      .
      .
  7FFFB000:7FFFEFFF
3,928,064, X'003BF000', bytes described in ASID(X'0015')

HEADER
  00000000:0000103F
4,160, X'00001040', bytes described in HEADER

COMPDATA(IARCDR01)
  00000000:00000FFF
4,096, X'00001000', bytes described in COMPDATA(IARCDR01)

DOMAIN(SUMDUMP)
  00001000:00006FFF
24,576, X'00006000', bytes described in DOMAIN(SUMDUMP)

ASID(X'0015') SUMDUMP
  01E8CDD8:01E8CDF3 COMMON SUMLIST
  01EB9000:01EB91FF COMMON SUMLIST
  7F702F80:7F702FFF SUMLIST
  7F704000:7F708FFF SUMLIST
  7FFFB040:7FFFB07F SUMLIST
  7FFFC008:7FFFC037 SUMLIST
21,260, X'0000530C', bytes described in ASID(X'0015') SUMDUMP

1 Dump described
```

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.

```
Source of Dump                                      Blocks              Bytes
ACTIVE                                                   2              8,320
  No title
  No symptoms

Source of Dump                                      Blocks              Bytes
DSNAME('C89.BLSRMVCL.SOC4DUMP')                     26,544        110,423,040
  Title=JOBNAME C89      STEPNAME SMPROC  SMPROC   SYSTEM 0C4
  Psym=RIDS/BLSRVEC3#L RIDS/BLSRMVCL PIDS/5752SC132 AB/S00C4 RIDS/BLSUSTAI#R V

Source of Dump                                      Blocks              Bytes
DSNAME('H44IPCS.R38A.PMR00137.B379.EH603')          12,762         53,089,920
  Title=COMPON=COMPONENT TRACE,COMPID=SCTRC,ISSUER=ITTAWRIT
  Psym=RIDS/NUCLEUS#L RIDS/ITTAWRIT PIDS/5752SCTRC AB/S001D RIDS/ITTAWRIT#R VA

Source of Dump                                      Blocks              Bytes
DSNAME('H44IPCS.R38A.PMR00137.B379.EH603A')        573,996      2,387,823,360
  Title=SLIP DUMP ID=0005
  Trap=SLIP SET,COMP=01D,NUCMOD=IARDS,DN=(3.*,15.SYSLOGR0),SD=(ALLNUC,PSA,SQA,

Source of Dump                                      Blocks              Bytes
DSNAME('H44IPCS.R38A.PMR00218.B677.DUMP1')          10,574         43,987,840
  Title=SLIP DUMP ID=X05C
  Trap=SLIP SET,C=05C,ID=X05C,A=SVCD,RE=308

Source of Dump                                      Blocks              Bytes
DSNAME('NHAN.FBS29K.DUMP')                         438,123      1,822,591,680
  Title='MVSPROD1 02/27/97'
  No symptoms

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/BSLRVEC3#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, Vol 2 (DCCB-ITZYRETC)* for information about the EDT.

Syntax

```
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)] ]

          [ GRPPTR[(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 5-223.
           [ 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.

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.

GRPPTR[(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.

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 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.

For information about using the storage map, see the *z/OS MVS IPCS User's Guide*.

Related Subcommands

DROPMAP
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 5-223.

```
             [ 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.

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 }  [ (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 5-223.

```
            [ 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 Appendix B, "IPCS Symbols" on page B-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.

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       1D418. ASID(X'0001') POSITION(-24) LENGTH(1248) STRUCTURE(CVT) DROP
MY#LONG#SYMBOLIC
               0. ASID(X'0078') LENGTH(96) AREA DROP
MYARRAY    F0000. ASID(X'0078') POSITION(+64) LENGTH(4) ENTRIES(52:77)
                  SIGNED DROP
MYCVT      1D418. ASID(X'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 usually displayed on one line. The format resembles that of the EQUATE subcommand parameters.
- When the symbol and the address would overlap, if both were 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 GMT and local time stamps using the time-zone adjustments from your dump.

Syntax

```
{ LISTTOD }
{ LTOD    }

             gmt-tod-value

             [ EXTENDED ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
             [ ACTIVE|MAIN|STORAGE    ]
             [ DSNAME(dsname)|DATASET(dsname) ]
             [ FILE(ddname)|DDNAME(ddname)   ]
             [ PATH(hfspath)                     ]

             [ FLAG(severity) ]

             [ PRINT | NOPRINT ]

             [ TERMINAL | NOTERMINAL ]

             [ TEST | NOTEST ]
```

Parameters

gmt-tod-value
Specifies the first 1-32 hexadecimal digits of a GMT TOD clock value associated with a dump.

EXTENDED
Specifies that the value should be treated as a 32-byte STCKE value rather than an 8-byte TOD clock value as would be stored by the STCK instruction.

Return Codes

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

Example

Display the translated TOD clock value from a dump.

– Action

```
listtod 123
```

– Result

```
ltod 123

11/07/2052 05:05:11.053312 GMT
11/07/2052 01:05:11.053312 LOCAL
```

The subcommand specifies the first three digits of the TOD clock value.

LISTUCB Subcommand — List UCBs

Use the LISTUCB subcommand as a convenient means to request the display of one or more unit control blocks (UCBs).

LISTUCB Subcommand

Syntax

```
{ LISTUCB }   (device-number-list)
{ LISTU   }
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ 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.
  - 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 desired 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 UCB for device number 8000.

– Action

```
COMMAND ===> LISTUCB 8000
```

– Result

```
UCBPRFIX: 01D0E020
   -0008  LOCK..... 00000000  IOQ...... 00000000
UCBOB: 01D0E028
   +0000  JBNR..... 00        FL5...... 00        ID....... FF
   +0003  STAT..... 00        CHAN..... 8000      FL1...... 00
   +0007  FLB...... 20        NXUCB.... 00000000  WGT...... 06
   +000D  NAME..... UCB       TBYT1.... 00        TBYT2.... 00
   +0012  DVCLS.... 41        UNTYP.... 01        FLC...... 00
   +0015  EXTP..... D0E001    CTCAL.... 00000000  CTCF1.... 00
   +001D  RV042.... 000000    CTCWA.... 00000000  RSV...... 00000000
UCBCMXT: 01D0E000
   +0000  ETI...... 19        STI...... 00        FL6...... 00
   +0003  ATI...... 2C        SNSCT.... 02        FLP1..... 00
   +0006  STLI..... 00        FL7...... 00        IEXT..... 01D55B68
   +000C  CHPRM.... 00        SATI..... 00        ASID..... 0000
   +0010  RSV...... 00        WTOID.... 000000    DDT...... 00FCCCC0
   +0018  CLEXT.... 00000000  DCTOF.... 0000      RSV...... 0000

UCBXPX: 01D55B68
   +0000  RSTEM.... 00        MIHKY.... 07        MIHTI.... 00
   +0003  HOTIO.... 00        IOQF..... 00000000  IOQL..... 00000000
   +000C  SIDA..... 0000      SCHNO.... 0000      PMCW1.... 0000
   +0012  MBI...... 0000      LPM...... 00        RSV...... 00
   +0016  LPUM..... 00        PIM...... 00        CHPID.... 00000000
   +001C            00000000  LEVEL.... 01        IOSF1.... 40
   +0022  IOTKY.... 00        MIHFG.... 00        LVMSK.... 00000001

 ACTUAL UCB COMMON SEGMENT ADDRESS 01D0E028

 DEVICE IS DYNAMIC
```

Note: Even if you are using captured UCBs, LISTUCB displays the actual UCB common segment address.

LITERAL Subcommand — Assign a Value to a Literal

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 ADDDUMP 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 5-223.

```
[ 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

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 5-223.
              [ 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 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

LPAMAP Subcommand

```
LPAMAP  [ EPA     ]
        [ MODNAME ]
        [ ALL     ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

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


[ FLAG(severity) ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ TEST | NOTEST ]
```

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.

```
           * * *  L I N K   P A C K   A R E A   M A P  * * *
           SORTED ALPHABETICALLY BY MODULE NAME

NAME      EPA       ADDRESS   LENGTH    MAJOR     NOTE
--------  --------  --------  --------  --------  ------------
ADYPRED   81F9C260  01F9C260  00000DA0
AHLACFV   81B9B95C                      AHLTVTAM
AHLDMPMD  81B49F5C                      AHLSETD
AHLDSP    81B8A968                      AHLTXSYS
AHLEXT    81F6765E                      AHLTSYSM
AHLFFP    81B44000                      AHLFVEC
AHLFIO    81F7C0CC                      AHLTSYFL
AHLFPI    81F7C1A2                      AHLTSYFL
AHLFRR    81F677E8                      AHLTSYSM
AHLFSSCH  81F7C0EC                      AHLTSYFL
AHLFSVC   81F7C17E                      AHLTSYFL
AHLFVEC   81B44000  01B44000  00004C38
AHLMCER   81B494C0                      AHLSETD
AHLPINT   81F67746                      AHLTSYSM
AHLREADR  81C7FBE8  01C7FBE8  00000418
AHLRNIO   81B8A858                      AHLTXSYS
AHLSBCU1  81B92F5E                      AHLWSMOD
AHLSBLOK  81B926C0                      AHLWSMOD
AHLSBUF   81B92A88                      AHLWSMOD
AHLSETD   81B49000  01B49000  000017B0
AHLSETEV  81B4B000  01B4B000  000019C0
AHLSFEOB  81B927FE                      AHLWSMOD
AHLSRB    81B8A9F4                      AHLTXSYS
AHLSRM    81B8AA68                      AHLTXSYS
AHLSTAE   81F678C6                      AHLTSYSM
AHLSVC    81F67618                      AHLTSYSM
AHLTACFV  81B9B968                      AHLTVTAM
AHLTCCWG  81B4D000  01B4D000  00002468
AHLTDIR   81B49AF8                      AHLSETD
AHLTDSP   81B5463E                      AHLTPID
AHLTEXT   81F75928  01F75928  000006D8
AHLTFCG   81B50000  01B50000  000016F0
AHLTFOR   81EBD570  01EBD570  00000A90
AHLTFRR   81EBD692                      AHLTFOR
AHLTLSR   81B547CA                      AHLTPID
AHLTPI    81B5445E                      AHLTPID
AHLTPID   81B54448  01B54448  00000BB8
AHLTRNIO  81EBD570                      AHLTFOR
AHLTSLIP  81B52000  01B52000  00002440
AHLTSRB   81B5475C                      AHLTPID
AHLTSRM   81EBD58A                      AHLTFOR
AHLTSTAE  81EBD7B4                      AHLTFOR
AHLTSVC   81B55000  01B55000  00002AC8
AHLTSYFL  81F7C0B0  01F7C0B0  000006F0
AHLTSYSM  81F67508  01F67508  00000AF8
AHLTUSR   81B84978  01B84978  00000688
AHLTVTAM  81B9B940  01B9B940  000006C0
```

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, GTFTRACE subcommands, or both, into chronological order in a single report. Use the MERGEEND subcommand to stop merging traces.

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.

```
********** 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=SI
 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

      MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
      --------  --------  ---------------  -----------
 02.  XEPEXIT   00000002  14:18:40.000001  External Entry Point Exit
      FUNC1... FLTAEPAG          Enabled Addr Space Page Faults
      JOBN1... EDWTR1   ASID1... 0014     PLOCKS.. 00000000 CPU.... 0001
      JOBN2... EDWTR1   ASID2... 0014     RLOCKS.. 00000000
 SRM              ASCB.... 00FD2E00 CPU..... 0001     JOBN.... *MASTER*
                  R15..... 00000000 R0...... 00010005 R1...... 00000000
 01.        GMT-07/23/90 14:18:40.120000    LOC-07/23/90 14:18:40.120000

 02.  XEPENTRY  00000001  14:18:40.130594  External Entry Point Entry
      FUNC1... GENIOCMP          General I/O Completion
      JOBN1... JES2     ASID1... 0012     PLOCKS.. 08000081 CPU.... 0001
      JOBN2... *ALL*    ASID2... FFFE     RLOCKS.. 08000000
 SRM              ASCB.... 00FD2E00 CPU..... 0001     JOBN.... *MASTER*
                  R15..... 00000000 R0...... 00010005 R1...... 00000000
 01.        GMT-07/23/90 14:18:40.142505    LOC-07/23/90 14:18:40.142505

 SRM              ASCB.... 00FD2E00 CPU..... 0001     JOBN.... *MASTER*
                  R15..... 00000000 R0...... 00010005 R1...... 00000000
 01.        GMT-07/23/90 14:18:40.814690    LOC-07/23/90 14:18:40.814690


 02.  PAGEA2R   0000001D 14:18:40.891890  Page Request Auxiliary to Real
      FUNC1... GENIOCMP          General I/O Completion
      FUNC2... FLTAEPAG          Enabled Addr Space Page Faults
      JOBN1... JES2     ASID1... 0012     PLOCKS.. 08004081 CPU.... 0001
      JOBN2... EDWTR1   ASID2... 0014     RLOCKS.. 08004000
 SRM              ASCB.... 00FD2E00 CPU..... 0001     JOBN.... *MASTER*
                  R15..... 00000000 R0...... 00010005 R1...... 00000000
 01.        GMT-07/23/90 14:18:40.901011    LOC-07/23/90 14:18:40.901011
```

```
02.  XEPEXIT   00000002  14:18:40.952534  External Entry Point Exit
     FUNC1... GENIOCMP          General I/O Completion
     JOBN1... JES2     ASID1... 0012     PLOCKS.. 08004081 CPU.... 0001
     JOBN2... *ALL*    ASID2... FFFE     RLOCKS.. 08004000
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       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 5-223.

```
            [ ACTIVE | MAIN | STORAGE           ]
            [ DSNAME(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')
```

– Result

NAME produces a listing that displays the address space, data space, or subspace associated with the hexadecimal STOKEN value, 11223344 55667788.

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 5-223.

```
            [ 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))
```

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 0 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))
```

– Results

The following output is produced.

```
System level
TOKEN.... SYSTLEV_NAME_003_token
NAME..... SYSTLEV_NAME_003_name
ASID..... 000F
Persistent
Created by authorized program
```

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
00001 00000000 ASID(X'0010')                          AREA
      Remarks:
```

– Action 4

  Add a pointer entry that has the address from field 1 in the NAMETOKN output.

– Results

```
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  E2E8E2F1  4BD3D6C7  D9C5C340  40404040  | SYS1.LOGREC |
01CE0030 TO 01CE004F (X'00000020' bytes)--All bytes contain X'40', C' '
```

NOTE Subcommand — Generate a Message

Use the NOTE subcommand to direct messages to the IPCSPRNT data set, your terminal, or both, 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.

Thus, a message may be truncated to a different length for each destination.

NOTE directs the message to the IPCSPRNT data set, your terminal, or both, depending on the PRINT and TERMINAL parameters. If you omit the PRINT and TERMINAL parameters, NOTE uses the current local defaults for these parameters.

You can also assign a message severity level, which determines whether or not 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 }    ['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 5-223.

```
             [ FLAG(severity) ]

             [ PRINT | NOPRINT ]

             [ TERMINAL | NOTERMINAL ]

             [ 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 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 data set.

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.

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 quotes if desired. 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 5-223.
              [ 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

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.

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 OMVSDATA subcommand.

Example

See *z/OS MVS Diagnosis: Reference* for examples of the OMVSDATA 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

-------- Data Selection Parameters ----------------------------
search-argument

-------- Result Distribution Parameters ----------------
            [ CLIST(var-list)     ]
            [ DIALOG(var-list)     ]
            [ REXX(var-list)        ]

            [ LIST ]
            [ NOLIST                  ]
            [ SCREEN               ]

-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.
            [ PRINT | NOPRINT ]

            [ TERMINAL | NOTERMINAL ]

            [ TEST | NOTEST ]
```

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.

CLIST(*var-list*)
DIALOG(*var-list*)
REXX(*var-list*)
Requests that the information retrieved be made available to a command procedure or ISPF dialog.

The syntax for *var-list* is as follows:

```
MNEMONIC(variable-name)
```

LIST
NOLIST
SCREEN
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.

Example

In z/OS V1R4, IPCS enhances the display of the multi-byte operation codes associated with z/Architecture. The split-opcode intructions beginning with 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          ]
       [ DSNAME(dslist) | DATASET(dslist) ]
       [ FILE(ddlist) | DDNAME(ddlist)    ]
       [ PATH(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 5-223.

```
       [ TEST | NOTEST ]

       [ CONFIRM | NOCONFIRM ]
```

Parameters

ACTIVE or MAIN or STORAGE

DSNAME(dslist) or DATASET(dslist)
FILE(ddlist | IPCSDDIR) or DDNAME(ddlist)

Specifies the source to be prepared for use. If one of these parameters in 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 on default values and restrictions for dump directories, see the CLOSE subcommand.

PATH(path-name ...)

Specifies one or more HFS paths to be processed. The PATH(path-name ...) option permits a a list of path names to be processed in addition to any *ddnames* and *dsnames* listed on the subcommand. Partially-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:

```
[ 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 quotes.

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 quotes.

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 and/or TOC data sets
– Data that has a data-type attribute of AREA

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 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.
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 would 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 prior to application of a patch, recording conclusions regarding that data in the dump directory prior to application of a patch. The PATCH subcommand does not attempt to locate and alter any such data. Some of this data may be affected via 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 via 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), via 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:

Qualifier	Function
ADD	Causes the PATCH subcommand to store a new patch. See “Adding or Replacing a Patch” on page 5-188 for more information. Existing, overlapping patches are considered to be an error and cause the new patch to be rejected.
DELETE	Causes the PATCH subcommand to delete patches. See “Deleting Patches” on page 5-188 for more information.
LIST	Causes the PATCH subcommand to list patches. See “Listing Patches” on page 5-189 for more information.
REPLACE	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 5-188 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.
STORE	Causes the PATCH subcommand to store a patch, replacing any

existing ones whose descriptions overlap the new one. See "Adding or Replacing a Patch" for more information.

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.

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

```
PATCH       LIST

            [ data-descr ]

            [ DETAIL ]
```

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

```
[ TEST | NOTEST ]
```

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 would be: – Specifying PATCH ADD processing for a location where a patch has already been applied. – Use of the TSO attention mechanism to terminate PATCH processing when IKJPARS solicits operand correction.
X'10'	Request not completed due to problems with the IPCS execution environment. Examples of such problems would be: – 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 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:

```
/*---------------------- IPCS Profile Data ------------------------*/
PROFILE NOLINESIZE            /* Limit for variable-width reports  */
PROFILE NOPAGESIZE            /* Line limit for print file pages   */
PROFILE STACK(NODUPLICATES)   /* Duplicate stack entry screening   */
```

Figure 5-1. 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,

Syntax

```
{ PROFILE } [ LINESIZE(nnn) | NOLINESIZE ]
{ PROF    }
            [ PAGESIZE(nnn) | NOPAGESIZE ]

            [ LIST | NOLIST ]

            [ STACK {(DUPLICATES | NODUPLICATES)} ]
```

-------- SETDEF-Defined Parameter ----------
Note: You can override the following SETDEF parameter.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
            [TEST | NOTEST ]
```

Parameters

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']xxx'), or binary (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']xxx'), or binary (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.

NOPAGESIZE is the default.

LIST or NOLIST

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 5-1 on page 5-191.

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:

```
/*---------------------- 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

```
{RENUM }    [ SUMMARY | NOSUMMARY ]
{REN   }
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

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

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)
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
- Data-in-virtual mapped range report
- Data space report
- Exception report
- Execution status report
- Expanded storage report
- High Virtual page report
- Real frames report
- RSM requests report
- RSM shared data report
- Subspace report
- Summary report
- 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

-------- Report Type Parameters ----------------------------
                { ADDRSPACE  }
                { DIVMAP     }
                { DSPACE     }
                { EXCEPTION  }
                { EXECUTION  }
                { EXPFRAME   }
                { HIGHVIRTUAL }
                { REALFRAME  }
                { RSMREQ     }
                { SHRDATA    }
                { SUBSPACE   }
                { SUMMARY    }
                { VIRTPAGE   }
```

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

```
[ COMMON ]

[ DATASPACES ]

[ PERMCOMM ]

[ RANGE(rangelist) ]

[ SAVEAREA(address) ]

[ SHARED ]

[ STATUS(statuslist) ]

[ TOKEN(token) ]

[ TOTONLY ]

[ VFETCH ]
```

-------- Address Space Selection Parameters ----------

```
[ ALL ]

[ ASIDLIST(asidlist) ]

[ CURRENT ]

[ ERROR ]

[ JOBLIST(joblist) | JOBNAME(joblist) ]
```

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

Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ 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. 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. 5-199 summarizes the parameters you can specify with a given report.

ADDRSPACE

Requests the RSM address spaces report. This report summarizes real and expanded 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 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:
- 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.

EXPFRAME

Requests the expanded storage report. This report displays information about the expanded storage frames within the system. The information may be sorted by ASID, by the range of the expanded storage frames, and by various statuses of the expanded storage frames. You may also specify that common area expanded storage frames or expanded storage frames associated with virtual fetch be displayed.

Usage Note: The only data selection parameters that apply to this report are COMMON, RANGE, SHARED, STATUS, TOTONLY and VFETCH.

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.

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, 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 requestor, 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, STATUS, and TOTONLY.

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 expanded 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, DATASPACES, 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 DATASPACES 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.

Report Type Parameter	ALL ASIDLIST CURRENT ERROR JOBLIST/ JOBNAME	COMMON	DATASPACES	PERMCOMM	RANGE
ADDRSPACE	X				
DIVMAP	X				
DSPACE	X				
EXCEPTION	X		X		
EXECUTION					
EXPFRAME	X	X			X
HIGHVIRTUAL	X				X
REALFRAME	X	X		X	X
RSMREQ	X	X			
SHRDATA	X	X			
SUBSPACE	X				X
SUMMARY					
VIRTPAGE	X	X	X	X	X

Report Type Parameter	SAVEAREA	SHARED	STATUS	TOKEN	TOTONLY	VFETCH
ADDRSPACE			X		X	
DIVMAP			X		X	
DSPACE					X	
EXCEPTION	X					
EXECUTION	X					
EXPFRAME		X	X		X	X
HIGHVIRTUAL			X		X	
REALFRAME		X	X		X	
RSMREQ		X	X		X	
SHRDATA			X	X	X	
SUBSPACE			X			
SUMMARY						
VIRTPAGE			X		X	

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.

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 EXPFRAME, 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.)

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 EXPFRAME, REALFRAME, SUBSPACE, and VIRTPAGE parameters.

The *rangelist* is one or more ranges. In each range, the lower and upper limits are separated by a colon.

The value to specify for *rangelist* depends on the report:

Report Parameter	Value for rangelist
EXPFRAME	Hexadecimal expanded frame numbers from 0 to the number of expanded frames in the system (up to 8 hexadecimal digits).
HIGHVIRTUAL	Hexadecimal virtual addresses from 80000000 to FFFFFFFF_FFFFFFFF. The default range for this report is 1_00000000:1_80000000. **Note:** Each range limit can be 17 characters each and may contain underscores.
REALFRAME	Hexadecimal real frame numbers from 0 to the number of real frames in the system (up to 8 hexadecimal digits).
VIRTPAGE	Hexadecimal virtual addresses from 0 to 7FFFFFFF.
SUBSPACE	Hexadecimal virtual addresses from 0 to 7FFFFFFF.

Note: Hexadecimal notation (X'n...') is optional, that is, 7FFF as opposed to X'7FFF'.

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.

SHARED
Requests that the report contain information about data defined as shared. Use SHARED to select data in the EXPFRAME, REALFRAME and RSMREQ reports.

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
Indicates that you want to see the address spaces that are non-swappable.

SWAUX
Indicates that you want to see the address spaces that are swapped to auxiliary storage.

SWAUXIP
Indicates that you want to see the address spaces that are in the process of being swapped to auxiliary storage.

SWEXP
Indicates that you want to see the address spaces that are swapped to expanded storage.

SWEXPIP
Indicates that you want to see the address spaces that are in the process of being swapped to expanded storage.

SWIN
Indicates that you want to see the address spaces that are swapped in.

SWINIP
Indicates that you want to see the address spaces that are in the process of being swapped in.

TERM
Indicates that you want to see the address spaces that are in the process of terminating.

– *Object states for DIVMAP report:*

MAPIP
Indicates that you want to see the data-in-virtual mapped ranges that are involved in a DIV MAP request

MAPRPIP
Indicates that you want to see the data-in-virtual mapped ranges that are involved in a DIV MAP-reprime request

UNMAPIP
Indicates that you want to see the data-in-virtual mapped ranges that are involved in a DIV UNMAP request

SAVEIP
Indicates that you want to see the data-in-virtual mapped ranges that are involved in a DIV SAVE request

RESETIP
Indicates that you want to see the data-in-virtual mapped ranges that are involved in a DIV RESET request

MAPPED
Indicates that you want to see the data-in-virtual mapped ranges that are not involved in a DIV request

– *Object states for EXPFRAME report:*

ALLOC
Indicates that you want to see the frames that are allocated.

ALLOCSM
Indicates that you want to see only frames backing pages of shared segments.

AVAIL
Indicates that you want to see available frames.

OFFLINE
Indicates that you want to see the frames that have been taken off-line

OFFINT
Indicates that you want to see the frames that will be taken off-line when freed from the current owner.

– *Object states for HIGHVIRTUAL report:*

AUX
Indicates that you want to see pages that have their most recent copies on a paging data set.

FREF
Indicates that you want to see pages that were in a first-reference state.

GUARD
Indicates that you want to see pages that are in the guard area of a memory object.

REAL
Indicates that you want to see pages that have their most recent copies in a real frame.

SIAI
Indicates that you want to see pages that are in the process of being swapped in from auxiliary storage.

SOAI
Indicates that you want to see pages that are in the process of being swapped out to auxiliary storage.

SWAX
Indicates that you want to see pages that have their most recent copies swapped to auxiliary storage.

– *Object states for REALFRAME report:*

ALLOC
Indicates that you want to see the frames that are allocated.

ALLOCSM
Indicates that you want to see only frames backing pages of shared segments.

ALLOCVR
Indicates that you want to see frames allocated to V=R jobs that are either running or waiting for additional frames.

AVAIL
Indicates that you want to see available frames.

OFFINT
Indicates that you want to see the frames that will be taken off-line when freed from the current owner.

OFFINTPL
Indicates that you want to see the frames that are off-line intercepted and currently in use by a job that is polluting the V=R area with a long term resident page.

OFFINTVR
Indicates that you want to see frames that are off-line intercepted and allocated to a V=R job.

OFFLINE
Indicates that you want to see frames that are off-line.

POLLUTE
Indicates that you want to see frames that are part of the V=R area, but are allocated to a long-term resident page that is not V=R.

VRINT
Indicates that you want to see frames that will be assigned to a waiting V=R job when freed from the current owner.

– *Object states for RSMREQ report:*

CANCEL
Indicates that you want to see any cancelled requests.

COMPLETE
Indicates that you want to see non-fast path PGSER FIX requests that have completed and are awaiting the corresponding PGSER FREE request.

DBLFRAME
Indicates that you want to see requests that are waiting for a real frame pair.

FAIL
Indicates that you want to see requests that had failures other than I/O or cross memory access failures.

FRAMEAA
Indicates that you want to see requests that are waiting for any type of real frame.

FRAMEAB
Indicates that you want to see requests that are waiting for a real storage frame that resides below 16 megabytes.

FRAMEPA
Indicates that you want to see requests that are waiting for a real frame that resides in the preferred area.

FRAMEPB
Indicates that you want to see requests that are waiting for a real frame that resides in the preferred area below 16 megabytes.

INPROGR
Indicates that you want to see 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
Indicates that you want to see requests that had I/O failures.

PGREAD
Indicates that you want to see requests that are waiting for a page to be read in from a paging data set, or some other data set.

PGWRITE
Indicates that you want to see requests that are waiting for a page to be written to a paging data set or some other data set.

XMFAIL
Indicates that you want to see requests that had cross memory access errors.

– *Object states for SHRDATA report:*

AUX
Indicates that you want to see pages that have their most recent copies on a paging data set.

DSN
Indicates that you want to see pages that have their most recent copies on a data set containing the data-in-virtual object of which the pages are a part.

EXP
Indicates that you want to see pages that have their most recent copies in an expanded frame.

FREF
Indicates that you want to see 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
Indicates that you want to see pages that reside in a real frame. They are either valid or have output paging I/O in progress.

– *Object states for SUBSPACE report:*

GLOBAL
Indicates that you want to see the storage that is addressable by all subspaces within this address space.

ASSIGN
Indicates that you want to see 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
Indicates that you want to see the storage in the address space that is not assigned to any subspace.

– *Object states for VIRTPAGE report:*

AUX
Indicates that you want to see pages that have their most recent copies on a paging data set.

DSN
Indicates that you want to see pages that have their most recent copies on a data set containing the data-in-virtual object of which the pages are a part.

EXP
Indicates that you want to see pages that have their most recent copies in an expanded frame.

FREF
Indicates that you want to see 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 using PGSER.
- It was released using DSPSERV.

MIG
Indicates that you want to see 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
Indicates that you want to see pages that have their most recent copies in a real frame.

SMEG
Indicates that you want to see pages that are part of a shared segment.

VIO
Indicates that you want to see 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
Indicates that you want to see pages that are in the process of being swapped in from auxiliary storage.

SIEI
Indicates that you want to see pages that are in the process of being swapped in from expanded storage.

SOAI
Indicates that you want to see pages that are in the process of being swapped out to auxiliary storage.

SOEI
Indicates that you want to see pages that are in the process of being swapped out to expanded storage.

SWAX
Indicates that you want to see pages that have their most recent copies swapped to auxiliary storage.

SWEX
Indicates that you want to see pages that have their most recent copies swapped to expanded storage.

SWMG
Indicates that you want to see 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.

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 ADDRSPACE, DIVMAP, DSPACE, EXPFRAME, REALFRAME, RSMREQ, SHRDATA, and VIRTPAGE tabular reports.

VFETCH
Requests that, for an EXPFRAME report, the expanded storage frames associated with virtual fetch should be included. Use VFETCH for an EXPFRAME report only.

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 ADDRSPACE, DIVMAP, DSPACE, EXCEPTION, EXPFRAME, 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

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(EXP)
```

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 and ERROR 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 CURRENT and ERROR address spaces.

– Action

```
COMMAND ===> RSMDATA RSMREQ
```

Example 4

Generate a report showing real and expanded 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
RUNCPOOL

Syntax

```
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 and/or parameters can accompany the CLIST, REXX exec, or IPCS subcommand. The symbol X will point to the current array entry prior to each EXEC invocation.

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.

RUNCHAIN Subcommand

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 | RUNC }

            [ data-descr | ADDRESS(X) ]

            [ AMASK(mask) ]

            [ CHAIN [(nnn|999)] ]

            [ DROP | NODROP ]

            [ EXEC((clist|rexx-exec|subcommand)) ]

            [ LINK(range)    ]

            [ MASK(mask) ]


            [ 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 5-223.

```
            [ 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. 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 and/or parameters can accompany the CLIST, REXX exec, or IPCS subcommand. The symbol X will point to the current control block on the chain prior to each EXEC invocation.

The EXEC parameter also accepts another RUNCHAIN invocation to process multiple levels of control blocks. See the BLSCRNC2 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)

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 is an unsigned integer. The end of the range may be omitted. 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 field before comparing it to the value specified with the NULL parameter. IPCS accepts 64-bit values and interprests 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 alphameric 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
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 thers 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.

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 could not 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 prior to processing the cells themselved, and an optional report may be formatted that identifies data areas used to manage the cell as well as 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  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           ]
[ 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

cpid-general-value
Specifies a fullword cell pool identifier (CPID used in conjunction with the CPOOL macro).

ASID(asid)
Specifies 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
Specifies whether cells in the pool that are in use are to be included in RUNCPOOL processing.

INDETERMINATE
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 and/or keywords can accompany the CLIST, REXX exec, or IPCS subcommand. The symbol X will point to the current cell on the chain prior to 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 - A Private Area CPOOL

Example 1 shows a small private area CPOOL in which all of the cells are currently unused.

```
runcpool 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 - Another 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.

```
runcpool x'008B6000' display

PPD at 7F7E8F10
  ASID(X'036E') CPID(X'008B6000') in loc(below) subpool(236)
  Csize(80) primary(101) secondary(102)
PXT at 008B6000

SPD at 7F7E8F48
SXT at 00887000
Cells(203) used(126)

CPOOLCELL - Cell in use
LIST 00887018 ASID(X'036E') LENGTH(80) AREA(CPOOLCELL)
ASID(X'036E') ADDRESS(00887018) KEY(10)
00887018.                                           C4E2C1C2 00887428 |                        DSAB.h..|
00887020. 008B7F68 00500000 008BC9C8 008A2070  00000000 00000000 0000CA00 00000000 |..".."..&;...IH.................|
00887040. 008F9E50 00000000 00000000 00000000  00000000 00000200 008A2080 00000066 |...&;..........................|
00887060. 00000200 00000000                                                       |........                        |

CPOOLCELL - Cell in use
LIST 00887068 ASID(X'036E') LENGTH(80) AREA(CPOOLCELL)
ASID(X'036E') ADDRESS(00887068) KEY(10)
00887068.                  C4E2C1C2 008870B8  008B7A18 00500000 008BC798 0089DB50 |        DSAB.h....:..&;...Gq.i.&|
00887080. 00000000 00000000 0000CA00 00000000  0086FD50 00000000 00000000 00000000 |................f.&;...........|
008870A0. 00000000 000003D8 0089DB60 00000067  000003D8 00000000                   |.......Q.i.-.......Q....        |

CPOOLCELL - Cell in use
LIST 008870B8 ASID(X'036E') LENGTH(80) AREA(CPOOLCELL)
ASID(X'036E') ADDRESS(008870B8) KEY(10)
008870B8.                                           C4E2C1C2 00887748 |                        DSAB.h..|
008870C0. 00887068 00500000 008BC7AC 0089D6A0  00000000 00000000 0000CA00 00000000 |.h...&;...G..iO................|
008870E0. 0086FD50 00000000 00000000 00000000  00000000 000003E0 0089D6B0 00000068 |.f.&;.................\.iO.....|
00887100. 000003E0 00000000                                                       |...\....                        |

CPOOLCELL - Cell in use
LIST 00887108 ASID(X'036E') LENGTH(80) AREA(CPOOLCELL)
ASID(X'036E') ADDRESS(00887108) KEY(10)
00887108.                  C4E2C1C2 00887158  008B76A8 00500000 008BC3C4 0089D1F0 |        DSAB.h.....y.&;...CD.iJ0|
00887120. 00000000 00000000 0000CA00 00000000  0088CE88 00000000 00000000 00000000 |................h.h...........|
00887140. 00000000 00000188 0089D200 00000069  00000188 00000000                   |.......h.iK........h....        |
⋮
CPOOLCELL - Cell in use
LIST 008B7F68 ASID(X'036E') LENGTH(80) AREA(CPOOLCELL)
ASID(X'036E') ADDRESS(008B7F68) KEY(10)
008B7F68.                  C4E2C1C2 00887018  008B7798 00500000 008BC9B4 008A25E0 |        DSAB.h.....q.&;...I....\|
008B7F80. 00000000 00000000 0000CA00 00000000  008F9E50 00000000 00000000 00000000 |..................&;...........|
008B7FA0. 00000000 000001F8 008A25F0 00000065  000001F8 00000000                   |.......8...0.......8....        |

IGV18094I 126 cells processed
```

Example 3 - A Common Area CPOOL

Example 3 shows a summary of a common area CPOOL.

```
runcpool a'2D37000'

PPD 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 04582000
SXT at 049CF000
SXT at 0273B000

Cells(5) used(0)

IGV18094I No cells processed
```

SCAN Subcommand — Validate System Data Areas

Use the SCAN subcommand to validate system data and make storage map entries for that data. Appendix D, “Control Blocks and Data Areas Scanned, Mapped, and Formatted” on page D-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.)
- 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 could add many entries to the map. If this control block is the CVT or an ASCB, one scan probe could 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 5-223.

```
            [ FLAG(severity) ]

            [ PRINT | NOPRINT ]

            [ TERMINAL | NOTERMINAL ]

            [ TEST | NOTEST ]
```

Parameters

limit
Specifies the maximum number of scan probes that SCAN is to perform. 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 due to 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.

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      [ 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 5-223.
            [ 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 are CURRENT and ERROR.

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:

When You Specify This Address Space Parameter	You Get This AREA(name) Storage Map Entry Attribute
CURRENT	AREA(CURRENT)
ERROR	AREA(ERROR)
TCBERROR	AREA(TCBERROR)
JOBLIST	AREA(JOBxxxx)

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

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:

```
ASID  JOBNAME   ASCBADDR  SELECTION CRITERIA
----  --------  --------  ------------------
0001  *MASTER*  00123456  CURRENT JOBNAME
0010  JES3      00234567  JOBNAME CURRENT
01BB  USERJOB   00789ABC  ERROR
```

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 as 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:

```
/*-------------- Default Values for IPCS Subcommands -------------*/
SETDEF NOPROBLEM                /* Current problem number         */
SETDEF NOPRINT   TERMINAL       /* Routing of displays            */
SETDEF FLAG(WARNING)            /* Optional diagnostic messages   */
SETDEF   CONFIRM                /* Double-checking major acts     */
SETDEF NOTEST                   /* IPCS application testing       */
SETDEF NODSNAME                 /* No data set name               */
SETDEF LENGTH(4)                /* Default data length            */
SETDEF   VERIFY                 /* 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(NOSTORAGE)       /* Include contents of storage    */
SETDEF DISPLAY(  SYMBOL)        /* Include associated symbol      */
```

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 }   [ LIST | NOLIST ]
{SETD   }

            [ LOCAL   ]
            [ GLOBAL  ]
```

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

```
            [ address-processing-parameters ]

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

            [ DISPLAY[(display-options)]   ]
            [ NODISPLAY[(display-options)] ]

            [ FLAG(severity) ]

            [ LENGTH(length) ]

            [ PRINT | NOPRINT ]

            [ TERMINAL | NOTERMINAL ]

            [ PROBLEM(prob-num) | NOPROBLEM ]

            [ 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:

– ADDDSN
– DELDSN
– DELPROB
– MODDSN
– 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

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-labelled 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 unallocate (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 unallocate 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 and/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.

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 ASID, virtual address, storage key, and absolute address of the data area being displayed. DISPLAY(MACHINE) and NODISPLAY(NOMACHINE) request this data.

Note: When processing a SUMDUMP record, you must use SETDEF CONFIRM DISPLAY(MACHINE). The CONFIRM parameter permits message BLS18160D

```
BLS18160D May summary dump data be used by dump access?
Enter Y to use, N to bypass
```

Entering N causes IPCS to display the storage keys. Entering Y causes all information **except** storage keys to be displayed for the SUMDUMP records.

NOMACHINE suppresses the ASID, virtual 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.

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.

```
FLAG  { (ERROR)            }
      { (INFORMATIONAL)    }
      { (SERIOUS | 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 could 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 if 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.

TERMINAL or NOTERMINAL

Specifies if 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.

NOTERMINAL 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.

PROBLEM(prob-num) or NOPROBLEM

Control the problem number.

PROBLEM specifies a problem number. The problem number is 5 decimal digits; leading zeros are optional.

NOPROBLEM sets the default problem number to a null value.

NOPROBLEM(00000) is the IPCS-defined default.

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 wish 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

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 NOPROBLEM                /* Current problem number         */
SETDEF NOPRINT   TERMINAL       /* Routing of displays            */
SETDEF FLAG(WARNING)            /* Optional diagnostic messages   */
SETDEF   CONFIRM                /* Double-checking major acts     */
SETDEF NOTEST                   /* IPCS application testing       */
SETDEF   DSNAME('D4.DMP.SVC20')
SETDEF LENGTH(4)                /* Default data length            */
SETDEF   VERIFY                 /* 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(NOSTORAGE)       /* 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 preallocated SMF (VSAM) data set.

The output data set *must*:

- Be preallocated to the data set with a ddname of SMFDATA
- Have the same control interval size as the defined SMF data sets
- Be large enough to accommodate all the SMF data in the dump
- Be 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 — 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 5-223.
              [ 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 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 Z*nnnnn* 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      [ data-descr | X ]
           [ DROP | NODROP ]
```

-------- SETDEF-Defined Parameter ----------
Note: You can override the following SETDEF parameter. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
           [ 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 usually examined during the initial part of the problem determination process.

STATUS produces different diagnostic information depending on the report type parameter or parameters entered: SYSTEM, CPU, WORKSHEET, and FAILDATA.

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.
- 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 5-223.
              [ 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 SYSTEM, CPU, WORKSHEET, and FAILDATA. For more information on defaults, see 5-238.

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
- 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 5-238 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

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 5-239 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.

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 5-241 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 5-243 for an example of the FAILDATA report.

NOFAILDATA suppresses formatting of the SDWA.

Defaults

The defaults for the STATUS report type parameters follow.

Parameters on the STATUS Subcommand	Reports Requested	For Example
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, and/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.

```
SYSTEM STATUS                                  1
Nucleus member name: IEANUC01                  2
 I/O configuration data:                       3
   IODF data set name: SYS0.IODF52
   IODF configuration ID: CONFIG00
   EDT ID: 00
Sysplex name: PLEX01                           4  5
TIME OF DAY CLOCK: A9B7540D 54AD1405 08/12/1994 10:54:28.305617 local
TIME OF DAY CLOCK: A9B789B2 3DAD1405 08/12/1994 14:54:28.305617 GMT
Program Producing Dump: SVCDUMP                6
Program Requesting Dump: DATSVY02
Incident token: LOCAL    S520     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.

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 would 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:                                                  1
 PSW=070C1000  83D00B72 (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
     DISABLED FOR PER                                        2
   ASID(X'0015') 03D00B72. DATSVY02+03CA IN EXTENDED PRIVATE
  ASCB21 at F9CD80, JOB(DAESVY01), for the home ASID   3
  ASXB21 at 6FE038 for the home ASID. No block is dispatched
  HOME ASID: 0015 PRIMARY ASID: 0015 SECONDARY ASID: 0015
  GPR VALUES                                              4
      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.                   5
  CONTROL REGISTER VALUES                                 6
      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
```

1 Identifies the STATUS report type, CPU. The CPU address is omitted because a virtual dump is being processed.

2 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.
  - *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, and/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* 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 would 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
```

– Result

The following output is produced.

STATUS Subcommand

```
                            MVS Diagnostic Worksheet      1

Dump Title: SERV2_DUMP1                                   2

CPU Model 3090 Version FF Serial no. 176280 Address 01    3
Date: 05/05/1994     Time: 16:59:35.414381 Local

Original dump dataset: SERV2.DMP00001

Information at time of entry to SVCDUMP:

HASID 0001  PASID 0001  SASID 0001  PSW 070C1000 83930B82

CML ASCB address 00000000  Trace Table Control Header address 7F731000

Dump ID: 001                                              4
Error ID: N/A

SDWA address N/A


 SYSTEM RELATED DATA                                      5

 CVT  SNAME (154) S520       VERID (-18) HBB5520 DRIVER08

      CUCB   (64) 00FCF8F8   PVTP  (164) 00FDDED0    GDA  (230) 01D551A0

      RTMCT (23C) 00FC0780   ASMVT (2C0) 00FD3250    RCEP (490) 014D9F80

 CSD  Available CPU mask: 4000  Alive CPU mask: 4000  No. of active CPUs: 0001

      System set non-dispatchable by SVC dump

 PROCESSOR RELATED DATA                                   6

  NAME             OFFSET  |  CPU 01
 ------ LCCA --------------+----------------------------------------
  IHR1   Recursion    208  | 00
  SPN1/2 Spin         20C  | 0000
  CPUS   CPU WSAVT    218  | 00F8AA48
  DSF1/2 Dispatcher   21C  | 0080
  CRFL   ACR/LK flgs  2B4  | 00000000
 ------ PSA ---------------+----------------------------------------
  TOLD   Curr TCB     21C  | 00000000
  AOLD   Curr ASCB    224  | 00F4A580
  SUPER  Super Bits   228  | 00000000
  CLHT   Lock Table   280  | 00FCDA18
  LOCAL  Local lock   2EC  | 00000000
  CLHS   Locks held   2F8  | 80000000
  CSTK   FRR stack    380  | 00000C00
  SMPSW  SRB Disp PSW 420  | 070C0000
                      424  | 8267AE00
  PSWSV  PSW Save     468  | 070C0000
                      46C  | 8264BFFC
  MODE   Indicators   49F  | 04
```

```
                 SDUMP Parameter List

+0000  FLAG0.... 1E        FLAG1.... A1        SDATA.... 9920      DCBAD.... 00000000  STORA.... 00000000
+000C  HDRAD.... 01E29BC0  ECBAD.... 036F4288  SRBAD.... 036F4288  CASID.... 0001      TASID.... 0001
+0018  ASIDP.... 00000000  SUMLP.... 00000000  SDDAT.... 00000000  FLAG2.... 00        CNTL1.... C0
+002A  TYP1..... 10        VERSN.... 03        SDTA2.... 65004000  EXIT..... 6500      SDAT3.... 40
+002F  SDAT4.... 00        SPLST.... 00000000  KYLST.... 00000000  RGPSA.... 00000000  DCBA..... 00000000
+0040  STRAL.... 00000000  HDRA..... 00000000  ASDLA.... 00000000  SMLA..... 00000000  SBPLA.... 00000000
+0054  KEYLA.... 00000000  LSTDP.... 00000000  LSTDA.... 00000000  SMLLP.... 00000000  SMLLA.... 00000000
+0068  PSWRP.... 00000000  PSWRA.... 00000000  SYMAD.... 00000000  SYMA..... 00000000  IDAD..... 00000000
+007C  IDA...... 00000000  SLADR.... 00000000  SLALT.... 00000000  ITADR.... 01E27578  ITALT.... 00000000
+0090  RMADR.... 00000000  RMALT.... 00000000  PDADR.... 00000000  PDALT.... 00000000  JLADR.... 00000000
+00A4  JLALT.... 00000000  DLADR.... 00000000  DLALT.... 00000000
```

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 The identifiers of the dump and the error.

5 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
```

6 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
```

– 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 00000004 00C13300  00C13304 00C13300 00C136E8 00C1349C
    AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
    Registers 8-15
    GR: 00C13350 00C13300 01E01260 81E0D04C  01E0E04B 01E01260 00000000 80AD5A88
    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
```

1 Identifies the report type, DIAGNOSTIC DATA REPORT.

2 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.

3 Indicates information that was not available because the recovery routine did not provide it. When this information is available, it would appear in section **2** under the title “Other Serviceability Information”.

4 Provides PSW, register, and ASID-related error information, along with failure reasons and environments and, if applicable, super or spin bit settings.

5 Presents second-level status information as indicated by the second set of registers and their corresponding PSW, which are located in the SDWA.

6 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

7 Indicates that the variable recording area is empty. If the area contained data, it would be 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.

STRDATA Subcommand

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

-------- 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)   ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ ACTIVE | MAIN | STORAGE            ]
[ DSNAME(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

STRDATA Subcommand

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

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 would be 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 would include 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 quotes, 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
ASSOCIATED REQUEST BLOCK REPORT
EVENT MONITOR CONTROLS REPORT
EVENT QUEUE CONTROLS REPORT
LIST NUMBER ENTRY POSITION SUMMARY REPORT
LOCK ENTRIES REPORT
USER CONTROLS 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
– 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 a 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
– 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.

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 would be 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 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:

LIST 2
Head of Queue
entry 1 key 1 entry 2 key 2 entry 3 key 2 entry 4 key 2 entry 5 key 3
Tail of Queue

To display the second and third entries for key 2 from the head of list 2, enter the following command:

```
COMMAND ===>  STRDATA LISTNUM(2) ENTRYPOS(2,3) ENTRYKEY(02) ORDER(HEAD)
```

The following shows how entries with the same key would be considered a separate queue, queue 2, so you would get back entry 3 as entry position 2 and entry 4 as entry position 3.

LIST 2
entry 1 key 1
Head of Queue
entry 2 key 2 position 1 entry 3 key 2 position 2 entry 4 key 2 position 3
Tail of Queue
entry 5 key 3

Another example is:

```
COMMAND ===>  STRDATA ENTRYKEY(02) LISTNUM(ALL) ENTRYPOS(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 }
{ SUMM    }

-------- Report Type Parameters ----------------------------
            [ KEYFIELD [REGISTERS | NOREGISTERS]  ]

            [ FORMAT ]

            [ TCBSUMMARY ]

            [ JOBSUMMARY ]


-------- Address Space Selection Parameters ----------------
            [ ALL ]

            [ CURRENT ]

            [ ERROR ]

            [ TCBERROR | ANOMALY ]

            [ ASIDLIST(asidlist) ]

            [ JOBLIST(joblist) | JOBNAME(joblist) ]


-------- SETDEF-Defined Parameters -------------------------
Note: You can override the following SETDEF parameters.
See
"SETDEF Subcommand — Set Defaults" on page 5-223.
             [ 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 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:

ASBC fields:

AFFN	FLG2
ASID	FWDP
ASSB	LOCK
ASXB	SRBS
CSCB	TSB
DSP1	

TCB fields:

BITS	NDSP
CMP	PKF
DAR	RTWA
DSP	STAB
FBYT1	STCB
JSCB	TSFLG
LMP	

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 & 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.

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
FBYT1 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 defaults are CURRENT and ERROR. 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 tMVS 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 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   * * * *
JOBNAME TC
2   SELECTED BY: CURRENT ERROR

 ASCB: 00F6AD00
    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  NDSP..... 00000000
    JSCB..... 007FFDFC  BITS..... 00000000  DAR...... 00
    RTWA..... 00000000  FBYT1.... 00
    Task non-dispatchability flags from TCBFLGS4:
     Top RB is in a wait

PRB: 007FFF98
   WLIC..... 00020001  OPSW..... 070C1000  810234C0
   LINK..... 017FE240
   EP....... IEAVAR00  ENTPT.... 82B6CED0

TCB: 007FF3B8
   CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
   TSFLG.... 00        STAB..... 007FF6B0  NDSP..... 00000000
   JSCB..... 007FFDFC  BITS..... 00000000  DAR...... 00
   RTWA..... 00000000  FBYT1.... 00
   Task non-dispatchability flags from TCBFLGS4:
    Top RB is in a wait
```

(continued on next page)

```
PRB: 007FF0A0
   WLIC..... 00020078  OPSW..... 070C2000  823E55D0
   LINK..... 017FF3B8
   EP....... IEAVTSDT  MAJOR.... IGC0005A  ENTPT.... 823E52D8

TCB: 007FF128
   CMP...... 00000000  PKF...... 80        LMP...... FF        DSP...... FF
   TSFLG.... 00        STAB..... 007FF620  NDSP..... 00000000
   JSCB..... 007FCC14  BITS..... 00000000  DAR...... 00
   RTWA..... 00000000  FBYT1.... 00
   Task non-dispatchability flags from TCBFLGS4:
    Top RB is in a wait

PRB: 007FCC30
   WLIC..... 00020001  OPSW..... 070C1000  80E11948
   LINK..... 017FCE30
   EP....... IEFSD060  ENTPT.... 80E08880

PRB: 007FCE30
   WLIC..... 00020006  OPSW..... 070C1000  80E1A706
   LINK..... 007FF128
   EP....... IEESB605  ENTPT.... 00E1A000
```

1 Indicates the report type.

2 Indicates the selection criteria that were met.

Example 2

Produce a FORMAT report.

– Action

```
COMMAND ===> summary format current
```

– Result

IPCS produces the following output:

```
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
```

```
3  SELECTED BY: CURRENT ERROR

JOB TC

ASCB: 00F6AD00
   +0000  ASCB..... ASCB      FWDP..... 00F6E800  BWDP..... 00F63D00
   +000C  R00C..... 00000000  SVRB..... 007FF808  SYNC..... 000000F0
   +0018  IOSP..... 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    CSCB..... 02DAE530  TSB...... 00000000
   +0040  EJST..... 00000000  126D0478
   +0048  EWST..... A7843AC5  96613D05            JSTL..... 0000023D
   +0054  ECB...... 807FFF98  UBET..... A7843AAF  TLCH..... 00000000
   +0060  DUMP..... 007FF3B8  AFFN..... FFFF      RCTF..... 01
   +0067  FLG1..... 00        TMCH..... 00000000  ASXB..... 007FE038
   +0070  SWCT..... 0024      DSP1..... 00        FLG2..... 00
   +0074  RSV...... 0000      SRBS..... 0000      LLWQ..... 00000000
   +007C  RCTP..... 007FE240  LOCK..... 00000000  LSWQ..... 00000000
   +0088  QECB..... 00000000  MECB..... 40000000  OUCB..... 01A720F8
   +0094  OUXB..... 01CAC7F8  FMCT..... 0000      LEVL..... 03
   +009B  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..... 0000C478
   +00C8  SRBT..... 00000000  0033F200            R0D0..... 00000000
   +00D4            00000000  TCBS..... 00000001  LSQT..... 00000000
   +00E0  WPRB..... 007FEA40  NDP...... 73        TNDP..... FF
   +00E6  NTSG..... FF        IODP..... FF        LOCI..... 00000000
   +00EC  CMLW..... 00000000  CMLC..... 00000000  SSO1..... 000000
   +00F7  SSO4..... 00        ASTE..... 0142E840  LTOV..... 7FFE2000
   +0100  ATOV..... 7FFE3CA8  ETC...... 0000      ETCN..... 0000
   +0108  LXR...... 0000      AXR...... 0000      STKH..... 007FEA50
   +0110  GQEL..... 00000000  LQEL..... 02E07990  GSYN..... 00000000
   +011C  XTCB..... 007FC378  CS1...... 00        CS2...... 00
   +0122  R122..... 0000      GXL...... 01B90FE0
   +0128  EATT..... 00000000  1038360E
   +0130  INTS..... A7843AAB  41D12702            LL1...... 00
   +0139  LL2...... 00        LL3...... 00        LL4...... 00
   +013C  RCMS..... 00000000  IOSC..... 00000006  PKML..... 0000
   +0146  XCNT..... 01F4      NSQA..... 00000000  ASM...... 01A82028
   +0150  ASSB..... 01A72280  TCME..... 00000000  GQIR..... 00000000
   +015C  R15C..... 00000000  00000000  00000000  CREQ..... 00000000
   +016C  RSME..... 01B90E20  AVM1..... 00        AVM2..... 00
   +0172  AGEN..... 0000      ARC...... 00000000  RSMA..... 01B90D10
   +017C  DCTI..... 00000041
```

```
   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
   +0019  XMF2..... 00        XMCC..... 0000      CBTP..... 00000000
   +0020  VSC...... 00000000  NVSC..... 00000000  ASRR..... 00000000
   +002C  DEXP..... 00000000  STKN..... 00000084  00000002
   +0038  BPSA..... 00000000  CSCT..... 00000000  BALV..... 7FFFBF00
   +0044  BALD..... 011D1F00  XMSE..... 00000000  TSQN..... 00000004
   +0050  VCNT..... 00000000  PALV..... 7FFD9E00  ASEI..... 00000000
   +005C  RMA...... 02DE10E0  HST...... 00000000  00000000
   +0068  IIPT..... 00000000  003AA100            ANEC..... 00000000
   +0074  SDOV..... 00000000  MCSO..... 00000000  DFAS..... 01AE6A40
   +0080  FLG0..... 00        FLG1..... 00        FLG2..... 00
   +0083  FLG3..... 00        ASCB..... 00F6AD00  ASRF..... 01A5C680
   +008C  ASRB..... 01B90E80  SSD...... 00000000  R094..... 00000000
   +0098  LASB..... 037EB360  80000000            SCH...... 00000000
   +00A4  FSC...... 00000000  JSAB..... 01B90C90  RCTW..... 01A6F440
   +00B0  CREQ..... 00000000  CRQA..... 00000001  TLMI..... 00000000
   +00BC  SDAS..... 7FFE9C98  TPIN..... 00000000  SPIN..... 00000000
   +00C8  ECT1..... 00000000  ECT2..... 00000000  MT#...... 00000000
   +00D4  DFP...... 00000000  R0D8..... 00000000  NTTP..... 00000000
   +00E0  OECB..... 00000000  OASB..... 009F23A0  XSBA..... 7FFFEEB0
   +00EC  R0EC..... 00000000  VAB...... 01D0B200  LMAB..... 00000000
   +00F8  R0F8..... 00000000  00000000            XRCT..... 00000000
   +0104  R104..... 00000000  R108..... 00000000  TPMA..... 00000000
   +0110  ROSU..... 00000000  TPMT..... 00000000  SSDT..... 00000000
   +011C  TAWQ..... 01A3D140  WCML..... 01A6F700  WS3S..... 01A6F1C0
   +0128  WSSS..... 00000000  CAPQ..... 00000000  R130..... 00000000
   +0134            00000000  00000000  00000000  00000000  00000000
   +0148            00000000  00000000            JNBI..... ........
   +0158  JBNS..... ........
No formatting support for JSAB
Insufficient storage for exit IAZJSABP
```

```
ASXB: 007FE038
   +0000  ASXB..... ASXB      FTCB..... 007FE240  LTCB..... 007FC378
   +000C  TCBS..... 0004      R00E..... 0000      MPST..... 00000000
   +0014  LWA...... 00000000  VFVT..... 00000000  SAF...... 00000000
   +0020  IHSA..... 007FE598  FLSA..... FE0000D8  00F77500  00FD1770
   +0030           812ED762  00000040  012EE761  00F77500  812ED896
   +0044           00000000  00FFA848  000000D8  00F77508  00F77400
   +0058           7FFE44F0  81161DC2  7FFE493C  00000C60  812B0132
   +006C  OMCB..... 00000000  SPSA..... 007FEA68  RSMD..... 00000000
   +0078  RCTD..... 007FE480  DECB..... 807FF0A0  OUSB..... 7FFFD1C0
   +0084  CRWK..... 00000000  PRG...... 00000000  00000000  00000000
   +0094           00000000  PSWD..... 00000000  00000000
   +00A0  SIRB..... 007FE3D8  ETSK..... 007FE240  FIQE..... 00000000
   +00AC  LIQE..... 00000000  FRQE..... 00000000  LRQE..... 00000000
   +00B8  FSRB..... 00000000  LSRB..... 00000000  USER..... TC
   +00C7  SFLG..... 00        SENV..... 007FCF58  R0CC..... 00000000
   +00D0  NSSA..... 7FFFCDE0  NSCT..... 0000000D  CRB1..... 00
   +00D9  CRB2..... 00        CRB3..... 00        CRB4..... 00
   +00DC  PT0E..... 00000000  R0E0..... 00000000  JSVT..... 00000000
   +00E8  DIVW..... 00000000  R0EC..... 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:

```
1        * * * * T C B  S U M M A R Y * * * *

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: 007FF128
    CMP...... 00000000  PKF...... 80        LMP...... FF        DSP...... FF
    TSFLG.... 00        STAB..... 007FF620  NDSP..... 00000000
    JSCB..... 007FCC14  BITS..... 00000000  FBYT1.... 00

TCB: 007FC378
   CMP...... 88522000  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:

```
1       * * * *  S Y S T E M  S U M M A R Y  * *
*** ACTIVE 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

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


2  *** JOB SUMMARY ***
 ---------------------------------------------------------------------

3  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
 ---------------------------------------------------------------------
 *** 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
```

1. Provides a summary of the system.
2. Indicates the report type.
3. Indicates the selection criteria that were meet.
4. Provides a problem list.

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 Appendix B, “IPCS Symbols” on page B-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 5-223.

```
[ 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) ]

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

             [ EXCLUDE(BR) ]

             [ START(mm/dd/yy,hh.mm.ss.dddddd) ]

             [ STOP(mm/dd/yy,hh.mm.ss.dddddd) ]

             [ CPU(cpu-address) ]

             [ TCB(TCB-list) ]

             [ TTCH(TTCH-address | LIST) ]

             [ WEB(WEB-list) ]

-------- Address Space Selection Parameters ----------------

             [ ALL ]

             [ CURRENT ]

             [ ERROR ]

             [ TCBERROR ]

             [ ASIDLIST(asidlist) ]

             [ JOBLIST(joblist) | JOBNAME(joblist) ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

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


[ FLAG(severity) ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL ]

[ TEST | NOTEST ]
```

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.

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.

EXCLUDE(BR)
Suppresses formatting of trace table entries for branch or mode tracing if any were present in the dump. When you do not specify EXCLUDE(BR), the formatted trace table shows all the types of trace table entries.

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
dddddd	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).

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
```

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)
Limits formatting to trace entries for the central processor identified by the CPU address. Use a Store CPU Address (STAP) instruction to obtain the CPU address. The CPU address is 1 through 15 and is specified in decimal (nn), hexadecimal (X'h'), or binary (B'bbbb'). The default is to format trace entries from all central processors.

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. Refer to "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:

```
TTCH      ASID   TCB        TIME
7F77D000  0022   006F4B90   4/23/1966 16:48:55.683827
7F786000  0021   006F4A98   4/23/1966 16:48:31.718191
7F78F000  0020   006F4B90   4/23/1966 16:48:31.171256
```

In the above example:

TTCH The TTCH column shows the address of the trace table snapshot in the dump.

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 (:).

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*.

TCBEXIT Subcommand — Run a TCB Exit Routine

Use the TCBEXIT subcommand to run an IBM-supplied or an installation-supplied exit routine.

Syntax

```
{ TCBEXIT }  { pgmname | * }
{ TCBX    }

             data-descr

             [ AMASK(mask) ]
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters.
See
“SETDEF Subcommand — Set Defaults” on page 5-223.

```
             [ FLAG(severity) ]

             [ PRINT | NOPRINT ]

             [ TERMINAL | NOTERMINAL ]

             [ TEST | NOTEST ]
```

Parameters

pgmname or *

Specifies an IBM-supplied or installation-supplied exit routine, which processes system control blocks.

The *pgname* 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.

Exit Routine	Data Processed
IECDAFMT	Data management control blocks

IECIOFMT Input/output supervisor (IOS) and execute channel program (EXCP) control blocks

IEAVTFMT Recovery termination management (RTM) control blocks

IEAVSSA1 Vector Facility data file.IEAVSSA1 exit routine

IEAVXD01 Access registers

IEAVD30 Linkage stack

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 on writing installation TCB exit routines, see *z/OS MVS IPCS Customization*.

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

Code	Explanation
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 1

Invoke an IBM-supplied TCB exit to display RTM-related control blocks.

– Action

```
COMMAND ===>  tcbexit ieavtfmt 21C.%
```

– 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

 EED1: 009FF750
    +0000  E2E3D2C5  009FEA68  00000000  00000000  | STKE.þ......... |
    +0010  0000001B  00000000  00000000  00000000  | ............... |
    +0020  00000000  00000000  00000000  00000000  | ............... |
    +0030  00000000  00000000  00AA0400  00000000  | ............... |
    +0040  009FBE00  009FAFB0  00000000  00006EF4  | .--..---......>4 |
    +0050  00000000  00006F40  00000000  00000000  | ......? ........ |
    +0060  00000000  00000000  00000000  00000000  | ............... |
    +0070  00000000  00000000  00000000  00000000  | ............... |
    +0080  00000000  00000000  00000000  00000000  | ............... |
    +0090  00000000  00000000  00000000  00000000  | ............... |
    +00A0  00000000  00000000  00000000  00000000  | ............... |
    +00B0  00000000  00000000  00000000  00000000  | ............... |
    +00C0  00000000  00000000  00000000  00000000  | ............... |
    +00D0  00000000  00000000  00000000  00000000  | ............... |
    +00E0  00000000  00000000  D4E2E3D9  40404040  | ........MSTR    |
    +00F0  D3D3C140  40404040  40404040  40404040  | LLA             |
    +0190  00000000  00000000  00000000  00000000  | ............... |
```

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. These subcommands can also be used to set traps when executing IPCS code that uses the exit service routines.

Related Subcommands

TRAPON
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:

Code	Exit Service Routine
ACC	Storage access service
ADS	Add symptom service
CBF	Control block formatter service
CBS	Control block status service
CQE	Contention queue element create service
CSI	CSVINFO macro
ECT	ECT exit service
EQS	Equate symbol service
FMT	Format model processor service
GTS	Get symbol service
MAP	Map service
NAM	Name service
NDX	Table of contents service
NTK	NAME/TOKEN lookup service
PRT	Standard print service
PR2	Expanded print service
SEL	Select ASID service
SYM	Symbol service
WHS	WHERE service

Return Codes

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

```
ACC   INPUT(ABDPL PARMS STOP) OUTPUT(RETC DATA PARMS STOP ERROR)
ADS   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
CBF   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
CBS   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
CQE   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
CSI   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
ECT   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
EQS   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
FMT   INPUT(ABDPL PARMS DATA STOP) OUTPUT(RETC PARMS STOP ERROR)
GTS   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
MAP   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
NAM   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
NDX   INPUT(ABDPL STOP) OUTPUT(RETC STOP )
NTK   INPUT(ABDPL STOP) OUTPUT(RETC STOP )
PRT   INPUT(ABDPL STOP) OUTPUT(RETC STOP )
PR2   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
SEL   INPUT(ABDPL PARMS STOP) OUTPUT(RETC DATA PARMS STOP ERROR)
SYM   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
WHS   INPUT(ABDPL PARMS STOP) OUTPUT(RETC PARMS STOP ERROR)
```

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.

```
ACC INPUT(ABDPL PARM STOP) OUTPUT(RETC PARM DATA STOP ERROR)
```

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. These subcommands can also be used to set traps when executing IPCS code that uses the exit service routines.

Related Subcommands

TRAPON
TRAPLIST
GO

Syntax

```
TRAPOFF   { ALL         }
          { code        }
          { (code-list) }
```

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:

Code	Exit Service Routine
ACC	Storage access service
ADS	Add symptom service
CBF	Control block formatter service
CBS	Control block status service
CQE	Contention queue element create service
CSI	CSVINFO macro
ECT	ECT exit service
EQS	Equate symbol service
FMT	Format model processor service
GTS	Get symbol service
MAP	Map service
NAM	Name service
NDX	Table of contents service
NTK	NAME/TOKEN lookup service
PRT	Standard print service
PR2	Expanded print service
SEL	Select ASID service
SYM	Symbol service
WHS	WHERE service

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*, use the TRAPON, TRAPOFF, and TRAPLIST subcommands to obtain diagnostic input and output information. These subcommands can also be used 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 would, therefore, also be 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

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:

Code	Exit Service Routines
ACC	Storage access service
ADS	Add symptom service
CBF	Control block formatter service
CBS	Control block status service
CQE	Contention queue element create service
CSI	CSVINFO macro
ECT	ECT exit service
EQS	Equate symbol service
FMT	Format model processor service
GTS	Get symbol service
MAP	Map service
NAM	Name service
NDX	Table of contents service
NTK	NAME/TOKEN lookup service
PRT	Standard print service
PR2	Expanded print service
SEL	Select ASID service
SYM	Symbol service
WHS	WHERE service

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:

Option	Processing
ABDPL	Displays the common exit parameter list and its extension that are passed to all services.
DATA	Displays data passed to a service in addition to basic parameters. The DATA option can be used only if the FMT code is specified.

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 5-275 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:

– 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 5-275 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:

- 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:

```
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 may enter installation-defined authorized commands, as well as 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.SBLSCLI0(BLSCALTL). For information about defining and using global variables, see *z/OS TSO/E CLISTs*.

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 | [%]rexxnm | 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.

rexxnm
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

Code	Explanation
12	Severe, an error condition or user request forced early ending of subcommand processing.
16	Ending error, an error condition from a called service routine forced an early end to the processing.
any	The return code is generated by the TSO/E command.

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 had not preceded the command name, STATUS, with TSO, the IPCS STATUS subcommand would have been processed.

Example 2

Send a data set to a node and userid.

– 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 5-223.

```
              [ 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 alphameric 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 on writing verb exit routines, see *z/OS MVS IPCS Customization*.

verbname

Specifies the name of a verb exit routine.

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 on 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 has a page reference or cross reference to another book, 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.

Verb Name	Component, Product, or Function	Where Documented
Note: Other products might have dump formatters available for use with IPCS. Check the related product documentation for information.		
ALCWAIT	Allocation	See “VERBEXIT ALCWAIT Subcommand — List Jobs Waiting for Devices” on page 5-287
AOMDATA	Asynchronous operations manager	*DFSMS/MVS DFSMSdfp Diagnosis Reference*
ASMDATA	Auxiliary storage management	See “VERBEXIT ASMDATA Subcommand — Format Auxiliary Storage Manager Data” on page 5-288
AVMDATA	Availability manager	See “VERBEXIT AVMDATA Subcommand — Format Availability Manager Data” on page 5-288
BLSAIPST	System initialization	See “VERBEXIT BLSAIPST Subcommand — Format System Initialization Data” on page 5-288
CBDATA	Component Broker	See “VERBEXIT CBDATA Subcommand — Format Component Broker Data” on page 5-288
CICSDATA	Customer Information Control System	*CICS Operations Guide*
DAEDATA	Dump analysis and elimination	See “VERBEXIT DAEDATA Subcommand — Format Dump Analysis and Elimination Data” on page 5-289
DSNWDMP	DATABASE 2	*DB2 Diagnosis Guide and Reference*
GRSTRACE	Global resource serialization	See “VERBEXIT GRSTRACE Subcommand — Format Global Resource Serialization Data” on page 5-291
HASMFMTM	JES2	*z/OS JES2 Diagnosis*

VERBEXIT Subcommand

Verb Name	Component, Product, or Function	Where Documented
IEFIVAWT	Allocation	See “VERBEXIT IEFIVAWT Subcommand — List Pending XCF Work for Tape Allocation” on page 5-292
IEFIVIGD	Allocation	See “VERBEXIT IEFIVIGD Subcommand — List Global Tape Device Information” on page 5-292
IMSDUMP	Information Management System (IMS)	– *IMS/VS Utilities Reference* – *IMS/VS Diagnosis Guide and Reference*
IRLM	Information Management System (IMS)	– *IMS/VS Diagnosis Guide and Reference* – *IMS/VS System Definition Referencel*
JESXCF	JES common coupling services MVS component (JES XCF)	See “VERBEXIT JESXCF Subcommand — Format Data for JES XCF Component” on page 5-293
JES3	JES3	*z/OS JES3 Diagnosis*
LEDATA	Language Environment	See “VERBEXIT LEDATA Subcommand — Format Language Environment Data” on page 5-294
LOGDATA	Logrec buffer records	See “VERBEXIT LOGDATA Subcommand — Format Logrec Buffer Records” on page 5-296
MMSDATA	MVS message service	See “VERBEXIT MMSDATA Subcommand — Format MVS Message Service Data” on page 5-299
MTRACE	Master trace table	See “VERBEXIT MTRACE Subcommand — Format Master Trace Entries” on page 5-299
NUCMAP	Modules in the nucleus	See “VERBEXIT NUCMAP Subcommand — Map Modules in the Nucleus” on page 5-302
SADMPMSG	Stand-alone dump message log	See “VERBEXIT SADMPMSG Subcommand — Format Stand-Alone Dump Message Log” on page 5-305
SMSDATA SMSXDATA	DFP Storage Management Subsystem	*z/OS DFSMSdfp Diagnosis Reference*
SRMDATA	System resource manager	See “VERBEXIT SRMDATA Subcommand — Format System Resource Manager Data” on page 5-305
SUMDUMP	SVC summary dump data	See “VERBEXIT SUMDUMP Subcommand — Format SVC Summary Dump Data” on page 5-306
SYMPTOM	Symptom string	See “VERBEXIT SYMPTOM Subcommand — Format Symptom String” on page 5-307
TCAMMAP	Telecommunications Access Method	*Advanced Communications Function for TCAM*
TSODATA	Time Sharing Option	*TSO/E V2 Diagnosis: Guide and Index*
VSMDATA	Virtual storage management	See “VERBEXIT VSMDATA Subcommand — Format Virtual Storage Management Data” on page 5-308
VTAMMAP	Virtual Telecommunications Access Method (VTAM)	*VTAM Diagnosis*

parameter
Specifies a parameter string to be passed to either an IBM-supplied or an installation-supplied verb exit routine.

If specifying more than one parameter, separate the parameters with commas. Enclose the parameter string in apostrophes. When IPCS passes the string to the exit routine, it omits the apostrophes.

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

Code	Explanation
12	Severe error, an error condition or user request forced early termination of the subcommand.
16	Terminating error, an error condition from a called service routine forced an early termination of the subcommand.
other	An exit generated return code.

Example

Invoke an installation-supplied verb exit represented by the verb name HISTORY.

– Action

```
COMMAND ===>  verbexit history 'rb,56b34'
```

– Result

The installation-supplied verb exit routine HISTORY would receive 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 on the VERBEXIT subcommand to format diagnostic data from the auxiliary storage manager (ASM).

Parameters

The VERBEXIT ASMDATA subcommand has no parameters.

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 OS/390. 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
- BGVT address - Component Broker Global Vector table
- ASR Table and ASR Table entries - Active Server Respository 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
- 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.



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/18F4B22100EF181BBF2FD0F0 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:

         RETAIN
MVS KEY   KEY     SYMPTOM DATA       EXPLANATION
-------  ------   ------------       -----------
MOD/     RIDS/    NUCLEUS            LOAD MODULE NAME
CSECT/   RIDS/    IARUVXCH           ASSEMBLY MODULE CSECT NAME
PIDS/    PIDS/    5752SC1CR          PRODUCT/COMPONENT IDENTIFIER
AB/S     AB/S     S00C4              ABEND CODE-SYSTEM
REXN/    RIDS/    IARRR              RECOVERY ROUTINE CSECT NAME
FI/      VALU/H   18F4B22100EF181BBF2FD0F0
                                     FAILING INSTRUCTION AREA
REGS/    REGS/    0A8D0              REG/PSW DIFFERENCE
HRC1/    PRCS/    00000004           REASON CODE
SUB1/    VALU/C   REAL#STORAGE#MANAGEMENT
                                     COMPONENT SUBFUNCTION



ADDITIONAL SYMPTOM DATA NOT USED BY DAE TO IDENTIFY THIS DUMP:

         RETAIN
MVS KEY   KEY     SYMPTOM DATA         EXPLANATION
-------  ------   ------------         -----------
VCBI2/   VALU/H   3C7800F2D80001B017C800000000011BE84401ACDC90
                                       CONTROL BLOCK ID AND DATA
CID1/    VALU/C   SC1CR                COMPONENT IDENTIFIER
AMD1/    VALU/C   04#14#87             MODULE ASSEMBLY DATE
VRS1/    VALU/C   HBB3310              VERSION-PRODUCT/PTF IDENTIFIER
RRL1/    FLDS/    IARRRCV              RECOVERY ROUTINE LABEL
CDB1/    VALU/C   5752                 BASE COMPONENT IDENTIFIER
HLH1/    VALU/H   0800C000             HIGHEST LOCK HELD INDICATOR
SUP1/    VALU/H   10000000             PSASUPER FLAGS
FRR1/    VALU/H   01ACFC90             FRR PARAMETER AREA
ASID1/   VALU/H   00DE                 TASK RELATED ASID
ORCC1/   PRCS/    0C4000               ORIGINAL COMPLETION CODE
ORRC1/   PRCS/    00000004             ORIGINAL REASON CODE

********************** END OF DATA ****************************
```

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.

Parameters

The VERBEXIT GRSTRACE, QCBTRACE, or Q subcommand has no parameters.

Example

For an example of GRSTRACE output, see the global serialization resource component in *z/OS MVS Diagnosis: Reference*.

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 AWTR Request Queue

IEFOAWTR: 7E721540
  ID....=...  Version.. 0001   Length... 0055
  Next...00000000  FuncVal.. 0004   Flags....0000
  Function: Merge
  SendMemT.E5010000 007FFBF8          MsgBufA.. 00B67618
  MsgBufl.. 982F3CAC  MsgBufSP .20    MsgBufKy. 00
  MsgBufT.. 010B0002
  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
```

– Result

```
TSRA IGDE Hash Table

Hash Value 0141

IEFZIGDE: 7EC57288
  ID....IGDE  Version.. 01   Length... 04D8
  GILen... 0060   SIELen... 0020
  HashVal.. 00000140  UCBAddr.. 7FFFFBAD  DevNum... 0000
  UHashVal.. 00
  DevType.. ...   EpiValue.00000000   EnqASID.. 0000
  EnqTCB... 00000000
  MinorNam.D5C5C440 F0F0F3F4 F9F0C2F4 F0E5E2E2 C7C1E5F3
       F4F9F0C1 E3E2C2F5 F7F10000 40404040
  UpdtETOD. 00000000 00000000 00000000 00000001
  AllSysID. 00
  AllASID.. 0000   AllSysmn. ......   AllJobN.. .......
  Device supports self-description.
  CurVol... ....   LastVol.. ....   FileSeqN. 0000

System Interest Entry 02
  DevNum... B571   DevType.. 349S
  UpdtETOD. 00B6A244 D554198F 07080000 00010002



Hash Value 01D5

IEFZIGDE: 7EC4D2B0
  ID....IGDE  Version.. 01   Length... 04D8
  GILen... 0060   SIELen... 0020
  HashVal.. 000001D5  UCBAddr.. 021B2420  DevNum... 05A9
  UHashVal.. A9
  Device is online AS on this system.
  DevType.. 3480   EpiValue. F3F4F8F0   EnqASID.. 0000
  EnqTCB... 00000000
  MinorNam.C4C5E540 F0F5C1F9 40404040 40404040 40404040
       40404040 40404040 40404040 40404040
  UpdtETOD. 00B6A247 FCC981AC 01080000 00010001
  AllSysID. 00
  AllASID.. 0000   AllSysmn. ......   AllJobN.. .......
  Device supports self-description.
  CurVol... ....   LastVol.. ....   FileSeqN. 0000

System Interest Entry 00
  DevNum... 05A9   DevType.. 348S
  UpdtETOD. 00B6A245 1DEFDE79 02080000 00010001

System Interest Entry 02
  DevNum... 0589   DevType.. 348S
  UpdtETOD. 00B6A244 D5475DD7 07080000 00010002
```

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

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]
                [CM]
                [MH]
                [CEEDUMP]
                [ALL]

     Data selection parameters:

                [DETAIL | EXCEPTION]

     Control block selection parameters:

                [CAA(caa-address)]
                [DSA(dsa-address)]
                [TCB(tcb-address)]
                [ASID(address-space-ID)]
```

Parameters

Report Type Parameters

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

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 Strorage 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.

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.

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.

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
_____________________________________________________________________________

                        * * * *  L O G D A T A  * * * *
DUMP FOR DATSVY02                                        2  11:12:05 11/29/94
_____________________________________________________________________________

TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT           DAY.YEAR
       (SVC 13)                                        REPORT DATE: 333.94
FORMATTED BY: IEAVTFDE  HBB5520                         ERROR DATE: 224.94
                            MODEL:   3090                           HH:MM:SS.TH
                            SERIAL:  176280                   TIME: 10:38:59.69

JOBNAME: *MASTER*
ERRORID: SEQ=00012  CPU=0041  ASID=0001  TIME=10:38:59.6

SEARCH ARGUMENT ABSTRACT

  AB/S00F4 PRCS/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: 8B2


SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE
  PROGRAM ID
  LOAD MODULE NAME
  CSECT NAME
  RECOVERY ROUTINE CSECT NAME
  RECOVERY ROUTINE LABEL
  DATE ASSEMBLED
  MODULE LEVEL
  SUBFUNCTION
```

```
TIME OF ERROR INFORMATION
  PSW: 075C1000 8251832E   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D
  FAILING INSTRUCTION TEXT: CCB418F6 0A0D4110 CC2C45E0
  REGISTERS 0-7
  GR: 6204000C 440F4000 00000000 7F70C658  00FCF420 6204000C 00000024 00FD1CD0
  AR: 015209B8 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  REGISTERS 8-15
  GR: 00000000 7F70C4C8 00FCF3EC 02518A7B  82517A7C 7F70C6A8 82518324 00000024
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  HOME ASID: 0001    PRIMARY ASID: 0001    SECONDARY ASID: 0001
  PKM: 8000          AX: 0001              EAX: 0000
  RTM WAS ENTERED BECAUSE AN SVC WAS ISSUED IN AN IMPROPER MODE.
  THE ERROR OCCURRED WHILE AN ENABLED RB WAS IN CONTROL.
  NO LOCKS WERE HELD.
  NO SUPER BITS WERE SET.


RECOVERY ENVIRONMENT
  RECOVERY ROUTINE TYPE: FUNCTIONAL RECOVERY ROUTINE (FRR)
  PSW AT ENTRY TO FRR: 070C0000 82098DD0
  FRR PARAMETER AREA ON ENTRY TO FRR:
  +00  C9C7E6C6 C5C6D740 7F70B028 7F70B250 00000000 00000000


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 7F70C4BC 00000000 7F70C658  7F70B250 7F70B0A8 7F70C858 7F70C830
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  REGISTERS 8-15
  GR: 006E5F1C 020C1598 00000001 02518A7B  82517A7C 7F70C6A8 7F70B478 8251AB48
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
```

```
HEXADECIMAL DUMP

  HEADER
  +000    40831820    00000000    0094224F    10385969    | C.......M.|....|
  +010    FF176280    30900000                            |........        |

  JOBNAME
  +000    5CD4C1E2    E3C5D95C                            |*MASTER*        |

  SDWA BASE
  +000    00000C60    440F4000    00000000    00000000    |...-.. .........|
  +010    00000000    00000000    6204000C    440F4000    |............. .|
  +020    00000000    7F70C658    00FCF420    6204000C    |....".F...4.....|
  +030    00000024    00FD1CD0    00000000    7F70C4C8    |.......}....".DH|
  +040    00FCF3EC    02518A7B    82517A7C    7F70C6A8    |..3....#B.:@".FY|
  +050    82518324    00000024    00000000    00000000    |B.C............|
  +060    00000000    00000000    075C1000    8251832E    |.........*..B.C.|
  +070    0002000D    03C96001    070C0000    82098DD0    |.....I-.....B..}|
  +080    0002000D    03C96001    1E050019    7F70C4BC    |.....I-.....".D.|
  +090    00000000    7F70C658    7F70B250    7F70B0A8    |....".F."..&"..Y|
  +0A0    7F70C858    7F70C830    006E5F1C    020C1598    |".H.".H..>^....Q|
  +0B0    00000001    02518A7B    82517A7C    7F70C6A8    |.......#B.:@".FY|
  +0C0    7F70B478    8251AB48    00000000    00000000    |"...B..........|
  +0D0    00000000    00000000    00000000    00000000    |................|
  +0E0    00000000    00000000    04040001    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    00F93AA8    |.............9.Y|
  +140    00000000    00000000    00000000    00000000    |................|
  +150    00000000    00000000    00000000    00000000    |................|
  +160    00000000    00000000    00000000    FFFF0006    |................|
  +170    00F93DF8    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    |................|
```

```
DUMP FOR DATSVY02                                        3  11:12:05 11/29/94

  +030    00000000    00000000    00000000    00000001    |................|
  +040    00000000    00000000    00000000    82098DD0    |............B..}|
  +050    C9C7E6C6    C5C6D740    7F70B028    7F70B250    |IGWFEFP "..."..&|
  +060    00000000    00000000    00000000    00000000    |................|
  +070    00000000    00000001    006EBB20    CCB418F6    |.........>.....6|
  +080    0A0D4110    CC2C45E0    00FCFF00    01FFE07F    |.......\......\"|
  +090    440F4000    00000024    5EB1EE40    01FFE07F    |.. .....;.. ..\"|
  +0A0    00786180    80000001    00010001    01756040    |../...........- |
  +0B0    FE000000    01FFE07F    00000000    00000000    |......\"........|
  +0C0    00000000    00000000    010E1233    01FFE07F    |..............\"|
  +0D0    DF881755    7F6A7158    015209B8    00000000    |.H.."...........|
  +0E0    00000000    00000000    00000000    00000000    |................|
  +0F0    00000000    00000000    00000000    00000000    |................|
  +100    00000000    00000000    00000000    00000000    |................|
  +110    00000000    00000000    00000000    00000000    |................|
  +120    00000000    00000000    00000000    00000000    |................|
  +130    00000000    00000000    00000000    00000000    |................|
  +140    00000000    00000000    00000000    00000000    |................|
  +150    00000000    00000000    01756040    00000000    |..........- ....|
  +160    00000000    00000000    0175E680    00000000    |..........W.....|
  +170    00000000    00000000    00000000    00000000    |................|
  +180    00000000    00000000    00000000    00000000    |................|
  +190    00000000    00000000    00000000    00000000    |................|
  +1A0    00000000    7F6A7158    00000000    00000000    |...."...........|
  +1B0    00000000    00000000    00000000    00000000    |................|
  +1C0    00000000    00000000                            |........        |


  SDWA SECOND RECORDABLE EXTENSION (SDWARC2)
  +000    00000000    00000000    00000000    00000000    |................|

  SDWA THIRD RECORDABLE EXTENSION (SDWARC3)
  +000    00000000    00000000    00000000    00000000    |................|
  +010    00000000    00000000    00000000    00000000    |................|
  ERRORID
  +000    000C0041    00010005    D9A4                    |........RU      |

IEA24050I LOGDATA processing completed successfully.
```

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

```
*** NIP MESSAGE TABLE ***

UNABLE TO ACCESS UCM - NIP MESSAGE TRACE TERMINATED


*** MASTER TRACE TABLE ***

TAG  IMM DATA  |------------------------------------------- MESSAGE DATA -------------------------------------------|

0001 00999BF8  N 0000000 NONAME   88251 13:31:45.65          00000090  IEF196I SRB    0MIN 00.00SEC VIRT   476K SYS
               88K EXT      12K SYS
0001 00999BF8  N 0000000 NONAME   88251 13:31:45.65          00000090  IEF196I 9176K
0001 00999BF8  N 0000000 NONAME   88251 13:31:45.66          00000090  IEF196I IEF375I  JOB /LLA     / START 88251.1331
0001 00999BF8  N 0000000 NONAME   88251 13:31:45.67          00000090  IEF196I IEF376I  JOB /LLA     / STOP  88251.1331
               CPU    0MIN 00.08SEC
0001 0099C478  N 0000000 NONAME   88251 13:31:45.67          00000090  IEF196I SRB    0MIN 00.00SEC
0001 0099C478  N 0000000 NONAME   88251 13:31:45.67          00000090  IEF196I IEF358I INITIATOR TERMINATED DUE TO
               OUTSTANDING CROSS MEMORY
0001 0099C478  N 0000000 NONAME   88251 13:31:45.68          00000090  IEF196I BIND.
0001 00999BF8  N 4020000 NONAME   88251 13:31:45.67          00000090  IEF358I INITIATOR TERMINATED DUE TO OUTSTANDING
               CROSS MEMORY BIND.
0001 0099C478  N 0000000 NONAME   88251 13:31:45.71          00000090  IEA989I SLIP TRAP ID=X33E MATCHED
0001 0099C478  N 8040000 NONAME   88251 13:31:45.83          00000090 *CSV218E LIBRARY LOOKASIDE CRITICAL FAILURE
               (ABEND=SFF0 U000, REASON=--NONE--)
0001 00999BF8  NC0000000 NONAME   88251 13:45:19.66       04 00000090  S LLA,SUB=MSTR
0001 00999BF8  N 0000000 NONAME   88251 13:45:20.36          00000090  IEF196I     1     //LLA      JOB MSGLEVEL=1
0001 00999BF8  N 0000000 NONAME   88251 13:45:20.37          00000090  IEF196I     2     //STARTING EXEC LLA
0001 00999BF8  N 0000000 NONAME   88251 13:45:20.42          00000090  IEF196I     3     XXLLA PROC LLA=
0001 00999BF8  N 0000000 NONAME   88251 13:45:20.42          00000090  IEF196I     4     XXLLA EXEC PGM=CSVLLCRE,
               PARM='LLA=&LLA'
0001 00999BF8  N 0000000 NONAME   88251 13:45:20.42          00000090  IEF196I           IEF653I SUBSTITUTION JCL -
               PGM=CSVLLCRE,PARM='LLA='
0001 0099C478  N 0000000 NONAME   88251 13:45:20.89          00000090  IEF196I IEF236I ALLOC. FOR LLA LLA
0001 0099C478  M 4000000 NONAME   88251 13:45:24.32          00000090 *IEA911E COMPLETE DUMP ON SYS1.DUMP06
0001 00999BF8  D                                         837 00000090  FOR ASID (0017)
0001 00999C58  E                                         837 00000090   ERROR ID = SEQ00039 CPU00 ASID0017
               TIME13.45.20.9
0001 00998530  N 4000000 NONAME   88251 13:45:24.32          00000090  IEA994E   ALL SYS1.DUMP DATA SETS ARE FULL
0001 0099C844  N 0000000 NONAME   88251 13:45:24.35          00000090  IEF196I IEA995I SYMPTOM DUMP OUTPUT
0001 0099A390  N 0000000 NONAME   88251 13:45:24.35          00000090  IEF196I SYSTEM COMPLETION CODE=FF0
0001 0099A390  N 0000000 NONAME   88251 13:45:24.36          00000090  IEF196I  TIME=13.45.20  SEQ=00039  CPU=0000
               ASID=0017
0001 0099A390  N 0000000 NONAME   88251 13:45:24.36          00000090  IEF196I  PSW AT TIME OF ERROR  070C1000
               80FE5CFC  ILC 2  INTC 0D
0001 0099C844  N 0000000 NONAME   88251 13:45:24.36          00000090  IEF196I    NO ACTIVE MODULE FOUND
0001 0099C844  N 0000000 NONAME   88251 13:45:24.36          00000090  IEF196I    DATA AT PSW  00FE5CF6 - 00181610
               0A0D4110  016D182F
0001 0099A390  N 0000000 NONAME   88251 13:45:24.36          00000090  IEF196I    GPR  0-3  80000000  80FF0000
               009FF5A0  00FC4E88
0001 0099A390  N 0000000 NONAME   88251 13:45:24.36          00000090  IEF196I    GPR  4-7  009F8E88  009FD358
               80FE5CD6  00F40400
0001 0099C844  N 0000000 NONAME   88251 13:45:24.37          00000090  IEF196I    GPR  8-11 00000000  80FE579C
               009FD418  7FFFE2C0
0001 0099C844  N 0000000 NONAME   88251 13:45:24.37          00000090  IEF196I    GPR 12-15 7FFE0000  00006730
               00FE6200  80014910
0001 0099C844  N 0000000 NONAME   88251 13:45:24.37          00000090  IEF196I  END OF SYMPTOM DUMP
0001 0099C844  M 0020000 NONAME   88251 13:45:24.35          00000090  IEA995I SYMPTOM DUMP OUTPUT
0001 0099A390  D                                         850 00000090  SYSTEM COMPLETION CODE=FF0
0001 0099A3F0  D                                         850 00000090   TIME=13.45.20  SEQ=00039  CPU=0000  ASID=0017
0001 0099CACC  D                                         850 00000090   PSW AT TIME OF ERROR  070C1000   80FE5CFC  ILC
               2  INTC 0D
0001 0099CB2C  D                                         850 00000090     NO ACTIVE MODULE FOUND
0001 00999E80  D                                         850 00000090     DATA AT PSW  00FE5CF6 - 00181610  0A0D4110
               016D182F
0001 00999EE0  D                                         850 00000090     GPR  0-3  80000000  80FF0000  009FF5A0
               00FC4E88
```

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.

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

```
                    * * * *  N U C L E U S   M A P  * * * *

NUCLEUS MAP SORTED NUMERICALLY BY EPA

  NAME        LOCATION  ATTR  LENGTH   CSECT-NAME                  NAME        LOCATION  ATTR  LENGTH   CSECT-NAME

IEAVFX00      00000000   10   001000                              IEATSELM      00FC55C0   00   00014C    IEATPC
IEATCBP       00000218   00   000DE8    IEAVFX00                  IEFQMWR       00FC5710   10   000038
IECVDDT4      00F4A000   10   000048                              IEFENFDM      00FC5748   11   000030
IECVPRNT      00F4A048   12   000438                              IEABEND       00FC5778   10   0000D0
PRTDSE        00F4A04E   02   000432    IECVPRNT                  IEAPATCH      00FC5848   10   0000D5
PRTSIO        00F4A054   02   00042C    IECVPRNT                  IEAIHASU      00FC5910   00   00000D    IEAPATCH
PRTEOS        00F4A05A   02   000426    IECVPRNT                  IEFJESCT      00FC5920   10   000080
PRTTRAP       00F4A060   02   000420    IECVPRNT                  IFDOLT0A      00FC59A0   10   00004C
PRTDDT        00F4A066   02   00041A    IECVPRNT                  IEASMFEX      00FC59F0   13   000370
DDT1403       00F4A1F4   02   00028C    IECVPRNT                  IEAVESVT      00FC5D80   10   000258
DDT3203       00F4A228   02   000258    IECVPRNT                  IEAGSMQ       00FC5DA0   00   000238    IEAVESVT
DDT3211       00F4A264   02   00021C    IECVPRNT                  IEAGSPL       00FC5DA4   00   000234    IEAVESVT
DDT3800       00F4A2A0   02   0001E0    IECVPRNT                  IEALSMQ       00FC5DA8   00   000230    IEAVESVT
DDT4248       00F4A2E8   02   000198    IECVPRNT                  ICYMMVT       00FC5FD8   10   000458
PRTCCW        00F4A324   02   00015C    IECVPRNT                  ICYMMVTC      00FC604C   00   0003E4    ICYMMVT
IECVDDTR      00F4A480   10   00022C                              ICYPATCH      00FC60B8   00   000378    ICYMMVT
DDTR3480      00F4A4F4   00   0001B8    IECVDDTR                  ICYIOSB       00FC61B8   00   000278    ICYMMVT
DDTR342C      00F4A534   00   000178    IECVDDTR                  ICYMMSV1      00FC6230   00   000200    ICYMMVT
DDTR3400      00F4A570   00   00013C    IECVDDTR                  ICYMMPT1      00FC6330   00   000100    ICYMMVT
TAPECCW       00F4A5AC   00   000100    IECVDDTR                  IECVAFP1      00FC6430   12   000230
IECVDDTP      00F4A6B0   10   000048                              AFPSIO        00FC6490   02   0001D0    IECVAFP1
IGGDDT01      00F4A6F8   10   0001F2                              AFPTRAP       00FC6496   02   0001CA    IECVAFP1
IGGDDTA1      00F4A734   00   0001B6    IGGDDT01                  AFPDDT        00FC649C   02   0001C4    IECVAFP1
IGGDDT02      00F4A778   00   000172    IGGDDT01                  DDTAFP1       00FC650C   02   000154    IECVAFP1
IGGDDT04      00F4A7AC   00   00013E    IGGDDT01                  AFPCCW1       00FC6554   02   00010C    IECVAFP1
IGGDDTA4      00F4A7E8   00   000102    IGGDDT01                  DDTAFP22      00FC6660   10   00004E
IGGDDT05      00F4A82C   00   0000BE    IGGDDT01                  VSIMFLIH      00FC66B0   10   00041A
IGGDDT30      00F4A860   00   00008A    IGGDDT01                  VSFLIH        00FC66B0   00   00041A    VSIMFLIH
IGGDDT3V      00F4A894   00   000056    IGGDDT01                  VSIMRTRN      00FC68A6   00   000224    VSIMFLIH
IECDPERF      00F4A8F0   10   0000D2                              IEAVBK00      00FC6AD0   10   000044
IECDPC01      00F4A8F0   00   0000D2    IECDPERF                  IEFLINK       00FC6AD4   00   000040    IEAVBK00
IECDPC02      00F4A900   00   0000C2    IECDPERF                  LINKDCB       00FC6AD4   00   000040    IEAVBK00
IECDPC03      00F4A910   00   0000B2    IECDPERF                  IEASVDCB      00FC6ADC   00   000038    IEAVBK00
IECDPC04      00F4A920   00   0000A2    IECDPERF                  SVCDCB        00FC6ADC   00   000038    IEAVBK00
IECDPC05      00F4A930   00   000092    IECDPERF                  IEAQLPAQ      00FC6B10   00   000004    IEAVBK00
IECDPC06      00F4A940   00   000082    IECDPERF                  VSIM          00FC6B18   10   00FAC2
IECVDDT5      00F4A9C8   10   000048                              VSIMGPRS      00FC7400   00   00F1DA    VSIM
IECVDDT7      00F4AA10   10   000048                              VSIMFPRS      00FC7440   00   00F19A    VSIM
IECVOPTB      00F4AA58   10   000118                              VMSKMODE      00FC7490   00   00F14A    VSIM
IECVDDTE      00F4AB70   10   000048                              SS            00FC7494   00   00F146    VSIM
IECVDDT2      00F4ABB8   10   000048                              PS            00FC7498   00   00F142    VSIM
IECZDTAB      00F4AC00   10   0000B0                              VCT           00FC749C   00   00F13E    VSIM
IOSUCB        00F4ACB0   10   07A0C0                              VIX           00FC74A0   00   00F13A    VSIM
IOSDCQ        00FC4D70   10   0000D8                              VIU           00FC74A4   00   00F136    VSIM
IEAVCVT       00FC4E48   10   000620                              VCBITS        00FC74A8   00   00F132    VSIM
IEACVT        00FC4E88   00   0005E0    IEAVCVT                   VMASK         00FC74AC   00   00F12E    VSIM
IEFAUSDM      00FC5468   10   000028                              VIPSW         00FC752C   00   00F0AE    VSIM
IEEMSER       00FC5490   10   0000D4                              VSIMCC        00FC7530   00   00F0AA    VSIM
BAIPL         00FC549C   00   0000C8    IEEMSER                   PSCOUNT       00FC7534   00   00F0A6    VSIM
IEAMSLNK      00FC54A0   00   0000C4    IEEMSER                   VINTCODE      00FC7538   00   00F0A2    VSIM
IEATPC        00FC5568   10   0001A4                              VSIMVECT      00FC8E18   00   00D7C2    VSIM
TPCMFTQE      00FC55A8   00   000164    IEATPC                    V00           00FC8E18   00   00D7C2    VSIM
V01           00FC9618   00   00CFC2    VSIM                      IGC045        00FDAA60   00   0000A4    IGC037
V02           00FC9E18   00   00C7C2    VSIM                      IEAMSWCB      00FDAB80   10   0015E0
V03           00FCA618   00   00BFC2    VSIM                      IEASTKH       00FDABA0   00   0015C0    IEAMSWCB
V04           00FCAE18   00   00B7C2    VSIM                      IEALSPC       00FDABB8   00   0015A8    IEAMSWCB
V05           00FCB618   00   00AFC2    VSIM                      IEAISSAT      00FDABC8   00   001598    IEAMSWCB
V06           00FCBE18   00   00A7C2    VSIM                      IEAMASCB      00FDAC00   00   001560    IEAMSWCB
V07           00FCC618   00   009FC2    VSIM                      IEAMSTCB      00FDAFA0   00   0011C0    IEAMSWCB
V08           00FCCE18   00   0097C2    VSIM                      IEAMPRB       00FDB260   00   000F00    IEAMSWCB
V09           00FCD618   00   008FC2    VSIM                      IEAMSIRB      00FDB360   00   000E00    IEAMSWCB
V10           00FCDE18   00   0087C2    VSIM                      IEAWASCB      00FDB538   00   000C28    IEAMSWCB
V11           00FCE618   00   007FC2    VSIM                      IEAWTCB       00FDB778   00   0009E8    IEAMSWCB
```

The nucleus map sorted numerically by entry point address (EPA) continues with data like the above.

```
                              * * * *  N U C L E U S   M A P  * * * *

NUCLEUS MAP SORTED ALPHABETICALLY BY NAME

  NAME        LOCATION  ATTR  LENGTH   CSECT-NAME                  NAME        LOCATION  ATTR  LENGTH   CSECT-NAME

ABN0000       011F012C   0E   000854    ISGJDI                    AVFFS         01157FC0   1E   000AD8
ACMSRALH      00FF41A0   0B   0004E0    IEAVELIT                  AVFFSFRR      01157FE0   0E   000AB8    AVFFS
ACRRETRY      01188FB4   0E   00049C    IEAVTCR1                  AVFIS         01158A98   1E   0008C8
ACRSTART      01187E16   0E   000D0A    IEAVTACR                  AVFISRB       01158AB8   0E   0008A8    AVFIS
ACRSUPER      01188FAE   0E   0004A2    IEAVTCR1                  AVFISXIT      01159008   0E   000358    AVFIS
ADISP         00FF3F00   0B   000780    IEAVELIT                  AVFIX         00FF43A4   0B   0002DC    IEAVELIT
AFPCCW1       00FC6554   02   00010C    IECVAFP1                  AVFRM         01159360   1E   000930
AFPDDT        00FC649C   02   0001C4    IECVAFP1                  AVFRMTRP      01159720   0E   000570    AVFRM
AFPSIO        00FC6490   02   0001D0    IECVAFP1                  AVFRMTRT      01159380   0E   000910    AVFRM
AFPTRAP       00FC6496   02   0001CA    IECVAFP1                  AVFTB         01159C90   1E   0009E8
AHLHEAD       00FDC978   02   000048    AHLMCIH                   AVFTBACK      01159CB0   0E   0009C8    AVFTB
AHLMCIH       00FDC710   12   0002B0                              AVFTBRTY      0115A452   0E   000226    AVFTB
AHLMCIHB      00FDC8B6   02   00010A    AHLMCIH                   AVFTC         0115A678   1E   000550
AHLSTCLS      00FDC94E   02   000072    AHLMCIH                   AVFTCLPE      0115A7D8   0E   0003F0    AVFTC
AHLVCCR8      00FDC94E   02   000072    AHLMCIH                   AVFTCLPM      0115A698   0E   000530    AVFTC
AOMAFRR       010982C8   1E   000818                              AVFTD         0115ABC8   1E   000900
AOMAIO        01097C18   1E   0006B0                              AVFTDCPB      0115ABE8   0E   0008E0    AVFTD
AOMARQ        010955D0   1E   0009C8                              AVFTDCPS      0115B020   0E   0004A8    AVFTD
AOMARQ2       0109593A   0E   00065E    AOMARQ                    AVFTE         0115B4C8   1E   000C70
AOMATTN       01095F98   1E   0011E0                              AVFTERTY      0115BDB6   0E   000382    AVFTE
AOMATTNP      01097178   1E   000AA0                              AVFTEXT1      0115B4E8   0E   000C50    AVFTE
AOMATTN1      01096410   0E   000D68    AOMATTN                   AVFTF         0115C138   1E   0008B8
AOMATTN2      010964C0   0E   000CB8    AOMATTN                   AVFTFINT      0115C158   0E   000898    AVFTF
AOMATTN3      0109686E   0E   00090A    AOMATTN                   AVFTFRNT      0115C3F0   0E   000600    AVFTF
AOMCBLKS      01211020   16   000128                              AVFTG         00FF1788   1B   000038
AOMGTRCE      01098AE0   1E   0004C8                              AVFTGLUE      00FF17A8   0B   000018    AVFTG
AOMMTERM      0109A9C8   1E   0001D0                              AVFTM         0115C9F0   1E   000BF0
AOMQERR       01098FA8   1E   000A68                              AVFTMAIN      0115CA10   0E   000BD0    AVFTM
AOMQMGR       01099A10   1E   000FB8                              AVFTMRTY      0115D252   0E   00038E    AVFTM
AOMSSSVT      012110C8   06   000080    AOMCBLKS                  AVFTS         0115D5E0   1E   0005D8
ASALLOC       00FF40B0   0B   0005D0    IEAVELIT                  AVFTSRBR      0115D600   0E   0005B8    AVFTS
ASMGLP        00FF4610   0B   000070    IEAVELIT                  AVPAG         00FF4254   0B   00042C    IEAVELIT
ASMP          00FF4550   0B   000130    IEAVELIT                  BAIPL         00FC549C   00   0000C8    IEEMSER
ASMVT         00FD8930   00   000500    ILRASM00                  CBBRANCH      00FFE7FC   0B   00000C    IGVVSCAP
ASRGLTAB      01205218   16   000040                              CLRDIE        011E1EB4   0E   000F8C    IOSRMIHR
ASRM          00FF40E0   0B   0005A0    IEAVELIT                  CMLCIOBT      00FE7232   0B   000ECE    IEAVESLK
ASRSERVA      011548DC   0E   001264    ASRSERVP                  CMLCOBT       00FE7244   0B   000EBC    IEAVESLK
ASRSERVE      011552EA   0E   000856    ASRSERVP                  CMLP          00FF4590   0B   0000F0    IEAVELIT
ASRSERVM      011552EC   0E   000854    ASRSERVP                  CMLREL        00FE7674   0B   000A8C    IEAVESLK
ASRSERVP      01153D88   1E   001DB8                              CMLUOBT       00FE7370   0B   000D90    IEAVESLK
ASRSERVR      01154AD0   0E   001070    ASRSERVP                  CMSCOBT       00FE7932   0B   0007CE    IEAVESLK
ASRSERVX      01154ACC   0E   001074    ASRSERVP                  CMSEDLK       00FDCA70   00   000048    IEAVESLA
ASRSERVZ      011554DE   0E   000662    ASRSERVP                  CMSFIRST      00FDCA60   00   000058    IEAVESLA
AVFFA         01155B40   1E   0007B8                              CMSFRSQH      00FDCA64   00   000054    IEAVESLA
AVFFAFRE      01155B60   0E   000798    AVFFA                     CMSLOCK       00FDCA80   00   000038    IEAVESLA
AVFFAFSE      01155E00   0E   0004F8    AVFFA                     CMSP          00FF45A0   0B   0000E0    IEAVELIT
AVFFD         011562F8   1E   000780                              CMSRALL       00FE7D8C   0B   000374    IEAVESLK
AVFFDIAG      01156318   0E   000760    AVFFD                     CMSRALLH      00FE7D8C   0B   000374    IEAVESLK
AVFFL         01156A78   1E   000620                              CMSREL        00FE7A00   0B   000700    IEAVESLK
AVFFLOOP      01156A98   0E   000600    AVFFL                     CMSRELI       00FE7A1C   0B   0006E4    IEAVESLK
AVFFR         01157098   1E   000F28                              CMSSMFLK      00FDCA60   00   000058    IEAVESLA
AVFFRR        011570B8   0E   000F08    AVFFR                     CMSSQH        00FDCA84   00   000034    IEAVESLA
CMSUOALL      00FE7AB4   0B   00064C    IEAVESLK                  CSVSBCHK      01162304   0E   00061C    CSVSBRTN
CMSUOBT       00FE7880   0B   000880    IEAVESLK                  CSVSBLDF      01161A38   0E   000EE8    CSVSBRTN
CMSUOBTI      00FE789C   0B   000864    IEAVESLK                  CSVSBLDL      01162100   0E   000820    CSVSBRTN
COFMESTA      0115DBB8   1E   0012C8                              CSVSBLFD      011620CC   0E   000854    CSVSBRTN
COFMIDEN      0115EE80   1E   000B68                              CSVSBMFD      01161E46   0E   000ADA    CSVSBRTN
COFMMTGR      0115F9E8   1E   0002E8                              CSVSBRTN      01161A38   1E   000EE8
CPUAFF        00FF74F4   0B   000624    IEAVESTS                  CSVSRCH       00FF17C0   1B   000072
CPUOBT        00FF2846   0B   000FDA    IEAVELK                   CSVSYNCH      01162920   1E   000570
CPUP          00FF4640   0B   000040    IEAVELIT                  CSVS2CDQ      011637F8   0E   000230    CSVS2RTN
CPUREL        00FF2862   0B   000FBE    IEAVELK                   CSVS2GWK      01163708   0E   000320    CSVS2RTN
CRBRANCH      00FFE7E4   0B   000024    IGVVSCAP                  CSVS2LIB      01163580   0E   0004A8    CSVS2RTN
CSRABEND      01030050   0E   000018    CSRABRTN                  CSVS2LSR      011636D4   0E   000354    CSVS2RTN
CSRABRTN      0102F050   1E   001018                              CSVS2RDQ      01163888   0E   0001A0    CSVS2RTN
CSRDWST       01205140   16   000040                              CSVS2RTN      01163580   1E   0004A8
CSRTABLE      011FFDB0   16   001000                              CSVVFEP1      01163AAA   0E   0009DE    CSVVFNDE
```

The nucleus map sorted alphabetically by name continues with data like the above.

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).

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*.

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 could exist in the buffers if an SVC dump was 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

```
STORAGE TYPE        RANGE START       RANGE END        ASID ATTRIBUTES
                          023BCD70          023BCD7F   001E (COMMON)
SUMLSTA RANGE --          017E8000          017E8FFF   0001 (COMMON)
SUMLSTA RANGE --          01F9B000          01F9CFFF   0001 (COMMON)
SUMLSTA RANGE --          02166000          02167FFF   0001 (COMMON)
PSA ------------          00000000          00001FFF   001E (COMMON)
PCCA -----------          00F43008          00F4324F   001E (COMMON)
LCCA -----------          00F82000          00F82A47   001E (COMMON)
LCCX -----------          021C7000          021C771F   001E (COMMON)
INT HANDLER DUCT          02232FC0          02232FFF   001E (COMMON)
I.H. LINKAGE STK          02262000          0226202F   001E (COMMON)
REGISTER AREA --          0000E000          00010FFF   001E
REGISTER AREA --          00FC4000          00FC6FFF   001E (COMMON)
REGISTER AREA -- 00000001_7F5AD000 00000001_7F5B0FFF   001E
REGISTER AREA --          7FFFE000          7FFFEFFF   001E
```

VERBEXIT SYMPTOM Subcommand — Format Symptom String

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 MVS Diagnosis: Procedures*.

If the user has the authority to update the dump data set, the user can use the IPCS add symptom service to add secondary symptom strings up to 2048 bytes; the actual length may be fewer bytes, depending on the space available in the dump header record. 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

```
                    * * * * S Y M P T O M * * * *

Primary Symptom String:

 RIDS/CSVLLCRE#L RIDS/CSVLLBLD PIDS/5655CICJ AB/S0FF0 RIDS/CSVLLBLD#R
 VALU/H016D182F REGS/09500 REGS/06026 VALU/COOKASIDE

 Symptom            Symptom data      Explanation
 ----------------   -------------     -----------
 RIDS/CSVLLCRE#L    CSVLLCRE#L        Routine identifier
 RIDS/CSVLLBLD      CSVLLBLD          Routine identifier
 PIDS/5655CICJ      5655CICJ          Component identifier
 AB/S0FF0           0FF0              ABEND code - system
 RIDS/CSVLLBLD#R    CSVLLBLD#R        Routine identifier
 VALU/H016D182F     016D182F          Error related hexadecimal value
 REGS/09560         09560             Program register
 REGS/06026         06026             Program register
 VALU/COOKASIDE     OOKASIDE          Error related character value


The dump does not contain a secondary symptom string.
```

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 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 explictly requested or implied by omission of all qualifying options.

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 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 defaults are CURRENT and ERROR. 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.

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 ore 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 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 or not 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(majorname) ]
                               [MINOR(minorname) ]

          [ 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 5-223.
           [ 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 SUMMARY.

Note: In the following descriptions, *vlfclass* is a class name consisting of 1 to 7 alphameric 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(majorname)
Requests a report containing details about 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* 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:
  - 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 upper case 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.

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 simply 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 LIST service aid to format and print the load module. The LIST 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 LIST.

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(SUBPOOL*sp*KEY(*key*), where

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).

However, 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 usually displayed followed by the name of the area.

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 }   data-descr
{ W     }
```

-------- SETDEF-Defined Parameters ----------
Note: You can override the following SETDEF parameters. See
"SETDEF Subcommand — Set Defaults" on page 5-223.

```
[ 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.

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 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.

```
IPCS OUTPUT STREAM ------------------------------------- LINE 0 COLS 1 78
COMMAND ===> _                                            SCROLL ===> CSR
 ****************************** TOP OF DATA ***********************************
      ASID(X'0058') 00FD2834. IEANUC01.IOSUCB+020D14 IN READ/WRITE NUCLEUS
 ****************************** END OF DATA ***********************************
```

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 had been used in this example, the item that contains the address would have been added to the pointer stack. If more than one item contained the address, the item with the smallest offset would have been 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:

```
===> SYSTRACE
```

Now, enter WHERE on the command line of the system trace table.

```
IPCS OUTPUT STREAM ---------------------------------------- LINE 471 COLS 1 78
COMMAND ===> WHERE 1D073D0                          SCROLL ===> CSR
 01 001B 009F8E88   SVCR   78 070C2000 81D07386  00000000 00000080 7FFFDD10
 01 001B 009F8E88   SVC    78 070C3000 81D073B2  0000E101 00000030 7FFFDD10
 01 001B 009F8E88   SVCR   78 070C3000 81D073B2  00000000 00000030 7FFFDD10
 01 001B 009F8E88   SVC    78 070C1000 81D073D0  0000E101 00000010 7FFFDD80
 01 001B 009F8E88   SVCR   78 070C1000 81D073D0  00000000 00000010 7FFFDD80
 01 001B 009F8E88   SVCR   2A 070C1000 8001B832  00000000 00000000 009F8CF0
 01 001B 009F8E88   SVC    78 070C2000 8001B93E  00000003 00000190 00006C60
 01 001B 009F8E88   SVCR   78 070C2000 8001B93E  00000000 00000190 00006C60
 01 001B 009F8E88   SVC    78 070C0000 80008B50  00000002 00000368 00000000
 01 001B 009F8E88   SVCR   78 070C0000 80008B50  00000000 00000368 00006A88
 01 001B 009F8E88   SVC    38 070C1000 80008C4E  00000000 00000006 00006C78
 01 001B 009F8E88   PC    ...   0      811520D2  00108
 01 001B 009F8E88   PT    ...   0      811520D2  001B
 01 001B 009F8E88   SVCR   38 070C1000 80008C4E  00000000 00000002 00F3FB14
 01 001B 009F8E88   SVC    78 070C0000 80019294  00000002 00000358 00000000
 01 001B 009F8E88   SVCR   78 070C0000 80019294  00000000 00000358 00006730
 01 001B 009F8E88  ?EXPL 0003                    77007B16 C3629700 00000003
 01 000E 009F6E88   SRB       070C0000 811A3180  0000000E 00F42200 00F4222C
                                                 80
 01 000E 00000000   SSRV    2          80FFB540  000A002C 7F000000 00000000
                                                 00000000
 01 000E 00000000   SSRV  12A          80007478  00B9E8B0 40000000 00000000
                                                 00000000
 01 000E 009F6E88   DSP       070C0000 800073C2  00000000 000070C0 00B9E8B0  00
 01 000E 009F6E88   SSRV    1          80007388  00B9E8B0 00000001 00000000
                                                 00000000
 01 001B 009F8E88   DSP       070C0000 800192C8  00000000 00000002 00006B48  00
 01 001B 009F8E88   SVC    78 070C2000 8001936C  0000E572 00001000 00000000
 01 001B 009F8E88   SVCR   78 070C2000 8001936C  00000000 00001000 7FFE0000
 01 001B 009F8E88   SVC    78 070C2000 800199B8  0000EF72 00000018 00000000
 01 001B 009F8E88   SVCR   78 070C2000 800199B8  00000000 00000018 01B1E640
 01 001B 009F8E88   SVC    F0 070C2000 80014912  80014910 00000000 0000677C
 01 001B 009F8E88  *SVC     D 070C1000 80FE5CFC  80014910 80000000 80FF0000
 01 001B 009F8E88   PC    ...   0      811808CE  00506
 01 001B 009F8E88   PC    ...   0      82106B2A  00503
 ***************************** END OF DATA ***********************************
```

– 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.

```
IPCS OUTPUT STREAM ------------------------------------- LINE 0 COLS 1 78
COMMAND ===> _                                            SCROLL ===> CSR
 ****************************** TOP OF DATA ***********************************
      ASID(X'001B') 01D073D0. IGC0004B+03D0 IN EXTENDED PLPA
 ****************************** END OF DATA ***********************************
```

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.

```
DSNAME('D46IPCS.DRVC400.SA00001') POINTERS  -----------------------------------
COMMAND ===> _                                            SCROLL ===> CSR
PTR   Address  Address space                            Data type
00001 00000000 ASID(X'0003')                            AREA
      Remarks:
00002 000006B0 ASID(X'0003')                            AREA
      Remarks:
00003 00FD7420 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
-------------------------------------------------------------------------------
|00004 01D07000 ASID(X'001B')                           MODULE(IGC0004B)      |
|      Remarks:                                                               |
-------------------------------------------------------------------------------
***************************** END OF POINTER STACK ******************************
```

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.

WHERE Subcommand

```
IPCS OUTPUT STREAM ------------------------------------- LINE 0 COLS 1 78
COMMAND ===> WHERE 120E298                                  SCROLL ===> CSR
 ****************************** TOP OF DATA ***********************************

                    * * * * * L O G D A T A * * * * *

 TYPE: SOFTWARE RECORD               REPORT: SOFTWARE EDIT REPORT   DAY.YEAR
       (SVC 13)                                         REPORT DATE 197.95
 SVC: VS 2 REL 3                                         ERROR DATE 158.95
                                     MODEL:3090                     HH:MM:SS.T
                                     SERIAL:170067                  13::1:44.9
 JOBNAME: D22AMSTR
 ERRORID: SEQ=17095 CPU=0041 ASID=0047 TIME=13:31:43.9

 < Some Output Not Shown >

 TIME OF ERROR INFORMATION

   PSW: 070C2000 8120E298   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D
   FAILING INSTRUCTION TEXT: 581097E6 0A0D4830 C0105800

   REGISTERS 0-7
   GR: 00000000 04C78000 006F72D8 0000E601 006F72D8 7FF12080 00000004 00EE5500
   AR: 0124E1B8 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

– 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.

```
IPCS OUTPUT STREAM ------------------------------------- LINE 0 COLS 1 78
COMMAND ===> _                                              SCROLL ===> CSR
 ****************************** TOP OF DATA ***********************************
      ASID(X'0058') 0120E298. IEANUC01.IGVSTSKT+0200 IN READ ONLY NUCLEUS
 ****************************** END OF DATA ***********************************
```

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 all of the detailed information in that register or PSW.

```
DSNAME('D46IPCS.DRVC400.SA00001') POINTERS  ------------------------------------
COMMAND ===> _                                              SCROLL ===> CSR
PTR   Address  Address space                            Data type
00001 00000000 ASID(X'0003')                            AREA
      Remarks:
00002 000006B0 ASID(X'0003')                            AREA
      Remarks:
00003 00FD7420 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
-------------------------------------------------------------------------------
|00004 0120E298 ASID(X'0058')                           MODULE(IEANUC01)      |
|      Remarks:                                                               |
-------------------------------------------------------------------------------
***************************** 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. IGC0006A+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
+0000  C9C7C3F0  F0F1F3C0                     | IGC0013{         |
```

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(subsystype)]
                            [,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 5-223.

```
[ 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 wish. 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.

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 ASDIs. 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     ]
```

-------- 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 5-223.

```
[ 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

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, as well as 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

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, as well as 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

I>

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.

Data Selection Parameter	REPORT TYPES							
	CACHE	CON-NEC-TION	FACIL-ITY	LIST	LOCK-MGR	LOCK RE-SOURCE	XES-STACK	LOCK
ASID	X	X		X	X	X		X
CFNAME	X	X	X	X	X	X		X
CONNAME	X	X	X	X	X	X		X
HASHVALUE				X	X			
JOBNAME	X	X		X	X	X		X
LISTNUM				X				
LOCKMGRCONID						X		
LTENTRY		X		X	X	X		
REQID	X			X				
REQUESTORCONID				X	X			
RNAME		X			X	X		
SOURCENAME		X						
STRNAME	X	X	X	X	X	X		X
SYSNAME		X						
TARGETNAME		X						

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.

An example is:

```
COMMAND ===>  XESDATA LOCKMGR LTENTRY(20)
```

REQID(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 REQID(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 resources 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.

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*.

Chapter 6. IPCS Dialog Controls

This chapter describes the IPCS dialog controls. Use these controls in the IPCS full-screen problem analysis dialog (called the IPCS dialog in this book), except in TUTORIAL. The controls are:

- IPCS primary commands
- Line commands
- Program function (PF) keys
- Command codes
- Selection codes
- ISPF primary commands

The *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:

Command	Function
ASCII	Display ISO-8 ASCII characters
CANCEL	End the BROWSE option
CBFORMAT	Format a control block
DOWN	Scroll data forward
EBCDIC	Display EBCDIC characters
END	End a subcommand or panel
EQUATE	Create a user_defined symbol
FIND	Search for a specified value
IPCS	Invoke an IPCS subcommand, CLIST, or REXX exec
LEFT	Scroll data left
LOCATE	Scroll the display to show specific data
MORE	Scroll data
OPCODE	Display mnemonic operation code
RENUM	Renumber symbol entries
RESET	Remove pending commands
RETURN	Display the IPCS Primary Option Menu
RFIND	Repeat the FIND command
RIGHT	Scroll data right
SELECT	Select a pointer to display storage

© Copyright IBM Corp. 1988, 2002

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 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 the line command.

The line commands are:

Command	Function
D	Delete screen output
E	Edit a pointer
F	Format a defined control block
I	Insert a pointer
R	Repeat a pointer
S	Select a pointer to display storage
S, F, or L	Show excluded screen output
X	Exclude screen output

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
- “Dump Display Reporter Panels” on page 6-4

- “IPCS Inventory Panel” on page 6-5
- “Storage Panel” on page 6-6.

Note: Commands identified as IPCS in the following tables are described in this chapter. Commands identified as ISPF are in *z/OS ISPF Dialog Tag Language Guide and Reference*

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

When You Want to:	Enter ===>	Use PF Key
Get **help**	HELP command (ISPF)	1 or 13
Split the screen	SPLIT command (ISPF)	2 or 14
End or cancel	END primary command (IPCS)	3 or 15
Return to IPCS Primary Option Menu	RETURN primary command (IPCS)	4 or 16
Swap screens	SWAP command (ISPF)	9 or 21
Move the **cursor** to the command/option line	CURSOR command (ISPF)	12 or 24
Invoke an IPCS subcommand, CLIST, or REXX exec	IPCS primary command (IPCS)	—

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

When You Want to:	Enter ===>	Use PF Key
Get **help**	HELP command (ISPF)	1 or 13
Reset entered commands	RESET primary command (IPCS)	—
Split the screen	SPLIT command (ISPF)	2 or 14
End processing	END primary command (IPCS)	3 or 15
Cancel processing	CANCEL primary command (IPCS)	—
Return to IPCS primary option menu	RETURN primary command (IPCS)	4 or 16
Search for a value	FIND primary command (IPCS)	—
Repeat the FIND command	RFIND primary command (IPCS)	5 or 17
Use **Symbols** to: • Create an IPCS defined symbol • Create a user-defined symbol • Renumber stack entries	STACK primary command (IPCS) EQUATE primary command (IPCS) RENUM primary command (IPCS)	6 or 18 — —

Table 6-2. Pointer and Storage Panels - Commands and PF Keys (continued)

When You Want to:	Enter ===>	Use PF Key
Scroll • Up (towards top) • Down (towards bottom)	UP primary command (IPCS) DOWN primary command (IPCS)	7 or 19 8 or 20
Swap screens.	SWAP command (ISPF)	9 or 21
Display a pointer or storage	LOCATE primary command (IPCS) SELECT primary command (IPCS)	—
Browse through a dump by positioning the cursor To a 24-bit address; the pointer is recorded on the pointer panel To a 31-bit address; the pointer is recorded on the pointer panel	STACK X primary command (IPCS) LOCATE CURSOR% primary command (IPCS) STACK X primary command (IPCS) LOCATE CURSOR? primary command (IPCS)	 10 or 22 11 or 23
Move the cursor to command/option line	CURSOR command (ISPF)	12 or 24
Format a control block	CBFORMAT primary command (IPCS)	—
Identify areas of storage that contain an address	WHERE primary command (IPCS)	—
Invoke an IPCS subcommand, CLIST, or REXX exec	IPCS primary command (IPCS)	—
Select a pointer and **display** storage addressed by that selected pointer	S (select) line command (IPCS)	—
Delete pointers	D (delete) line command (IPCS)	—
Edit a selected pointer	E (edit) line command (IPCS)	—
Format a pointer with a data-type of STRUCTURE	F (format) line command (IPCS)	—
Insert pointers	I (insert) line command (IPCS)	—
Replicate existing pointers	R (repeat) line command (IPCS)	—

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

When You Want to:	Enter ===>	Use PF Key
Get **help**	HELP command (ISPF)	1 or 13
Reset entered commands	RESET primary command (IPCS)	—
Split the screen	SPLIT command (ISPF)	2 or 14
End processing	END primary command (IPCS)	3 or 15
Return to IPCS primary option menu	RETURN primary command (IPCS)	4 or 16
Search for a value	FIND primary command (IPCS)	—

Table 6-3. Dump Display Reporter Panel - Commands and PF Keys (continued)

When You Want to:	Enter ===>	Use PF Key
Repeat the FIND command	RFIND primary command (IPCS)	5 or 17
Scroll		6 or 18
To next full screen	MORE primary command (IPCS)	7 or 19
Up (towards top)	UP primary command (IPCS)	8 or 20
Down (towards bottom)	DOWN primary command (IPCS)	10 or 22
Left	LEFT primary command (IPCS)	11 or 23
Right	RIGHT primary command (IPCS)	—
To specific data	LOCATE primary command (IPCS)	
Swap screens	SWAP command (ISPF)	9 or 21
Move the cursor to the command/option line	CURSOR command (ISPF)	12 or 24
Format a control block	CBFORMAT primary command (IPCS)	—
Identify areas of storage that contain an address	WHERE primary command (IPCS)	—
Invoke an IPCS subcommand, CLIST, or REXX exec	IPCS primary command (IPCS)	—
Delete screen lines permanently	D (delete) line command (IPCS)	—
Exclude screen lines.	X (exclude) line command (IPCS)	—
Display excluded screen lines For excluded lines For the first line of excluded text For the last line of excluded text	S, F, or L (show) line command (IPCS)	—

IPCS Inventory Panel

The following 2-character command codes are used to manage the inventory panel.

Code	Function Performed
BR	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.
CL	Close the source. Resources that were obtained by dump OPEN processing are immediately released.
DD	Delete the source description of the indicated source from the dump directory.
DT	Delete translation records from the source description in the dump directory.
LA	List the source description, with storage attributes.
LB	List the source description, with record locations.
LD	List the source description, with dumped storage summary.
LT	List the source description, with translation results.

Code	Function Performed
LZ	List the source description, with all the information from the other LIST options.
OP	OPEN the source for processing.
SD	Establish a data set as the default source.

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*.

Selection Code	Actions by IPCS
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 on 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.

– 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 towards the bottom of data.

Syntax

DOWN Primary Command

```
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 usually 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 towards 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

```
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

```
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

```
{ EQUATE }   symbol
{ EQU    }
{ EQ     }
           [ 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

Chapter 3, “Data Description Parameter” on page 3-1 has more information on 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

- 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’.

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 All Options Except BROWSE and TUTORIAL

```
{ FIND }   [relational-operator]
{ F    }
           value

          [ col [ col ]]

          [ FIRST    ]
          [ LAST     ]
          [ NEXT     ]
          [ PREVIOUS ]
```

Parameters

relational-operator
Specifies one of the following symbolic or programming operators to be used with the value operand:

[<|LT|<=|LE|¬>|NG|=|EQ|>=|GE|¬<|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.

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

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.

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 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 NOBREAK, the default is BREAK.

MASK(mask)
NOMASK
MASK defines a value that is logically ANDed with both operands prior to 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.

FIRST
LAST
NEXT
PREVIOUS
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.

– 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 on page 6-14 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 6-14 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 ---------------------------------------- LINE 0 COLS 1 78
---------------------
|COMMAND ===> f dsp_ |                                SCROLL ===> CSR
---------------------
 ****************************** TOP OF DATA ***********************************

     * * * *   K E Y F I E L D S   * * * *
 JOBNAME RASP
     SELECTED BY: CURRENT

  ASCB: 00F09E00
     FWDP..... 00F09C00  ASID..... 0003      TRQP..... 80F09301
     CSCB..... 00F1D3C8  TSB...... 00000000  AFFN..... FFFF
     ASXB..... 00AFDF00  DSP1..... 00        FLG2..... C4
     SRBS..... 0000      LOCK..... 00000000  ASSB..... 01AB6D00

  TCB: 00AFE178
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 00AFD300  NDSP..... 00000000
     JSCB..... 00AFDD84  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00
     Task non-dispatchability flags from TCBFLGS4:
      Top RB is in a wait

  PRB: 00AFDD60
     WLIC..... 00020001  FLCDE.... 00C12630  OPSW..... 070C1000  810D7C20
```

Figure 6-1. Using FIND on the Dump Display Reporter Panel

```
IPCS OUTPUT STREAM ---------------------------------- FOUND IN LINE 16 COL 17
COMMAND ===>                                            SCROLL ===> CSR
 ****************************** TOP OF DATA ***********************************

     * * * *   K E Y F I E L D S   * * * *
 JOBNAME RASP
     SELECTED BY: CURRENT

  ASCB: 00F09E00
     FWDP..... 00F09C00  ASID..... 0003      TRQP..... 80F09301
     CSCB..... 00F1D3C8  TSB...... 00000000  AFFN..... FFFF
-----------------------------------------------------------------------
|    ASXB..... 00AFDF00: DSP1..... 00        FLG2..... C4            |
-----------------------------------------------------------------------
     SRBS..... 0000      LOCK..... 00000000  ASSB..... 01AB6D00

  TCB: 00AFE178
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 00AFD300  NDSP..... 00000000
     JSCB..... 00AFDD84  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00
     Task non-dispatchability flags from TCBFLGS4:
      Top RB is in a wait

  PRB: 00AFDD60
     WLIC..... 00020001  FLCDE.... 00C12630  OPSW..... 070C1000  810D7C20
```

Figure 6-2. Result of Using FIND

```
IPCS OUTPUT STREAM ---------------------------------- FOUND IN LINE 16 COL 39
COMMAND ===>                                           SCROLL ===> CSR
 ****************************** TOP OF DATA ***********************************

     * * * *   K E Y F I E L D S   * * * *
 JOBNAME RASP
     SELECTED BY: CURRENT

  ASCB: 00F09E00
     FWDP..... 00F09C00  ASID..... 0003      TRQP..... 80F09301
     CSCB..... 00F1D3C8  TSB...... 00000000  AFFN..... FFFF
     ASXB..... 00AFDF00  DSP1..... 00        FLG2..... C4
     SRBS..... 0000      LOCK..... 00000000  ASSB..... 01AB6D00

  TCB: 00AFE178
-------------------------------------------------------------------------------
|    CMP...... 00000000  PKF...... 00        LMP.....  FF       DSP.....FF|
-------------------------------------------------------------------------------
     TSFLG.... 00        STAB..... 00AFD300  NDSP..... 00000000
     JSCB..... 00AFDD84  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00
     Task non-dispatchability flags from TCBFLGS4:
      Top RB is in a wait

  PRB: 00AFDD60
     WLIC..... 00020001  FLCDE.... 00C12630  OPSW..... 070C1000  810D7C20
```

Figure 6-3. Result of Using PF5/RFIND

IPCS Primary Command — Invoke an IPCS Subcommand, CLIST, or REXX Exec

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.

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 prior to 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 towards 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 usually 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

  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.
  - **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.

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 in the selection field prior to a word or is placed over the specific halfword where the instruction of interest originates.
- The operation code beginning in the first halfword shown on the screen when the provious 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.

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 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 towards the last, or right-most, column of the data.

RIGHT Primary Command

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 usually 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.

```
 DSNAME('D83DUMP.DUMPC.PB00465') POINTERS -------------------------------------
-------------------------------------------------------------------------------
|COMMAND ===> select 3_                                      SCROLL ===> CSR |
-------------------------------------------------------------------------------

 ASID(X'0014') is the default address space
 PTR   Address  Address space                            Data type
 00001 00000000 HEADER                                   AREA
       Remarks: Comment 1
 00002 00FD7BC8 ASID(X'0014')                            AREA
       Remarks: Comment 2
 00003 00000210 ASID(X'0014')                            AREA
       Remarks:
 00004 00FD7BA0 ASID(X'0001')                            STRUCTURE(CVT)
       Remarks: Communications Vector Table
 *************************** END OF POINTER STACK ****************************
```

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 [col1 [col2][A | D][...]]

     [X | NX]
```

Parameter

col1
Specifies 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
D 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.

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 on 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 towards the top of data.

Syntax

```
UP          [ amount ]
```

Parameter

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 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 usually 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     }
```

Parameter

data-descr

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 on 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, but the same results would occur if the command were 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 000006B0 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
 ****************************** TOP OF DATA ***********************************
      ASID(X'0003') 000006B0. STRUCTURE(Psa)+06B0 IN PSA
      ASID(X'0003') 000006B0. IEAVFX00+06B0 IN PSA
 ****************************** 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.

```
DSNAME('D46IPCS.DRVC400.SA00001') POINTERS  ----------------------------------
COMMAND ===> _                                            SCROLL ===> CSR
PTR   Address  Address space                          Data type
00001 00000000 ASID(X'0003')                          AREA
      Remarks:
00002 000006B0 ASID(X'0003')                          AREA
      Remarks:
00003 00FD7420 ASID(X'0001')                          STRUCTURE(Cvt)
      Remarks: Communications Vector Table
-------------------------------------------------------------------------------
|00004 00000000 ASID(X'0003')                         STRUCTURE(Psa)        |
|      Remarks:                                                             |
-------------------------------------------------------------------------------
*************************** END OF 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 Represents 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, prior to 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.

```
IPCS OUTPUT STREAM ---------------------------------------- LINE 0 COLS 1 78
COMMAND ===>                                                SCROLL ===> CSR
 ******************************** TOP OF DATA ********************************
     * * * *   K E Y F I E L D S   * * * *
 JOBNAME D58PXE1
    SELECTED BY: CURRENT


------------------
|d4ASCB: 00920200|
------------------


     FWDP..... 00914E00  ASID..... 00B3      CSCB..... 00920D48
     TSB...... 00922178  AFFN..... FFFF      ASXB..... 005FDC20  DSP1..... 00
     FLG2..... 00        SRBS..... 0000      LOCK..... 00000000
     ASSB..... 01929980

  TCB: 005FDE40
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 005FDDF8  NDSP..... 00000000
     JSCB..... 005FDAA4  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00        STCB..... 7FFFECB0

  PRB: 005FDAD8
     WLIC..... 00020001  FLCDE.... 00BF9458  OPSW..... 070C1000  810203A0
     LINK..... 015FDE40

  CDE: 00BF9458
```

This screen shows the result of using the D line command.

```
IPCS OUTPUT STREAM ---------------------------------------- LINE 0 COLS 1 78
COMMAND ===> _                                              SCROLL ===> CSR
 ******************************** TOP OF DATA ********************************
     * * * *   K E Y F I E L D S   * * * *
 JOBNAME D58PXE1
     SELECTED BY: CURRENT

     ASSB..... 01929980

  TCB: 005FDE40
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 005FDDF8  NDSP..... 00000000
     JSCB..... 005FDAA4  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00        STCB..... 7FFFECB0

  PRB: 005FDAD8
     WLIC..... 00020001  FLCDE.... 00BF9458  OPSW..... 070C1000  810203A0
     LINK..... 015FDE40

  CDE: 00BF9458
     NAME..... IEAVAR00  ENTPT.... 81B3E120

  TCB: 005FD080
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 005FDDD0  NDSP..... 00000000
     JSCB..... 005FDAA4  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00        STCB..... 7FFFE890
```

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

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.

```
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

-------------------------------------------------------------------------
|e0002 00FD7BC8 ASID(X'0014')                           AREA            |
|      Remarks: Comment 2                                               |
-------------------------------------------------------------------------

00003 00000210 ASID(X'0014')                            AREA
      Remarks:
00004 00FD7BA0 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

Figure 6-4. Using E on the Pointer Panel

```
EDIT DSNAME('D83DUMP.DUMPC.PB00465') POINTER 00002
COMMAND ==]

 Enter/verify attributes of the pointer.

 Use ENTER to view updated definition,
     END to save pointer and return,
     CANCEL to return without saving changes.

 Address       ===] 00FD7BC8
 Address space ===] ASID(X'0014')
 Data type     ===] AREA
 Remarks       ===] The remark text of this pointer is being changed to
show how the comments can be truncated when the pointer stack is
displayed._
```

Figure 6-5. Pointer Editing Panel

```
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
-------------------------------------------------------------------------
|e0002 00FD7BC8 ASID(X'0014')                           AREA           |
|      Remarks: Comment 2                                              |
-------------------------------------------------------------------------

00002 00FD7BC8 ASID(X'0014')                            AREA
      Remarks: The remark text of this pointer is being changed to show
00003 00000210 ASID(X'0014')                            AREA
      Remarks:
00004 00FD7BA0 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

Figure 6-6. Result of Using Edit

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.

```
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
00002 00FD7BC8 ASID(X'0014')                            AREA
      Remarks: Comment 2
00003 00000210 ASID(X'0014')                            AREA
      Remarks:
-------------------------------------------------------------------------
|f0004 00FD7BA0 ASID(X'0001')                           STRUCTURE(Cvt) |
|      Remarks: Communications Vector Table                            |
-------------------------------------------------------------------------
*************************** END OF POINTER STACK ****************************
```

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

-------------------------------------------------------------------
|i0002 00FD7BC8 ASID(X'0014')                            AREA      |
|      Remarks: Comment 2                                          |
-------------------------------------------------------------------

00003 00000210 ASID(X'0014')                            AREA
      Remarks:
00004 00FD7BA0 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

- Result

  The following screen shows the results of using an I line command.

```
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
00002 00FD7BC8 ASID(X'0014')                            AREA
      Remarks: Comment 2

-------------------------------------------------------------------
|00003 00000000 ASID(X'0014')                           AREA      |
|     Remarks:                                                    |
-------------------------------------------------------------------

00004 00000210 ASID(X'0014')                            AREA
      Remarks:
00005 00FD7BA0 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

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  }
{ Rn }
```

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.

```
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

----------------------------------------------------------------------
|r2002 00FD7BC8 ASID(X'0014')                           AREA         |
|  _   Remarks: Comment 2                                            |
----------------------------------------------------------------------

00003 00000210 ASID(X'0014')                            AREA
      Remarks:
00004 00FD7BA0 ASID(X'0001')                            STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

– Result

R Line Command

The following screen shows 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
00002 00FD7BC8 ASID(X'0014')                          AREA
      Remarks: Comment 2
-------------------------------------------------------------------------------
|00003 00FD7BC8 ASID(X'0014')                         AREA                |
|      Remarks: Comment 2                                                  |
|00004 00FD7BC8 ASID(X'0014')                         AREA                |
|      Remarks: Comment 2                                                  |
-------------------------------------------------------------------------------
00005 00000210 ASID(X'0014')                          AREA
      Remarks:
00006 00FD7BA0 ASID(X'0001')                          STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

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.

```
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
00002 00FD7BC8 ASID(X'0014')                          AREA
      Remarks: Comment 2

-----------------------------------------------------------------------
|s0003 00000210 ASID(X'0014')                         AREA          |
|      Remarks:                                                      |
-----------------------------------------------------------------------

00004 00FD7BA0 ASID(X'0001')                          STRUCTURE(Cvt)
      Remarks: Communications Vector Table
*************************** END OF POINTER STACK ****************************
```

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-hand 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, prior to 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.

```
IPCS OUTPUT STREAM ---------------------------------------- LINE 0 COLS 1 78
COMMAND ===>                                                SCROLL ===> CSR
 ******************************** TOP OF DATA ********************************
     * * * *   K E Y F I E L D S   * * * *
 JOBNAME D58PXE1
     SELECTED BY: CURRENT

  ASCB: 00920200
     FWDP..... 00914E00  ASID..... 00B3      CSCB..... 00920D48
     TSB...... 00922178  AFFN..... FFFF      ASXB..... 005FDC20  DSP1..... 00
     FLG2..... 00        SRBS..... 0000      LOCK..... 00000000
     ASSB..... 01929980

  TCB: 005FDE40
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 005FDDF8  NDSP..... 00000000

-------------------------------------------------------------------------
|  f2_- - - - - - - - - - - - - - - - - - - - - - - - 5 LINE(S) NOT DISPLAYED|
-------------------------------------------------------------------------

     LINK..... 015FDE40

  CDE: 00BF9458
     NAME..... IEAVAR00  ENTPT.... 81B3E120

  TCB: 005FD080
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 005FDDD0  NDSP..... 00000000
     JSCB..... 005FDAA4  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00        STCB..... 7FFFE890
```

– Result

The following screen shows the resulting display output after pressing the ENTER key.

```
IPCS OUTPUT STREAM ---------------------------------------- LINE 0 COLS 1 78
COMMAND ===> _                                              SCROLL ===> CSR
 ******************************** TOP OF DATA ********************************
     * * * *   K E Y F I E L D S   * * * *
 JOBNAME D58PXE1
     SELECTED BY: CURRENT

  ASCB: 00920200
     FWDP..... 00914E00  ASID..... 00B3      CSCB..... 00920D48
     TSB...... 00922178  AFFN..... FFFF      ASXB..... 005FDC20  DSP1..... 00
     FLG2..... 00        SRBS..... 0000      LOCK..... 00000000
     ASSB..... 01929980

  TCB: 005FDE40
     CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 005FDDF8  NDSP..... 00000000

-------------------------------------------------------------------------------
|       JSCB..... 005FDAA4  BITS..... 00000000  DAR...... 00                  |
|       RTWA..... 00000000  FBYT1.... 00        STCB..... 7FFFECB0            |
|  - - -   - - - - - - - - - - - - - - - - - - - - - - 3 LINE(S) NOT DISPLAYED|
-------------------------------------------------------------------------------

     LINK..... 015FDE40

  CDE: 00BF9458
```

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

```
{ X      }
{ Xn     }
{ XX-XX  }
```

Parameters

n Represents a decimal number in the range of 1 through 9999.

- Represents an inclusive number of lines.

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

```
 IPCS OUTPUT STREAM ---------------------------------------- LINE 0 COLS 1 78
 COMMAND ===>                                                SCROLL ===> CSR
  ******************************** TOP OF DATA ********************************
      * * * *   K E Y F I E L D S   * * * *
  JOBNAME D58PXE1
      SELECTED BY: CURRENT

   ASCB: 00920200
      FWDP..... 00914E00  ASID..... 00B3      CSCB..... 00920D48
      TSB...... 00922178  AFFN..... FFFF      ASXB..... 005FDC20  DSP1..... 00
      FLG2..... 00        SRBS..... 0000      LOCK..... 00000000
      ASSB..... 01929980

   TCB: 005FDE40
      CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
      TSFLG.... 00        STAB..... 005FDDF8  NDSP..... 00000000

-------------------------------------------------------------
|xx   JSCB..... 005FDAA4  BITS..... 00000000  DAR...... 00|
-------------------------------------------------------------

      RTWA..... 00000000  FBYT1.... 00        STCB..... 7FFFECB0

   PRB: 005FDAD8
-------------------------------------------------------------------------------
|xx_ WLIC..... 00020001  FLCDE.... 00BF9458  OPSW..... 070C1000  810203A0  |
-------------------------------------------------------------------------------
      LINK..... 015FDE40

   CDE: 00BF9458
```

This screen shows the result of using the X line command.

```
 IPCS OUTPUT STREAM ---------------------------------------- LINE 0 COLS 1 78
 COMMAND ===> _                                              SCROLL ===> CSR
  ******************************** TOP OF DATA ********************************
      * * * *   K E Y F I E L D S   * * * *
  JOBNAME D58PXE1
      SELECTED BY: CURRENT

   ASCB: 00920200
      FWDP..... 00914E00  ASID..... 00B3      CSCB..... 00920D48
      TSB...... 00922178  AFFN..... FFFF      ASXB..... 005FDC20  DSP1..... 00
      FLG2..... 00        SRBS..... 0000      LOCK..... 00000000
      ASSB..... 01929980

   TCB: 005FDE40
      CMP...... 00000000  PKF...... 00        LMP...... FF        DSP...... FF
      TSFLG.... 00        STAB..... 005FDDF8  NDSP..... 00000000
-------------------------------------------------------------------------------
 |- - -   - - - - - - - - - - - - - - - - - - - - - 5 LINE(S) NOT DISPLAYED|
-------------------------------------------------------------------------------
      LINK..... 015FDE40

   CDE: 00BF9458
```

Chapter 7. IPCS CLISTs and REXX EXECs

This section 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.SBLSCLI0 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 section 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 on 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

When You Want to	Use the
Obtain a stand-alone dump screening report	"BLSCSCAN CLIST — Obtain a Stand-Alone Dump Screening Report" on page 7-12.
Obtain an SVC dump screening report	"BLSCBSVB CLIST — Obtain an SVC Dump Screening Report" on page 7-5.
Obtain a SYSMDUMP dump screening report	"BLSCBSYB CLIST — Obtain a SYSMDUMP Dump Screening Report" on page 7-6.
Format save area chain	"BLSCEPTR CLIST — Run a Save Area Chain" on page 7-9.
List entry points with the same name	"BLSXWHER REXX EXEC — Find All Modules with the Same Entry Point Name" on page 7-13.

Customize an IPCS Session

When You Want to	Use the
Create or allocate a user dump directory or a sysplex dump directory	"BLSCDDIR CLIST — Create a Dump Directory" on page 7-8.
Define IPCS dialog libraries to ISPF	BLSCLIBD CLIST in *z/OS MVS IPCS Customization*.
Define the SYS1.SBLSCLI0 library to IPCS dialog	BLSCALTL CLIST in *z/OS MVS IPCS Customization*.

© Copyright IBM Corp. 1988, 2002

Print Dump Analysis Reports

When You Want to	Use the
Print a stand-alone dump screening report	“BLSCBSAA CLIST — Print a Stand-Alone Dump Screening Report”.
Print an SVC dump screening report	“BLSCBSVA CLIST — Print an SVC Dump Screening Report” on page 7-4.
Print a SYSMDUMP dump screening report	“BLSCBSYA CLIST — Print a SYSMDUMP Dump Screening Report” on page 7-6.
Print a stand-alone dump detailed report	“BLSCBSAP CLIST — Print a Stand-Alone Dump Detailed Report” on page 7-3.
Print an SVC dump detailed report	“BLSCBSVP CLIST — Print an SVC Dump Detailed Report” on page 7-5.
Print a SYSMDUMP dump detailed report	“BLSCBSYP CLIST — Print a SYSMDUMP Dump Detailed Report” on page 7-7.

Print Storage Data

When You Want to	Use the
Print common storage areas	“BLSCPCSA CLIST — Print Common Storage Areas” on page 7-9.
Print nucleus storage areas	“BLSCPNUC CLIST — Print Nucleus Storage Areas” on page 7-10.
Print one or more storage areas	“BLSCPRNT CLIST — Print a Dump” on page 7-10.
Print private storage areas	“BLSCPRIV CLIST — Print Private Storage Areas” on page 7-10.
Print global system queue areas	“BLSCPSQA CLIST — Print Global System Queue Areas” on page 7-12.

Sample CLISTs and REXX EXECs

For This Subcommand	See the Example
COMPARE	5-46.
EVALDEF	5-93.
EVALDUMP	5-96.
EVALMAP	5-100. “BLSXWHER REXX EXEC — Find All Modules with the Same Entry Point Name” on page 7-13.
EVALSYM	5-105.
RUNCHAIN	5-213.

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 IEFRDER. 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 IEFRDER. 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

By default the BLSJIPCS cataloged procedure invokes the BLSCBSVA CLIST.

```
//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.

See “BLSCBSVB CLIST — Obtain an SVC Dump Screening Report” and *z/OS MVS IPCS User’s Guide* for other ways to obtain information from an SVC dump.

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.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,CLIST=BLSCBSVP,DUMP='svcdump.dsname'
```

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 "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 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.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,CLIST=BLSCBSYA,DUMP='sysmdump.dsname'
```

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.

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.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,CLIST=BLSCBSYP,DUMP='sysmdump.dsname'
```

CLIST Listing

See the BLSCBSYP member of SYS1.SBLSCLI0.

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 on 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) ]

           [ 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.

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.

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').

The subcommands in this CLIST create the following symbols in the IPCS symbol table:

EP*nnn* 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.

EPID*nnn* The entry point identifier string for the entry point represented by EP*nnn*.

SA*nnn* The save area holding the entry point address represented by EP*nnn*.

Syntax

```
%BLSCEPTR [TCB(address)]
```

Parameter

TCB(address)
The address of the TCB that BLSCEPTR uses to start chaining the save areas. If you do not specify a TCB address, BLSCEPTR uses the address found in PSATOLD (PSA+X'21C').

CLIST Listing

See the BLSCEPTR 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* for more information about writing a CLIST that uses BLSCPCSA to create a custom dump report.

Syntax

```
%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* for more information about writing a CLIST that uses BLSCPNUC to create a custom dump report.

Syntax

```
%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).

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.

```
ALLOCATE DDNAME(IPCSTOC) SYSOUT(x)
ALLOCATE DDNAME(IPCSPRNT) SYSOUT(x)
SETDEF DSNAME('dump.dsname')
%BLSCPRNT NUCLEUS,SQA,CSA,CURRENT,JOBNAME(jobname)
CLOSE PRINT
```

Example of IPCS Batch Invocation

```
//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=*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN    DD   *
%BLSCDDIR DSNAME(userid.ddir) VOLUME(volid)...    (optional)
IPCS
SETDEF DSN('dump.dsname') 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* for more information about writing a CLIST that uses BLSCPSQA to create a custom dump report.

Syntax

```
%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.)
    ===] SCR006

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.

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

===] %BLSCSCAN
```

CLIST Listing

See the BLSCSCAN member of SYS1.SBLSCLI0.

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 on specifying *asid*.

BLSXWHER REXX EXEC

IPCS Dialog Invocation

BLSXWHER finds the storage map entries for load module ILRPGEXP in the default address space, if any exist.

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

===] %BLSXWHER ILRPGEXP
```

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*.

```
//IPCSJOB  JOB 'acctinfo','PGMR output',MSGLEVEL=(1,1),
//          MSGCLASS=A,CLASS=J,NOTIFY=PGMR
//* -------------------------------------------------------
//*
//*   Input: dump in data set 'PGMR.DUMP1.DUMP'
//*   Output:
//*     - IPCS dump directory data set for the input dump
//*        (IPCSDDIR DD)
//*     - Formatted output (SYSTSPRT DD)
//*     - TSO/E messages (SYSTSPRT DD)
//*     All of the output will have message identifiers
//*     printed (due to PROFILE MSGID command in SYSTSIN)
//* -------------------------------------------------------
//IPCS     EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=1500K
//IPCSDDIR DD DSN=PGMR.DUMP.DIR,DISP=(OLD,KEEP)
//*
//SYSPROC  DD DSN=SYS1.SBLSCLI0,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
PROFILE MSGID
IPCS NOPARM
SETDEF DSN('PGMR.DUMP1.DUMP') LIST NOCONFIRM
%BLSCSCAN
END
/*
```

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

© Copyright IBM Corp. 1988, 2002

Batch Mode

- 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.')
//*
//*                        INPUT DATA SETS
//*
//IPCSDUMP DD DSN=&DUMP,DISP=SHR       DUMP OR TRACE DATA SET
//SYSPROC  DD DSN=SYS1.SBLSCLI0,DISP=SHR CLIST PROCEDURES
//SYSTSIN  DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80) TSO/E COMMANDS
//*
//*                        FORMATTED OUTPUT
//*
//SYSTSPRT DD SYSOUT=A                 BATCH TSO/E SESSION LOG
//IPCSTOC  DD SYSOUT=A                 PRINT FILE TABLE OF CONTENTS
//IPCSPRNT 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 7-2
- “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

Appendix A. Problem Management Subcommands

This section describes the IPCS subcommands you can use to manage problems and their associated data sets. For more information on the use of these subcommands, see the appendix on managing problems and data sets in *z/OS MVS IPCS User's Guide*.

However, IBM makes the following recommendations:

- If you are not currently using a problem management tool to report and track problems detected in your installation's data processing environment, use the IBM Information/Management facility instead of the IPCS problem management subcommands.
- If you currently use the IPCS problem management subcommands to report and track problems, consider migrating to the IBM Information/Management facility.

IBM does not plan to enhance or update the IPCS problem management subcommands, and you should be aware that these subcommands do not support all of the current codes for authorized program analysis reports (APARs) and program temporary fixes (PTFs).

For more information about the IBM Information/Management facility, see *Information/Management Library: Problem, Change, and Configuration Management User's Guide*.

ADDDSN Subcommand — Add a Data Set Name to a Problem

Use the ADDDSN subcommand to associate a data set or a partitioned data set member with a problem.

You can specify that the subcommand require your confirmation before proceeding if a data set attribute conflict occurs. These conflicts occur if the data set specified on this subcommand is already associated with a problem and the attributes specified when the data set was associated with the other problem conflict with the attributes specified on this subcommand. (The attributes of a data set are recorded only once, regardless of the number of problems with which that data set is associated.) The subcommand displays the attributes previously recorded (the current attributes) and requests your instructions.

If the specified data set is already associated with the specified problem, the subcommand ensures the accuracy of the association and then ends.

Any user can run ADDDSN at any time during an IPCS session. This subcommand does not change the current source data set unless you specify the DEFAULT parameter.

Related Subcommands

DELDSN
LISTDSN
MODDSN

Syntax

© Copyright IBM Corp. 1988, 2002

ADDDSN Subcommand

```
{ ADDDSN }    [ CONFIRM | NOCONFIRM ]
{ AD     }

            [ DATASET(dsn) ]
            [ DSNAME(dsn)  ]

            [ DEFAULT ]

            [ DESCRIPTION('text') ]

            [ MANAGED | UNMANAGED ]

            [ PROBLEM(prob-num) ]

            [ TEST | NOTEST ]

            [ TYPE( {DUMP | PRINT | UDEF} ) ]
```

Parameters

CONFIRM or NOCONFIRM

CONFIRM specifies that the subcommand is to request your confirmation before processing a data set attribute conflict.

1. The subcommand displays the current data set attributes and the problem or problems with which the data set is associated. It then requests your confirmation.
2. If you enter Y, the subcommand associates the specified data set with the specified problem and leaves the data set attributes as they are.
3. If you enter N, the subcommand ends without associating the data set with the problem, leaves the data set attributes as they are in the data set directory, and ignores the DEFAULT parameter, if specified.

NOCONFIRM specifies that the subcommand is not to request your confirmation if a data set attribute conflict occurs. The subcommand associates the specified data set with the specified problem and leaves the data set attributes as they are in the data set directory.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default 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.

DATASET or DSNAME(dsn)

The name of the data set to be associated with the problem. The subcommand ignores a password if specified with the data set name, unless you specify DEFAULT.

If you omit both DATASET and DSNAME, the subcommand uses the current data set name.

DEFAULT

Specifies that the data set named in this 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, if you enter N to resolve a data set attribute conflict, or if the subcommand fails, the current source is not changed in the defaults.

DESCRIPTION(text)
A description of the data set. The description is a maximum of 60 characters of text.

If you omit this parameter, and the data set is not already associated with a problem, there is no description recorded for this data set.

If you omit this parameter and the data set is already associated with a problem, the subcommand uses the description previously provided for this data set.

MANAGED or UNMANAGED
MANAGED directs IPCS to manage the data set. This allows IPCS to scratch the data set when it is no longer associated with any problem.

UNMANAGED directs IPCS not to manage the data set. IPCS does not scratch the data set when it is no longer associated with a problem.

If you omit both MANAGED and UNMANAGED and the data set is not already associated with a problem, the default is MANAGED.

If you omit both MANAGED and UNMANAGED and the data set is already associated with a problem, the subcommand uses the current management attribute for this data set.

IPCS does not manage partitioned data set members. If you specify MANAGED for a partitioned data set member, it is forced to UNMANAGED. If you associate a partitioned data set member with a problem and you omit this parameter, the default is UNMANAGED.

PROBLEM(prob-num)
Specifies the problem number with which the data set is to be associated. This problem number must already exist in the problem directory. The problem number is five decimal digits; leading zeros are optional.

If you omit this parameter, the subcommand associates the data set with the current problem.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

TYPE({DUMP|PRINT|UDEF})
The data set category:

TYPE(DUMP)
The data set is an unformatted system dump.

TYPE(PRINT)
The data set is printable.

TYPE(UDEF)
The data set type is user-defined.

If you omit the TYPE parameter and the data set is not already associated with a problem, the default is TYPE(DUMP).

If you omit this parameter and the data set is associated with a problem, the subcommand uses the current type attribute for this data set.

Return Codes

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

ADDPROB Subcommand — Add a Problem to IPCS

Use the ADDPROB subcommand to add a new problem to IPCS. IPCS creates a unique identifier for the new problem, or assigns an identifier specified by you, and issues the message:

```
BLS05100I PROBLEM prob-id HAS BEEN ADDED TO THE PROBLEM DIRECTORY
```

If you enter the subcommand with no parameters, it creates a problem and sets default problem attributes as described for each parameter. ADDPROB creates a new problem status record and includes this record in the problem directory. Any user can run ADDPROB at any time during an IPCS session. This subcommand does not change the current problem unless the DEFAULT parameter is specified.

Related Subcommands

ADDDSN
DELPROB
LISTPROB
MODPROB

Syntax

```
{ ADDPROB } [ ABSTRACT('text') ]
{ AP      }
           [ APARID(apar-id) ]

           [{ CLIST  } (PROBLEM(var-name))  ]
           [{ DIALOG }                      ]
           [{ REXX   }                      ]

           [ COMPID(component-id) ]
           [ DATE(date) ]
           [ DEFAULT ]

           [ DESCRIPTION('text') ]
           [ DSDESCRIPTION(dsn)  ]

           [ FIXID('fix-id') ]

           [ FIXSTATUS({NONE  }) ]
           [           {RQST  }  ]
           [           {RCVD  }  ]
           [           {INST  }  ]
           [           {ACPT  }  ]
           [           {REJD  }  ]

           [ GROUP(group-id) ]
```

```
[ IBMSTATUS({ NONE     )   ]
[           { REPORTED }   ]
[           { APARED   }   ]
[           { PSRR     }   ]
[           { ABY      }   ]
[           { ACK      }   ]
[           { CAN      }   ]
[           { DOC      }   ]
[           { DUA      }   ]
[           { DUB      }   ]
[           { DUU      }   ]
[           { MCH      }   ]
[           { PRE      }   ]
[           { PRS      }   ]
[           { RET      }   ]
[           { ROP      }   ]
[           { SUG      }   ]
[           { UR1      }   ]
[           { UR2      }   ]
[           { UR3      }   ]
[           { UR4      }   ]
[           { UR5      }   ]
[           { USE      }   ]



[ OWNER(tsologonid) ]
[ PROBLEM(prob-num) ]

[ PSTATUS({ INITIAL   })  ]
[         { ACTIVE    }   ]
[         { INACTIVE  }   ]
[         { CLAPAR    }   ]
[         { CLDUP     }   ]
[         { CLFIX     }   ]
[         { CLOTHRE   }   ]
[         { CLPTF     }   ]
[         { CLNTF     }   ]



[ PTFID(ptf-id) ]

[ PTFSTATUS( {NONE  }) ]
[            {RQST  }  ]
[            {RCVD  }  ]
[            {INST  }  ]
[            {ACPT  }  ]
[            {REJD  }  ]



[ SEVERITY({ 0 })  ]
[          { 1 }   ]
[          { 2 }   ]
[          { 3 }   ]
[          { 4 }   ]

[ SYSTEM(system-id)]

[ TEST | NOTEST ]

[ TIME(tod) ]
[ USER(user-data) ]
```

Parameters

ABSTRACT(‘text’)
The problem abstract. The abstract is a maximum of 128 characters of text providing a brief description of the problem.

If you omit this parameter, there is no abstract recorded for this problem.

APARID(apar-id)
The identifier assigned to the APAR for the problem. The APAR identifier must be 7 alphameric characters and the first character must be alphabetic.

If you omit this parameter, the subcommand sets this field to blanks.

CLIST or DIALOG or REXX (PROBLEM(var-name))
The problem number added is formatted and stored in the CLIST, REXX, or ISPF function pool dialog variable var-name.

COMPID(component-id)
The suspected failing component. The component identifier is a maximum of 10 characters. You can specify any character except blanks, commas, semicolons, and tabs. Parentheses are allowed but must be balanced.

If you omit this parameter, the subcommand sets this field to blanks.

DATE(date)
The date the problem occurred in mm/dd/yy format. The slashes must be specified; leading zeros are optional for month and day.

If you omit this parameter, the subcommand supplies the current date.

DEFAULT
Specifies that the problem being added is to become the current problem.

IPCS changes the current problem in both the local and global defaults. If you omit this parameter or if the subcommand fails, the current problem is not changed in the defaults.

DESCRIPTION(‘text’)
The description of the problem. The problem description may contain a maximum of 9,360,000 characters.

If you omit this parameter and if you omit DSDESCRIPTION, there is no description recorded for this problem.

DSDESCRIPTION(dsn)
The name of a cataloged sequential data set containing the description of the problem. The specified data set must be fixed format containing blocked or unblocked records with a logical record length of 80. The subcommand copies the first 72 characters of each logical record into the problem description. The problem description may contain a maximum of 9,360,000 characters.

If you omit this parameter and if you omit DESCRIPTION, there is no description recorded for this problem.

FIXID(‘fix-id’)
The identifier assigned to the fix for the problem. The fix identifier is a maximum of 60 characters.

If you omit this parameter, the subcommand sets this field to blanks.

FIXSTATUS(value)
The status of a local fix, circumvention, or bypass.

If you specify a value for this parameter, the subcommand records the date and time of that specification.

If you omit this parameter, the default is NONE and the subcommand sets this attribute field and its date and time fields to blanks.

The FIXSTATUS values and their meanings are:

NONE
You do not know of a fix that solves the problem. If you specify NONE, the subcommand sets this attribute field to blanks but records the current date and time.

RQST
The fix has been requested but has not yet arrived.

RCVD
The fix has been received but has not been installed.

INST
The fix has been installed and is being tested to determine whether or not it solves the problem.

ACPT
The fix solves the problem and is accepted.

REJD
The fix does not solve the problem and is rejected.

GROUP(group-id)
Specifies the department name or number responsible for the problem. The group identifier is a maximum of 8 alphameric characters.

If you omit this parameter, the subcommand uses the default group identifier from the IPCS session parameters member. If there is no default group identifier in the IPCS session parameters member, the subcommand sets this field to blanks.

IBMSTATUS(value)
Specifies the status of a problem that has been reported to IBM.

If you specify a value for this parameter, the subcommand records the date and time of that specification.

If you omit this parameter, the default is NONE and the subcommand sets this attribute field and its date and time fields to blanks.

The IBMSTATUS values and their meanings are:

NONE
The problem has not been reported to IBM. If you specify NONE, the subcommand sets this attribute field to blanks but records the current date and time.

REPORTED
The problem has been reported to the IBM Field Engineering (FE) division representative; help in solving the problem has been requested.

APARED
An APAR has been submitted for the problem.

PSRR
The problem has been submitted as a Programming Service Requirement Request.

ABY
APAR placed in abeyance to test a fix.

ACK
APAR acknowledged.

CAN
APAR canceled.

DOC
APAR closed, documentation error.

DUA
APAR closed, duplicate of a resolved, unacceptable APAR, or a duplicate of an APAR which was closed more than 10 days ago.

DUB
APAR closed, duplicate of resolved acceptable APAR received within ten days of the original APAR closing.

DUU
APAR closed, duplicate of an unresolved APAR.

MCH
APAR closed, machine error.

PER
APAR closed, program error.

PRS
APAR closed, permanent restriction.

RET
APAR closed, used for APARs which cannot be resolved without additional input from the field.

ROP
APAR previously in abeyance is reopened.

SUG
APAR closed, suggestion.

UR1
APAR closed, unable to reproduce (or known to be corrected) on the next release available from IBM. Written against a release that was supported at the time the APAR was received.

UR2
Same as UR1 except written against a release that was not supported at the time the APAR was received.

UR3
APAR closed, unable to reproduce (or known to be corrected) on a currently supported release. Written against a release supported at the time the APAR was received.

UR4
Same as UR3 except written against a release that was not supported at the time the APAR was received.

UR5
APAR closed, unable to reproduce on the same level system as reported. Not used if UR1 - UR4 are more appropriate.

USE
APAR closed, user error.

OWNER(tsologonid)
Specifies the TSO/E userid of the person responsible for resolving the problem.

If you omit this parameter, the subcommand sets this field to your TSO/E userid.

Restriction:

When using IPCS in the background, you must specify the OWNER parameter. If you do not, the owner field for the problem will contain unpredictable information.

PROBLEM(prob-num)
The problem number of the problem to be added to the IPCS data base. This problem number must not exist in the problem directory. The problem number is five decimal digits; leading zeros are optional.

If you omit this parameter, IPCS uses the value found in the seed record of the problem directory, in conjunction with the data base contents, to determine the problem number.

PSTATUS(value)
Specifies the status of the problem.

IPCS records the date and time of the last change to this field.

If you omit this parameter, the default is INITIAL.

The problem status values and their meanings are:

INITIAL
The problem is new and its status is not known.

ACTIVE
The problem appears to be valid and its resolution is being actively pursued.

INACTIVE
The problem appears to be valid but its resolution is not being actively pursued. In most cases, this is due to a lack of problem data or resources.

CLAPAR
The problem has been closed because an APAR has been submitted for it.

CLDUP
The problem has been closed because it is a duplicate of another. The problem identifier of the original problem should be entered into the problem description.

CLFIX
The problem has been closed because a local fix, circumvention, or bypass has been applied and accepted.

CLOTHER
The problem has been closed for a reason other than those indicated by CLAPAR, CLDUP, CLFIX, CLPTF, and CLNTF. You should describe the reason for closure in the problem description.

CLPTF
The problem has been closed due to a PTF being applied and accepted.

CLNTF
The problem has been closed since, on further investigation, no trouble was found.

PTFID(ptf-id)
Specifies the identifier assigned to the PTF for the problem. The PTF identifier must be 7 alphameric characters and the first character must be alphabetic. If you omit this parameter, the subcommand sets this field to blanks.

PTFSTATUS(value)
Specifies the status of the PTF that you identify as the one which solves the problem. If you specify a value for this parameter, the subcommand records the date and time of that specification. If you omit this parameter, the default is NONE and the subcommand sets this attribute field and its date and time fields to blanks.

The PTFSTATUS values and their meanings are:

NONE
You do not know of a PTF that solves the problem. If you specify NONE, the subcommand sets this attribute field to blanks but records the current date and time.

RQST
The PTF has been requested but has not yet arrived.

RCVD
The PTF has been received but has not been installed.

INST
The PTF has been installed and is being tested to determine whether or not it solves the problem.

ACPT
The PTF solves the problem and is accepted.

REJD
The PTF does not solve the problem and is rejected.

SEVERITY(n)
Specifies the problem's severity, and should be consistent with IBM APAR severity codes listed below. 0 specifies a problem of undetermined severity. If you specify 0, the subcommand sets this field to blanks. If you omit this parameter, the default is 0. The severity codes and their meanings are:

Code	Explanation
01	The user is unable to use the program, resulting in a critical impact on his operations.
02	The user is able to use the program but is severely restricted.
03	The user is able to use the program with limited functions which are not critical to the overall operations.
04	The user or the system programmer has found a way to circumvent the problem. However, the APAR will be evaluated and action taken as dictated by the problem.

SYSTEM(system-id)
Specifies the system on which the problem occurred. The system identifier is a maximum of 8 alphameric characters.

If you omit this parameter, the subcommand uses the default system identifier from the IPCS session parameters member. If there is no default system identifier in the session parameters member, the subcommand sets this field to blanks.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

TIME(tod)
Specifies the time of day the problem occurred in hh:mm:ss format. The colons and leading zeros must be specified. If you omit this parameter, the subcommand supplies the current time.

USER(user-data)
Specifies a user-data field. This field may contain a maximum of 8 alphameric characters. You (or your installation) may define the meaning of this field.

If you omit this parameter, the subcommand sets this field to blanks.

Return Codes

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

DELDSN Subcommand — Delete a Data Set from a Problem

Use the DELDSN subcommand to dissociate a data set or a partitioned data set member from a problem.

On the DELDSN subcommand, you can ask IPCS to prompt you for confirmation before continuing with the DELDSN function. In IPCS line mode, DELDSN will display the names and attributes of the data sets to be dissociated before requesting confirmation and continuing with the DELDSN function. In the IPCS dialog, however, you should use the LISTPROB subcommand to display the names and attributes of the data sets before entering the DELDSN subcommand.

You should not use DELDSN to delete a dump data set. Use the DROPDUMP subcommand instead.

If the data set is not managed by IPCS, it is dissociated from the specified problem but is not scratched.

IPCS scratches and uncatalogs a managed data set when the following three conditions are met:

- It is cataloged.
- It is no longer associated with any problem in the IPCS problem directory.
- Its name does not begin with ‘SYSn.’, where n is a decimal number from 0 through 9.

IPCS never scratches a partitioned data set member.

DELDSN Subcommand

DELDSN dissociates a data set from a problem even if it cannot scratch the data set (for example, if the data set doesn't exist or you can't supply the correct password).

A data set can be dissociated from a problem only by the problem owner or by the person specified in the IPCS session parameters member as having administrative authority. This subcommand does not change the current data set, even if it scratches the current data set.

During certain phases of processing, DELDSN ignores attention interrupts. If you generate an attention interrupt during such a phase, the interrupt is stacked and processed when the subcommand finishes the protected phase. If you interrupt and end a subcommand that modifies the problem directory or the data set directory, the modification to the data set may be incomplete.

Restriction:

When using IPCS in the background, you must use the TMP provided with the TSO/Extensions Program and must specify the JCL "USERID" parameter so that DELDSN can verify that the user is authorized to dissociate the specified data set.

Related Subcommands

ADDDSN
LISTDSN
MODDSN

Syntax

```
{ DELDSN }  [ CONFIRM  | NOCONFIRM ]
{ DD     }

           [ DATASET(dsn) ]
           [ DSNAME(dsn)  ]
           [ ALL          ]

           [ PROBLEM(prob-num) ]

           [ TEST | NOTEST ]
```

Parameters

CONFIRM

Specifies that the subcommand is to request your confirmation before proceeding with the dissociation of the data set or data sets.

The subcommand displays the current data set attributes and then requests your confirmation.

If you enter Y, the subcommand dissociates the specified data sets from the specified problem and may scratch them.

If you enter N, the subcommand ends without dissociating the data sets from the specified problem.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this parameter.

NOCONFIRM

Specifies that the subcommand is not to request your confirmation before

dissociating the specified data sets. The subcommand dissociates the specified data sets from the specified problem and, optionally, scratches the data sets.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this parameter.

DATASET or DSNAME(dsn)
Specifies the name of the data set to be dissociated from the specified problem. If the data set is password-protected and is to be scratched, its password can be entered as part of the data set name. If the password is not specified but is needed, you are prompted for it.

If you specify a problem number but neither a data set name nor ALL, the command dissociates the current data set from the specified problem.

If you do not specify a problem number, a data set name, or ALL, the subcommand dissociates the current data set from the current problem.

ALL
Specifies that IPCS is to dissociate all data sets associated with the specified problem. If a data set is to be scratched and is password protected, the data set name is displayed and you are prompted for the password.

If you specify a problem number but neither a data set name nor ALL, the subcommand dissociates the current data set from the specified problem.

If you do not specify a problem number, a data set name, or ALL, the subcommand dissociates the current data set from the current problem.

PROBLEM(prob-num)
Specifies the problem from which the subcommand is to dissociate the specified data set or data sets. This problem number must already exist in the problem directory. The problem number is five decimal digits; leading zeros are optional.

If you omit this parameter, the subcommand dissociates the specified data set or data sets from the current problem.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

Return Codes

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

DELPROB Subcommand — Delete a Problem from IPCS

Use the DELPROB subcommand to dissociate all data sets from the specified problem and delete the problem from IPCS.

You can request that the subcommand display the problem’s current status and the data sets associated with the problem, and then require your confirmation before continuing with the DELPROB function.

If a data set is associated with the specified problem but is not managed by IPCS, the subcommand dissociates it from the problem but does not scratch it.

If a data set is associated with other problems, the subcommand dissociates it from the specified problem but does not scratch it.

IPCS scratches and uncatalogs a managed data set when the following three conditions are met:

- It is cataloged.
- It is no longer associated with any problem in the IPCS problem directory.
- Its name does not begin with 'SYSn.', where n is a decimal number from 0 through 9.

IPCS never scratches a partitioned data set member.

DELPROB dissociates a data set from a problem even if it cannot scratch the data set (for example, if the data set doesn't exist or you can't supply the correct password).

If someone is specified in the IPCS session parameters member as having delete authority, only that person can use the DELPROB subcommand. If no one has delete authority, only the problem owner and the person with administrative authority (if any) can process DELPROB.

This subcommand does not change the default problem, even if it deletes the default problem, nor does it change the current data set, even if it scratches that data set.

During certain phases of processing, DELPROB ignores attention interrupts. If you generate an attention interrupt during such a phase, the interrupt is stacked and processed when the subcommand finishes the protected phase. If you interrupt and end a subcommand that modifies the problem directory or the data set directory, the modification to the data set may be incomplete.

Restriction:

When using IPCS in the background, you must use the TMP provided with TSO/E and must specify the JCL "USERID" parameter so that DELPROB can verify that the user is authorized to delete the specified problem.

Related Subcommands

ADDPROB
LISTPROB
MODPROB

Syntax

```
{ DELPROB }   [ CONFIRM | NOCONFIRM ]
{ DP      }

             [ PROBLEM(prob-num) ]

             [ TEST | NOTEST ]
```

Parameters

CONFIRM

Specifies that the subcommand is to request your confirmation before

proceeding with the problem deletion. The subcommand displays the current status of the specified problem and a list of its associated data sets. It then requests your confirmation.

If you enter Y, the subcommand deletes the specified problem and dissociates and may scratch its associated data sets except as noted above. If you enter N, the subcommand ends without deleting the problem or dissociating or scratching its data sets. If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this parameter.

NOCONFIRM
Specifies that the subcommand is not to request your confirmation before deleting the specified problem. The subcommand deletes the specified problem and dissociates and scratches its associated data sets except as noted above. If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this parameter.

PROBLEM(prob-num)
Specifies the problem number being deleted. This number must currently exist in the problem directory. The problem number is five decimal digits; leading zeros are optional. If you omit this parameter, the subcommand deletes the default problem.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

Return Codes

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

LISTDSN Subcommand — List Data Set Attributes

Use the LISTDSN subcommand to list the attributes (type, management, and description) of a data set or data sets. You can also request a list of the problems with which the data set is associated. You can run LISTDSN any time during an IPCS session.

Related Subcommands
ADDDSN
DELDSN
MODDSN

Syntax

```
{ LISTDSN }  [ DATASET(dsn) ]
{ LD      }  [ DSNAME(dsn)  ]
             [ ALL          ]

             [ PRINT | NOPRINT ]

             [ TERMINAL | NOTERMINAL ]

             [ PROBLEMS | NOPROBLEMS ]

             [ TEST | NOTEST ]
```

Parameters

DATASET or DSNAME(dsn)
Specifies the name of the data set whose attributes are to be displayed.

If you omit DATASET, DSNAME, and ALL, the subcommand displays the attributes of the current data set.

ALL
Specifies that the subcommand is to display the attributes of all data sets in the data set directory.

If you omit DATASET, DSNAME, and ALL, the subcommand displays the attributes of the current data set.

PRINT or NOPRINT
PRINT routes the subcommand listing to the print data set.

NOPRINT suppresses the listing.

TERMINAL or NOTERMINAL
TERMINAL routes the subcommand listing to the terminal.

NOTERMINAL suppresses the listing.

PROBLEMS
Specifies that the subcommand is to list the problems with which each data set is associated.

If you omit both PROBLEMS and NOPROBLEMS, the default is NOPROBLEMS.

NOPROBLEMS
Specifies that the subcommand is not to list the problems with which each data set is associated.

If you omit both PROBLEMS and NOPROBLEMS, the default is NOPROBLEMS.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

Return Codes

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

LISTPROB Subcommand — List Problems

Use the LISTPROB subcommand to display a list of problems defined to IPCS.

The problems included in the report are specified with the selection parameters: COMPID, FIXSTATUS, GROUP, IBMSTATUS, OWNER, PROBLEMS, ALL, PSTATUS, PTFSTATUS, SEVERITY, SYSTEM, and USER. The subcommand examines the problems specified with the PROBLEMS or the ALL parameter and includes them in the listing if their attributes match the attributes specified by the COMPID, FIXSTATUS, GROUP, IBMSTATUS, OWNER, PSTATUS, PTFSTATUS, SEVERITY, SYSTEM and USER parameters. You can specify several values for each of these attribute parameters. However, to include a problem in the listing, its attributes must match at least one value specified for each attribute parameter.

If you specify no selection parameters, the subcommand lists the default problem.

With the exception of OWNER, if you omit an attribute parameter, the subcommand does not consider that attribute when examining the problems. Thus, a problem is eligible for inclusion in the listing regardless of the omitted attribute’s value.

The required type of report is specified by the LIST, ABSTRACT, STATUS, DESCRIPTION, DATASETS, and DSNAMES parameters.

You can run LISTPROB any time during an IPCS session.

Restriction:

When using IPCS in the background, LISTPROB requires that you establish the owner of the problems of interest in one of two ways:

- You may use the TMP provided with TSO/E and specify the JCL “USERID” parameter so that a default owner can be established.
- You may use the “OWNER” parameter on the LISTPROB subcommand.

Related Subcommands

ADDPROB
DELPROB
MODPROB

Syntax

```
{ LISTPROB } [ LIST         ]
{ LP       } [ ABSTRACT     ]
             [ STATUS       ]
             [ DESCRIPTION  ]

             [ DATASETS   [( DESCRIPTION )]  ]
             [ DSNAMES    [( DESCRIPTION )]  ]
             [ NODATASETS                    ]
             [ NODSNAMES                     ]

             [ COMPID(component-id ... )]



             [ FIXSTATUS( {NONE   }...)  ]
             [            {RQST   }      ]
             [            {RCVD   }      ]
             [            {INST   }      ]
             [            {ACPT   }      ]
             [            {REJD   }      ]

             [ GROUP(group-id ... )]
```

```
[ IBMSTATUS(   NONE...)        ]
[          {  REPORTED }       ]
[          {  APARED   }       ]
[          {  PSRR     }       ]
[          {  CLOSED   }       ]
[          {  ABY      }       ]
[          {  ACK      }       ]
[          {  CAN      }       ]
[          {  DOC      }       ]
[          {  DUA      }       ]
[          {  DUB      }       ]
[          {  DUU      }       ]
[          {  MCH      }       ]
[          {  PER      }       ]
[          {  PRS      }       ]
[          {  RET      }       ]
[          {  ROP      }       ]
[          {  SUG      }       ]
[          {  UR1      }       ]
[          {  UR2      }       ]
[          {  UR3      }       ]
[          {  UR4      }       ]
[          {  UR5      }       ]
[          {  USE      }       ]



[ OWNER(  {tsologonid}...  )]
[         {ALL       }      ]

[ PRINT | NOPRINT ]

[ TERMINAL | NOTERMINAL  ]



[ PROBLEMS(prob-num  [: prob-num]  ... )  ]
[ ALL                                     ]

[ PSTATUS({INITIAL   }... )  ]
[         {ACTIVE    }       ]
[         {INACTIVE  }       ]
[         {OPEN      }       ]
[         {CLOSED    }       ]
[         {CLAPAR    }       ]
[         {CLDUP     }       ]
[         {CLFIX     }       ]
[         {CLOTHER   }       ]
[         {CLPTF     }       ]
[         {CLNTF     }       ]
```

```
[ PTFSTATUS({NONE  }... )   ]
[           {RQST  }        ]
[           {RCVD  }        ]
[           {INST  }        ]
[           {ACPT  }        ]
[           {REJD  }        ]

[ SEVERITY( {0  }... )  ]
[          {1  }        ]
[          {2  }        ]
[          {3  }        ]
[          {4  }        ]

[ SYSTEM(system-id ... )]

[ TEST | NOTEST ]

[ USER(user-data ... ) ]
```

Parameters

LIST

Requests a display of the attributes listed below.

The information for each problem is displayed on one line, in column format.

If you omit LIST, ABSTRACT, DESCRIPTION, and STATUS, the default is STATUS.

The LIST parameter requests the following attributes:
- Problem identifier
- Date the problem occurred
- Problem severity
- Suspected failing component identifier
- Problem owner
- Problem status
- IBM status
- PTF status
- FIX status
- Date of the most recent change to any of the four status fields. (This item is a single date, the most recent date that any status field was changed.)

ABSTRACT

Requests the same information as LIST, plus the problem abstract. The subcommand lists the abstract on one or two lines, depending on its length.

If you omit LIST, ABSTRACT, DESCRIPTION, and STATUS, the default is STATUS.

STATUS

Requests the same information as LIST, plus the attributes listed below.

The information for each problem is displayed on several lines.

If you omit LIST, ABSTRACT, DESCRIPTION, and STATUS, the default is STATUS.

The STATUS parameter requests the following attributes in addition to the LIST attributes:
- Problem abstract
- Date the problem was reported
- Time the problem was reported

- Group identifier
- System identifier
- Time the problem occurred
- APAR identifier
- PTF identifier
- Fix identifier
- Date and time of the most recent change to each of the four status fields (This item contains multiple dates and times, one date and time for each status field.)
- User data

DESCRIPTION
Requests the same information and format as STATUS, plus the problem description.

DATASETS or DSNAMES [(DESCRIPTION)]
Requests the information below about the data sets associated with each problem in the report. If you omit DATASETS, DSNAMES, NODATASETS, and NODSNAMES, the default is NODSNAMES.

The DATASETS or DSNAMES parameters request the following information about the data sets:
- Data set name
- Type
- Management

DESCRIPTION requests, in addition to the above information, a one line description of each data set.

NODATASETS or NODSNAMES
Suppresses data set information. No information about the data sets associated with a problem appears in the report. If you omit DATASETS, DSNAMES, NODATASETS, and NODSNAMES, the default is NODSNAMES.

COMPID(component-id ...)
Specifies that problems whose component identifier matches one of the values specified are eligible for inclusion in the report. The component identifier is a maximum of 10 characters. For each identifier, you can specify any characters except blanks, commas, semicolons, and tabs. Parentheses are allowed but must be balanced.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their component identifier.

FIXSTATUS(value ...)
Specifies that problems whose fix status attribute matches one of the specified values are eligible for inclusion in the report.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their FIXSTATUS attribute.

The FIXSTATUS values and their meanings are:

NONE
The problem's fix status has not been assigned.

RQST
The fix has been requested but has not yet arrived.

RCVD
The fix has been received but has not been installed.

INST
The fix has been installed and is being tested to determine whether or not it solves the problem.

ACPT
The fix solves the problem and is accepted.

REJD
The fix does not solve the problem and is rejected.

GROUP(group-id ...)
Specifies that problems whose group attribute matches one of the values specified are eligible for inclusion in the report. The group identifier is a maximum of 8 alphameric characters.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their group attribute.

IBMSTATUS(value ...)
Specifies that problems whose IBM status attribute matches one of the values specified are eligible for inclusion in the report.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their IBMSTATUS attribute.

The IBM status values and their meanings are:

NONE
The problem has not been reported to IBM.

REPORTED
The problem has been reported to the IBM Field Engineering (FE) division representative; help in solving the problem has been requested.

APARED
An APAR has been submitted for the problem.

PSRR
The problem has been submitted as a Programming Service Requirement Request.

CLOSED
Problems whose IBMSTATUS is CAN, DOC, DUA, DUB, DUU, MCH, PER, PRS, RET, SUG, UR1, UR2, UR3, UR4, UR5, and USE.

ABY
APAR placed in abeyance to test a fix.

ACK
APAR acknowledged.

CAN
APAR canceled.

DOC
APAR closed, documentation error.

DUA
APAR closed, duplicate of a resolved, unacceptable APAR, or duplicate of an APAR that was closed more than 10 days ago.

DUB
APAR closed, duplicate of resolved, acceptable APAR received within ten days of the original APAR closing.

DUU
APAR closed, duplicate of an unresolved APAR.

MCH
APAR closed, machine error.

PRE
APAR closed, program error.

PRS
APAR closed, permanent restriction.

RET
APAR closed, used for APARs that cannot be resolved without additional input from the field.

ROP
APAR previously in abeyance is reopened.

SUG
APAR closed, suggestion.

UR1
APAR closed, unable to reproduce (or known to be corrected) on the next release available from IBM. Written against a release that was supported at the time the APAR was received.

UR2
Same as UR1 except written against a release that was not supported at the time the APAR was received.

UR3
APAR closed, unable to reproduce (or known to be corrected) on a currently supported release. Written against a release supported at the time the APAR was received.

UR4
Same as UR3 except written against a release that was not supported at the time the APAR was received.

UR5
APAR closed, unable to reproduce on the same level system as reported. Not used if UR1 - UR4 are more appropriate.

USE
APAR closed, user error.

OWNER(tsologonid ...)
Specifies that problems whose owner attribute matches one of the values specified are eligible for inclusion in the report.

ALL
Specifies that problems owned by any user are eligible for inclusion in the report.

If you omit this parameter and all other selection parameters (COMPID, FIXSTATUS, GROUP, IBMSTATUS, PROBLEMS, ALL, PSTATUS, PTFSTATUS, SEVERITY, SYSTEM, and USER), the subcommand lists the default problem, regardless of its owner.

If you omit this parameter and specify one or more of the other selection parameters, the subcommand lists the problems owned by you that meet the specified selection criteria.

PRINT or NOPRINT
PRINT routes the subcommand listing to the print data set.

NOPRINT suppresses the listing.

TERMINAL or NOTERMINAL
TERMINAL routes the subcommand listing to the terminal.

NOTERMINAL suppresses the listing.

PROBLEMS(prob-num [: prob-num] ...)
Specifies the problem numbers that are eligible for inclusion in the report. A problem number is five decimal digits; leading zeros are optional.

Two problem numbers separated by a colon indicate an inclusive range of problem numbers. You may specify lists of problem numbers, ranges of problem numbers, lists of ranges of problem numbers, and any combination of these.

If you omit this parameter and all other selection parameters (COMPID, FIXSTATUS, GROUP, IBMSTATUS, OWNER, ALL, PSTATUS, PTFSTATUS, SEVERITY, SYSTEM, and USER), the subcommand lists the specified attributes for the default problem regardless of owner.

If you omit this parameter and specify one or more of the other selection parameters, the subcommand lists the specified attributes of all problems that meet the specified selection criteria.

ALL
Specifies that all problems in the problem directory are eligible for inclusion in the report.

If you omit this parameter and all other selection parameters (COMPID, FIXSTATUS, GROUP, IBMSTATUS, OWNER, PROBLEMS, PSTATUS, PTFSTATUS, SEVERITY, SYSTEM, and USER), the subcommand lists the specified attributes for the default problem regardless of owner.

If you omit this parameter and specify one or more of the other selection parameters, the subcommand lists the specified attributes of all problems that meet the specified selection criteria.

PSTATUS(value ...)
Specifies that problems whose problem status attribute matches one of the specified values are eligible for inclusion in the report.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their PSTATUS attribute.

The problem status values and their meanings are:

INITIAL
The problem is new and its status is not known.

ACTIVE
The problem appears to be valid and its resolution is being actively pursued.

INACTIVE
The problem appears to be valid but its resolution is not being actively pursued. In most cases, this is due to a lack of problem data or resources.

OPEN
Problems whose status is INITIAL, ACTIVE, or INACTIVE.

CLOSED
Problems whose status is CLAPAR, CLDUP, CLFIX, CLOTHER, CLPTF, or CLNTF.

CLAPAR
The problem has been closed because an APAR has been submitted for it.

CLDUP
The problem has been closed because it is a duplicate of another. The problem identifier of the original problem should be entered into the problem description.

CLFIX
The problem has been closed because a local fix, circumvention, or bypass has been applied and accepted.

CLOTHER
The problem has been closed for a reason other than those indicated by CLAPAR, CLDUP, CLFIX, CLPTF, and CLNTF.

CLPTF
The problem has been closed because a PTF has been applied and accepted.

CLNTF
The problem has been closed since, on further investigation, no trouble was found.

PTFSTATUS(value ...)
Specifies that problems whose PTF status attribute matches one of the values specified are eligible for inclusion in the report.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their PTF status attribute.

The PTF status attribute values and their meanings are:

NONE
You do not know of a PTF that solves the problem.

RQST
The PTF has been requested but has not yet arrived.

RCVD
The PTF has been received but has not been installed.

INST
The PTF has been installed and is being tested to determine whether or not it solves the problem.

ACPT
The PTF solves the problem and is accepted.

REJD
The PTF does not solve the problem and is rejected.

SEVERITY(n ...)

Specifies that problems whose severity matches one of the specified values are eligible for inclusion in the report.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their severity.

The severity values and their meanings are:

Code	Explanation
00	The problem's severity has not been determined.
01	The user is unable to use the program, resulting in a critical impact on his operations.
02	The user is able to use the program but is severely restricted.
03	The user is able to use the program with limited functions which are not critical to the overall operations.
04	The user or a system programmer has found a way to circumvent the problem. However, the APAR will be evaluated and action taken as dictated by the problem.

SYSTEM(system-id ...)

Specified that problems whose system identifier matches one of the values specified are eligible for inclusion in the report. The system identifier is a maximum of eight alphameric characters.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their system identifier.

TEST or NOTEST

TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

USER(user-data ...)

Specifies that problems whose user data matches one of the specified values are eligible for inclusion in the report. The user data may be a maximum of eight alphameric characters.

If you omit this parameter, problems are eligible for inclusion in the report regardless of their user data.

Return Codes

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

MODDSN Subcommand — Modify Data Set Attributes

Use the MODDSN subcommand to alter the attributes of a data set or a partitioned data set member. The data set must be associated with the specified or current problem and you must own that problem or have administrative authority. The data set may be associated with other problems that you may or may not own.

You can request the subcommand to require your confirmation before proceeding if data set attribute conflicts occur. These conflicts occur if the data set attributes currently associated with the data set conflict with the attributes specified on this subcommand. (The attributes of a data set are recorded only once regardless of the

number of problems with which that data set is associated.) The subcommand displays the attributes previously recorded (the current attributes) and requests your instructions.

Any user can run MODDSN at any time during an IPCS session to modify the attributes of data sets associated with problems he owns. A person with administrative authority can run MODDSN to modify the attributes of any data set defined to IPCS. This subcommand does not change the current source unless you specify the DEFAULT parameter.

Restriction:

When using IPCS in the background, you must use the TMP provided with TSO/E and must specify the JCL “USERID” parameter so that MODDSN can verify that the user is authorized to change the attributes of the specified data set.

Related Subcommands

ADDDSN
DELDSN
LISTDSN

Syntax

```
{ MODDSN }   [ CONFIRM | NOCONFIRM ]
{ MD     }

            [ DATASET(dsn) ]
            [ DSNAME(dsn)  ]

            [ DEFAULT ]
            [ DESCRIPTION('text') ]

            [ MANAGED | UNMANAGED ]

            [ PROBLEM(prob-num) ]

            [ TEST | NOTEST ]

            [ TYPE({DUMP  })   ]
            [      {PRINT }    ]
            [      {UDEF  }    ]
```

Parameters

CONFIRM

Specifies that the subcommand is to request your confirmation before proceeding if a data set attribute conflict occurs.

The subcommand displays the current data set attributes and the problem or problems with which the data set is associated. It then requests your confirmation.

If you enter Y, the subcommand stores the new attributes.

If you enter N, the subcommand ends without making changes to the data set’s attributes and ignores the DEFAULT parameter, if specified.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this parameter.

NOCONFIRM

Specifies that the subcommand is not to request your confirmation if a data set attribute conflict occurs but is to make the specified modifications.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this parameter.

DATASET or DSNAME(dsn)
Specifies the name of the data set whose attributes are to be modified. The subcommand ignores a password if one is specified as part of the data set name unless you specify DEFAULT.

If you omit both DATASET and DSNAME, the subcommand uses the current data set name.

DEFAULT
Specifies that the data set named 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, if you enter N to resolve a data set attribute conflict, or if the subcommand fails, the current source is not changed in the defaults.

DESCRIPTION('text')
Specifies a description of the data set. The description is a maximum of 60 characters of text.

If specified, the subcommand replaces the existing description with the new description.

If you omit this parameter, the subcommand does not change the data set's current description.

MANAGED
Specifies that IPCS is to manage the data set. This allows IPCS to scratch the data set when it is no longer associated with any problem.

If you omit both MANAGED and UNMANAGED, the subcommand does not change the data set's current management attribute.

IPCS does not manage partitioned data set members. If you specify MANAGED for a partitioned data set member, it is forced to UNMANAGED.

UNMANAGED
Specifies that IPCS is not to manage the data set. IPCS does not scratch the data set when it is no longer associated with a problem.

If you omit both MANAGED and UNMANAGED, the subcommand does not change the data set's current management attribute.

PROBLEM(prob-num)
Specifies a problem number with which the data set is associated. This problem number must already exist in the problem directory. A problem number is five decimal digits; leading zeros are optional.

If you omit this parameter, the subcommand uses the default problem.

Whether you specify the problem number or accept the default, you must own that problem or have administrative authority.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

TYPE(value)
Specifies the type of data set.

If you omit this parameter, the subcommand does not change the data set's current type attribute.

The type attribute values and their meanings are:

DUMP
The data set is an unformatted dump.

PRINT
The data set is printable.

UDEF
The data set type is user-defined. IPCS, therefore, does not know the format of the data set.

Return Codes

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

MODPROB Subcommand — Modify Problem Attributes

Use the MODPROB subcommand to modify the attributes of a problem defined to IPCS. A problem's attributes can be altered only by the problem owner or by the person specified in the IPCS session parameters member as having administrative authority. However, anyone may add text to the problem description.

This subcommand does not change the default problem unless you specify the DEFAULT parameter.

Restriction:

When using IPCS in the background, you must use the TMP provided with TSO/E and must specify the JCL "USERID" parameter so that MODPROB can verify that the user is authorized to change the attributes of the specified problem.

Related Subcommands

ADDPROB
DELPROB
LISTPROB

Syntax

```
{ MODPROB } [ ABSTRACT('text') ]
{ MP      }
            [ APARID(apar-id) ]
            [ COMPID(component-id) ]
            [ DATE(date) ]
            [ DEFAULT ]

            [ DESCRIPTION('text')  ]
            [ DSDESCRIPTION(dsn)   ]

            [ FIXID('fix-id') ]

            [ FIXSTATUS ({NONE  }) ]
            [            {RQST  }  ]
            [            {RCVD  }  ]
            [            {INST  }  ]
            [            {ACPT  }  ]
            [            {REJD  }  ]

            [ GROUP(group-id) ]



            [IBMSTATUS({ NONE    })     ]
            [          { REPORTED}      ]
            [          { APARED  }      ]
            [          { PSRR    }      ]
            [          { ABY     }      ]
            [          { ACK     }      ]
            [          { CAN     }      ]
            [          { DOC     }      ]
            [          { DUA     }      ]
            [          { DUB     }      ]
            [          { DUU     }      ]
            [          { MCH     }      ]
            [          { PER     }      ]
            [          { PRS     }      ]
            [          { RET     }      ]
            [          { ROP     }      ]
            [          { SUG     }      ]
            [          { UR1     }      ]
            [          { UR2     }      ]
            [          { UR3     }      ]
            [          { UR4     }      ]
            [          { UR5     }      ]
            [          { USE     }      ]



            [ OWNER(tsologonid) ]

            [ PROBLEM(prob-num) ]

            [ PSTATUS({ INITIAL   })   ]
            [         { ACTIVE    }    ]
            [         { INACTIVE  }    ]
            [         { CLAPAR    }    ]
            [         { CLDUP     }    ]
            [         { CLFIX     }    ]
            [         { CLOTHER   }    ]
            [         { CLPTF     }    ]
            [         { CLNTF     }    ]
```

```
[ PTFID(ptf-id) ]

[ PTFSTATUS({ NONE  }) ]
[           { RQST  }  ]
[           { RCVD  }  ]
[           { INST  }  ]
[           { ACPT  }  ]
[           { REJD  }  ]



[ SEVERITY({ 0  })  ]
[          { 1  }   ]
[          { 2  }   ]
[          { 3  }   ]
[          { 4  }   ]

[ SYSTEM(system-id) ]

[ TEST | NOTEST ]

[ TIME(tod) ]
[ USER(user-data) ]
```

Parameters

ABSTRACT('text')
Specifies the problem abstract. The abstract is a maximum of 128 characters of text providing a brief description of the problem.

If specified, the subcommand replaces the existing abstract with the new abstract. If you omit this parameter, the subcommand does not change the current abstract.

APARID(apar-id)
Specifies the identifier assigned to the APAR for the problem. The APAR identifier must be 7 alphameric characters and the first character must be alphabetic.

If you omit this parameter, the subcommand does not change the current APAR identifier.

COMPID(component-id)
Specifies the suspected failing component. The component identifier may be a maximum of 10 characters. For each component identifier, you can specify any characters except blanks, commas, semicolons, and tabs. Parentheses are allowed but must be balanced.

If you omit this parameter, the subcommand does not change the current component identifier.

DATE(date)
Specifies the date the problem occurred in mm/dd/yy format. The slashes must be specified; leading zeros are optional.

If you omit this parameter, the subcommand does not change the date currently recorded for this problem.

DEFAULT
Specifies that the problem being modified is to become the current problem.

IPCS changes the current problem in both the local and global defaults. If you omit this parameter or if the subcommand fails, the current problem is not changed in the defaults.

DESCRIPTION('text')

Specifies text to be appended to the existing problem description. The problem description may contain a maximum of 9,360,000 characters.

If you omit this parameter and you omit DSDESCRIPTION, the subcommand does not change the current description of the problem.

This parameter may be used by any user and is not restricted to the problem owner.

DSDESCRIPTION(dsn)

Specifies the name of a cataloged sequential data set containing the description of the problem. The specified data set must be fixed format containing blocked or unblocked records with a logical record length of 80. The subcommand erases the current problem description and replaces it with the new description. The subcommand copies the first 72 characters of each logical record from the specified data set into the problem description. If the specified data set can not be found, the subcommand leaves the problem with no description. The problem description may contain a maximum of 9,360,000 characters.

This parameter can be used only by the problem owner or the person with administrative authority.

If you omit this parameter and if you omit DESCRIPTION, the subcommand does not change the current description for this problem.

FIXID('fix-id')

Specifies the identifier assigned to the fix for the problem. The fix identifier is a maximum of 60 characters. There are no restrictions on the characters you can use for this parameter.

If you omit this parameter, the subcommand does not change the current fix identifier.

FIXSTATUS(value)

Specifies the status of a local fix, circumvention, or bypass.

If you specify a value for this parameter, the subcommand records the date and time of that specification. If you omit this parameter, the subcommand does not change the current FIXSTATUS and does not update its date and time fields.

The FIXSTATUS values and their meanings are:

NONE

You do not know of a fix that solves the problem.

If you specify NONE, the subcommand sets this field to blanks and records the current date and time.

RQST

The fix has been requested but has not yet arrived.

RCVD

The fix has been received but has not been installed.

INST

The fix has been installed and is being tested to determine whether or not it solves the problem.

ACPT

The fix solves the problem and is accepted.

REJD
The fix does not solve the problem and is rejected.

GROUP(group-id)
Specifies the department name or number responsible for the problem. The group identifier is a maximum of 8 alphameric characters.

If you omit this parameter, the subcommand does not change the current group identifier.

IBMSTATUS(value)
Specifies the status of a problem that has been reported to IBM.

If you specify a value for this parameter, the subcommand records the date and time of that specification.

If you omit this parameter, the subcommand does not change the current IBMSTATUS and does not update its date and time fields.

The IBMSTATUS values and their meanings are:

NONE
The problem has not been reported to IBM.

If you specify NONE, the subcommand sets this field to blanks and records the current date and time.

REPORTED
The problem has been reported to the IBM Field Engineering (FE) division representative; help in solving the problem has been requested.

APARED
An APAR has been submitted for the problem.

PSRR
The problem has been submitted as a Programming Service Requirement Request.

ABY
APAR placed in abeyance to test a fix.

ACK
APAR acknowledged.

CAN
APAR canceled.

DOC
APAR closed, documentation error.

DUA
APAR closed, duplicate of a resolved, unacceptable APAR, or duplicate of an APAR which was closed more than 10 days ago.

DUB
APAR closed, duplicate of resolved acceptable APAR received within ten days of the original APAR closing.

DUU
APAR closed, duplicate of an unresolved APAR.

MCH
APAR closed, machine error.

PER
APAR closed, program error.

PRS
APAR closed, permanent restriction.

RET
APAR closed, closing code used for APARs which cannot be resolved without additional input from the field.

ROP
APAR previously in abeyance is reopened.

SUG
APAR closed, suggestion.

UR1
APAR closed, unable to reproduce (or known to be corrected) on the next release available from IBM. Written against a release that was supported at the time the APAR was received.

UR2
Same as UR1 except written against a release that was not supported at the time the APAR was received.

UR3
APAR closed, unable to reproduce (or known to be corrected) on a currently supported release. Written against a release that was supported at the time the APAR was received.

UR4
Same as UR3 except written against a release that was not supported at the time the APAR was received.

UR5
APAR closed, unable to reproduce on the same level system as reported. Not used if UR1 - UR4 are more appropriate.

USE
APAR closed, user error.

OWNER(tsologonid)
Specifies the TSO/E userid of the person responsible for resolving the problem.

If you omit this parameter, the subcommand does not change the current owner.

PROBLEM(prob-num)
Specifies the problem number of the problem whose attributes are being modified. This problem number must already exist in the problem directory. The problem number is five decimal digits; leading zeros are optional.

If you omit this parameter, the subcommand modifies the attributes of the default problem.

PSTATUS(value)
Specifies the status of the problem.

If you specify a value for this parameter, the subcommand records the date and time of that specification.

If you omit this parameter, the subcommand does not change the current problem status and does not update its date and time fields.

The problem status values and their meanings are:

INITIAL
The problem is new and its status is not known.

ACTIVE
The problem appears to be valid and its resolution is being actively pursued.

INACTIVE
The problem appears to be valid but its resolution is not being actively pursued. In most cases, this is due to a lack of problem data or resources.

CLAPAR
The problem has been closed due to an APAR submitted for it.

CLDUP
The problem has been closed because it is a duplicate of another. The problem identifier of the original problem should be entered into the problem description.

CLFIX
The problem has been closed because a local fix, circumvention, or bypass has been applied and accepted.

CLOTHER
The problem has been closed for a reason other than those indicated by CLAPAR, CLDUP, CLFIX, CLPTF, and CLNTF. You should describe the reason for closure in the problem description.

CLPTF
The problem has been closed because a PTF has been applied and accepted.

CLNTF
The problem has been closed since, on further investigation, no trouble was found.

PTFID(ptf-id)
Specifies the identifier assigned to the PTF for the problem. The PTF identifier must be 7 alphameric characters and the first character must be alphabetic.

If you omit this parameter, the subcommand does not change the current PTF identifier.

PTFSTATUS(value)
Specifies the status of the PTF that you identify as the one which solves the problem.

If you specify a value for this parameter, the subcommand records the date and time of that specification.

If you omit this parameter, the subcommand does not change the current PTF status and does not update its date and time fields.

The PTFSTATUS values and their meanings are:

NONE
You do not know of a PTF that solves the problem.

If you specify NONE, the subcommand sets this attribute field to blanks and records the current date and time.

RQST
The PTF has been requested but has not yet arrived.

RCVD
The PTF has been received but has not been installed.

INST
The PTF is installed and is being tested to see whether or not it solves the problem.

ACPT
The PTF solves the problem and is accepted.

REJD
The PTF does not solve the problem and is rejected.

SEVERITY(n)
Specifies the problem's severity, consistent with IBM APAR severity codes listed below.

If you specify 0, the subcommand sets this field to blanks.

If you omit this parameter, the subcommand does not change the current severity value.

The severity codes and their meanings are:

Code	Explanation
01	The user is unable to use the program, resulting in a critical impact on his operations.
02	The user is able to use the program but is severely restricted.
03	The user is able to use the program with limited functions which are not critical to the overall operations.
04	The user or a system programmer has found a way to circumvent the problem. However, the APAR will be evaluated and action taken as dictated by the problem.

SYSTEM(system-id)
Specifies the system on which the problem occurred. The system identifier may be a maximum of eight alphameric characters.

If you omit this parameter, the subcommand does not change the current system identifier.

TEST or NOTEST
TEST nullifies IPCS error recovery for ABENDs.

NOTEST activates IPCS error recovery for ABENDs.

TIME(tod)
Specifies the time of day the problem occurred in hh:mm:ss format. The colons and leading zeros must be specified.

If you omit this parameter, the subcommand does not change the time currently recorded for this problem.

USER(user-data)
Specifies the user-data field. This field can contain a maximum of 8 alphameric characters.

If you omit this parameter, the subcommand does not change the current user-data field.

Return Codes

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

Appendix B. 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 would 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

Symbol	Associated Data	Data Type Definition
AFT	The ASN-first-table control block	STRUCTURE(AFTE)
ASCBnnnnn	The address space control block for address space nnnnn	STRUCTURE(ASCB)
ASMVT	The system auxiliary storage management vector table	STRUCTURE(ASMVT)
ASTnnnn	The ASN-second-table control block for address space group nnnn	STRUCTURE(ASTE)
ASTEnnnnn	The ASN-second-table control block entry for address space nnnnn	STRUCTURE(ASTE)

© Copyright IBM Corp. 1988, 2002

IPCS Symbols

Symbol	Associated Data	Data Type Definition
ASVT	The system address space vector table	STRUCTURE(ASVT)
ASXBnnnnn	The address space extension block for address space nnnnn	STRUCTURE(ASXB)
BLSQXBT	Table of system materials built from parmlib members BLSCECT, BLSCUSER, ...	STRUCTURE(BLSQXBT)
BLSQXBTnnnnn	Table of materials used by IPCS in ASID nnnnn for processing of dumps and traces generated by an ESA-mode system.	STRUCTURE(BLSQXBT)
BLSQXBTG	Table of materials used by IPCS in ASID nnnnn for processing of dumps and traces generated by a system supporting z/Architecture.	STRUCTURE(BLSQXBT)
CDEpgmname	A contents directory entry for entry point pgmname	STRUCTURE(CDE)
COMMON	The system common area	AREA(COMMON)
CSA	The common system area	AREA(CSA)
CSD	The common system data area	STRUCTURE(CSD)
CURSOR	A fullword pointer identified by the position of the cursor on the display terminal	
CVT	The system communications vector table	STRUCTURE(CVT)
CVTVSTGX	The virtual storage address extension to the system communications vector table	STRUCTURE(CVTVSTGX)
CVTXTNT2	The system communications vector table extension	STRUCTURE(CVTXTNT2)
DAESYMPTOMS	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.	CHARACTER
DATOFFNUCLEUS	The portion of the system nucleus that is used with dynamic address translation turned off	AREA(DATOFFNUCLEUS)
DIB	A control block maintained to support the data-in-virtual function	STRUCTURE(DIB)
DIBX	A control block maintained to support the data-in-virtual function	STRUCTURE(DIBX)
DUMPINGPROGRAM	The name of the program that produced the dump	CHARACTER
DUMPORIGINALDSNAME	The name of the original data set to which the dump was written	CHARACTER
DUMPREQUESTOR	The name of the program that requested the dump	CHARACTER
DUMPTIMESTAMP	The time from the time-of-day (TOD) clock presented in the following format: `mm/dd/yyyy hh:mm:ss.ffffff`	CHARACTER

Symbol	Associated Data	Data Type Definition
DUMPTOD	The time from the time-of-day (TOD) clock in a bit string	STRUCTURE(TODCLOCK)
ECSA	The extended common system area	AREA(ECSA)
EFLPA	The extended fixed link pack area	AREA(EFLPA)
EMLPA	The extended modified link pack area	AREA(EMLPA)
ENUCLEUS	The extended nucleus	AREA(ENUCLEUS)
EPnnnnn	Entry point nnnnn in an entry point trace	MODULE
EPIDnnnnn	Entry point identifier nnnnn in an entry point trace	CHARACTER
EPLPA	The extended pageable link pack area	AREA(EPLPA)
ERRORID	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.	STRUCTURE(ERRORID)
ESQA	The extended system queue area	AREA(ESQA)
FINDAREA	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.	
FLPA	The fixed link pack area	AREA(FLPA)
GDA	The global data area	STRUCTURE(GDA)
IEAVESLA	The system lock area	STRUCTURE(IEAVESLA)
IEFJESCTPX	Pageable JESCT extension	STRUCTURE(IEFJESCTPX)
IEFZB445	Device allocation default table	STRUCTURE(IEFZB445)
IHSAnnnnn	The interrupt handler save area for address space nnnnn	STRUCTURE(IHSA)
INCIDENTTOKEN	The incident token for all dumps initiated by a single dump request	STRUCTURE(IEAINTKN)
ISGGVT	The global resource serialization vector table	STRUCTURE(ISGGVT)
ISGGVTX	The global resource serialization vector table extension	STRUCTURE(ISGGVTX)
ISGQHTG	The global resource serialization queue hash table for global resources	STRUCTURE(ISGQHT)
ISGQHTL	The global resource serialization queue hash table for local resources	STRUCTURE(ISGQHT)
ISGRSV	The global resource serialization ring status vector	STRUCTURE(ISGRSV)
ITTCTAB	Component trace anchor block	STRUCTURE(ITTCTAB)
ITTCTQE name	Component name CTRACE queue entry	STRUCTURE(ITTCTQE)
LCCAnn	The logical configuration communication area for processor nn	STRUCTURE(LCCA)

IPCS Symbols

Symbol	Associated Data	Data Type Definition
LCCAVT	The LCCA vector table	STRUCTURE(LCCAVT)
LCCXnn	LCCA entension for cpu nn	STRUCTURE(LCCX)
LDAnnnnn	LDA for ASID nnnnn	STRUCTURE(LDA)
LPDEpgmname	The link pack directory entry for pgmname	STRUCTURE(LPDE)
MLPA	The modified link pack area	AREA(MLPA)
NUCLEUS	The nucleus	AREA(NUCLEUS)
NVT	The nucleus initialization program (NIP) vector table	STRUCTURE(NVT)
PART	The page address resolution table. This symbol is defined only by the ASMCHECK subcommand.	STRUCTURE(PART)
PCCAnn	The physical configuration communication area for processor nn	STRUCTURE(PCCA)
PCCAVT	The PCCA vector table	STRUCTURE(PCCAVT)
PFT	The system page frame table	STRUCTURE(PFT)
pgmname	A load module or portion of a load module originating at entry point pgmname	MODULE(pgmname)
PGTnnnnnaaaaa	The page table for address space nnnnn, segment aaaaa The page table for segment 0 of address space 1 is PGT00001AAAAA; for segment 1, PGT00001AAAAB, ...	STRUCTURE(PGTE)
PLPA	The pageable link pack area	AREA(PLPA)
PRIMARYSYMPTOMS	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.	CHARACTER
PRIVATE	The private area	AREA(PRIVATE)
PRIVATEX	The extended private area	AREA(PRIVATEX)
PSAnn	The prefixed storage area for processor nn	STRUCTURE(PSA)
PSAVALID	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.	STRUCTURE(PSA)
PSW	The program status word at or near the error point in a virtual dump	STRUCTURE(PSW)
PSWnn	The program status word for CPU nn in a stand-alone dump	STRUCTURE(PSW)
PVT	The system paging vector table	STRUCTURE(PVT)
REGACC	Access registers at or near the error point in a virtual dump	STRUCTURE(REGACC)

Symbol	Associated Data	Data Type Definition
REGACCnn	Access registers for CPU nn in a stand-alone dump	STRUCTURE(REGACC)
REGCTL	Control registers at or near the error point in a virtual dump	STRUCTURE(REGCTL)
REGCTLnn	Control registers for CPU nn in a stand-alone dump	STRUCTURE(REGCTL)
REGFLT	Floating point registers at or near the error point in a virtual dump	STRUCTURE(REGFLT)
REGFLTnn	Floating point registers for CPU nn in a stand-alone dump	STRUCTURE(REGFLT)
REGFPC	Floating point control register at or near the error point in an unformatted dump	STRUCTURE(REGFLT)
REGFPCnn	Floating point control register for CPU nn in a stand-alone dump	STRUCTURE(REGFLT)
REGGEN	General purpose registers at or near the error point in a virtual dump	STRUCTURE(REGGEN)
REGGENnn	General purpose registers for CPU nn in a stand-alone dump	STRUCTURE(REGGEN)
REGG64H	High-order halves (bits 0-31) of 64-bit general registers	STRUCTURE(REGG64H)
REGG64Hnn	High-order halves (bits 0-31) of 64-bit general registers for cpu nn	STRUCTURE(REGG64H)
REMOTEDUMP	Indicator that dumps on other systems in the sysplex were requested: • The request for this dump also requested dumps on other systems • This is a dump requested by another system	CHARACTER
RONUCLEUS	The read-only portion of the nucleus	AREA(RONUCLEUS)
RTCT	The recovery termination control table	STRUCTURE(RTCT)
SAnnnnn	Save area nnnnn in an entry point or 72-byte save area trace	STRUCTURE(REGSAVE)
SCCB	The service call control block	STRUCTURE(SCCB)
SCVT	The secondary CVT	STRUCTURE(SCVT)
SDWAHDR	The SDWA saved in a dump header record	STRUCTURE(SDWAHDR)
SECONDARYSYMPTOMS	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.	CHARACTER
SGTnnnnn	The segment table for address space nnnnn	STRUCTURE(SGTE)
SLIPTRAP	The SLIP command that requested the dump. If the actual command is longer than 256 bytes, it is truncated.	CHARACTER

IPCS Symbols

Symbol	Associated Data	Data Type Definition
TCBCURRENT	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.	STRUCTURE(TCB)
TCBnnnnnaaaaa	The task control block for address space nnnnn, in position aaaaa in the priority queue The highest priority TCB in address space 1 is TCB00001AAAAA; the next TCB on the queue is TCB00001AAAAB, ... The last 2 characters in this name are alphabetic and range from AAAAA through AZZZZ, BAAAA, ... BZZZZ, ...	STRUCTURE(TCB)
TITLE	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.	CHARACTER
UCBdddd	The unit control block for device dddd. The dddd designates the device number in hexadecimal.	STRUCTURE(UCB)
UCM	The unit control module	STRUCTURE(UCM)
X	The "current address" in a dump. This symbol is defined by most IPCS subcommands in all types of dumps supported by IPCS.	
XLpgmname	An extent list for entry point pgmname	STRUCTURE(XTLST)
Znnnnn	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.	

Appendix C. 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 upper case 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 upper case letter, A through F

Table C-1. IPCS Special Symbols

SYMBOL	Minimum	Maximum	Symbol Description
ASCBnnnnn	ASCB1	ASCB99999	The address space control block for address space nnnnn.
ASTnnnn	AST0	AST9999	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*16 through nnnn*16+15.)
ASTEnnnnn	ASTE1	ASTE9999	The address space second table entry for address space nnnnn.
ASXBnnnnn	ASXB1	ASCB99999	The address space extension block for address space nnnnn.
IHSAnnnnn	IHSA1	IHSA9999	The interrupt handler save area for address space nnnnn.
LCCAnn	LCCA0	LCCA99	The logical configuration communication area for processor nn.
PCCAnn	PCCA0	PCCA99	The physical configuration communication area for processor nn.
PGTnnnnnaaaaa	PGT1A	PGT99999ZZZZZ	The page table for segment aaaaa (base 26 number) in address space nnnnn.
PSAnn	PSA0	PSA99	The prefixed storage area for processor nn.
PSWnn	PSW0	PSW99	The program status word for processor nn.
REGACCnn	REGACC0	REGACC99	The access registers for processor nn.
REGCTLnn	REGCTL0	REGCTL99	The control registers for processor nn.
REGFLTnn	REGFLT0	REGFLT99	The floating point registers for processor nn.
REGFPCnn	REGFPC0	REGFPC99	The floating point control register for processor nn.
REGGENnn	REGGEN0	REGGEN99	The general purpose registers for processor nn.
SGTnnnnn	SGT1	SGT99999	The segment table for address space nnnnn.
TCBnnnnnaaaaa	TCB1A	TCB99999ZZZZZ	The task control block in position aaaaa (base 26 number) on the priority chain in address space nnnnn.
UCBdddd	UCB0	UCBFFFF	The unit control block for the device number dddd.
Znnnnn	Z1	Z99999	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.

© Copyright IBM Corp. 1988, 2002

Appendix D. 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.

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
ACE	no	no	yes	
AFT	yes	yes	no	
AFTE	yes	yes	no	1
AIA	no	no	yes	
ALE	no	no	yes	
AMDCPMAP	no	no	yes	
AR	no	no	yes	
ASCB	yes	yes	yes	12
ASEI	no	no	yes	
ASMHD	no	no	yes	
ASMVT	no	yes	yes	
ASPCT	no	no	yes	
ASSB	yes	yes	yes	
AST	yes	yes	no	12
ASTE	yes	yes	no	2, 12
ASVT	yes	yes	no	
ASXB	yes	yes	yes	12
CACHE	no	no	yes	
CDE	yes	yes	yes	
CDEMAJOR	yes	yes	yes	3
CDEMINOR	yes	yes	yes	3
CLTE	no	no	yes	
CQE	yes	yes	no	
CSD	yes	yes	yes	
CSRCPOOL	no	no	yes	
CVT	yes	yes	yes	
CVTVSTGX	yes	yes	yes	
CVTXTNT2	yes	yes	yes	
DCB	no	yes	no	
DEIB	no	no	yes	

© Copyright IBM Corp. 1988, 2002

Control Blocks and Data Areas

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
DEIE	no	no	yes	
DIB	yes	yes	yes	
DIBX	yes	yes	yes	
DOA	yes	yes	yes	
DOM	no	no	yes	
DSAB	no	no	yes	
DSNT	no	no	yes	15
EED	no	no	yes	
FRRS	no	no	yes	
GDA	yes	yes	no	
GEPL	no	no	yes	
HED	no	no	yes	
HTBL	no	no	yes	
IATYDAT	no	no	yes	10
IATYDMC	no	no	yes	10
IATYDSS	no	no	yes	10
IATYFCT	no	no	yes	10
IATYIOP	no	no	yes	10
IATYMEME	no	no	yes	10
IATYMEMH	no	no	yes	10
IATYMPC	no	no	yes	10
IATYOSD	no	no	yes	10
IATYOSED	no	no	yes	10
IATYOSEF	no	no	yes	10
IATYOSEV	no	no	yes	10
IATYRQCI	no	no	yes	10
IATYRQCM	no	no	yes	10
IATYRQFX	no	no	yes	10
IATYRQGM	no	no	yes	10
IATYRQMD	no	no	yes	10
IATYRQOS	no	no	yes	10
IATYSEL	no	no	yes	10
IATYSPB	no	no	yes	10
IATYSVT	no	yes	yes	10
IATYTVT	no	no	yes	10
IEAVESLA	no	yes	no	
IEFJESCT	no	yes	yes	
IEFJESCTPX	yes	yes	no	
IEFJSCVT	no	yes	yes	

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
IEFJSSVT	no	yes	yes	
IEFOAWTR	no	no	yes	
IEFZAGT	no	no	yes	
IEFZB41C	no	no	yes	
IEFZB445	yes	yes	yes	
IEFZIGDE	no	no	yes	
IEFZTSRA	no	no	yes	
IHSA	yes	yes	yes	12
IORB	no	no	yes	
IOSB	no	no	yes	
IRB	yes	yes	no	6
ISGGVT	yes	yes	no	9
ISGGVTX	yes	yes	no	9
ISGQCB	yes	yes	no	9
ISGQEL	yes	yes	no	9
ISGQHT	yes	yes	no	9
ISGRPT	no	yes	no	
ISGRSV	yes	yes	no	9
ISGSAHT	no	yes	no	
IXCYERE	no	no	yes	
IXCYEVE	no	no	yes	
IXCYWRE	no	no	yes	
IXGARTE	no	no	yes	
IXGDIRCT	no	no	yes	
IXGINV	no	no	yes	
IXGIPSTK	no	no	yes	14
IXGLBCB	no	no	yes	
IXGLCB	no	no	yes	
IXGLCBVT	no	no	yes	
IXGLCCB	no	no	yes	
IXGLSAB	no	no	yes	
IXGPCNTL	no	no	yes	
IXGRQE	no	no	yes	
IXGSTRCB	no	no	yes	
JCT	no	no	yes	15
JCTX	no	no	yes	15
JESCT	no	yes	no	
JFCB	no	no	yes	15
JFCBE	no	no	yes	15

Control Blocks and Data Areas

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
JFCBX	no	no	yes	15
JQE	no	yes	no	
JSCB	no	no	yes	
LCCA	yes	yes	yes	12
LCCAVT	yes	yes	no	
LDA	yes	yes	yes	
LGE	no	no	yes	
LGVT	no	no	yes	
LGVTE	no	no	yes	
LLE	no	no	yes	
LPDE	yes	yes	yes	3
LPDEFINAL	yes	yes	yes	3
LPDEMAJOR	yes	yes	yes	3
LPDEMINOR	yes	yes	yes	3
LPDENULL	yes	yes	yes	3
LS	no	no	yes	
LSE	no	no	yes	
LSEH	no	no	yes	
LSET	no	no	yes	
LSSD	yes	yes	yes	
LSSG	yes	yes	yes	
MEPL	no	no	yes	
MGCRE	no	no	yes	
NVT	yes	yes	no	
ORE	yes	yes	no	
OUCB	yes	yes	yes	
OUSB	no	yes	no	
OUXB	no	yes	no	
PART	no	no	yes	
PARTE	no	no	yes	
PAT	no	no	yes	
PCB	no	yes	no	
PCCA	yes	yes	yes	12
PCCAVT	yes	yes	no	
PFT	no	yes	no	4
PFTE	no	yes	no	
PGT	yes	yes	no	5, 12
PGTE	yes	yes	no	
PPD	yes	yes	no	

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
PRB	yes	yes	no	6
PSA	yes	yes	yes	12
PSW	no	yes	yes	12
PVT	yes	yes	no	
PXT	no	yes	no	
RB	yes	yes	yes	
RDCM	no	no	yes	
REGACC	no	no	yes	12
REGCTL	no	no	yes	12
REGFLT	no	no	yes	12
REGGEN	no	no	yes	12
REGS	no	no	yes	
REGSAVIM	no	no	yes	
RMCT	no	yes	no	
RQE	no	yes	no	
RSMHD	no	yes	no	
RTCT	yes	yes	yes	
RTM2WA	no	yes	yes	
RT1W	yes	yes	no	
SART	no	no	yes	
SARTE	no	no	yes	
SAT	no	no	yes	
SBC	no	no	yes	
SCB	yes	yes	yes	
SCCB	yes	yes	yes	
SCCW	no	no	yes	
SCFS	no	no	yes	
SCT	no	no	yes	15
SCTX	no	no	yes	15
SCVT	yes	yes	no	
SDCT	no	no	yes	
SDUMP	no	no	yes	
SDWA	no	no	yes	
SDWAHDR	no	no	yes	
SEPL	no	no	yes	
SGT	yes	yes	no	7, 12
SGTE	yes	yes	no	
SIOT	no	no	yes	15
SIRB	yes	yes	no	6

Control Blocks and Data Areas

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
SPD	yes	yes	no	
SPQE	no	yes	no	
SRB	no	yes	yes	
SSRB	no	no	yes	
STCB	yes	yes	yes	
STKE	no	no	yes	
SUPVT	no	no	yes	
SVRB	yes	yes	no	6
SVT	no	no	yes	
SVTX	yes	no	yes	
SXT	no	yes	no	
TCB	yes	yes	yes	12
TDCM	no	no	yes	
TIAB	no	no	yes	
TIOT	no	yes	yes	11
TIOTE	no	no	yes	11
TIRB	yes	yes	no	6
TODCLOCK	no	no	yes	
TODCNULL	no	no	yes	
TODC4	no	no	yes	
TQE	no	yes	no	
TSB	no	yes	no	
UCB	yes	yes	no	12
UCBCTC	yes	yes	no	8
UCBDA	yes	yes	no	8
UCBEXT	no	yes	no	
UCBGFX	yes	yes	no	8
UCBTAPE	yes	yes	no	8
UCBTP	yes	yes	no	8
UCBUR	yes	yes	no	8
UCB3270	yes	yes	no	8
UCM	yes	yes	yes	
UCME	no	no	yes	
VCOM	no	no	yes	
VF	no	no	yes	
VSWK	no	no	yes	
WCB	yes	yes	yes	
WEB	yes	no	yes	
WEE	yes	no	yes	

Control Block or Data Area	Scanned	Storage Map Entry	Formatted	Notes (following the table)
WQE	yes	yes	yes	
WSAVTC	no	yes	no	
WSAVTG	no	yes	no	
WSMA			yes	
XCFSTACK	no	no	yes	13
XESSTACK	no	no	yes	13
XSB	no	no	yes	
XTLST	yes	yes	yes	

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, ISGRPT, ISGRSV, and ISGSAHT are referenced, without the prefix ISG, in the following books. For example, ISGGVT is listed under GVT.
   - *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*
   - *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*
   - *z/OS MVS Data Areas, Vol 4 (RD-SRRA)*
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 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 Appendix C, “IPCS Special Symbols for System Control Blocks” on page C-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.

Appendix E. 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 E-1. AMDPRDMP - IPCS Conversion Summary

Print Dump Control Statement or Function	IPCS Equivalent
ASMDATA control statement	ASMDATA verb exit • Use VERBEXIT ASMDATA to format certain ASM control blocks. • See “VERBEXIT Subcommand — Run an Installation-Supplied or an IBM-Supplied Verb Exit Routine” on page 5-284 and “VERBEXIT ASMDATA Subcommand — Format Auxiliary Storage Manager Data” on page 5-288.
AVMDATA control statement	AVMDATA verb exit • Use VERBEXIT AVMDATA to format the contents of accessible availability manager control blocks. • See “VERBEXIT Subcommand — Run an Installation-Supplied or an IBM-Supplied Verb Exit Routine” on page 5-284 and “VERBEXIT AVMDATA Subcommand — Format Availability Manager Data” on page 5-288.
Copy and clear a source SYS1.DUMP data set	COPYDUMP • Use COPYDUMP CLEAR to clear a SYS1.DUMP data set after copying. • See “COPYDUMP Subcommand — Copy Dump Data” on page 5-51 • For sample JCL to print, offload, and clear a dump, see the *z/OS MVS IPCS User’s Guide*.
CPUDATA control statement	STATUS DATA subcommand • Use the STATUS subcommand to gather processor-related debugging information. • See “STATUS Subcommand — Describe System Status” on page 5-234.
CVT control statement	EQUATE subcommand • Use EQUATE CVT address when you want to associate the address of the CVT control block with a symbol. • See “EQUATE Subcommand — Create a Symbol” on page 5-89.
CVTMAP control statement	CBFORMAT subcommand • Use the CBFORMAT subcommand to display the contents of the CVT control block. • See “CBFORMAT Subcommand — Format a Control Block” on page 5-30.
DAEDATA control statement	DAEDATA verb exit • Use VERBEXIT DAEDATA to format DAE dump data. • See “VERBEXIT Subcommand — Run an Installation-Supplied or an IBM-Supplied Verb Exit Routine” on page 5-284 and “VERBEXIT DAEDATA Subcommand — Format Dump Analysis and Elimination Data” on page 5-289.
Dumped storage summary	LISTDUMP subcommand • Use the LISTDUMP subcommand to provide a summary of the storage in one or more dumps. • LISTDUMP is described under “LISTDUMP Subcommand — List Dumps in Dump Directory” on page 5-147.

© Copyright IBM Corp. 1988, 2002

Print Dump to IPCS Conversion

Table E-1. AMDPRDMP - IPCS Conversion Summary (continued)

Print Dump Control Statement or Function	IPCS Equivalent
EDIT control statement	GTFTRACE subcommand • 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 5-120.
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 5-88.
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 5-255.
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. Refer to 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 5-284 and “VERBEXIT GRSTRACE Subcommand — Format Global Resource Serialization Data” on page 5-291.
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. • The IMSDUMP formatter is described in *IMS/VS Utilities Reference Manual* and *IMS/VS Diagnosis Guide and Reference*.
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 section called “Print and Table of Contents Files” in the *z/OS MVS IPCS User’s Guide*.
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 5-129.

Table E-1. AMDPRDMP - IPCS Conversion Summary (continued)

Print Dump Control Statement or Function	IPCS Equivalent
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. • The IRLM dump formatter is described in *IMS/VS Diagnosis Guide and Reference* and *IMS/VS System Definition Reference Manual*.
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*.
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*.
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 5-284 and “VERBEXIT LOGDATA Subcommand — Format Logrec Buffer Records” on page 5-296.
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 5-167.
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 5-284 and “VERBEXIT MTRACE Subcommand — Format Master Trace Entries” on page 5-299.
NEWDUMP control statement NEWTAPE 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 5-223.
NUCMAP 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 5-284 and “VERBEXIT NUCMAP Subcommand — Map Modules in the Nucleus” on page 5-302.
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 E-1. AMDPRDMP - IPCS Conversion Summary (continued)

Print Dump Control Statement or Function	IPCS Equivalent
PRINT CSA, SQA, NUCLEUS control statements	LIST subcommand and CLISTS • Use these symbols on the LIST subcommand to format and display information for the CSA, SQA, and NUCLEUS: **CSA, ECSA** CSA storage above and below 16 megabytes. **SQA, ESQA** SQA storage above and below 16 megabytes. **NUCLEUS** **ENUCLEUS** **RONUCLEUS** **DATOFFNUCLEUS** Read/write nucleus storage below and above 16 megabytes; read-only nucleus storage; the DAT-OFF portion of the nucleus. **PRIVATE, PRIVATEX** Private area below and above 16 megabytes. • Use the BLSCPCSA, BLSCPNUC, BLSCPRIV, and BLSCPSQA CLISTs to print information from these system areas. • See “LIST Subcommand — Display Storage” on page 5-145 for information about the LIST subcommand. • See Chapter 8, “IPCS Batch Mode” on page 8-1 for a description of these CLISTs.
PRINT STORAGE, REAL control statements	LIST subcommand • Use the LIST subcommand to display storage contents. • See “LIST Subcommand — Display Storage” on page 5-145.
PRINT CURRENT,JOBNAME control statements	BLSCPRNT CLIST • 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. • See “BLSCPRNT CLIST — Print a Dump” on page 7-10 for a description of BLSCPRNT and its operands.
Q or QCBTRACE control statements	See GRSTRACE control statement.
RSMDATA control statement	RSMDATA subcommand • Use the RSMDATA subcommand to format information about the real storage management component. • See “RSMDATA Subcommand — Analyze Real Storage Manager Data” on page 5-195.
SADMPMSG control statement	VERBEXIT SADMPMSG subcommand • Use the SADMPMSG verb name of the VERBEXIT subcommand to format the SADMP execution-time virtual storage dump message log. • See “VERBEXIT Subcommand — Run an Installation-Supplied or an IBM-Supplied Verb Exit Routine” on page 5-284 and “VERBEXIT SADMPMSG Subcommand — Format Stand-Alone Dump Message Log” on page 5-305.
SEGTAB control statement	EQUATE subcommand • Use the SGT symbol with an address on the EQUATE subcommand to associate the segment table with its address and storage attributes. • See “EQUATE Subcommand — Create a Symbol” on page 5-89.

Table E-1. AMDPRDMP - IPCS Conversion Summary (continued)

Print Dump Control Statement or Function	IPCS Equivalent
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 5-284 and “VERBEXIT SRMDATA Subcommand — Format System Resource Manager Data” on page 5-305.
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 5-284 and “VERBEXIT SUMDUMP Subcommand — Format SVC Summary Dump Data” on page 5-306.
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 JOBSUMMARY control statement produced. If you want to do this, use the LISTDUMP subcommand; see “LISTDUMP Subcommand — List Dumps in Dump Directory” on page 5-147. • See “SUMMARY Subcommand — Summarize Control Block Fields” on page 5-255.
TCAMMAP control statement	VERBEXIT TCAMMAP subcommand • Use the TCAMMAP verb name on the VERBEXIT subcommand to format information for ACF/TCAM problem determination. • See *Advanced Communications Function for TCAM*.
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 5-183.
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 5-268.
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*.
VSMDATA control statement	VERBEXIT VSMDATA subcommand • Use the VSMDATA 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 5-284 and “VERBEXIT VSMDATA Subcommand — Format Virtual Storage Management Data” on page 5-308.

Table E-1. AMDPRDMP - IPCS Conversion Summary (continued)

Print Dump Control Statement or Function	IPCS Equivalent
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*.

Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it 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 Volume 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.

© Copyright IBM Corp. 1988, 2002

Notices

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

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, 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.

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

© Copyright IBM Corp. 1988, 2002

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This book documents information NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- CICS
- CICS/ESA
- DB2
- DFSMSdfp
- DFSMS/MVS
- Extended Services
- IBM
- IBMLink
- IMS
- IMS/ESA
- Language Environment
- MVS
- MVS/DFP
- MVS/ESA
- MVS/SP
- NetView
- OS/390
- Resource Link
- RETAIN
- System/370
- System/390
- VTAM
- WebSphere
- zSeries
- z/Architecture
- z/OS

- z/OS.e
- 3090

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

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

Index

Special characters

* sign
  literal value notation 2-6

A

ABEND command
  to cancel IPCS processing 1-3
ABSOLUTE parameter
  in IPCS data description parameter 3-8
access register
  formatting related dump data
    IEAVD30 exit routine 5-273
    IEAVXD01 exit routine 5-273
access register data
  analyzing dumps 5-23
accessibility F-1
ACTIVE parameter
  of SETDEF IPCS subcommand 5-226
ADDDSN IPCS subcommand
  description A-1
ADDDUMP IPCS subcommand
  description 5-7
  examples 5-8
ADDPROB IPCS subcommand
  description A-4
address
  identifying where an address resides in a dump 5-313
address expression
  in IPCS data description parameter 3-3
ADDRESS parameter
  in IPCS data description parameter 3-3
address pointer entry
  renumbering 5-193
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 5-220
  identifying in dump through an STOKEN 5-172
address type
  floating-point in IPCS 3-5
  general-purpose in IPCS 3-5
  indirect in IPCS 3-6
  literal in IPCS 3-5
  relative in IPCS data description parameter 3-4
  symbolic in IPCS data description parameter 3-4
address-processing-parameter
  of SETDEF IPCS subcommand 5-226
AFT symbol
  for IPCS B-1
ALCWAIT IPCS verb name
  description 5-287
allocatable device
  analyzing dumps 5-9
allocate queue 5-20
allocation
  obtaining dump output 5-292
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 5-285
AOMDATA IPCS verb name 5-285
APPC/MVS (Advanced Program-to-Program Communications/MVS)
  APPC/MVS server 5-18
    obtaining diagnosis data 5-20
  APPC/MVS transaction scheduler
    obtaining diagnosis data 5-27
  obtaining diagnosis data 5-18
APPCDATA IPCS subcommand
  description 5-18
  SERVERDATA report 5-20
ARCHECK IPCS subcommand
  description 5-23
AREA parameter
  in IPCS data description parameter 3-13
array parameter
  description 3-15
ASCB (address space control block)
  analyzing with CBSTAT IPCS subcommand 5-37
  displaying address for an address space in a dump 5-220
  displaying using SUMMARY IPCS subcommand 5-255
ASCB exit routine
  running an installation-supplied routine 5-26
ASCBEXIT IPCS subcommand
  description 5-26
  return codes 5-27
  testing installation-supplied exits 5-231
ASCBnnnnn symbol
  for IPCS B-1
ASCHDATA IPCS subcommand
  description 5-27
ASCII character string
  notation 2-5
ASCII IPCS primary command
  description 6-6
ASID (address space identifier)
  displaying for each address space in a dump 5-220
ASID parameter
  in IPCS data description parameter 3-8

© Copyright IBM Corp. 1988, 2002

ASM (auxiliary storage manager)
analyzing data in dump 5-29
formatting data in dump 5-288
ASMCHECK IPCS subcommand
description 5-29
ASMDATA IPCS verb name
description 5-288
ASMVT symbol
for IPCS B-1
ASTEnnnnn symbol
for IPCS B-1
ASTnnnn symbol
for IPCS B-1
ASVT symbol
for IPCS B-2
ASXBnnnnn symbol
for IPCS B-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 5-288
AVMDATA IPCS verb name
description 5-288

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
BIT parameter
in IPCS data description parameter 3-13
BLOCK parameter
in IPCS data description parameter 3-8
BLS18451I message
information from WHERE IPCS subcommand or primary command 5-316
BLS9 command of TSO/E 4-2
TASKLIB parameter 4-2
TEST parameter 4-2
BLS9CALL command of TSO/E 4-3
HEADING parameter 4-3
LIBRARY parameter 4-4
MEMBER parameter 4-4
NOHEADING parameter 4-3
NOTITLE parameter 4-3
PAGE parameter 4-4
parm parameter 4-3
program parameter 4-3
STATUS parameter 4-4
SYSIN parameter 4-4
BLS9CALL command of TSO/E *(continued)*
SYSLIB parameter 4-4
SYSLIN parameter 4-4
SYSLMOD parameter 4-4
SYSPRINT parameter 4-4
SYSPUNCH parameter 4-4
SYSTERM parameter 4-4
SYSUT1 parameter 4-4
SYSUT2 parameter 4-4
SYSUT3 parameter 4-4
SYSUT4 parameter 4-5
TITLE parameter 4-3
BLSAIPST IPCS verb name
description 5-288
BLSCALTL CLIST 7-1
BLSCBSAA CLIST
description 7-2
BLSCBSAP CLIST
description 7-3
BLSCBSVA CLIST
description 7-4
BLSCBSVB CLIST
description 7-5
BLSCBSVP CLIST
description 7-5
BLSCBSYA CLIST
description 7-6
BLSCBSYB CLIST
description 7-6
BLSCBSYP CLIST
description 7-7
BLSCCOMP CLIST
use of COMPARE subcommand 5-46
BLSCDDIR CLIST
description 7-8
BLSCEDUM CLIST
description 5-96
BLSCEMAP CLIST
description 5-100
BLSCEPTR CLIST
description 7-9
BLSCESYM CLIST
description 5-105
BLSCLIBD CLIST 7-1
BLSCPCSA CLIST E-4
description 7-9
BLSCPNUC CLIST E-4
description 7-10
BLSCPRIV CLIST E-4
description 7-10
BLSCPRNT CLIST E-4
description 7-10
BLSCPSQA CLIST E-4
description 7-12
BLSCRNCH CLIST
description 5-213
BLSCSCAN CLIST
description 7-12
BLSCSETD CLIST
description 5-93

BLSJIPCS cataloged procedure
  description 8-1
  use with BLSCBSAA CLIST 7-2
  use with BLSCBSAP CLIST 7-3
  use with BLSCBSVA CLIST 7-4
  use with BLSCBSVP CLIST 7-5
  use with BLSCBSYA CLIST 7-6
  use with BLSCBSYP CLIST 7-7
BLSQXBT symbol
  for IPCS B-2
BLSQXBTnnnnn symbol
  for IPCS B-2
BLSQZBTG symbol
  for IPCS B-2
BLSXWHER REXX EXEC
  description 7-13
BROWSE option
  using CANCEL IPCS primary command 6-7
  using CBFORMAT IPCS primary command 6-7

C

CANCEL IPCS primary command
  description 6-7
cancel processing 1-3
captured UCB pages
  formatting with IOSCHECK subcommand 5-131
CB (Component Broker)
  obtaining formatted data 5-288
CBDATA IPCS verb name
  description 5-288
CBFORMAT IPCS primary command
  description 6-7
CBFORMAT IPCS subcommand
  description 5-30
  return codes 5-32
CBSTAT IPCS subcommand
  description 5-35
  return codes 5-37
CDEpgmname symbol
  for IPCS B-2
cell
  add entry to symbol table 5-214
  search a CPOOL 5-214
central processor
  displaying status in a dump 5-234
  instruction address trace 5-64
central storage
  diagnosis information 5-195
chain
  using RUNARRAY subcommand 5-208
  using RUNCHAIN subcommand 5-209
character
  symbols used in picture strings 2-4
CHARACTER parameter
  in IPCS data description parameter 3-13
character string
  notation 2-3, 2-5
CICS (Customer Information Control System)
  formatting dump data 5-285
CICSDATA IPCS verb name 5-285
CLIST
  add notes to output 5-177
  description 7-1
  identifying libraries 4-1
  invoked from IPCS dialog 6-15
  invoking with TSO subcommand of IPCS 5-282
  retrieving dump directory information 5-94
  retrieving information into variables 5-91
  retrieving storage map information into
    variables 5-97
  retrieving symbol table information into
    variables 5-100
  variables 5-105, 5-127, A-4
CLOSE subcommand
  description 5-38
  example 5-40
COMCHECK IPCS subcommand
  description 5-40
command
  syntax conventions 1-5
command code
  for IPCS inventory panel 6-5
common storage tracking function
  analyzing dumps 5-310
COMMON symbol
  for IPCS B-2
COMMTASK (communications task)
  formatting dump data using COMCHECK IPCS
    subcommand 5-40
COMPARE IPCS subcommand
  description 5-43
  example used in a CLIST 5-46
  return codes 5-46
COMPDATA parameter
  in IPCS data description parameter 3-9
component trace
  analyzing records in a dump or trace data set 5-65
  copying entries 5-56
CONFIRM parameter
  of SETDEF IPCS subcommand 5-226
console management
  formatting dump data using COMCHECK IPCS
    subcommand 5-40
contention data
  analyzing dumps 5-9
control block
  add entry to symbol table 5-209
  formatting and displaying using CBFORMAT IPCS
    subcommand 5-30
  formatting with CBFORMAT IPCS primary
    command 6-7
  IPCS symbol C-1
  key fields from dump
    using SUMMARY IPCS subcommand 5-255
  list of control blocks scanned, mapped and formatted
    by IPCS D-1
  search a chain 5-209
  search an array 5-208
  status using CBSTAT IPCS subcommand 5-35
  TCB-related 5-272

control block *(continued)*
validating using the SCAN IPCS subcommand 5-218
control register
formatting in a dump 5-236
copy
CTRACE entries 5-56
GTF records 5-56
trace entries 5-56
COPYCAPD IPCS subcommand
description 5-46
return codes 5-48
COPYDDIR IPCS subcommand
description 5-49
COPYDUMP IPCS subcommand
description 5-51
return codes 5-55
COPYTRC IPCS subcommand
description 5-56
return codes 5-60
COUPLE IPCS subcommand
description 5-60
coupling facility
formatting structure data 5-245
CPOOL
using RUNCPOOL subcommand 5-214
CPU parameter
in IPCS data description parameter 3-9
CPUTRACE IPCS subcommand
description 5-64
cross system extended services
formatting dump information 5-323
CSA (common storage area)
common storage tracking function analyzing dumps 5-310
print storage from a dump 7-9
CSA symbol
for IPCS B-2
CSD data area
formatting with CBFORMAT IPCS subcommand 5-32
CSD symbol
for IPCS B-2
CSECT
information from WHERE IPCS subcommand or primary command 5-315
CTRACE IPCS subcommand
description 5-65
merging formatted trace entries 5-170
current address
in a dump formatted by IPCS B-6
X symbol in IPCS 3-4
CURSOR symbol
for IPCS B-2
CVT (communication vector table)
formatting with CBFORMAT IPCS subcommand 5-32
CVT symbol
for IPCS B-2
CVTVSTGX symbol
for IPCS B-2
CVTXTNT2 symbol
for IPCS B-2

D

D IPCS line command
description 6-28
DAE (dump analysis and elimination)
formatting data in dump 5-289
DAEDATA IPCS verb name
description 5-289
DAESYMPTOMS symbol
for IPCS B-2
DAT-off nucleus
print storage from a dump 7-10
data area
list of data areas scanned, mapped and formatted by IPCS D-1
validating using the SCAN IPCS subcommand 5-218
data description parameter
address processing parameters 3-7
array parameters 3-15
attribute parameters 3-13
description 3-1
NOREMARK parameter 3-16
REMARK parameter 3-16
data entry panel
IPCS and ISPF primary commands and PF keys 6-3
data management control blocks
formatting related dump data IECDAFMT exit routine 5-272
data set
close to IPCS processing 5-38
dump directory 1-2
obtaining contents summary report 5-147
opening for IPCS processing 5-183
data space
describing in a dump 3-1
displaying 3-10
identifying in dump through an STOKEN 5-172
data-in-virtual
formatting dump data 5-76
DATASET parameter
of SETDEF IPCS subcommand 5-227
DATOFFNUCLEUS symbol
for IPCS B-2
DB2 (DATABASE 2)
formatting dump data 5-285
DDNAME parameter
of SETDEF IPCS subcommand 5-227
decimal number
converting to hexadecimal 5-127
notation on subcommands 2-2
default value
displaying IPCS default values using SETDEF IPCS subcommand 5-223
PROFILE-defined 5-190
setting IPCS default values using SETDEF IPCS subcommand 5-223

delayed issue queue
branch entry and NIP time messages 5-299
DELDSN IPCS subcommand
description A-11
deletion
storage map records 5-85
DELPROB IPCS subcommand
description A-13
device allocation data
obtaining formatted output 5-287
DFP (Data Facility Product)
formatting dump data 5-285
diagnostic worksheet
displaying from dump 5-234
DIB symbol
for IPCS B-2
DIBX symbol
for IPCS B-2
DIMENSION parameter
in IPCS data description parameter 3-15
disability F-1
displaying operation code
using OPCODE IPCS primary command 6-20
DIVDATA IPCS subcommand
description 5-76
DLF (data lookaside facility)
obtaining diagnosis data 5-80
DLFDATA IPCS subcommand
description 5-80
documents, licensed xiv
DOMAIN parameter
in IPCS data description parameter 3-9
DOWN IPCS primary command
description 6-7
DROPDUMP IPCS subcommand
description 5-82
examples 5-84
DROPMAP IPCS subcommand
description 5-85
DROPSYM IPCS subcommand
description 5-86
examples 5-88
DSNAME parameter
of SETDEF IPCS subcommand 5-227
DSNWDMP IPCS verb name 5-285
dump
analyzing
access register data 5-23
data-in-virtual data 5-76
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 5-65
analyzing system status 5-234
close to IPCS processing 5-38
copying 5-46, 5-51
dump *(continued)*
data comparison 5-43
displaying storage 5-145
displaying symbol definitions 5-159
displaying title in dump data set 4-7
extracting a single dump from a string of dumps 5-51
instruction address trace 5-64
multiple dumps in a single data set 5-51
printing 7-10
reducing the size of a dump 5-51
retrieving data into variables 5-105
retrieving dump directory information 5-94
suppressing output 5-232
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 5-82
displaying list of sources 5-147
example of creating and initializing 1-2
freeing space 5-86
initialize using IPCSDDIR command 4-6
opening for IPCS processing 5-183
retrieving information 5-94
dump display reporter panel
IPCS and ISPF primary commands and PF keys 6-4
dump header
formatting title 5-146
dump title
displaying using LIST IPCS subcommand 5-145
displaying using SYSDSCAN command 4-7
formatting 5-146
obtaining a summary of dump titles 5-51
DUMPINGPROGRAM symbol
for IPCS B-2
DUMPORIGINALDSNAME symbol
for IPCS B-2
DUMPREQUESTOR symbol
for IPCS B-2
DUMPTIMESTAMP symbol
for IPCS B-2
DUMPTOD symbol
for IPCS B-3
dynamic configuration change
displaying EDT data 5-154

E

E IPCS line command
description 6-29
EBCDIC IPCS primary command
description 6-8

ECSA (extended common storage area)
  print storage from a dump 7-9
ECSA symbol
  for IPCS B-3
EDT (eligible device table)
  displaying data 5-154
  primary EDT 5-154
  secondary EDT 5-154
EFLPA symbol
  for IPCS B-3
EMLPA symbol
  for IPCS B-3
END IPCS primary command
  description 6-9
END IPCS subcommand
  description 5-88
  return codes 5-88
ENQ contention
  analyzing dumps 5-9
ENTRY parameter
  in IPCS data description parameter 3-15
ENUCLEUS symbol
  for IPCS B-3
EPLPA symbol
  for IPCS B-3
EQUATE IPCS primary command
  description 6-9
EQUATE IPCS subcommand
  return codes 5-89
ERROR parameter
  of SETDEF IPCS subcommand FLAG
    parameter 5-230
ERRORID symbol
  for IPCS B-3
ESQA (extended global system queue area)
  print storage from a dump 7-12
ESQA symbol
  for IPCS B-3
EVALDEF IPCS subcommand
  description 5-91
  example used in a CLIST 5-93
EVALDUMP IPCS subcommand
  description 5-94
  example used in a CLIST 5-96
EVALMAP IPCS subcommand
  description 5-97
  example used in a CLIST 5-100
EVALSYM IPCS subcommand
  description 5-100
  example used in a CLIST 5-105
EVALUATE IPCS subcommand
  description 5-105
  return codes
    CLIST, REXX, or DIALOG return codes 5-108
exit routine
  for analyzing ASCBs in a dump 5-26
  invoking
    using ASCBEXIT IPCS subcommand 5-26
  invoking TCB exit routine
    using TCBEXIT IPCS subcommand 5-272
exit routine *(continued)*
  invoking using VERBEXIT IPCS
    subcommands 5-284
  testing installation-supplied exits 5-231
exit service
  activating traps 5-278
  deactivating traps 5-276
  listing status of traps 5-274
extended MCS console
  formatting dump data using COMCHECK IPCS
    subcommand 5-40
extended private storage area
  printing storage from a dump 7-10
extended read-write nucleus
  print storage from a dump 7-10
external interrupt
  formatting GTF trace records 5-123

F

F IPCS line command
  description 6-31, 6-34
F parameter
  in IPCS data description parameter 3-13
FILE parameter
  of SETDEF IPCS subcommand 5-227
FIND IPCS primary command
  description 6-10
  example using quoted-string notation 2-5
FIND IPCS subcommand
  description 5-109
  example using picture strings 2-4
FINDAREA symbol 5-109
  for IPCS B-3
FINDMOD IPCS subcommand
  description 5-114
FINDUCB IPCS subcommand
  description 5-115
FLAG parameter
  of SETDEF IPCS subcommand 5-229
floating-point address type
  in IPCS data description parameter 3-5
FLPA symbol
  for IPCS B-3
full-screen display 6-1
  using IPCS commands 6-1
fullword pointer
  notation 2-3

G

GDA symbol
  for IPCS B-3
general purpose register
  formatting in a dump 5-236
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

general value *(continued)*
hexadecimal string notation 2-6
notation types 2-3
picture string notation 2-4
quoted string notation 2-5
text string notation 2-6
word notation 2-6
general-purpose address type
in IPCS data description parameter 3-5
global resource serialization
formatting queue information 5-117
obtaining dump output 5-291
GO IPCS subcommand
description 5-117
GRSDATA IPCS subcommand
description 5-117
GRSTRACE IPCS verb name
description 5-291
GTF (generalized trace facility)
copying trace records 5-56
formatting trace records 5-120
GTFTRACE IPCS subcommand
description 5-120
merging formatted trace entries 5-170

H

HASMFMTM IPCS verb name 5-285
HEADER parameter
in IPCS data description parameter 3-10
HEADING parameter of BLS9CALL command 4-3
HELP IPCS subcommand
description 5-126
hexadecimal number
converting to decimal 5-127
notation on subcommands 2-2
HEXADECIMAL parameter
in IPCS data description parameter 3-13
hexadecimal string
notation 2-6

I

I IPCS line command
description 6-31
I/O contention
analyzing dumps 5-9
IBM status attribute A-21
IEAVD30 exit routine
running 5-273
IEAVESLA symbol
for IPCS B-3
IEAVTFMT exit routine
running 5-273
IEAVXD01 exit routine
running 5-273
IECDAFMT exit routine
running 5-272
IECIOFMT exit routine
running 5-273
IEFIVAWT IPCS verb name
description 5-292
IEFIVIGD IPCS verb name
description 5-292
IEFJESCTPX symbol
for IPCS B-3
IEFRDER file 7-2, 7-3
IEFZB445 symbol
for IPCS B-3
IHSAnnnnn symbol
for IPCS B-3
IKJEFT01 program
invoked by BLSJIPCS cataloged procedure 8-1
IMS (Information Management System)
formatting dump data 5-285
IMSDUMP IPCS verb name 5-285
INCIDENTTOKEN symbol
for IPCS B-3
indirect address
in IPCS data description parameter 3-6
information
online help 5-126
INFORMATIONAL parameter
of SETDEF IPCS subcommand FLAG
parameter 5-230
instruction address trace
formatting trace entries 5-64
integer
converting decimal to hexadecimal 5-127
converting hexadecimal to decimal 5-127
positive value notations 2-2
signed value notations 2-2
INTEGER IPCS subcommand
description 5-127
inventory panel
IPCS command codes 6-5
IOS (input/output supervisor)
formatting related dump data
IECIOFMT exit routine 5-273
IOSCHECK IPCS subcommand 5-129
IOSCHECK IPCS subcommand
description 5-129
IPCS (interactive problem control system)
access line mode 1-2
batch mode processing 8-1
description of subcommands 5-1
directing output 1-2
end a session 5-88
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 1-5
task directory for subcommands 5-2
TSO/E commands 4-1
IPCS command of TSO/E
description 4-5
PARM parameter 4-5

IPCS command of TSO/E *(continued)*
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 7-5
SYSMDUMP dump analysis 7-6
IPCS primary command
description 6-15
IPCS subcommand
invoking from IPCS dialog 6-15
IPCSDATA IPCS subcommand
description 5-135
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 5-183
output for COPYDDIR IPCS subcommand 5-49
IPCSINDD ddname 5-51
IPCSPRnn parmlib member
specified on IPCS command 4-5
IPCSPRNT data set
add notes to output 5-177
opening for IPCS processing 5-183
output from BLSCBSAA CLIST 7-2
output from BLSCBSAP CLIST 7-3
output from BLSCBSVA CLIST 7-4
output from BLSCBSVP CLIST 7-5
output from BLSCBSYA CLIST 7-6
output from BLSCBSYP CLIST 7-7
routing subcommand listing 5-230
IPCSTOC data set
opening for IPCS processing 5-183
IPLDATA IPCS subcommand
description 5-143
IRLM IPCS verb name 5-285
ISGGVT symbol
for IPCS B-3
ISGGVTX symbol
for IPCS B-3
ISGQHTG symbol
for IPCS B-3
ISGQHTL symbol
for IPCS B-3
ISGRSV symbol
for IPCS B-3
ISPEXEC IPCS subcommand
description 5-144
ISPF (Interactive System Productivity Facility)
function pool dialog variables 5-97
retrieving dump directory information 5-94
ISPF (Interactive System Productivity Facility) *(continued)*
function pool dialog variables *(continued)*
retrieving session default values into variables 5-91
retrieving storage map information into variables 5-97
retrieving symbol table information into variables 5-100, 5-105
invoking under IPCS 5-282
request a dialog service 5-144
request IPL reports 5-143
ITTCTAB symbol
for IPCS B-3

J

JES XCF component
formatting dump data 5-285
obtaining dump output 5-293
JES2 subsystem
formatting dump data 5-285
JES3 IPCS verb name 5-285
JES3 subsystem
formatting dump data 5-285
JESXCF IPCS verb name 5-285
description 5-293
job
displaying name for address space in a dump 5-220

K

key
PF
used in IPCS dialog 6-2
keyboard F-1
keyword parameter
data description 3-1
description for IPCS 1-5

L

L IPCS line command
description 6-34
latch contention
analyzing dumps 5-15
LCCAnn symbol
for IPCS B-3
LCCAVT symbol
for IPCS B-4
LE (Language Environment)
formatting dump data 5-285
obtaining formatted data 5-294
LE IPCS verb name 5-285
LEDATA IPCS verb name
description 5-294
LEFT IPCS primary command
description 6-16
LENGTH parameter
in IPCS data description parameter 3-2, 3-6

LENGTH parameter *(continued)*
of SETDEF IPCS subcommand 5-230
LIBRARY parameter of BLS9CALL command 4-4
licensed documents xiv
line command
IPCS dialog 6-1
task directory 6-2
removal using RESET primary command 6-21
line size
adjusting 5-190
list
dump title 5-145
LIST IPCS subcommand
description 5-145
LISTDSN IPCS subcommand
description A-15
LISTDUMP IPCS subcommand
description 5-147
LISTEDT IPCS subcommand
description 5-154
example 5-157
LISTMAP IPCS subcommand
description 5-157
example 5-159
LISTPROB IPCS subcommand
description A-16
LISTSYM IPCS subcommand
description 5-159
examples 5-161
LISTTOD IPCS subcommand
description 5-162
example 5-163
LISTUCB IPCS subcommand
description 5-163
literal
assign value 5-165
literal address
in IPCS data description parameter 3-5
LITERAL IPCS subcommand
description 5-165
literal value
finding in a dump 5-109
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 5-315
load module library
specify search order for IPCS analysis programs 4-5
LOCATE IPCS primary command
description 6-17
LOGDATA IPCS verb name
description 5-296
LOGGER IPCS subcommand
description 5-167
logrec
obtaining formatted output from a dump 5-296
LookAt message retrieval tool xiv
lowercase letter
symbol used in picture strings 2-4
LPA (link pack area)
list entry points in a dump 5-167
LPAMAP IPCS subcommand
description 5-167
dump output 5-169
LPDEpgmname symbol
for IPCS B-4

M

MACHINE parameter
of SETDEF IPCS subcommand DISPLAY parameter 5-228
MAIN parameter
of SETDEF IPCS subcommand 5-226
master trace
formatting table from a dump 5-299
MCS console
formatting dump data using COMCHECK IPCS subcommand 5-40
MEMBER parameter of BLS9CALL command 4-4
MERGE IPCS subcommand
description 5-170
MERGEEND IPCS subcommand
description 5-170
message
eliminating from IPCS reports 5-229
filtering by severity 5-229
generating using NOTE IPCS subcommand 5-177
issued during an IPCS session 1-4
message queue
formatting dump data using COMCHECK IPCS subcommand 5-40
message retrieval tool, LookAt xiv
MLPA (modified link pack area)
list entry points in a dump 5-167
MLPA symbol
for IPCS B-4
MMS (MVS message service)
obtaining dump output 5-299
MMSDATA IPCS verb name
description 5-299
MODDSN IPCS subcommand
description A-25
MODPROB IPCS subcommand
description A-28
module
find in a dump 5-114
MODULE parameter
in IPCS data description parameter 3-14
MORE IPCS primary command
description 6-20
MTRACE IPCS verb name
description 5-299
MULTIPLE parameter
in IPCS data description parameter 3-15

N

name
  for IPCS symbols B-1
NAME IPCS subcommand
  description 5-172
NAMETOKN IPCS subcommand
  description 5-174
NIP hardcopy message buffer 5-299
NOCONFIRM parameter
  of SETDEF IPCS subcommand 5-226
NOCPU parameter
  in IPCS data description parameter 3-10
NOHEADING parameter of BLS9CALL command 4-3
NOLIST parameter
  on the NAME subcommand 5-173
  on the NAMETOKEN subcommand 5-175
NOMACHINE parameter
  of SETDEF IPCS subcommand DISPLAY
    parameter 5-228
non-blank character
  symbol used in picture strings 2-4
non-numeric character
  symbol used in picture strings 2-4
NOPRINT parameter
  of SETDEF IPCS subcommand 5-230
NOPROBLEM parameter
  of SETDEF IPCS subcommand 5-231
NOREMARK parameter
  description 3-16
  in IPCS data description parameter 3-16
  of SETDEF IPCS subcommand DISPLAY
    parameter 5-228
NOREQUEST parameter
  of SETDEF IPCS subcommand DISPLAY
    parameter 5-229
NOSTORAGE parameter
  of SETDEF IPCS subcommand DISPLAY
    parameter 5-229
NOSUMMARY parameter
  on the COPYDDIR subcommand 5-50
  on the DROPDUMP subcommand 5-83
  on the DROPMAP subcommand 5-85
  on the DROPSYM subcommand 5-87
  on the LISTDUMP subcommand 5-148
  on the LISTMAP subcommand 5-159
NOSYMBOL parameter
  of SETDEF IPCS subcommand DISPLAY
    parameter 5-229
NOTE IPCS subcommand
  description 5-177
NOTERMINAL parameter
  of SETDEF IPCS subcommand 5-231
NOTEST parameter
  of SETDEF IPCS subcommand 5-231
Notices G-1
NOTITLE parameter of BLS9CALL command 4-3
NOVERIFY parameter
  of SETDEF IPCS subcommand 5-232
nucleus
  obtaining formatted output 5-302
  print storage from a dump 7-10
NUCLEUS symbol
  for IPCS B-4
NUCMAP IPCS verb name
  description 5-302
number
  converting decimal to hexadecimal 5-127
  converting hexadecimal to decimal 5-127
numeric character
  symbol used in picture strings 2-4
numeric comparison
  by COMPARE IPCS subcommand 5-43
NVT symbol
  for IPCS B-4

O

OMVSDATA IPCS subcommand
  description 5-180
online help
  for IPCS subcommands 5-126
OPCODE IPCS primary command
  description 6-20
OPCODE IPCS subcommand
  description 5-182
OPEN IPCS subcommand
  description 5-183
OPEN/CLOSE/EOV
  formatting GTF trace records 5-124
operator communications activity 5-40
output
  suppressing IPCS screen output lines
    using X line command 6-37
output routing
  SETDEF IPCS subcommand parameters 5-230

P

PAGE parameter of BLS9CALL command 4-4
page size
  adjusting 5-190
parameter
  in IPCS data description 3-1
  types in IPCS 1-5
parameter string
  format when passed to verb exit routine 5-286
parm parameter of BLS9CALL command 4-3
PARM parameter of IPCS command 4-5
PART symbol
  for IPCS B-4
PATCH subcommand 5-186, 5-190
PCCAnn symbol
  for IPCS B-4
PCCAVT symbol
  for IPCS B-4
PF key
  used in IPCS dialog 6-2
PFT symbol
  for IPCS B-4
pgmname symbol
  for IPCS B-4

PGTnnnnnaaaaa symbol
for IPCS B-4
picture string
classes of characters 2-4
notation 2-4
PLPA (pageable link pack area)
list entry points in a dump 5-167
PLPA symbol
for IPCS B-4
pointer
displaying using LOCATE IPCS primary command 6-17
duplicating using R IPCS line command 6-33
edit using E IPCS line command 6-29
formatting under the IPCS BROWSE option 6-31
inserting using I IPCS line command 6-31
selecting using S IPCS line command 6-34
selecting using SELECT IPCS primary command 6-22
sorting using SORT IPCS primary command 6-23
stack symbol, Znnnnn B-6
pointer panel
IPCS and ISPF primary commands, IPCS line commands, and PF keys 6-3
POINTER parameter
in IPCS data description parameter 3-14
pointer stack
renumbering entries 5-193
renumbering symbols in IPCS BROWSE 6-20
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 E-1
primary command
invoking for IPCS through PF keys 6-2
IPCS 6-1
IPCS dialog 6-1
task directory 6-2
removal using RESET primary command 6-21
PRIMARYSYMPTOMS symbol
for IPCS B-4
print data set
opening for IPCS processing 5-183
routing subcommand listing 5-230
print dump
conversion to IPCS E-1
PRINT parameter 1-2
of SETDEF IPCS subcommand 5-230
printed report
setting line and page size 5-190
printing
storage areas from a dump 7-10
private storage area
printing 7-10
PRIVATE symbol
for IPCS B-4
PRIVATEX symbol
for IPCS B-4
problem determination
displaying system status from dump 5-234
PROBLEM parameter
of SETDEF IPCS subcommand 5-231
PROFILE IPCS subcommand
description 5-190
example 5-193
program interrupt
formatting GTF trace records 5-124
program parameter of BLS9CALL command 4-3
PSAnn symbol
for IPCS B-4
PSAVALID symbol
for IPCS B-4
PSW symbol
for IPCS B-4
PSWnn symbol
for IPCS B-4
PVT symbol
for IPCS B-4

Q

Q IPCS verb name
description 5-291
QCBTRACE IPCS verb name
description 5-291
quoted string
notation 2-5

R

R IPCS line command
description 6-33
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 5-255
RBA parameter
in IPCS data description parameter 3-10
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

REGACC symbol
for IPCS B-4
REGACCnn symbol
for IPCS B-5
REGCTL symbol
for IPCS B-5
REGCTLnn symbol
for IPCS B-5
REGFLT symbol
for IPCS B-5
REGFLTnn symbol
for IPCS B-5
REGFPC symbol
for IPCS B-5
REGFPCnn symbol
for IPCS B-5
REGGEN symbol
for IPCS B-5
REGGENnn symbol
for IPCS B-5
relative address
in IPCS data description parameter 3-4
REMARK parameter
description 3-16
in IPCS data description parameter 3-16
of SETDEF IPCS subcommand DISPLAY parameter 5-228
REMOTEDUMP symbol
for IPCS B-5
RENUM IPCS primary command
description 6-20
RENUM IPCS subcommand
description 5-193
REQUEST parameter
of SETDEF IPCS subcommand DISPLAY parameter 5-229
RESET IPCS primary command
description 6-21
resource contention data
analyzing dumps 5-9
return code
for most IPCS subcommands 5-2
from the EVALUATE subcommand
CHECK option 5-109
default option 5-109
RETURN IPCS primary command
description 6-21
REXX exec
description 7-1
format a value in a variable 5-127
identifying libraries 4-1
invoked from IPCS dialog 6-15
invoking with TSO subcommand of IPCS 5-282
retrieving dump directory information 5-94
retrieving information into variables 5-91
retrieving storage map information into variables 5-97
retrieving symbol table information into variables 5-100
variables 5-94, 5-97, 5-127, A-4
RFIND IPCS primary command
description 6-21
RIGHT IPCS primary command
description 6-21
RIM (resource initialization manager)
analyzing with CBSTAT IPCS subcommand 5-37
RONUCLEUS symbol
for IPCS B-5
RSM (real storage manager)
displaying diagnostic information 5-195
RSMDATA IPCS subcommand
description 5-195
report/parameter matrix 5-199
RTCT symbol
for IPCS B-5
RTM (recovery termination manager)
formatting related dump data
IEAVTFMT exit routine 5-273
running 5-273
RUNARRAY IPCS subcommand
description 5-208
RUNCHAIN IPCS subcommand
description 5-209
example used in a CLIST 5-213
RUNCPOOL IPCS subcommand
description 5-214

S

S IPCS line command (select)
description 6-34
S IPCS line command (show)
description 6-34
SADMP message log
obtaining formatted output 5-305
SADMPMSG IPCS verb name
description 5-305
SAM/PAM/DAM
formatting GTF trace records 5-124
save area
CLIST to follow the forward chain 7-9
SCALAR parameter
in IPCS data description parameter 3-16
SCAN IPCS subcommand
description 5-218
SCCB symbol
for IPCS B-5
screen output
displaying excluded lines
using S, F, or L IPCS line command 6-34
for IPCS
using D IPCS line command 6-28
suppressing lines
using X IPCS line command 6-37
scrolling dump data
by changing scroll amount field 6-7
using DOWN IPCS primary command 6-7
using LEFT IPCS primary command 6-16
using LOCATE IPCS primary command 6-17
using MORE IPCS primary command 6-20
using RIGHT IPCS primary command 6-21

scrolling dump data *(continued)*
using UP IPCS primary command 6-25
SCVT symbol
for IPCS B-5
SDWAHDR symbol
for IPCS B-5
search
for a module in a dump 5-114
for a value
using FIND IPCS primary command 6-10
repeat a search
using RFIND IPCS primary command 6-21
search value
repeat 2-6
SECONDARYSYMPTOMS symbol
for IPCS B-5
SELECT IPCS primary command
description 6-22
SELECT IPCS subcommand
description 5-220
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 5-230
SERVERDATA report 5-20
session default value
displaying IPCS default values using SETDEF IPCS subcommand 5-223
retrieving information into variables 5-91
setting IPCS default values using SETDEF IPCS subcommand 5-223
SETDEF IPCS subcommand
description 5-223
SEVERE parameter
of SETDEF IPCS subcommand FLAG parameter 5-230
SGTnnnnn symbol
for IPCS B-5
shortcut keys F-1
SHRDATA IPCS subcommand
description 5-198
signed integer
binary notation 2-2
hexadecimal notation 2-2
notation types 2-2
SIGNED parameter
in IPCS data description parameter 3-13
SLIP trace records
formatting GTF trace records 5-124
SLIPTRAP symbol
for IPCS B-5
SMF (system management facilities)
displaying diagnostic information 5-232
SMFDATA IPCS subcommand
description 5-232
SMS (Storage Management Subsystem)
formatting dump data 5-285
SMSDATA IPCS verb name 5-285
SMSXDATA verb name 5-285
SORT IPCS primary command
description 6-23
source
types for IPCS processing 1-1
spin loop
formatting trace entries 5-64
SQA (global system queue area)
print 7-12
SRM (system resources manager)
formatting GTF trace records 5-124
obtaining control blocks from dump 5-305
SRMDATA IPCS verb name
description 5-305
SSIDATA IPCS subcommand
description 5-233
stack entry
controlling duplication 5-190
creating 5-233
STACK IPCS primary command
description 6-24
STACK IPCS subcommand
description 5-233
stand-alone dump
CLIST for initial analysis
description 7-12
printed 7-2
CLIST to print storage 7-3
source for IPCS processing 1-1
stand-along dump
formatting message log from dump 5-305
STATUS IPCS subcommand
description 5-234
STATUS parameter
in IPCS data description parameter 3-10
STATUS parameter of BLS9CALL command 4-4
STOKEN (space token)
decipher using NAME subcommand 5-172
storage
describe in a dump 3-1
displaying from dump 5-145
obtaining contents summary report 5-147
printing from a dump 7-10
storage area
IPCS symbol C-1
storage display
using LOCATE IPCS primary command 6-17
storage key
displaying current default values 5-223
setting IPCS default values using SETDEF IPCS subcommand 5-223
storage map
creating entries for address spaces 5-220
deleting records using DROPMAP IPCS subcommand 5-85
displaying entries 5-157
list of entries created for control blocks or data areas by IPCS D-1
retrieving information into variables 5-97

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 5-226
of SETDEF IPCS subcommand DISPLAY parameter 5-229
STRDATA IPCS subcommand
description 5-245
string comparison
by COMPARE IPCS subcommand 5-44
STRUCTURE parameter
in IPCS data description parameter 3-14
information from WHERE IPCS subcommand or primary command 5-316
subcommand
IPCS online help 5-126
literal values 2-1
of IPCS 5-1
subcommand listing
routing to print data set 5-230
subspace
identifying in dump through an STOKEN 5-172
subsystem information
displaying through IPCS 5-233
SUMDUMP IPCS verb name
description 5-306
SUMDUMP parameter
in IPCS data description parameter 3-11
SUMMARY IPCS subcommand
description 5-255
suppress
IPCS screen output
using X line command 6-37
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 5-306
symbol
creating IPCS-defined
using STACK IPCS primary command 6-24
using STACK IPCS subcommand 5-233
creating user-defined
using EQUATE IPCS primary command 6-9
using EQUATE IPCS subcommand 5-89
deleting from the symbol table 5-86
displaying definitions for a dump 5-159
displaying using SYMDEF IPCS subcommand 5-267
for dump storage areas C-1
IPCS naming conventions B-1
IPCS special symbols C-1
renumbering pointers in IPCS BROWSE 6-20
X (current address) in IPCS 3-4
SYMBOL parameter
of SETDEF IPCS subcommand DISPLAY parameter 5-229
symbol table
add entries for a cell chain 5-214
add entries for a control block chain 5-209
adding symbol 5-233
deleting entries 5-86
displaying for IPCS 5-267
IPCS naming conventions B-1
renumbering stack symbol entries 5-193
retrieving entry information into variables 5-100
retrieving information through EVALSYM IPCS subcommand 5-100
symbols established by BLSCEPTR CLIST 7-9
symbolic address
in IPCS data description parameter 3-4
SYMDEF IPCS subcommand
description 5-267
SYMPTOM IPCS verb name
description 5-307
symptom string
obtaining formatted output 5-307
SYMPTOMS IPCS verb name
description 5-307
syntax
conventions in this book 1-5
SYS1.DUMPnn data set
clear E-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
SYSLIN parameter of BLS9CALL command 4-4
SYSLMOD parameter of BLS9CALL command 4-4
SYSMDUMP dump
CLIST for initial analysis 7-6
printed form 7-6
CLIST to print storage 7-7
source for IPCS processing 1-1
sysplex
obtaining status of systems from a dump 5-60
sysplex dump directory
adding source description 5-7
copying source description 5-49
creating with BLSCDDIR CLIST 7-8
deleting record 5-82
displaying list of sources 5-147
initialize using IPCSDDIR command 4-6
retrieving information 5-94
SYSPRINT parameter of BLS9CALL command 4-4
SYSPUNCH parameter of BLS9CALL command 4-4
system console
formatting dump data using COMCHECK IPCS subcommand 5-40
system data
validating using the SCAN IPCS subcommand 5-218

system initialization data
obtaining formatted output 5-288
system logger address space
dump 5-167
system status
displaying in a dump 5-234
system trace
formatting entries 5-268
SYSTERM parameter of BLS9CALL command 4-4
SYSTRACE IPCS subcommand
description 5-268
SYSUT1 parameter of BLS9CALL command 4-4
SYSUT2 parameter of BLS9CALL command 4-4
SYSUT3 parameter of BLS9CALL command 4-4
SYSUT4 parameter of BLS9CALL command 4-5

T

task directory
IPCS line commands 6-2
IPCS primary commands 6-2
IPCS subcommands 5-2
TSO/E commands for IPCS 4-1
TASKLIB parameter of IPCS command 4-2, 4-5
TCAM (Telecommunications Access Method)
formatting dump data 5-285
TCAMMAP IPCS verb name 5-285
TCB (task control block)
analyzing with CBSTAT IPCS subcommand 5-37
CLIST to run the save area chain 7-9
displaying using SUMMARY IPCS
subcommand 5-255
TCB exit routine
IBM-supplied exit routine 5-272
installation-supplied exit routine 5-272
TCBCURRENT symbol
for IPCS B-6
TCBEXIT IPCS subcommand
description 5-272
return codes 5-273
testing installation-supplied exits 5-231
TCBnnnnnaaaaa symbol
for IPCS B-6
TDCM control block
formatting dump data using COMCHECK IPCS
subcommand 5-40
TERMINAL parameter 1-2
of SETDEF IPCS subcommand 5-231
TERMINATING parameter
of SETDEF IPCS subcommand FLAG
parameter 5-230
TEST parameter
of SETDEF IPCS subcommand 5-231
TEST parameter of BLS9 command 4-2
TEST TSO/E command
testing ASCBEXIT, TCBEXIT, or VERBEXIT
subcommands 5-231
text string
notation 2-6
TIME command
during IPCS attention processing 1-3, 1-4
time-of-day clock
formatting from a dump 5-234
title
displaying using LIST IPCS subcommand 5-145
TITLE parameter of BLS9CALL command 4-3
TITLE symbol
for IPCS B-6
TOC (table of contents) data set
opening for IPCS processing 5-183
TOC entry
generating 5-177
trace
formatting GTF records 5-120
formatting master trace table 5-299
merging formatted entries 5-170
trace data set
analyzing component trace entries 5-65
source for IPCS processing 1-1
trace entry
formatting instruction address trace 5-64
trap
activating for IPCS exit service routines 5-278
deactivating for IPCS exit service routines 5-276
listing status for IPCS exit service routines 5-274
trap processing
resume 5-117
TRAPLIST IPCS subcommand
description 5-274
TRAPOFF IPCS subcommand
description 5-276
TRAPON IPCS subcommand
description 5-278
TSO subcommand of IPCS
description 5-281
return codes 5-283
TSO/E (Time Sharing Option Extensions)
commands for IPCS 4-1
enter commands from an IPCS session 5-282
formatting dump data 5-285
invoking commands from IPCS 4-1
IPCS command 4-5
IPCSDDIR command 4-6
SYSDSCAN command 4-7
TSO/E ALTLIB command 4-1
TSO/E BLS9 command 4-2
TSO/E BLS9CALL command 4-3
using from an IPCS session 5-281
TSO/E command
authorized command
running in an IPCS environment 5-283
for IPCS
task directory 4-1
TSODATA IPCS verb name 5-285
TTR parameter
in IPCS data description parameter 3-11

U

UCB (unit control block)
displaying using LISTUCB IPCS
subcommand 5-163

UCB (unit control block) *(continued)*
finding with FINDUCB IPCS subcommand 5-115
formatting captured UCB pages with IOSCHECK 5-131
formatting with CBFORMAT IPCS subcommand 5-32
UCBdddd symbol
for IPCS B-6
UCM symbol
for IPCS B-6
UNSIGNED parameter
in IPCS data description parameter 3-14
UP IPCS primary command
description 6-25
uppercase letter
symbol used in picture strings 2-4
user dump directory
adding source description 5-7
copying source description 5-49
creating with BLSCDDIR CLIST 7-8
deleting record 5-82
displaying list of sources 5-147
freeing space 5-86
initialize using IPCSDDIR command 4-6
retrieving information 5-94

V

value
finding literal values in a dump 5-109
general notations 2-3
repeat a search
using RFIND IPCS primary command 6-21
searching for using FIND IPCS primary command 6-10
Vector Facility
displaying registers 5-234
formatting related dump data
IEAVSSA1 exit routine 5-273
verb exit routine
invoking using VERBEXIT IPCS subcommands 5-284
VERBEXIT ALCWAIT IPCS subcommand
description 5-287
VERBEXIT ASMDATA IPCS subcommand
description 5-288
VERBEXIT AVMDATA IPCS subcommand
description 5-288
VERBEXIT BLSAIPST IPCS subcommand
description 5-288
VERBEXIT CBDATA IPCS subcommand
description 5-288
VERBEXIT DAEDATA IPCS subcommand
description 5-289
VERBEXIT IPCS subcommand
description 5-284
return codes 5-287
testing installation-supplied exits 5-231
VERBEXIT LEDATA IPCS subcommand
description 5-294
VERBEXIT LOGDATA IPCS subcommand
description 5-296
dump output 5-296
VERBEXIT MMSDATA IPCS subcommand
description 5-299
VERBEXIT MTRACE IPCS subcommand
description 5-299
dump output 5-300
VERBEXIT NUCMAP IPCS subcommand
description 5-302
dump output 5-302
VERBEXIT SADMPMSG IPCS subcommand
description 5-305
dump output 5-305
VERBEXIT SRMDATA IPCS subcommand
description 5-305
VERBEXIT SUMDUMP IPCS subcommand
description 5-306
dump output 5-306
VERBEXIT SYMPTOM IPCS subcommand
description 5-307
dump output 5-307
VERBEXIT VSMDATA IPCS subcommand
description 5-308
VERIFY parameter
of SETDEF IPCS subcommand 5-232
VLF (virtual lookaside facility)
obtaining diagnosis data for 5-311
VLFDATA IPCS subcommand
description 5-311
VSAM (virtual storage access method)
formatting GTF trace records 5-124
VSAM data set
access as relative byte address group 3-10
access by control interval 3-8
access data portion of cluster 3-11
access index portion of cluster 3-11
source for IPCS processing 1-1
VSM (virtual storage management)
obtaining formatted data 5-308
VSMDATA IPCS verb name
description 5-308
VTAM (Virtual Telecommunications Access Method)
format dump data 5-285
formatting GTF trace records 5-124
VTAMMAP IPCS verb name 5-285

W

wait state message area
displaying using STATUS IPCS subcommand 5-234
WARNING parameter
of SETDEF IPCS subcommand FLAG parameter 5-230
WHERE IPCS primary command
description 6-26
WHERE IPCS subcommand
description 5-313
WLM (workload manager)
obtaining diagnosis data 5-321

WLMDATA IPCS subcommand
  description 5-321
word notation 2-6
worksheet
  diagnostic 5-234

X

X IPCS line command
  description 6-37
X symbol
  for IPCS B-6
XCF (cross-system coupling facility)
  obtaining diagnosis data for 5-60
XESDATA IPCS subcommand
  description 5-323
XLpgmname symbol
  for IPCS B-6

Z

z/OS UNIX System Services
  obtaining diagnostic data 5-180
Znnnnn symbol
  for IPCS B-6

Readers' Comments — We'd Like to Hear from You

z/OS
MVS Interactive Problem
Control System (IPCS)
Commands

Publication No. SA22-7594-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SA22-7594-03

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

SA22-7594-03

Cut or Fold
Along Line

Program Number: 5694-A01, 5655-G52

Printed in U.S.A.

SA22-7594-03