File No. S370-37
Order No. GC34-2006-1

Systems

# OS/VS2 MVS Interactive Problem Control System User's Guide and Reference

IBM

File No. S370-37
Order No. GC34-2006-1

Systems

# OS/VS2 MVS Interactive Problem Control System User's Guide and Reference

SUID 5752-857

IBM

SECOND EDITION (October 1979)

This is a major revision of, and obsoletes GC34-2006-0. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change. This edition applies to the Interactive Problem Control System (IPCS) Selectable Unit (SU57) for use with OS/VS2 MVS Release 3.7 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Refer to the Summary of Amendments for a description of the primary changes to the publication. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 Bibliography, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Service Systems Publications, Dept. Z59, PO Box 390, Poughkeepsie, NY U.S.A. 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

# Preface

This publication has two purposes:

- To describe how to use the Interactive Problem Control System (IPCS) commands and subcommands, and what to use them for (the user's guide).
- To describe the function, syntax, and operands of the IPCS commands and subcommands (the reference).

Chapters 1 and 2 of this publication provide general information which all users of IPCS should know.

Chapters 3 and 4 describe how to use IPCS subcommands to perform the two major functions of IPCS: to manage problems and their associated data sets and to analyze dumps.

Chapter 5 describes the function, syntax, and operands of the three TSO commands supplied with IPCS and of the IPCS subcommands. The TSO commands are presented in alphabetic order as are the IPCS subcommands.

Chapter 6 provides additional information useful to people responsible for installing IPCS and managing the installation.

The Appendix defines all acronyms for control blocks, tables, and work areas used in this book.

Prerequisite publications for this book are:

OS/MVT and OS/VS2 TSO Terminals, GC28-6762

OS/VS2 TSO Terminal User's Guide, GC28-0645

Corequisite publications for this book are:

OS/VS2 MVS Interactive Problem Control System: Messages and Codes, GC34-2007

OS/VS2 TSO Command Language Reference, GC28-0646

OS/VS2 MVS System Programming Library: Debugging Handbook, GC28-0708

OS/VS2 MVS System Programming Library: Service Aids, GC28-0674

OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-0648

OS/VS2 System Programming Library: Supervisor, GC28-0628

OS/VS2 Access Method Services, GC26-3841

OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838

OS/VS2 MVS JCL, GC28-0692

# Contents


Summary of Amendments. . . . . 8

Chapter 1. Introduction. . . . . .11
  Problem Management . . . . . .11
  Data Management. . . . . .11
  Dump Analysis. . . . . .12
  Commands and Subcommands . . . . . .13
  Security . . . . . .14

Chapter 2. IPCS Session Control. . . . . .15
  Allocating Your Dump Directory . . . . . .15
  Allocating Your Print Output Data Set. . . . . .15
  Starting and Ending an IPCS Session. . . . . .17
    Starting the Session . . . . . .17
    Ending the Session . . . . . .17
  Executing TSO Commands and CLISTs. . . . . .17
  Setting Defaults . . . . . .18
  TEST and NOTEST Keywords . . . . . .19
  Processing Dumps . . . . . .19
  Listing and Deleting Dump Directory Information. . . . . .20
  Generating Attention Interrupts. . . . . .20
  Using the HELP Subcommand. . . . . .21
  Using a Background Terminal Monitor Program. . . . . .21

Chapter 3. Managing Problems and Data Sets . . . . . .23
  Managing Problems. . . . . .23
    Problem Attributes . . . . . .24
    Adding and Deleting Problems . . . . . .25
    Modifying and Listing Problem Attributes . . . . . .26
  Managing Data Sets Associated with Problems. . . . . .27
    Attributes of Data Sets. . . . . .27
      Dump Data Sets . . . . . .28
      Print Data Sets. . . . . .29
      User-Defined Data Sets . . . . . .29
    Data Set Attribute Conflicts . . . . . .29
    Associating and Dissociating Data Sets and Problems. . . . . .30
    Modifying and Listing Data Set Attributes. . . . . .31

Chapter 4. Analyzing Dumps . . . . . .33
  Using DSPL3270 . . . . . .33
    Data Display Area. . . . . .35
    The Current Address. . . . . .36
    Selecting Addresses. . . . . .36
      Implicit Scrolling . . . . . .36
      Indirect Addressing. . . . . .36
      Selecting Stack Elements . . . . . .36
    Scrolling Data . . . . . .36
    Skipping Data. . . . . .37
    Formatting Data. . . . . .37
    Stacking Addresses . . . . . .38
    Executing IPCS Subcommands, TSO Commands, and CLISTs . . . . . .38
    Terminating DSPL3270 . . . . . .39
  Checking System Components . . . . . .39
  Using Data Description Operands. . . . . .40
    Specifying Data Description Operands . . . . . .42
    Resolving Default Data Description Operands. . . . . .43
  Using the Symbol Table . . . . . .43
    Naming Conventions . . . . . .44
    Creating, Listing, and Deleting Symbol Table Entries . . . . . .45
  Using the Storage Map. . . . . .48

Creating Storage Map Entries . . . . . .48
Validating Structures. . . . . .49
Listing and Deleting Map Entries . . . . . .51
Examining Dump Data. . . . . .51
Executing User-Written Routines. . . . . .53
Using Subcommands in CLISTs. . . . . .53
Analyzing Dumps Containing Bad Data. . . . . .54
Establishing Definitions . . . . . .54
IPCS Data Validation . . . . . .55

Chapter 5. IPCS Commands and Subcommands . . . . . .57
TSO Commands for IPCS. . . . . .57
IPCS Command - Start an IPCS Session . . . . . .59
IPCSDDIR Command - Initialize a Dump Directory . . . . . .61
SYSDSCAN Command - Display Titles in SYS1.DUMP Data Sets . . . .63
Data Description Operands. . . . . .65
IPCS Subcommands . . . . . .73
ADDDSN - Add a Data Set Name to a Problem. . . . . .75
ADDPROB - Add a Problem to IPCS. . . . . .79
ASCBEXIT - Execute an ASCB Exit. . . . . .87
ASMCHECK - Analyze Auxiliary Storage Manager . . . . . .89
COMCHECK - Analyze Communications Task . . . . . .93
COMPARE - Logical Compare of Data Items. . . . . .97
COPYDUMP - Copy a Dump . . . . . 101
DELDSN - Delete a Data Set from a Problem. . . . . 107
DELPROB - Delete a Problem from IPCS . . . . . 111
DROPDUMP - Delete All Dump Records for a Dump. . . . . 115
DROPMAP - Delete Map Records for a Dump. . . . . 117
DROPSYM - Delete Symbols for a Dump. . . . . 119
DSPL3270 - Full-Screen Display . . . . . 121
END - End an IPCS Session. . . . . 127
ENQCHECK - Analyze ENQ Component . . . . . 129
EQUATE - Create a Symbol . . . . . 133
EVALUATE - Retrieve Dump Data for a CLIST. . . . . 135
FIND - Locate Data in a Dump . . . . . 137
FINDMOD - Locate a Module Name . . . . . 143
FINDUCB - Locate a UCB . . . . . 149
HELP - Get Information about Subcommands . . . . . 155
IOSCHECK - Analyze I/O Component . . . . . 157
LIST - Display Storage. . . . . 161
LISTDSN - List Data Set Attributes . . . . . 165
LISTDUMP - List Dumps in Dump Directory. . . . . 169
LISTMAP - List Storage Map Entries . . . . . 171
LISTPROB - List Problems . . . . . 177
LISTSYM - List Symbol Table Entries. . . . . 187
MODDSN - Modify Data Set Attributes. . . . . 191
MODPROB - Modify Problem Attributes. . . . . 195
NOTE - Generate a Message. . . . . 203
RUNCHAIN - Search a Chain of Control Blocks. . . . . 207
SCAN - Validate System Data Areas. . . . . 215
SETDEF - Set Defaults. . . . . 221
STATUS - Describe System Status. . . . . 229
Output for STATUS. . . . . 231
SUMMARY - Summarize Control Block Fields . . . . . 235
TCBEXIT - Execute a TCB Exit . . . . . 243
TSO - Execute a TSO Command. . . . . 247
VERBEXIT - Execute a User Control Statement Exit . . . . . 249

Chapter 6. Installation Planning and Use . . . . . 251
Data Sets Used by IPCS . . . . . 251
Session Parameters Member. . . . . 251
Processing Session Parameters Members. . . . . 253
Data Set Directory . . . . . 254
Problem Directory. . . . . 256
Dump Directory . . . . . 258

Print Output Data Set. . . . . 258
Dump Data Set. . . . . 258
Limiting Access to TSO Commands and CLISTs . . . . . 259

Appendix. Control Block Acronyms . . . . . 261

Index. . . . . 265

# Summary of Amendments

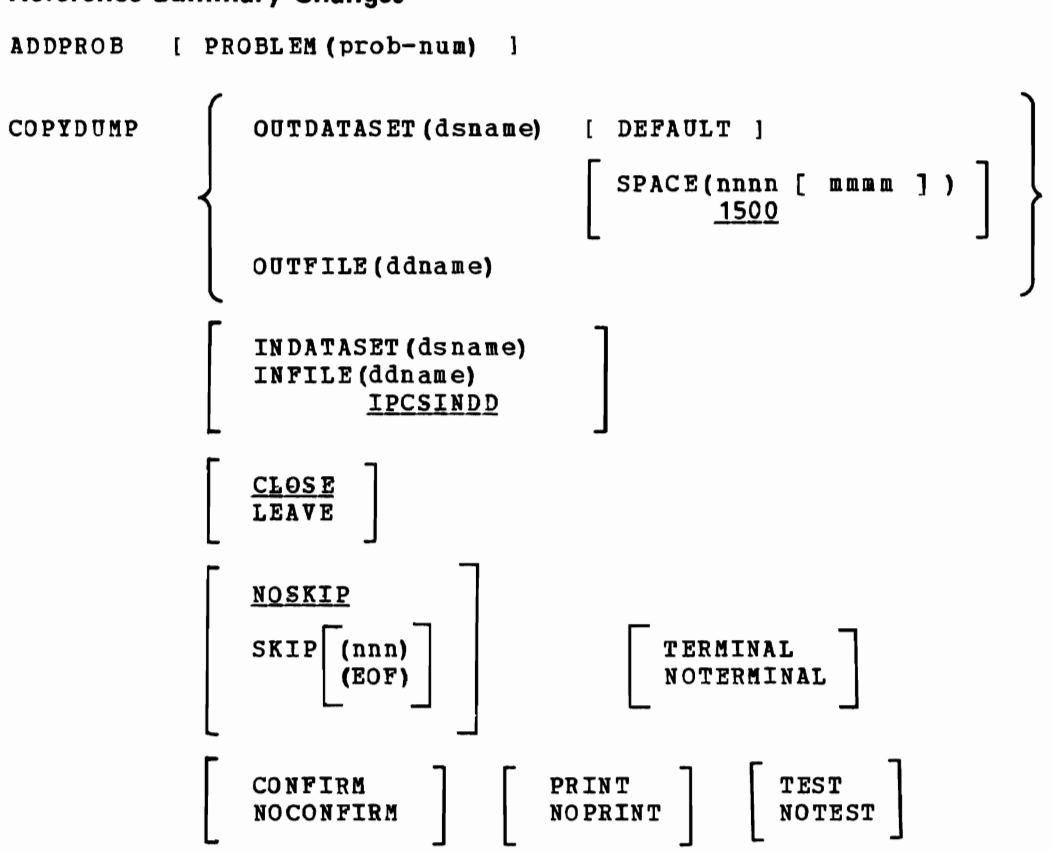
## Changes to Commands and Subcommands

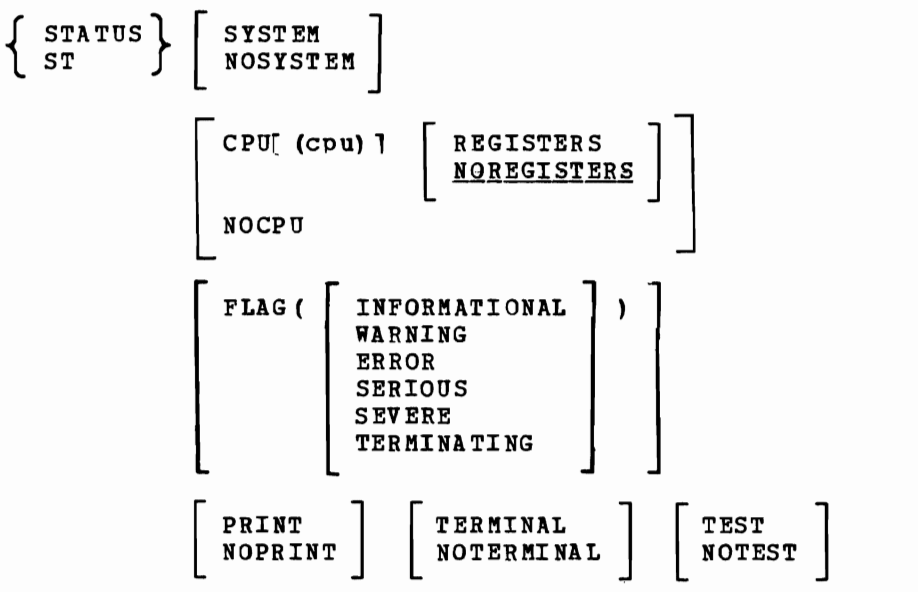
- ADDPROB has been modified to permit you to designate the problem number to be assigned.
- COPYDUMP is a new subcommand that allows you to select and extract a single dump from a sequential data set containing several dumps. A data set is produced that may be processed by MVS/IPCS dump data examination subcommands.
- STATUS is a new subcommand that aids you in initial problem determination, extracting and displaying system status current at the time a dump was produced.
- SUMMARY has been modified to allow you to selectively display TCBs. Also, the registers saved in each TCB/RB that is processed may be displayed by using a new option, REGISTERS.
- SYSDSCAN has been modified to allow you to display the date and time when dumps in the SYS1.DUMPnn data sets were produced. Also, long dump titles are now split at word boundaries and continued on additional lines if necessary.

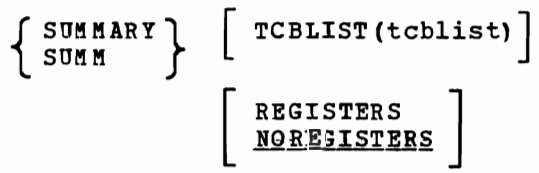
## Miscellaneous Changes

- You may use MVS/IPCS dump analysis subcommands prior to the association of a dump data set with a problem.
- Chapter 6 is now called "Installation Planning and Use," and has new sections on IPCS dump data sets and how to limit access to TSO commands in the IPCS environment.
- Other minor technical corrections and editorial changes have been made throughout the document.

## Reference Summary Changes



# Chapter 1. Introduction

The Interactive Problem Control System (IPCS) is an interactive, online facility for diagnosing software failures and managing problem information and status. With IPCS you can examine dumps without having to print them. IPCS provides subcommands that allow you to view a dump at your terminal, display data from it, locate modules and control blocks, validate control blocks, and check certain system components.

IPCS helps you manage and solve problems that occur at your installation. IPCS records information about problems in a problem directory. It records information about data sets associated with those problems in a data set directory. As you examine a dump of a software failure, IPCS accumulates information about the dump in a dump directory. This information makes future processing of that dump faster and easier.

IPCS is a TSO command processor. When you execute it you start an IPCS session. During that session, you can execute IPCS subcommands to manage problems, manage their associated data sets, and analyze data sets containing dumps.

IPCS subcommands fall into two broad categories: management and dump analysis. The management subcommands are further divided into problem management and data management subcommands.

## Problem Management

Problem management is an IPCS function that establishes an inventory of reported problems, which are listed by unique identifiers. This inventory is called the IPCS problem directory.

Problem management subcommands allow you to add problems to IPCS, list them, modify their attributes, and delete them. When you add a problem to the IPCS problem directory, IPCS assigns it a unique identifier. The 8-character identifier consists of three EBCDIC characters followed by five decimal digits. Your installation specifies the three EBCDIC characters; IPCS generates the five-digit sequence number, or you may assign this sequence number yourself.

Every problem defined to IPCS must have an owner. Anyone can add a problem and specify its owner. The owner can be the person adding the problem or someone else. You may specify the problem's status or resolution and its description when you add it to the problem directory or you may omit it. If you omit this information, only the problem owner and the person with administrative authority can specify or change it later. Similarly, only the owner of a problem or the person with delete authority can delete it. Any user can list a problem's attributes and the data set names associated with it.

## Data Management

Data management is an IPCS function that establishes an inventory of information which is contained in data sets. This inventory is called the IPCS data set directory. Each of these data sets is related to a

problem in the problem directory. The information in the data set directory combined with the information in the problem directory forms a useful tool for the management of your software problems.

This problem and data management information can be combined with information you obtain from analyzing dumps (see "Dump Analysis" immediately following), to provide you with readily available records of all data associated with all software problems you report to IPCS.

Data management subcommands allow you to associate data sets with problems, list them, modify their attributes, and dissociate them. IPCS does not assign data set identifiers as it does for problems. You must associate a data set with an already-defined problem. You can associate several data sets with a problem and you can associate a data set with several problems. You can associate a data set with a problem whether or not you own that problem.

You may specify the data set's attributes when you associate it with a problem. If you omit this information, it can be specified or modified later only by the owner of a problem the data set is associated with or by the person with administrative authority. Similarly, a data set can be dissociated from a problem only by the owner of that problem or by the person with administrative authority. Any user can list a data set's attributes and the problems it is associated with.

## Dump Analysis

Dump analysis is an IPCS function that enables you to examine dumps of software failures, and save the critical pieces of data that you find. The dump data is stored in a dump directory.

You can perform dump analysis before you do problem management or data management. In this way, if you determine that a dump data set is related to an already existing problem, you do not have to create a new problem entry for it, but can simply associate it with the known problem. If you determine that the dump represents a unique problem, you can perform the management functions of updating the problem directory and data set directory, and associating the dump data set with the new problem.

Dump analysis subcommands allow you to examine, search for, verify, and validate data in dumps which may or may not be associated with problems. You can associate any data set you want with a problem but only dumps are processed by the dump analysis subcommands. Specifically, the dumps must be unformatted, cataloged, and on direct access storage. You can use IPCS to copy dumps from one data set to another data set (for example, from a tape volume to a direct access volume) before you use the dump analysis subcommands.

IPCS dump analysis subcommands process unformatted stand-alone dumps (produced by the high-speed option of AMDSADMP) and virtual dumps, including SYSMDUMPs (produced by the MVS SDUMP facility). Using the dump analysis subcommands you can examine the data in a dump, locate and verify control blocks associated with certain functions or system components, trace and verify chains of control blocks, scan for abnormally terminating tasks, locate modules, execute user-written exits for certain control blocks, and keep a list for future reference of the names and locations of control blocks and areas of the dump that you consider important.

## Commands and Subcommands

The following list shows the TSO commands supplied with IPCS and the subcommands of IPCS, with a general statement of the functions provided. The commands are listed first, followed by the subcommands.

TSO Commands

- IPCS -- Start an IPCS session.
- IPCSDDIR -- Prepare a dump directory data set for dump analysis.
- SYSDSCAN -- Display the titles of dumps in SYS1.DUMP data sets.

Subcommands of the IPCS Command

Problem and Data Management Subcommands:

- ADDDSN -- Add a data set name to a problem.
- ADDPROB -- Add a problem.
- DELDSN -- Delete a data set name from a problem.
- DELPROB -- Delete a problem.
- LISTDSN -- List data set attributes.
- LISTPROB -- List reported problems.
- MODDSN -- Modify data set attributes.
- MODPROB -- Modify problem attributes.

DSPL3270 Subcommand:

- DSPL3270 -- Display selected portions of a dump with full-screen operation.

Analysis Subcommands:

- ASMCHECK -- Analyze the auxiliary storage manager (ASM).
- COMCHECK -- Analyze the communication task component.
- ENQCHECK -- Summarize ENQ/DEQ resource management chains.
- IOSCHECK -- Analyze the I/O supervisor component.
- STATUS -- Display system status at time of dump.
- SUMMARY -- Summarize selected control block fields.

Symbol Table and Storage Map Subcommands:

- DROPDUMP -- Delete all records that describe a particular dump from the dump directory.
- DROPMAP -- Delete records of control blocks that have been located in a dump.
- DROPSYM -- Delete symbols from the symbol table.
- EQUATE -- Create a symbol.
- LISTDUMP -- List dumps represented in the dump directory.
- LISTMAP -- List control blocks that have been located in a dump.
- LISTSYM -- List attributes of symbols in the symbol table.
- SCAN -- Validate key MVS control blocks.

Dump Viewing Subcommands:

- FIND -- Locate data in a dump.
- FINDMOD -- Search for a module by name.
- FINDUCB -- Search for a UCB.
- LIST -- Display storage.
- RUNCHAIN -- Search through a chain of control blocks.

AMDPRDMP Exit Support Subcommands:

- ASCBEXIT -- Pass control to an exit routine that is compatible with the AMDPRDMP ASCB exit.
- TCBEXIT -- Pass control to an exit routine that is compatible with the AMDPRDMP TCB exit.
- VERBEXIT -- Pass control to an exit routine that is compatible with the AMDPRDMP user control statement exit.

CLIST Support Subcommands:

- COMPARE -- Perform logical data comparison.
- EVALUATE -- Obtain dump data for CLIST processing.
- NOTE -- Produce messages and control spacing and pagination.

Miscellaneous Subcommands:

- COPYDUMP -- Copy a dump from one data set to another.
- END -- Terminate an IPCS session.
- HELP -- Provide descriptive information about the IPCS command and its subcommands.
- SETDEF -- Set, change, and display IPCS session defaults.
- TSO -- Invoke a non-IPCS TSO command or subcommand function.

## Security

Security of information at your installation is as normally available through MVS supported facilities, such as data set password protection and TSO LOGON verification. Programs such as the IBM Resource Access Control Facility (RACF) Program Product may be used at your option to provide additional security as required. It should be noted that your installation must continue to take specific action to prevent unauthorized access to dumps containing sensitive data.

# Chapter 2. IPCS Session Control

The preparations for your IPCS session consist of allocating your dump directory and print output data set and specifying the session parameters member. Once you have started your session, you can execute IPCS subcommands, TSO commands, and CLISTs, and use attention interrupts.

The following examples demonstrate how you might use TSO commands to prepare for an IPCS session. The sample commands and subcommands use the TSO command syntax and data set naming conventions. For a complete description of these conventions, see the OS/VS2 TSO Command Language Reference. In the examples, user-entered input lines are shown in lowercase and system responses, when they are shown, are in uppercase.

## Allocating Your Dump Directory

An IPCS dump directory contains information used by dump analysis subcommands to efficiently process dumps. Each IPCS user has at least one dump directory but can use only one during an IPCS session. Each dump directory contains information about one or more dumps.

Your dump directory is a VSAM key-sequenced cluster. It must be created with Access Method Services and must be initialized before its first use (see "Dump Directory" in Chapter 6). Once initialized, a dump directory data set does not need to be initialized again.

Once created and initialized, you must allocate your dump directory before you execute the IPCS command. You must allocate this data set using the ddname IPCSDDIR. To ensure the data set's integrity, specify a disposition of OLD.

IPCS initialization opens the data set allocated to IPCSDDIR. If you didn't allocate it or if IPCS can't open it, the IPCS command terminates. Once opened, IPCS does not close the data set for the duration of the session. Thus, you cannot free or reallocate your dump directory during an IPCS session.

If your TSO user-prefix is IPCSU1, a sample allocation is:

```
allocate ddname(ipcsddir) dataset(dir.debug)
```

This command allocates the data set IPCSU1.DIR.DEBUG to the ddname IPCSDDIR. Since the data set already exists, the disposition defaults to OLD.

## Allocating Your Print Output Data Set

This data set contains the output generated by IPCS subcommands for which you specify the PRINT keyword. The IPCS print output data set is a sequential data set containing blocked variable-length records with ASA control characters. This data set is optional. You can allocate no more than one print output data set for an IPCS session. You must allocate the print output data set using the ddname IPCSPRNT prior to executing the IPCS command.

If you didn't allocate it or if IPCS can't open it, IPCS displays a message informing you that the data set is not allocated. This is a warning and does not prevent you from continuing with the IPCS session, but IPCS subcommands ignore the PRINT keyword during this session. IPCS does not close the data set for the duration of the session. Thus you cannot free or reallocate your print output data set until you end the IPCS command.

A sample ALLOCATE command for the print data set is:

```
allocate ddname(ipcsprnt) sysout(a)
```

This command allocates the print data set to the ddname IPCSPRNT and assigns it to SYSOUT class A.

If you allocate your print output data set to other than SYSOUT, use the following attributes:

- DDNAME or FILE - IPCSPRNT
- DISP - NEW, OLD, or MOD (not SHR)
- DSORG - PS
- RECFM - V B A
- LRECL - value

  The minimum value for LRECL is 83, the maximum is 255. If you specify a value less than 83 or greater than 255, IPCS initialization cancels the PRINT option.

  If not specified before the data set is opened, IPCS initialization uses the value specified in the IPCS session parameters (see "Session Parameters Member" in Chapter 6.) If there is no value specified in the session parameters, the default value is 137.

- BLKSIZE - value

  The value must be greater than or equal to the LRECL, plus four. If you specify a value that does not meet this condition, IPCS initialization cancels the PRINT option.

  If not specified, IPCS initialization specifies the value as:

  ```
  BLKSIZE = 4 + (14 * LRECL)
  ```

- SPACE - value

  You must determine the amount of space the data set requires.

Another example, assuming your TSO user-prefix is IPCSU1, is:

```
attrib list1 dsorg(ps) recfm(v b a) lrecl(125) blksize(1254)

allocate ddname(ipcsprnt) dataset(sess7.print) new keep
  space(10,5) tracks using(list1)
```

These commands allocate a new data set named IPCSU1.SESS7.PRINT. The data set's attributes are specified in LIST1.

For convenience, you may want to create a TSO CLIST to allocate your dump directory and print output data set. Such a CLIST might be:

```
free file(ipcsddir ipcsprnt)
allocate ddname(ipcsddir) dataset(dir.debug)
allocate ddname(ipcsprnt) sysout(a)
```

This CLIST ensures that the two files are free and then allocates them.

## Starting and Ending an IPCS Session

After you've allocated the dump directory and, optionally, the print output data set, you are ready to enter the IPCS command.

### *Starting the Session*

Start the session by entering the IPCS command. The only operand you can use on this command is the keyword PARM specifying a one or two-digit decimal number indicating the session parameters (see "Session Parameters Member," in Chapter 6). IPCS appends the number you specify to IPCSPR, forming the name of a member of SYS1.PARMLIB. That member contains the parameters for this session.

The operand is optional. If you don't specify it, IPCS uses the member named IPCSPR00. If that member doesn't exist, the IPCS command terminates. IPCS does not allow you to specify more than one IPCS session parameters member on the command. A typical IPCS command is:

```
ipcs
```

The IPCS command signals the completion of its initialization by displaying

```
IPCS
```

on the next line on your terminal.

You are now in IPCS mode and can enter an IPCS subcommand, a TSO command, or a CLIST.

### *Ending the Session*

To end your IPCS session, enter the END subcommand. END takes no parameters. It closes any data sets opened during the session and frees any files allocated by IPCS initialization. It then terminates the IPCS session and returns you to the TMP (terminal monitor program). You may free the data sets you allocated to the session unless you want to use them later in your TSO session.

## Executing TSO Commands and CLISTs

While in IPCS mode you can still execute most TSO commands and CLISTs.

To execute a TSO command or CLIST whose name does not match an IPCS subcommand, type the TSO command or CLIST name and press enter.

IPCS first tries to locate a command processor by the specified name. If the name is not found, IPCS processes the command as an implied CLIST. Since this process may produce a write-to-programmer message (IEA703I), you may wish to prefix a known CLIST name with a percent sign (%) to speed processing and avoid the message.

To execute a TSO command whose name matches an IPCS subcommand, use the IPCS subcommand named TSO (see the TSO subcommand in Chapter 5). Type the command name as the operand to the TSO subcommand and press enter. To execute a CLIST whose name matches an IPCS subcommand, use the percent sign (%). Type the CLIST name after the percent sign and press enter. For more information on executing CLISTs, see the OS/VS2 TSO Terminal User's Guide.

Note that the TSO CALL, TEST, and any command requiring APF (Authorized Program Facility) authorization cannot be executed under IPCS.

IPCS simulates the TSO terminal monitor program (TMP) when handling command names and CLIST names. It provides for a command name validation exit, as the TMP does. For information on using your own command name validation routine, refer to "Limiting Access to TSO Commands and CLISTS," in Chapter 6.

## Setting Defaults

When you begin an IPCS session, IPCS establishes various default values which remain in effect until explicitly changed by you. IPCS sets these values from the session parameters member and from the initial values of the SETDEF subcommand keywords.

The values obtained from your session parameters member determine the default system and group identifier. These values are used when you add a problem and omit the SYSTEM or GROUP keyword.

By specifying default keywords with the SETDEF subcommand, you save yourself the trouble of specifying them on each subcommand to which they apply. Before using the dump analysis subcommands, you must define the applicable dump data set as the current IPCS data set. Dump analysis subcommands process the current data set and you specify the current data set with the SETDEF subcommand or with the ADDDSN, MODDSN, or COPYDUMP subcommands. (Hereafter in this document the terms current data set and default data set are used interchangeably. This is because the DEFAULT keyword can be used to indicate the current data set).

These default keywords fall into five categories:

- Keywords that control the data produced by a subcommand. These keywords are CONFIRM, DISPLAY, FLAG, TEST, and VERIFY.
- Keywords that direct the data produced by a subcommand to your terminal or to a print data set. These keywords are TERMINAL and PRINT.
- The keyword that specifies the default problem. This keyword is PROBLEM.

- The keyword that specifies the current data set. This keyword is DATASET (or DSNAME).
- Keywords that describe the dump data being accessed from the current data set. These keywords are ASID, CPU, and LENGTH.

Additionally, you can specify the default problem with the DEFAULT keyword on the ADDPROB and MODPROB subcommands and you can specify the current data set with the DEFAULT keyword on the ADDDSN, MODDSN, and COPYDUMP subcommands.

To save time, you can include the IPCS command and IPCS subcommands in a CLIST. Such a CLIST might be:

```
free file(ipcsddir ipcsprnt)
allocate ddname(ipcsddir) dataset(dir.debug)
allocate ddname(ipcsprnt) sysout(a)
ipcs
setdef dataset('sys1.dump04') print
dspl3270
```

This CLIST allocates your dump directory (previously defined and initialized) and print data sets, starts your IPCS session, sets two defaults (the current data set name and the PRINT keyword), and invokes the DSPL3270 subcommand.

## TEST and NOTEST Keywords

The TEST keyword of IPCS subcommands places IPCS in a mode designed to support interactive testing of code that operates in the IPCS environment. It is not recommended for use for any other purpose.

If you anticipate an abnormal termination 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 TEST facilities to isolate the cause of any problems, you should use the TEST keyword. When TEST is in effect, IPCS allows the TMP, the TSO TEST ESTAI functions, or both, to gain control when an abnormal termination occurs.

TEST mode also activates error-detection functions which have been developed to isolate dump data examination problems. Errors detected cause IPCS to abnormally terminate so that problems may be trapped close to the point of error.

The NOTEST keyword of IPCS subcommands places IPCS in the production mode of operation. This is the default. Automatic error recovery is attempted should errors occur in the IPCS environment.

When the NOTEST keyword is in effect, IPCS automatically recovers from most abnormal terminations without permitting TSO TEST to gain control.

## Processing Dumps

You can use IPCS to process cataloged, unformatted dumps that are in direct access data sets, as described in Chapter 1 above under "Dump Analysis." The first time during an IPCS session that you execute a dump analysis subcommand against a data set, IPCS verifies that the data set meets the criteria for a dump data set as described under "Dump Data Sets" in Chapter 3 below. If the data set does not meet the criteria,

IPCS terminates the dump analysis subcommand and issues a message. You must define the dump data set as the current data set before using the dump analysis subcommands.

If the current (or default) data set is not represented in the dump directory allocated to this session, IPCS: creates a description of the dump in the dump directory; creates storage map entries for the CVT and certain data areas, modules, and structures pointed to from the CVT; and creates symbol table entries for the CVT, private area, common area, and the dump title.

You can associate dump data sets with problems either before or after analyzing the dump.

## Listing and Deleting Dump Directory Information

Your dump directory may contain symbol table and storage map entries for several dumps. To get a list of all the dumps represented in your dump directory, use the LISTDUMP subcommand. The subcommand

```
listdump
```

lists the names of all dump data sets represented in the dump directory you are using for this session.

To delete all records pertaining to a particular dump from your dump directory data set, use DROPDUMP. The subcommand

```
dropdump 'schedlr.stand.alone.dump'
```

deletes all symbol table and storage map records for SCHEDLR.STAND.ALONE.DUMP from the dump directory you are using for this session. It does not affect the SCHEDLR.STAND.ALONE.DUMP data set.

## Generating Attention Interrupts

During an IPCS session you can interrupt the current processing at your terminal by generating an attention interrupt. You can use an attention interrupt to interrupt a long-running subcommand or user exit routine. For a complete description of attention interrupts, see OS/VS2 TSO Terminal User's Guide.

If you press ENTER after the attention interrupt, the interrupted processing resumes.

If you enter any IPCS subcommand, TSO command (except TIME), or CLIST, the interrupted processing is terminated and the new subcommand, command, or CLIST is executed.

If you enter the TSO TIME command, the command executes without terminating the interrupted processing and the attention interrupt is still in effect.

If you enter ABEND, the IPCS command is abnormally terminated with a user ABEND code of 114 (x'72'). The abnormal termination produces a dump if you have a SYSABEND, SYSUDUMP, or (if SU33 is installed) SYSMDUMP data set allocated to your session. The dump produced on the SYSMDUMP data set can be processed with IPCS subcommands, while the two other types can not be processed by IPCS.

During certain phases of their processing, the DELDSN and DELPROB subcommands ignore 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 terminate a subcommand that modifies the problem directory or the data set directory, the modification to the data set 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 TMP (terminal monitor program) level.

## Using the HELP Subcommand

During an IPCS session, you can use the HELP subcommand to receive information explaining the use of any IPCS subcommand. The information is displayed at your terminal.

To receive a list of all the IPCS subcommands in the HELP data set along with a description of each, enter the HELP subcommand as follows:

```
help
```

You can also get information on a specific subcommand by entering a subcommand name as an operand on the HELP subcommand. For example:

```
help delprob
```

## Using a Background Terminal Monitor Program

The IPCS commands and subcommands do, in general, execute under the TSO terminal monitor program (TMP) that is executing as a background job. However, certain management subcommands require a TSO userid as part of this TSO environment in order to execute correctly. The Resource Access Control Facility (RACF), for example, provides such an environment. The TSO userid does not appear in the title line for output directed to the print output data set unless it is specified as part of the job.

The management subcommands and their restrictions are listed below:

- ADDDSN - Subcommand provides restricted functions.
- ADDPROB - Subcommand provides restricted functions.
- DELDSN - Subcommand fails.
- DELPROB - Subcommand fails.
- LISTDSN - Subcommand executes normally.
- LISTPROB - Subcommand provides restricted functions.
- MODDSN - Subcommand fails.

- MODPROB - Subcommand provides restricted functions or fails, depending on its use.

ADDDSN requires that you not specify CONFIRM. Specify NOCONFIRM either on this subcommand or on the SETDEF subcommand.

ADDPROB requires that you specify the OWNER keyword. If you do not, the owner field for the problem will contain unpredictable information.

DELDSN fails because the subcommand, in the absence of a TSO userid, can't verify if the user is authorized to dissociate the specified data set.

DELPROB fails because the subcommand, in the absence of a TSO userid, can't verify if the user is authorized to delete the specified problem.

LISTDSN executes normally since it does not use a TSO userid.

LISTPROB requires that you specify the OWNER keyword if you specify any other problem selection keywords.

MODDSN fails because the subcommand, in the absence of a TSO userid, can't verify if the user is authorized to modify the specified data set.

MODPROB only allows you to add text to the problem description. In the absence of a TSO userid, the subcommand can't verify if the user is authorized to modify the problem in any other way.

# Chapter 3. Managing Problems and Data Sets

The following examples demonstrate how you can use IPCS subcommands to manage your problems and the data sets associated with them. The sample commands and subcommands use the TSO command syntax and data set naming conventions. For a complete description of these conventions, see the OS/VS2 TSO Command Language Reference.

In the examples, user-entered input lines are shown in lowercase and system responses, when they are shown, are in uppercase.

## Managing Problems

Managing problems consists of adding them to the IPCS problem directory, specifying their attributes, modifying their attributes, listing them, and deleting them from the problem directory.

Problems defined to IPCS must be owned by someone. A problem owner usually is the person responsible for the problem and its solution. You specify the owner when you add the problem to the IPCS problem directory. You can specify the owner's TSO userid or let it default to the TSO userid you are using when you add the problem.

Restrictions on processing problems are imposed through ownership of the problems. You can modify or delete a problem only if you own it. Optionally, your installation can designate a person as having administrative authority and a person (possibly the same one) as having delete authority (see "Session Parameters Member" in Chapter 6.)

The person with administrative authority has the same authority as a problem owner but has that authority over all problems in the problem directory. The person with delete authority is the only person who can delete problems from the problem directory. If your installation designates someone with delete authority, neither problem owners nor the person with administrative authority can delete problems.

If no one at your installation has administrative authority or delete authority, the owner of a problem is the only person who can modify the problem's attributes and delete it.

Note: The concepts of problem ownership, administrative authority, and delete authority are disciplines established by IPCS. Since IPCS executes as a problem program (non-authorized), these disciplines cannot be fully enforced. Access to data sets, however, is controlled through conventional MVS security facilities.

## *Problem Attributes*

A problem's attributes give a complete description of it. The attributes for a problem are:

| | |
|---|---|
| Problem identifier | Problem status |
| Problem occurrence date | FIX status |
| Problem occurrence time | IBM status |
| Problem reported date | PTF status |
| Problem reported time | APAR identifier |
| Group | PTF identifier |
| Owner | FIX identifier |
| System | Abstract |
| Component | Description |
| Severity | User |

IPCS uses the problem identifier to uniquely identify a problem. The problem identifier is composed of a three-character prefix and a five-decimal-digit suffix. The prefix is used to differentiate between problems with similar numbers in different problem directories. It is specified in your session parameters member, but you never specify it on an IPCS subcommand. When IPCS lists problems, it displays their complete problem identifiers, including prefixes. The prefix is not stored in the problem directory and can be changed without affecting previous work. IPCS assigns the suffix when you add a problem, or you may assign it yourself using the ADDPROB subcommand. Once the problem identifier is stored, you cannot change it. You specify the suffix on any IPCS subcommand needing identification of the problem.

The date and time of problem occurrence allow you to specify when a problem occurred. IPCS sets the date and time of problem reporting when you add the problem to the problem directory. You can modify the problem occurrence attributes but you cannot modify the problem reported attributes.

The group and owner identify both the group and the individual responsible for tracking and resolving the problem. The group is the department or organization name or any other identification you want to use for your installation. You can change the group at any time. You can specify an owner for the problem by specifying a TSO userid, or IPCS supplies your TSO userid as the default. The owner is significant because certain IPCS subcommands can be executed against a problem only by its owner. As owner of a problem, you can modify its attributes and, unless restricted, delete it. You can transfer ownership of a problem by modifying this field.

The system and component attributes identify the environment of the problem. The system identifier refers to the hardware or software system on which the problem occurred. The component identifier refers to the software component you suspect has caused the failure.

The severity attribute allows you to specify a numerical estimation of the severity of the problem, consistent with standard IBM APAR severity codes.

The status and identifier attributes allow you to provide information about the status and resolution of the problem. The status attributes -- PROBLEM, FIX, IBM, and PTF -- allow you to indicate the status of the problem and the status of its resolution. You can indicate the status from your point of view (PROBLEM and FIX status) and from IBM's point of view (IBM and PTF status). For each status attribute, IPCS records the date and time of the last change to that attribute.

The identifier attributes -- APAR, PTF, and FIX -- allow you to indicate the APAR for the problem and the PTF or FIX that might resolve it or that does resolve it.

The abstract and description allow you to describe the problem, first briefly in the abstract, then at length in the description.

The user data attribute allows you to specify additional information about the problem. It is defined by the user or by the installation.

### *Adding and Deleting Problems*

Adding a problem means reporting it to IPCS. Deleting a problem means removing any reference to it from IPCS. When you delete a problem, IPCS dissociates any data sets still associated with it.

To add a problem to IPCS, use the ADDPROB subcommand. You may specify an identifier for the problem or allow IPCS to assign it for you. The subcommand adds the problem to the problem directory data set, and informs you of the new problem identifier with a message.

If you specify no keywords for the ADDPROB subcommand, IPCS assigns default attributes. You can use the MODPROB subcommand to specify or change the information about a problem you own after it has been added to the problem directory.

A typical ADDPROB subcommand is:

```
addprob date(1/9/78) time(01:02:03) system(168C) pstatus(active)
  severity(3) compid(5752-sc1de) abstract('vsam problem')

BLS05100I PROBLEM ABC00016 HAS BEEN ADDED TO THE PROBLEM DIRECTORY
```

The problem is added to the IPCS problem directory and is assigned ABC00016 as its identifier. ADDPROB puts the TSO userid in use when this subcommand is executed in the problem-owner field. The problem identifier is important; you will need it later in order to do any work on this problem and its data sets.

To delete a problem from the IPCS problem directory, use the DELPROB subcommand. For example:

```
delprob problem(625)
```

If the problem you are deleting has data sets associated with it (see "Managing Data Sets Associated with Problems" later in this chapter), DELPROB dissociates those data sets.

In order to delete a problem, you must have delete authority or, if no one has delete authority, you must have administrative authority or own the problem.

Delete authority and administrative authority are specified in the session parameters (see "Session Parameters Member" in Chapter 6).

After you define a problem and as additional information about it becomes available, you, as problem owner, should update the problem attributes to reflect the changes. To do this, use the MODPROB subcommand. You can modify the attributes of only those problems that you own. The person (if any) with administrative authority can modify the attributes of any problem in your installation's problem directory.

Suppose you own problem ABC00061 and an APAR has been written against it. The following subcommand makes the appropriate changes:

```
modprob problem(61) aparid(oz12345) ibmstatus(apared)
  description('apar submitted 2/5/78')
```

If the corresponding fields are empty, they are filled with the new attributes. If the corresponding fields contain previously specified attributes, they are overlaid with the new attributes, except DESCRIPTION, which is appended to the existing description.

The subcommand

```
modprob owner(ipcsu11) severity(3)
```

transfers the ownership of the default problem to IPCSU11 and sets its severity to 3.

You have considerable flexibility in listing problems from the IPCS problem directory. LISTPROB has two types of operands: those specifying which problems are listed and those specifying which type of listing format is used. You do not have to own problems to list them.

To be included in the listing a problem must satisfy at least one value for each attribute you specify. You can specify the following problem attributes: COMPID, FIXSTATUS, GROUP, IBMSTATUS, OWNER, PROBLEMS, PSTATUS, PTFSTATUS, SEVERITY, SYSTEM, and USER.

If you specify

```
listprob pstatus(active) severity(2)
```

since OWNER is not specified in this example, a problem must be active, have a severity of 2, and be owned by you in order to be included in the listing.

If you specify

```
listprob owner(ipcsmjh ipcsrbr) severity(2 3 4)
```

a problem must be owned either by IPCSMJH or by IPCSRBR and it must have a severity of 2 or 3 or 4 to be included in the listing.

You can specify one problem, a list of problems, a range of problems, or a list of ranges. For example:

```
listprob problems(1 13:19 3:5 9 7)
```

Since OWNER is not specified in this example, information about problems 1, 13 through 19, 3 through 5, 9 and 7 is listed if they are owned by you. The subcommand displays the problems in the order in which you specify their numbers.

The LISTPROB subcommand produces information about the problems that meet your specifications and no others.

LISTPROB only selects problems with the attributes you specify, with the exception of OWNER. If you omit an attribute, a problem is eligible for inclusion in the list regardless of that attribute's value.

You control the amount and type of information in the listing with the LIST, ABSTRACT, STATUS, DESCRIPTION, and DSNAMES keywords.

## Managing Data Sets Associated with Problems

Managing data sets associated with IPCS problems consists of associating them with a problem, specifying and modifying their attributes, listing their attributes, and dissociating them from problems. Any kind of data set that you find meaningful can be associated with a problem. The problem must be defined to IPCS before you can associate a data set with it.

Unlike problem ownership, IPCS does not establish data set ownership. Restrictions on processing data sets are imposed through the ownership of the problems the data sets are associated with. You can associate a data set with a problem whether or not you own that problem. You can list the attributes of a data set whether or not you own the problem or problems it is associated with.

You can modify a data set's attributes only if you own one of the problems it is associated with. You can dissociate a data set from a problem only if you have administrative authority or if you own the problem.

### *Attributes of Data Sets*

There are three attributes for any data set associated with a problem:

- Description.
- Management.
- Type.

The description is a brief explanation of the data set's contents. The management attribute specifies whether or not IPCS attempts to scratch the data set when it is no longer associated with any problem. The type attribute identifies the kind and format of data in the data set.

When you associate a data set with a problem you can let IPCS manage it. If you let IPCS manage a data set, IPCS will, under certain circumstances, scratch the data set when it is no longer associated with any problem in the problem directory.

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.

If the data set's name is SYS1.DUMPnn, where nn is a decimal number from 00 through 99, IPCS initializes the data set for reuse rather than scratches it. In order for IPCS to initialize a SYS1.DUMPnn data set, it must be cataloged and be on direct access storage.

Data sets that may be associated with problems are any data sets that you consider valuable or useful in solving a problem. Such data sets can be dumps, assembler and compiler listings, utility and service aid outputs, etc.

These data sets are optional. It is possible to define a problem to IPCS and record its status and resolution without using a single dump or listing or any other data set except those used by IPCS itself.

IPCS classification supports three types of data sets associated with problems:

- Dump data sets.
- Print data sets.
- User-defined data sets.

### Dump Data Sets

Dump data sets contain a dump of all or part of the operating system. To be eligible for processing by IPCS dump analysis functions, the dump data sets must be sequential data sets containing dumps created by the SDUMP macro or by the high-speed option of the stand-alone dump program (AMDSADMP). IPCS dump analysis functions can only process the first dump in a dump data set containing more than one dump. However, the COPYDUMP subcommand can be used to extract any specific dump from a dump data set and transcribe it into another data set where it will be the first (or only) dump in the data set.

Another criterion to be met for IPCS dump analysis is that the dump data set must be the default IPCS data set. There are several ways to define the default data set. One way is to use the SETDEF subcommand. Or, you could use the DEFAULT keyword on the ADDDSN or MODDSN subcommand to establish the default data set. A third method is to use the DEFAULT keyword on the COPYDUMP subcommand to specify the default data set at the same time you are transcribing the dump into a working data set, if that is needed.

The physical characteristics of the dump data set represent another criterion in order for IPCS dump analysis to be used to process the dump. The dump data sets must be:

- In core image form
- Of fixed record format with blocksize of 4104.

- On direct access storage.
- Cataloged.

If a dump data set doesn't meet all of these conditions, you can associate it with a problem but the IPCS dump analysis subcommands cannot process it. When the above conditions have been met, you can use the dump analysis subcommands. When you execute a dump analysis subcommand, IPCS dynamically allocates and opens the dump data set. When you specify a different dump, IPCS closes and frees the old data set and then allocates and opens the new one. Since the dumps are cataloged, IPCS needs only their names to locate them.

### Print Data Sets

Print data sets contain printable data related to a problem. These data sets may contain formatted dumps or the output of utilities, service aids, assemblers, compilers, installation-written routines, etc.

You can associate print data sets with a problem but IPCS does not process them with the dump analysis subcommands.

### User-Defined Data Sets

User-defined data sets are data sets associated with a problem that do not qualify as dumps or print data sets. They can have any data set organization and can contain any data related to a problem.

You can associate user-defined data sets with a problem but IPCS does not process them with the dump analysis subcommands.

#### *Data Set Attribute Conflicts*

If you modify a data set's attributes or if you associate a data set with two or more problems, IPCS compares the attributes already recorded in the data set directory (the current attributes) with the newly specified attributes. If the attributes don't match, you have a data set attribute conflict.

You can use the CONFIRM keyword on the ADDDSN subcommand to have the subcommand inform you of data set attribute conflicts. If there is a conflict, you can associate the data set with the problem and leave the current attributes as they are or you can cancel the subcommand. You cannot change a data set's attributes with the ADDDSN subcommand.

If you specify NOCONFIRM, or if it is the default, the subcommand associates the data set with the problem but leaves the current attributes as they are.

You can also use the CONFIRM keyword on the MODDSN subcommand to have the subcommand inform you of conflicts, but the resolution of them is different. If there is a data set attribute conflict with the MODDSN subcommand, you can override the current attributes or you can leave them as they are. If you specify NOCONFIRM, or if it is the default, the subcommand makes the specified changes to the data set's attributes without informing you of any conflicts.

Associating a data set with a problem means informing IPCS that the data set is related to that problem. It also means specifying the data set's attributes. A problem may have several data sets associated with it and a data set may be associated with several problems. You do not have to own a problem in order to associate a data set with it.

Since you cannot examine print or user-defined data sets with IPCS subcommands, they are never opened by IPCS. However, if you want IPCS to manage them, the data sets must be cataloged. The advantage of associating these kinds of data sets with a problem is that their names are recorded in IPCS and the names of all data sets pertaining to a problem can be listed easily.

To associate a data set with a problem, use the ADDDSN subcommand. Assuming your TSO user-prefix is IPCSU1, a sample ADDDSN subcommand is:

```
adddsn dsname(prob31.dump) problem(31) confirm
  description('stand alone dump for scheduler abend')
```

This subcommand associates data set IPCSU1.PROB31.DUMP with problem ABC00031. If this data set is not already associated with a problem, its default attributes are DUMP and MANAGED. If it is already associated with a problem, it keeps its current attributes.

The subcommand

```
adddsn dsname('compiler.listing') unmanaged type(print) confirm
```

associates the data set COMPILER.LISTING with the default problem. If this data set is not already associated with a problem, it is given the UNMANAGED and PRINT attributes. If it is already associated with a problem and its current attributes are not UNMANAGED and PRINT, you are informed of the data set attribute conflict and asked to resolve it. If there is no data set attribute conflict, the data set is associated with the problem and no messages are issued.

Dissociating a data set from a problem means notifying IPCS that the data set is no longer linked with that problem. You must own a problem in order to dissociate data sets from it. You can dissociate a specific data set from a problem or you can dissociate all data sets from a problem.

If IPCS is managing a cataloged data set and you dissociate it from the last problem with which it was associated, IPCS scratches the data set unless its name begins with 'SYSn.'. The data set is dissociated from the problem even if the scratch fails (if, for example, you can't supply the password or the data set can't be located). If IPCS is not managing the data set, it does not attempt to scratch it even if you dissociate it from its last problem.

When you scratch a dump data set, you should also execute a DROPDUMP subcommand for that data set to delete its symbol table and storage map records from your dump directory.

A sample DELDSN subcommand is:

```
deldsn confirm dataset(sadump.dump) problem(681)
```

This subcommand displays the name and attributes of the data set IPCSU1.SADUMP.DUMP and requests your confirmation before dissociating it from problem ABC00681. You must own problem ABC00681.

The subcommand

```
deldsn all noconfirm
```

dissociates all data sets from the default problem. The subcommand does not display the data set attributes and does not request confirmation from you before dissociating the data sets.

### *Modifying and Listing Data Set Attributes*

To modify the attributes of a data set, you must own at least one of the problems the data set is associated with. The data set may be associated with other problems and those problems may be owned by other users. Since the data set attributes are recorded once, if you change them, you affect the data set for all the problems the data set is associated with.

To change the attributes of a data set, use the MODDSN subcommand. If your TSO user-prefix is IPCSU1 and if you own problem ABC00031, a sample MODDSN subcommand is:

```
moddsn noconfirm problem(31) dataset(prob31.dump) unmanaged
```

This subcommand changes the management attribute of data set IPCSU1.PROB31.DUMP to UNMANAGED. The subcommand verifies that you own problem ABC00031 and that IPCSU1.PROB31.DUMP is associated with that problem before changing the attribute.

The subcommand

```
moddsn dataset('program.listing') type(print) default
```

first verifies that PROGRAM.LISTING is associated with the default problem and that you own that problem. It then changes the type attribute of the data set PROGRAM.LISTING to print and makes this data set the current data set. If CONFIRM is the default, the subcommand confirms any data set attribute conflicts before modifying those attributes.

You can list the attributes (type, management, and description) of any data set in the data set directory. You do not have to own a problem the data set is associated with in order to list the attributes of that data set. Optionally, you can list the problems the data set is associated with.

The subcommand that lists data set attributes is LISTDSN. Assuming your TSO user-prefix is IPCSU1, a sample subcommand is:

```
listdsn dataset(dump.summary) problems
```

This subcommand displays the type, management, and description attributes of data set IPCSU1.DUMP.SUMMARY and the problems it is associated with.

The subcommand

```
listdsn all problems
```

lists the type, management, and description attributes of every data set in the data set directory and the problems they are associated with.

The subcommand

```
listdsn
```

lists the type, management, and description attributes of the current data set. It does not list the problems that data set is associated with.

# Chapter 4. Analyzing Dumps

The following examples demonstrate how you might use IPCS subcommands to view, verify, and analyze dumps. You do not have to own a problem to examine a dump associated with it. The sample commands and subcommands use the TSO command syntax and data set naming conventions. For a complete description of these conventions, see the OS/VS2 TSO Command Language Reference.

IPCS provides full-screen capability (DSPL3270) for display of dump data as well as line-oriented capability to analyze and display dump data. All IPCS subcommands except DSPL3270 are line-oriented. Within an IPCS session you may switch between full-screen mode and line-oriented mode.

In the examples, user-entered input lines are shown in lowercase and system responses, when they are shown, are in uppercase.

## Using DSPL3270

DSPL3270 is a full-screen subcommand. It allows you to display virtual data from a dump, to alter the display, and to keep track of already-located information, control blocks, etc.

All DSPL3270 examples assume you are using a 3275 or 3277 Model 2 display terminal. DSPL3270 provides several standard operations that you can execute with a program function (PF) key, if your terminal has them, or with the cursor and ENTER key. You can also use the cursor and ENTER key to specify dump data to display.

An example of the subcommand is:

```
dspl3270
```

Figure 4-1 shows the screen as it might look after you execute the subcommand.

```
3END  7<-SCROLL+>8  9STACK  10<-SKIP+>11
SK 00E090
RF CVT
ADDR 000008 ASID 0014 FMT X AREA A LINES/AREA: A 19 B 00 C 00 D 00 SKIP 000200
SUBCMND/CLIST:
000000                   00000000 00000000 0000E090 00000000 070C1000 00D4A832
000020  070C2000 00D4A0CE 070C0000 00D4A000 040C1000 0001259A 070E0000 00000000
000040  0006BE78 0C000001 000088A0 0000E090 B4F35AFF 00000000 040C0000 0004555E
000060  040C0000 000308AA 000C0000 000367C4 00080000 0000AF18 040C0000 00045766
000080  00000000 00001004 00020001 00D4A000 00000000 00000000 00000000 00000000
0000A0  00000000 00000000 20000000 001FE298 FFFFFFFF 00000000 000001D3 00000000
0000C0  00000000 00000000 00000000 00000000 00000000 00000000 7FFFFFEE 4523CFF8
0000E0  8B26520F 19744000 00000000 00000000 00000000 00000000 00000000 00000000
000100  03000148 00000000 00000000 00410080 00000000 00000000 00000000 00000000
000120  00000000 00000000 00000000 00000000 FFFFFFFF 0000002A FFFFFFFF FFFFFFFF
000140  FFFFFFFF FFFFFFFF 01010523 447EE000 23323339 42483849 00000000 00000000
000160  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000180  00FFE398 00FFE1F0 00FF6634 00FFDE28 800123B0 00012070 00000508 000051B0
0001A0  0001306E 0001406C 00007F80 00FF9828 00008110 00FFB3E8 00005080 00000000
0001C0  C080EC40 0F1D7C00 FC000000 00000000 00000000 00000000 00000000 00000000
0001E0  00000000 00000000 00000000 00000000 00000000 00000000 0FC00000 001E6560
000200  D7E2C140 00000040 00FFE1F0 001FE1F0 00FFDE28 001FDE28 00A258F8 00A258F8
000220  00FED098 00FED098 00000000 00B74340 00000000 00000000 00000000 00B74E40
000240  040C1000 00D4A832 00000000 00000000 070C0000 00D4EA76 040C0000 0001A8EC
```

**Figure 4-1. DSPL3270 Screen Format**

The top five lines of the screen define the data displayed and provide the standard functions. The bottom 19 lines are the data display area. The initial data in the display area always starts at the current address, the last address referenced by an IPCS subcommand. Since other IPCS subcommands set and change the current address, the initial data displayed when you execute DSPL3270 against a given dump varies with the current address when you execute DSPL3270.

Figure 4-1 shows the default screen format.

DSPL3270 uses the following values for the default screen format:

- ADDR - the current address (in hexadecimal).
- ASID - the ASID associated with the current address.
- FMT - X, hexadecimal format.
- AREA - A.
- LINES/AREA - A 19.
- SKIP - 200, X'200' bytes.

If you use DSPL3270 to view a dump, terminate DSPL3270, and later, in the same or another session, view that dump again, DSPL3270 recreates the screen format as it was defined when you terminated the previous examination. The only exception is the data shown in the data display area, which is determined by the current address when you execute DSPL3270. Otherwise, the screen format is remembered from your last examination of that dump.

*Data Display Area*

The data display area is the bottom 19 lines of the screen. Each line of data consists of the 6-character hexadecimal address of the first word of data on that line, followed by the data itself, in either EBCDIC or hexadecimal format.

You can treat this as a single area or you can divide it into as many as four subareas. The subareas are named A, B, C, and D. In this example, the data display area is a single 19-line area named A.

To divide it, type the number of lines you want for each area after the corresponding area name on line four. If the total number of lines is less than 19, DSPL3270 displays only the number of lines you specified. If the total exceeds 19, DSPL3270 truncates the area exceeding 19 and assigns zero lines to the remaining areas. An area is not displayed if it has no lines assigned.

The areas are displayed in alphabetical order from top to bottom in the display area.

As you manipulate the data in the areas, their contents change and allow you to view up to four different areas of the dump. Figure 4-2 shows three areas, A, C, and D, with 6, 4, and 2 lines respectively. Since B is not defined, it is not shown.

```
3END  7<-SCROLL+>8  9STACK  10<-SKIP+>11
SK 00E090
RF CVT
ADDR 000000 ASID 0014 FMT X AREA A LINES/AREA: A 06 B 00 C 04 D 02 SKIP
SUBCMND/CLIST:
000000  040C0000 00017EBA 00000000 00000000 0000E090 00000000 070C1000 00D4A832
000020  070C2000 00D4A0CE 070C0000 00D4A000 040C1000 0001259A 070E0000 00000000
000040  0006BE78 0C000001 000088A0 0000E090 B4F35AFF 00000000 040C0000 0004555E
000060  040C0000 000308AA 000C0000 000367C4 00080000 0000AF18 040C0000 00045766
000080  00000000 00001004 00020001 00040011 00D4A000 00000000 00000000 00000000
0000A0  00000000 00000000 20000000 001FE298 FFFFFFFF 00000000 000001D3 00000000


0001E0  00000000 00000000 00000000 00000000 00000000 00000000 EFC00000 001E6560
000200  D7E2C140 00000040 00FFE1F0 001FE1F0 00FFDE28 001FDE28 00A258F8 00A258F8
000220  00FED098 00FED098 00000000 00B74E40 00000000 00000000 00000000 00B74E40
000240  070C1000 00D4A832 00000000 00000000 070C0000 00D4EA76 040C0000 0001A8EC


000140  FFFFFFFF FFFFFFFF 01010523 447EE000 23323339 42483849 00000000 00000000
000160  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Figure 4-2. DSPL3270 Screen With Three Subareas

If you've defined more than one area, one of them must be designated the current data display area. The name following AREA on line four designates the current data display area. Designation of the current data display area is important because the data selection, scroll, skip, and format options apply only to the current data display area. Likewise, the ADDR and ASID fields describe the data in the current data display area. To make a defined area the current data display area, type its name after AREA and press ENTER.

### *The Current Address*

The current address is displayed after ADDR on line four. The current address is the beginning address of the data displayed in the current display area.

You can directly specify the current address by typing an address after ADDR or an address space identifier after ASID, or both, and pressing ENTER. The address and address space identifier must be in hexadecimal; leading zeros are not required. After you press ENTER, DSPL3270 displays the data at the specified address in the current data display area.

The current address is updated by selecting an address, scrolling, or skipping. In each case, DSPL3270 changes the current address to reflect the beginning address of the new data in the current data display area.

### *Selecting Addresses*

You can specify data to be displayed in the current data display area by positioning the cursor under the address of the data and pressing ENTER.

#### Implicit Scrolling

If you position the cursor under any address at the left margin of the screen and press ENTER, DSPL3270 moves that line to the top of the current data display area and makes that address the current address. This is done even if the address you selected is not in the current area but is in one of the other areas.

#### Indirect Addressing

If you position the cursor under any displayed word in the dump and press ENTER, DSPL3270 treats the contents of the selected word as an address and displays the data at that address in the current data display area and makes that address the current address. This also is done even if the word is not in the current area.

#### Selecting Stack Elements

If you position the cursor under an address in the stack and press ENTER, DSPL3270 makes that address the current address and displays the data at that address in the current data display area.

### *Scrolling Data*

To scroll data forward (toward the end of the dump) either press PF8 or position the cursor under the right half of SCROLL on the top line and press ENTER. DSPL3270 scrolls the current data display area forward by

the number of lines in that area.

To scroll data backward (toward the beginning of the dump), either press PF7 or position the cursor under the left half of SCROLL and press ENTER. DSPL3270 scrolls the current data display area backward by the number of lines in that area.

### *Skipping Data*

You can scan through the dump, skipping a specified number of bytes at a time, by using the SKIP feature. Position the cursor after SKIP on line four and type the number of bytes (in hexadecimal) that you want to skip. Then, to skip forward (toward the end of the dump) either press PF11 or position the cursor under the right half of SKIP on line one and press ENTER. DSPL3270 adds the specified number to the current address, displays the data at the new address in the current data display area, and makes the new address the current address.

To skip backward (toward the beginning of the dump), either press PF10 or position the cursor under the left half of SKIP on line one and press ENTER. DSPL3270 subtracts the specified number from the current address, displays the data at the new address in the current data display area, and makes the new address the new current address.

### *Formatting Data*

You can display the data in the current data display area in either hexadecimal or character format, but not both. To display the data in hexadecimal format (the default), type X after FMT on line four. To display the data in character format, type C after FMT. In either case, the specified format applies only to the current data display area.

When you specify hexadecimal format, each line of the current data display consists of:

- The 6-character hexadecimal address of that line of data.
- The hexadecimal representation of 8 words of data. A blank separates each word of data.

  The first line of the area is intensified and may contain less than 8 words of data.

  If the data you requested is not available in the dump, each display line consists of the address of that line followed by 'N/A'. Data availability is determined in page increments.

When you specify character format, each line of the current data display consists of:

- The 6-character hexadecimal address of that line of data.
- The EBCDIC character representation of 16 words of data. Any byte that cannot be displayed as a character is shown as a period. A vertical bar separates consecutive 8-byte fields of data.

  The first line of the area is intensified and may contain less than 16 words of data.

If the data you requested is not available in the dump, each display line consists of the address of that line followed by 'N/A'. Data availability is determined in page increments.

### *Stacking Addresses*

Lines two and three of the display are an 11-element stack in which you can save addresses. Line two displays the addresses. Line three displays an optional 6-character reference note for each stack element. DSPL3270 also saves, but does not display, the ASID for each stack address.

On entry to DSPL3270 using the initial format, the stack is primed with the address of the CVT.

The address added to the stack is always the current address (the address after ADDR on line four). This address is explicitly added to the stack if you use the STACK function; it is added to the stack even if it duplicates an address already in the stack. To explicitly add the current address to the stack, either press PF9 or position the cursor under STACK on the top line and press ENTER.

Under other circumstances, the current address is automatically added to the stack when it (the current address) is changed. Automatic stacking depends on how the current address was <u>obtained</u>, not how it is <u>changed</u>.

If the current address is obtained by one of the following means, it is <u>not</u> added to the stack:

- Skipping (SKIP function).
- Explicit scrolling (SCROLL function).
- Implicit scrolling (selecting the address of a line of data in the data display area.)
- Selecting an address already in the stack.

If the current address was obtained by any other means (such as typing a new address, setting the current address with another IPCS subcommand or CLIST, using indirect addressing, etc.) it is automatically added to the stack.

New stack entries are added at the right. If the stack is full, DSPL3270 deletes the oldest (leftmost) element, shifts the remaining elements to the left, and adds the new element at the right. You can delete a stack element by typing a slash (/) in the first position of its note field and pressing ENTER.

### *Executing IPCS Subcommands, TSO Commands, and CLISTs*

To exit DSPL3270 and execute an IPCS subcommand, TSO command, or CLIST, type the subcommand or CLIST name and its operands or parameters on line five and press ENTER. When the subcommand, command, or CLIST terminates, DSPL3270 is invoked and redisplays the screen as it was before the subcommand, command, or CLIST executed, with one possible exception. If the subcommand, command, or CLIST altered the current address, the current data display area displays data beginning at the new address and that address is shown after ADDR.

To display the storage represented by symbols in your symbol table, use this line to execute an IPCS subcommand that sets the current address to the address of the symbol you want displayed. When control returns to DSPL3270, the current data display area displays the storage at the current address.

DSPL3270 performs validity checks on input on line 4. If it detects errors, the subcommand stops processing the input, intensifies the erroneous field or fields, and displays an error message on the top line of the screen.

*Terminating DSPL3270*

To terminate DSPL3270, either press PF3 or position the cursor under END on the top line and press ENTER. The DSPL3270 subcommand removes any dump data from your screen and returns you to line oriented mode.

## Checking System Components

Certain major system components involve many control blocks that might be important in analyzing a dump. IPCS provides five subcommands to analyze certain control blocks associated with various components.

To verify auxiliary storage manager control blocks, use ASMCHECK. This subcommand locates and validates control blocks used for paging, determines the location of each paging device, and accumulates statistics about paging requests. An example of this subcommand is:

```
asmcheck
```

The COMCHECK subcommand locates and validates control blocks for the communications task and accumulates statistics about messages and consoles. An example of this subcommand is:

```
comcheck
```

Another system component involving several control blocks is ENQ/DEQ. The ENQCHECK subcommand locates, validates, and displays the ENQ/DEQ control blocks. An example of this subcommand is:

```
enqcheck
```

The IOSCHECK subcommand checks the control blocks and queues used by the I/O supervisor. An example of this subcommand is:

```
ioscheck
```

The SUMMARY subcommand locates and validates the ASCB, TCBs, and RBs for one or a list of address spaces. An example of this subcommand is:

```
summary anomaly
```

The ANOMALY keyword restricts the display to TCBs that have abnormally terminated or that are terminating. If you do not specify ANOMALY, the subcommand displays data for all TCBs and RBs.

## Using Data Description Operands

Certain dump analysis subcommands require a description of the data they are to process. Data description operands provide such descriptions. For a complete description of these operands, see "Data Description Operands" in Chapter 5. With these operands you tell a dump analysis subcommand where the data item is, how big it is, what kind of processing to do to the address, what format to use when displaying or printing the data, and whether it is an array or a scalar. You can also associate a descriptive remark with the data item.

Such attributes are used by dump analysis subcommands to process the data and are also used by you to describe the data for displaying, printing, and your own future reference.

You would specify these attributes when you are displaying or printing data, creating a symbol table entry for it, or using it in an operation performed by a subcommand. The format attributes and the remark are usually used only when displaying or printing the data or when creating a symbol table entry for it (see "Using the Symbol Table" later in this chapter). This latter case allows you to include the data item's attributes in the symbol table entry for future reference.

Instead of specifying data description operands you can, if appropriate, use the X symbol (see "Using the Symbol Table" later in this chapter). X is the symbol defined by IPCS to represent the current address in the current dump. Various subcommands modify X as they process. If the storage you want to process is addressed by X, you may use it instead of more complicated data description operands.

Data description operands fall into five categories:

- Address and length keywords - these specify the location and size of the data.
- Address processing keywords - these specify the processing the subcommand is to do to the address: whether to simulate prefixing, dynamic address translation, etc.
- Attribute keywords - these describe the type of data at the specified address and how it is to be displayed or printed.
- Array keywords - these describe data structures, arrays, and scalars.
- Remark keyword - this associates a text string with the data item.

The address and length operands specify the location and size of the data item. The following are all valid addresses or address expressions:

- ADDR1 - is a symbolic address, the address associated with symbol ADDR1.
- X - is a symbolic address, the current address in the dump.
- +73 - is a relative address, X'73' bytes beyond X, the current address.

- 4C. - is a literal address, location X'4C'.
- 7R - is a general-purpose register, register number 7.
- 6D - is a floating point register with double precision, register number 6.
- 6C.% - is an indirect address, the location pointed from location X'6C'.
- 10R%% - is an indirect address, the location pointed to from the location pointed to from general-purpose register 10.
- ADDR1-73n+4C - is an address expression, ADDR1 minus 73 (decimal) plus X'4C'.

Two typical uses of the full range of data display operands are the EQUATE and LIST subcommands. For example:

```
equate abc a72f4. asid(1) length(80) area scalar
  remark('input params from exec card')
```

creates a symbol table entry for the symbol ABC. The symbol is associated with an 80-byte area beginning at X'A72F4'. ASID indicates that this address is virtual and IPCS simulates dynamic address translation for ASID(1). The AREA, SCALAR, and REMARK keywords specify information about this area that you want for future reference. AREA indicates that it is neither a module nor a control block. SCALAR indicates it is a single block of storage, not an array. The REMARK is your description of the 80-byte area.

Having created the symbol, you can now display its storage with the LIST subcommand. For example:

```
list abc
```

displays the 80-byte buffer.

If you have not defined the symbol but still want to display that storage, specify

```
list a72f4. length(x'50') area scalar
```

This subcommand displays the same data (except the remark), in the same format, as the previous example.

If you want to see the 20-byte field following the buffer and you want it displayed in character format, specify

```
list abc+80n length(20) c
```

Another way to accomplish the same thing is:

```
equate x abc+80n length(x'14')

list x c
```

This example sets X, the current address, to the end of the buffer and specifies a length of X'14'. The LIST subcommand then displays this area in character format.

Suppose you have located a segment table at X'5A2400'. If you want to display the sixth through tenth entries (zero origin), specify

```
list 5a2418. length(4) entries(6:10)
```

This displays the five segment table entries beginning at X'5A2418' (each segment table entry is four bytes). The total length of the five entries is 20 bytes.

To list the contents of the general-purpose registers as they were at the time of the dump, specify:

```
list 0r:15r terminal
```

this displays all 16 general-purpose registers at your terminal.

If you specify:

```
list 0d:6d
```

you display the four floating-point double precision registers in hexadecimal.

Note that when a symbol is assigned an offset, the offset is stored separately and is not added to the base address. Thus,

```
equ a 1000.+10
```

creates a symbol record called A representing an address of 1000 and an offset of 10. In a similar manner, then,

```
equ b a+20
```

uses only the address portion of A (1000) and creates a symbol, B, with an address of 1000 and an offset of 20.

This design is used for the concept of fields within control blocks. Each field is considered at an offset from the beginning of the control block, not at an offset from another field.

### *Specifying Data Description Operands*

The way you specify data description operands varies slightly depending on the syntax of the subcommand. If the syntax is:

```
FIND ADDRESS(data-descr)
```

you can put only the address expression within the parentheses. Any other data description operands you specify must be outside the parentheses. For example:

```
find address(54.%%) length(9) asid(22)
```

If the subcommand syntax is:

```
LIST data-descr
```

there are no parentheses except those required by the data description operands themselves. Thus you could specify:

```
list 54.%% length(9) asid(22)
```

*Resolving Default Data Description Operands*

An IPCS subcommand has several sources of defaults for data description operands. A subcommand resolves the operands by the following steps, in descending priority order:

1. The operands specified on the subcommand.

   If the default ASID is 2 and you specify ASID 7, the subcommand uses ASID 7.

2. The attributes of the symbol, if one is used.

   Suppose the default ASID is 6 and you execute the following subcommands:

   ```
   equate aaa 84. asid(2) length(30)

   list aaa length(20)
   ```

   LIST displays 20 bytes starting from X'84' in address space 2. Even though the default ASID is 6 for both subcommands, since you specified ASID 2 for the symbol AAA, when you omit the ASID keyword on the LIST subcommand, it uses the attribute of the symbol.

   If you specify:

   ```
   list aaa+4
   ```

   since location AAA+4 is within the area represented by AAA, you would get 26 bytes starting from location X'88' in address space 2.

   However, if you specified:

   ```
   list aaa+50
   ```

   since AAA+50 is outside the area represented by AAA, the number of bytes displayed is determined by the default length rather than the length associated with AAA. The data displayed begins at location X'D4'.

3. The defaults specified by the SETDEF subcommand.

   If the default length is 50 and you create a symbol and specify no length, the EQUATE subcommand records a length of 50.

4. Defaults built into the IPCS subcommand, such as the AREA and SCALAR attributes.

## Using the Symbol Table

IPCS maintains a symbol table for every dump that you process with the dump analysis subcommands. The symbol table is kept in the dump directory data set and contains information that makes future references to that dump faster and more efficient. When you first process a new dump (one you have never processed before), IPCS reads the dump and initializes its symbol table. This occurs for independent dump data examination as well as for dumps associated with problems.

A symbol is a label associated with a location in a dump. The label can be a maximum of 31 alphameric characters and the first character must be

alphabetic. Symbols are recorded in a symbol table.

Also recorded in the symbol table, for each symbol, are attributes of the storage it is associated with. The attributes include the address, size, address space identifier, CPU, etc. The symbol itself can have the DROP or NODROP attribute. Symbols with the NODROP attribute can be deleted with the DROPSYM subcommand only if you use the PURGE keyword. You do not need PURGE to delete a symbol with the DROP attribute. IPCS uses the symbol X to represent the current address in the current dump. X is automatically given the NODROP attribute. You can specify X as an address operand to many dump analysis subcommands. Various dump analysis subcommands might modify X as a result of their processing.

When processing a stand-alone dump, the IPCS dump analysis subcommands might have to simulate dynamic address translation. There are many control blocks that the subcommands use to do this translation. In general, any dump analysis subcommand that takes a data description operand may locate and validate the control blocks it needs to perform its function (if they are not already located) and make entries for them in the symbol table and storage map (see "Validating Structures" later in this chapter).

### *Naming Conventions*

If a dump analysis subcommand needs a control block, it automatically locates the control block, validates it, and creates an entry for it in the symbol table and storage map (see "Validating Structures" later in this chapter). When it does this, the subcommand uses the following conventions to generate the symbol name for the control block. All numbers, designated "nnn", are decimal numbers, except where specified differently.

The naming conventions are:

- ASCBnnn - The address space control block for address space nnn, STRUCTURE(ASCB).
- ASMVT - The system auxiliary storage management vector table, STRUCTURE(ASMVT).
- ASVT - The system address space vector table, STRUCTURE(ASVT).
- ASXBnnn - The address space extension block for address space nnn, STRUCTURE(ASXB).
- CDEpgmname - A contents directory entry for entry point pgmname, STRUCTURE(CDE).
- COMMON - The system common area, AREA(COMMON).
- CSD - The common system data area, STRUCTURE(CSD).
- CVT - The system communications vector table, STRUCTURE(CVT).
- FINDAREA - The area currently being searched by the FIND subcommand. This may be explicitly changed with the EQUATE subcommand and implicitly changed with the FIND subcommand.

- IOCOM - The I/O communications area, STRUCTURE(IOCOM).
- IOX - The I/O communications area extension, STRUCTURE(IOX).
- LPDEpgmname - The link pack directory entry for pgmname, STRUCTURE(LPDE).
- PART - The page address resolution table, STRUCTURE(PART). Defined only by the ASMCHECK subcommand.
- 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).
- PGTnnnaa - The page table for address space nnn, segment aa, STRUCTURE(PGTE).

  The page table for segment 0 of address space 1 is PGT001AA; for segment 1, PGT001AB, etc.
- PRIVATE - The private area, AREA(PRIVATE).
- PVT - The system paging vector table, STRUCTURE(PVT).
- SGTnnn - The segment table for address space nnn, STRUCTURE(SGTE).
- TCBnnnaa - The task control block for address space nnn, in position aa in the priority queue, STRUCTURE(TCB).

  The highest priority TCB in address space 1 is TCB001AA; the next TCB on the queue is TCB001AB, etc.

  The last two characters in this name are alphabetic and range from AA through AZ, BA, ... BZ, ... etc.
- UCBddd - The unit control block for device ddd, STRUCTURE(UCB). ddd designates the device address in hexadecimal.
- UCM - The unit control module, STRUCTURE(UCM).
- XLpgmname - An extent list for entry point pgmname, STRUCTURE(XTLST).

If you explicitly create or modify one of the symbols listed above, rather that let IPCS create or modify it, you might bypass IPCS's validity checking process. For example, if you create the symbol UCB00E with the following subcommand:

```
equate ucb00e 4140.
```

and later use the FINDUCB subcommand to locate the UCB for device 00E, the FINDUCB subcommand finds the symbol in the symbol table and displays the storage at the address associated with that symbol. Since your EQUATE subcommand did not specify STRUCTURE(UCB), the storage at X'4140' was not validity checked to ensure that it is a UCB.

### *Creating, Listing, and Deleting Symbol Table Entries*

The following subcommands create or delete entries in the symbol table. Some subcommands allow you to create or delete the symbols, others let IPCS create the symbols.

If you specify

```
 list asvt structure(asvt) ...
```

you are asking the subcommand to find and display the ASVT wherever it is and however the subcommand can find it.

If the symbol ASVT is not already in the symbol table, LIST locates it, verifies it, enters it in the symbol table with the attribute of STRUCTURE(ASVT), and lists it.

But, if you specify

```
 list 5820.% structure(asvt) ...
```

LIST displays the storage at the location pointed to from X'5820'. The subcommand does not know if this really is the ASVT. It does not create a symbol table entry for ASVT nor does it use such an entry if it already exists. In this example, you are telling the subcommand that the storage at the specified location is the ASVT. The subcommand verifies that the storage is the ASVT but does not make a symbol table entry for the ASVT.

To make an entry in the symbol table, use the EQUATE subcommand. This subcommand allows you to specify the symbol name and its address, size, and attributes.

A sample EQUATE is

```
 equate failingtcb 51368. length(360) structure(tcb)
   x remark('tcb that caused the dump')
```

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. Since the NODROP keyword is not specified, this name can be deleted from the symbol table.

The subcommand

```
 equate jstcb
```

creates a symbol table entry for JSTCB. By default, the address and attributes associated with JSTCB are those associated with X, the current address.

You can set X to a specific address by:

```
 equate x 522836.
```

This sets X to address X'522836'.

You can also use EQUATE to change the attributes of symbols already defined. For example:

```
 equate buffer1 55280. length(80) asid(3) drop

 equate buffer1 buffer1 nodrop cpu(2)
```

The first EQUATE creates the symbol BUFFER1 and gives it certain attributes. The second EQUATE changes the DROP attribute to NODROP and specifies a CPU. You can change the attributes of any symbol in the symbol table whether you created it or whether IPCS subcommands created it for you.

The FINDMOD, FINDUCB, RUNCHAIN, SCAN, and SUMMARY subcommands locate modules and control blocks in the current dump. FINDMOD locates a module, FINDUCB locates a UCB, RUNCHAIN processes a specified chain of control blocks, SUMMARY locates ASCBs, TCBs, and RBs. As each subcommand finds what it is looking for, it generates a name for it and puts that name in the symbol table along with the attributes of the module or control block it represents.

You can display some or all of the entries in the symbol table for a given dump. The LISTSYM subcommand lists the specified names from the current dump data set. A example of LISTSYM is:

```
listsym * print terminal
```

This subcommand causes all the symbols associated with the current dump data set to be displayed at your terminal and directed to the IPCSPRNT data set. The keywords PRINT and TERMINAL are not necessary unless they override the defaults.

The subcommand

```
 listsym x
```

lists the address and attributes currently associated with X.

Symbol table records for each dump are kept in your dump directory data set. If the space allocated for that data set is exhausted, you may want to delete some entries from your symbol table.

The DROPSYM subcommand can delete some or all of the symbol table entries for a dump. For example:

```
 dropsym * purge
```

deletes every entry in the symbol table, including X, for the current dump. If you omit the PURGE keyword, this example deletes all symbols except those with the NODROP attribute.

If you drop X, IPCS may recreate it with an address of zero, using the default ASID and length attributes.

The subcommand

```
 dropsym tcb
```

deletes the name TCB. Since PURGE is not specified, it does not delete TCB if it has the NODROP attribute.

The subcommand

```
 dropsym (cde workarea1:workarea7)
```

deletes the name CDE. It also deletes the names WORKAREA1 through WORKAREA7, inclusive, and names beginning with WORKAREA1 through names beginning with WORKAREA7.

If you specify a single symbol or a list of symbols, DROPSYM deletes those symbols. If you specify a range of symbols, DROPSYM deletes all symbols whose names begin with the first character string through all symbols whose names begin with the second character string. If any such names have the NODROP attribute, the above example does not delete them.

## Using the Storage Map

IPCS maintains a storage map for every dump that you process with dump analysis subcommands. The storage map is kept in the dump directory data set and contains information that makes future references to that dump faster and more efficient.

The storage map is a map of storage in a dump, arranged in ascending address order. Each storage map entry contains a pointer to the dump records that contain the corresponding storage. IPCS organizes the storage map for a dump by address space, address within the address space, and data type.

### *Creating Storage Map Entries*

The storage map describes only storage that contains AREA(xxx), MODULE(yyy), or STRUCTURE(zzz) data areas. Dump initialization creates entries in the storage map (see "Processing Dumps" in Chapter 2). Various dump analysis subcommands also create storage map entries when you specify an area with a data description keyword of AREA(name), MODULE(name), or STRUCTURE(name) (see "Data Description Operands" in Chapter 5) or when the subcommand locates or validates such an area itself (see "Naming Conventions" earlier in this chapter and "Validating Structures" later in this chapter).

The subcommand

```
equate failingtcb 51368. length(360) structure(tcb)
```

creates an entry in the storage map for address X'51368' (if the data at X'51368' is validated as a TCB), and in the symbol table, for FAILINGTCB. The keywords AREA and MODULE force storage map entries also.

The SCAN subcommand is especially powerful in mapping the storage in a dump. SCAN works within the specified range of addresses in the storage map. SCAN only validates entries in the storage map with the STRUCTURE data type, though it creates storage map entries for AREA, MODULE, and STRUCTURE. Since dump initialization creates the original storage map entries for a dump, SCAN always has some entries to work with.

Suppose the map for a new dump contains only the entries for the CVT and the private area. The subcommand

```
scan
```

causes SCAN to make one pass through the storage map. If it finds a structure listed in "Validating Structures" later in this chapter, the subcommand validates it. SCAN never validates a specific storage map entry a second time. Subsequent passes through the map only validate those entries not already completely validated.

If you specify

```
scan range(0.:500000.)
```

the subcommand processes only storage map entries originating between the specified addresses. An entry in that range may point to an AREA, MODULE, or STRUCTURE outside that range. SCAN validates that AREA, MODULE, or STRUCTURE and makes a map entry for it but does not initiate

further validations from the new map entry since it doesn't fall in the specified range.

If you specify

```
 scan range(8187C.) length(x'1000')
```

SCAN processes the storage map entries between X'8187C' and X'8287C', inclusive.

If you specify

```
 scan asid(5)
```

it is equivalent to

```
 scan range(0.:ffffff.) asid(5)
```

If you specify

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

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

If you specify

```
 scan range(private) area(private)
```

SCAN ensures that the symbol PRIVATE is in the symbol table. It then scans the private area. In fact, it is probably a rescan since the private area was scanned during dump initialization.

### *Validating Structures*

In the process of automatically locating various control blocks or structures and generating names for them (see "Naming Conventions" earlier in this chapter), IPCS also validates them. You can cause IPCS to validate a control block by using the STRUCTURE keyword with a structure type from the list in this section. Validating a control block means checking various attributes and characteristics to ensure that it is the control block it is supposed to be. This means checking the control block's boundary alignment and the offset of certain special fields in it; such as acronyms, special values, masks, counts, pointers to other control blocks (which may themselves be validated), etc.

If you enter the subcommand

```
 equate mytcb 522c0. structure(tcb)
```

the EQUATE subcommand 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.

If you specify:

```
 scan 73260. length(x'2000')
```

the subcommand processes existing storage map entries in the range X'73260' through X'75260'. In validating each entry not already completely validated, SCAN generates new storage map entries for AREAs, MODULEs, and STRUCTUREs it finds. When it finds a structure that it can validate, SCAN validates it and in the process may generate yet more storage map entries.

IPCS subcommands do not validate every structure they find. There are some structures they make storage map entries for but do not validate.

The following list shows the control blocks the subcommands validate and make storage map entries for:

| | | |
|---|---|---|
| ASCB | LPDEMINOR (3) | RB |
| ASVT | PCCA | SGTE |
| ASXB | PCCAVT | SIRB (2) |
| CDE | PFT | SVRB (2) |
| CDEMAJOR (1) | PFTE | TCB |
| CDEMINOR (1) | PGTE | TIRB (2) |
| CVT | PRB (2) | UCB |
| CVTXTNT2 | PSA | UCBCTC (6) |
| IOCOM | PVT | UCBDA (6) |
| IOX | QCB | UCBGFX (6) |
| IRB (2) | QCBMAJOR (4) | UCBTAPE (6) |
| LCCA | QCBMINOR (4) | UCBTP (6) |
| LCCAVT | QEL | UCBUR (6) |
| LPDE | QELTYPEL (5) | UCB3270 (6) |
| LPDEMAJOR (3) | QELTYPER (5) | XTLST |

Notes:

(1) These control blocks are validated as if they were specified as CDE and the correct structure type is stored in the symbol table and storage map.

(2) These control blocks are validated as if they were specified as RB and the correct structure type is stored in the symbol table and storage map.

(3) These control blocks are validated as if they were specified as LPDE and the correct structure type is stored in the symbol table and storage map.

(4) These control blocks are validated as if they were specified as QCB and the correct structure type is stored in the symbol table and storage map.

(5) These control blocks are validated as if they were specified as QEL and the correct structure type is stored in the symbol table and storage map.

(6) These control blocks are validated as if they were specified as UCB and the correct structure type is stored in the symbol table and storage map.

The following list shows the control blocks the subcommands do not validate but make storage map entries for:

| | | |
|---|---|---|
| CSD | OUSB | SPQE |
| CSCB | OUXB | SRB |
| DCB | PCB | TIOT |
| GDA | RMCT | TQE |
| JQE | RQE | TSB |
| JESCT | RSMHD | UCBEXT |
| JSCB | RTCT | UCM |
| LDA | RTM2WA | WSAVTC |
| OUCB | SCVT | WSAVTG |

*Listing and Deleting Map Entries*

You can display some or all of the entries in the storage map for a given dump. The LISTMAP subcommand lists the entries for a specified address space or range of addresses.

The subcommand

```
listmap range(5000.:10000.) terminal noprint
```

requests a display, at your terminal only, of the map entries for the current dump that originate between the addresses X'5000' and X'10000'.

If the space allocated to your dump directory is exhausted, you may want to delete some entries from your storage map. You may also delete storage map entries because the dump they represent is no longer associated with a problem you are working on (the dump may still be associated with a problem another user is working on).

The DROPMAP subcommand can delete some or all of the storage map entries for a dump. For example:

```
dropmap range(0.:50000.)
```

deletes all storage map entries for the current dump that originate between the virtual addresses X'0' and X'50000' inclusive.

## Examining Dump Data

The symbol table and storage map reflect the location and attributes of named entities already located in a dump. IPCS provides subcommands to look at those areas or at any other areas in a dump.

The LIST subcommand displays the storage at a specified address for the specified length. For example:

```
list 51634. length(50)
```

displays 50 (decimal) bytes of data beginning at X'51634'. You can omit the LENGTH keyword if the default length is satisfactory. By default, the data is displayed four words on a line, first in hexadecimal then in character format.

If you want to locate a character or hexadecimal string in a dump, use the FIND subcommand. FIND can be used repeatedly to locate multiple occurrences of the string.

The subcommand

```
find c'mymod1' address(40000.:100000.)
```

searches for the character string MYMOD1 between the addresses X'40000' and X'100000', inclusive.

If FIND locates MYMOD1 and you want the next occurrence of it, enter

```
find
```

By specifying no operands, FIND continues searching for MYMOD1, beginning this search one byte after the address where the previous search ended. Since you did not alter the search range, the subcommand uses the same ending address specified on the previous FIND.

If you entered

```
find address(61354.)
```

the subcommand searches for MYMOD1 but starts the search at X'61354' and continues to X'FFFFFF'.

When you execute FIND, the subcommand sets X (the current address) and the symbol FINDAREA to the beginning of the search range. Thus, if you enter

```
find x'0000000c' address(62b00.:8d000.)
```

FIND sets X and FINDAREA to X'62B00'. If FIND locates the search argument, it sets X to the address of the found data. If FIND does not locate the search argument, it leaves X at X'62B00'.

Assuming MYMOD1, in an earlier example, is the name of a module, an easier way to locate it is the FINDMOD subcommand. For example:

```
findmod mymod1
```

searches, in order:

- The symbol table (in case the module has already been found).
- The master CDE chain (modules in the MLPA).
- The link pack area directory (modules in the LPA).

If it finds the module on the CDE chain or in the link pack area directory, the subcommand makes entries for the module in the symbol table and storage map. It creates an entry for MYMOD1, and either CDEMYMOD1 and XLMYMOD1, or LPDEMYMOD1, depending on where it found the module.

To locate the unit control block for a device, use FINDUCB. The subcommand

```
finducb 00e
```

locates the UCB for device 00E (leading zeros optional). If the symbol UCB00E exists in the symbol table, it is used. Otherwise, the subcommand creates entries for the UCB in the symbol table and storage map. In this case, it uses the name UCB00E.

The RUNCHAIN subcommand processes a chain of control blocks using a chain pointer that you specify. Optionally, you can request it to create symbol table entries for each control block in the chain using a character string that you specify.

Suppose you have located the ASCB for a failing address space. Using the ASCB you get the address of the highest priority TCB in that address space. Now use the RUNCHAIN subcommand to scan the chain of all TCBs in the address space. If the first TCB is at X'728C0', the subcommand

```
runchain address(728c0.) link(x'74') name(ascb1tcb)
```

starts with the TCB at X'728C0' and, using the TCBTCB field at offset X'74', traces the chain of TCBs. The subcommand produces a name for each TCB using the character string ASCB1TCB and adding a sequence number for each TCB. In this example, the length of the chain defaults to 999, therefore RUNCHAIN generates the names ASCB1TCB001, ASCB1TCB002, .... If there are more than 999 TCBs in the chain RUNCHAIN stops at 999.

## Executing User-Written Routines

The AMDPRDMP service aid, which prints various kinds of dumps, permits you to specify exit routines to be executed when it encounters a user-specified verb in the input stream, or when it encounters an ASCB or a TCB in the dump. IPCS provides the same interface, as documented in OS/VS2 MVS System Programming Library: Service Aids. Your routines must be in a library available to IPCS, such as a step library, job library, or system library.

Use the ASCBEXIT subcommand to invoke your ASCB exit. The subcommand

```
ascbexit chekascb 7
```

invokes the routine named CHEKASCB and passes it ASID 7.

The TCBEXIT subcommand invokes your TCB exit. The subcommand

```
tcbexit testtcb 715b0.
```

invokes routine TESTTCB, passing it the TCB address X'715B0'.

The VERBEXIT subcommand invokes your user control statement exit. The subcommand

```
verbexit history 'rb,56b34'
```

invokes the routine HISTORY, passing it the parameter string RB,56B34.

## Using Subcommands in CLISTs

All IPCS subcommands can be entered directly or invoked from CLISTs. Three subcommands are designed especially for CLISTs but can also be entered directly.

The COMPARE subcommand compares two pieces of data, sets a return code, and, optionally, displays a message. For example:

```
compare value(x'c1d2f3') with(address(1a3564.))
```

compares C1D2F3 with three bytes of data at location X'1A3564' and sets the return code to indicate the results. In this example, it does not display a message on your terminal because the LIST keyword is omitted.

The EVALUATE subcommand retrieves data from the dump and puts it in register 15. To process the retrieved data, use the CLIST symbol &LASTCC. The subcommand

```
evaluate a9362. length(2)
```

retrieves two bytes from location X'A9362' and puts them in register 15, right-justified and padded on the left with zeros.

The NOTE subcommand allows you to annotate a listing you are producing on the IPCSPRNT data set, to display messages on your terminal as a CLIST executes, or both. You can use other operands to cause page ejects or to underline messages on the IPCSPRNT data set. For example

```
note 'new dump starts here' space(1) print noterminal
```

inserts a blank line followed by the text NEW DUMP STARTS HERE in the IPCSPRNT data set.

## Analyzing Dumps Containing Bad Data

IPCS dump data examination subcommands require access to key system data. Under normal conditions IPCS generally defines the appropriate symbols on demand and enters them in the symbol table for repetitive reference to the data. Occasionally, however, dumps are generated in which damage to low core or other factors prevent IPCS from successfully locating such data.

When a dump data examination subcommand cannot locate a key block of storage, the subcommand terminates execution and produces a message identifying the block which could not be accessed. In cases where you can identify the storage location of the required block, you may use the EQUATE subcommand to define its location for entry into the IPCS symbol table.

### *Establishing Definitions*

In the following manner, you may define any symbol that is normally defined automatically by IPCS dump data examination.

Assuming the CVT is located in a dump at location X'F3C0' in absolute storage, you may define the CVT by entering:

```
equate cvt f3c0. absolute
```

Similarly, if the master segment table is located at X'5D7C00' in absolute storage, with a length of 4 and a dimension of 256, you may define the master segment by entering:

```
equate sgt001 5d7c00. absolute length(4) dim(256)
```

ASMCHECK, COMCHECK, IOSCHECK, and SUMMARY processing will only accept

| explicit symbol definitions in virtual storage. The remainder of IPCS
| will honor all definitions established using the above process. For
| more information on symbol definitions, see "Naming Conventions" earlier
| in this chapter.

### | *IPCS Data Validation*

| In most cases where IPCS has defined a symbol on demand, validation of
| the block of storage represented by that symbol is automatically
| performed prior to placing the symbol in the symbol table. (For more
| information on validation, see "Validating Structures" earlier in this
| chapter.) You may request validation of the data defined by a symbol by
| the use (or omission) of an explicit designation of data type. In the
| following example validation is performed:

```
| equate CVT f3c0. absolute structure(cvt)
```

| If serious defects are detected during validation, the symbol is not
| defined. In the example above, those IPCS subcommands that require
| access to the CVT would document the need for that symbol and terminate,
| avoiding the use of damaged data.

| You may use the LIST subcommand to examine the storage represented by
| the block you are trying to define in order to determine the extent of
| damage.

```
| list f3c0. absolute length(1260)
```

| If you determine that the damage will not impact the results of the IPCS
| subcommands to be run, you can force the definition into the symbol
| table by omitting the explicit data type:

```
| equate cvt f3c0. absolute structure
```

| IPCS subcommands use the established definition without validation, and
| the results of those subcommands are valid as long as the subcommands
| use fields that are not damaged. If damaged data is used, misleading
| diagnostic messages may be produced.

# Chapter 5. IPCS Commands and Subcommands

This chapter describes the TSO commands related to IPCS, the data description operands for IPCS subcommands and the IPCS subcommands.

The command syntax and notational conventions are the same as for TSO. For a complete description of these syntax and notational conventions, see the OS/VS2 TSO Command Language Reference.

The description of each subcommand includes its return codes, if that subcommand provides them. Return codes may be accompanied with messages. For more information on messages generated by IPCS subcommands, see the OS/VS2 MVS Interactive Problem Control System (IPCS) Messages and Codes.

For examples showing the use of the commands and subcommands, refer to Chapters 2, 3, and 4, and to the Index under "Examples."

## TSO Commands for IPCS

This section contains descriptions of three TSO commands supplied with IPCS. You can execute these commands at any time during a TSO session. With the exception of the IPCS command, you can also execute them during an IPCS session.

Use the IPCS command to start an IPCS session. IPCS is a TSO command processor that initializes the IPCS environment and allows you to issue the IPCS subcommands.

Prior to executing the IPCS command, you must allocate the dump directory. If printed output from the subcommands is desired, you must also allocate the IPCSPRNT data set.

RELATED SUBCOMMANDS:

- END
- SETDEF

---

```
IPCS [ PARM(nn) ]
```

---

PARM(nn) specifies the member of SYS1.PARMLIB that IPCS uses as its initialization parameters for this session. The 1 or 2 digit decimal number, appended to IPCSPR, produces the name of the member of SYS1.PARMLIB to be used. When specifying the number, leading zeros are optional.

If you omit this keyword, the default is IPCSPR00.

Use the IPCSDDIR command to prepare a dump directory data set for dump analysis.

Dump analysis commands use the dump directory in update mode. In this case, the directory must contain at least one record. IPCSDDIR writes two records to the dump directory: one with a key of binary zeros (0), another with a key of binary ones (1).

IPCSDDIR should only be used when you first define a dump directory data set. Once initialized by IPCSDDIR, the dump directory does not require reinitialization.

---

```
IPCSDDIR  dsn
```

---

dsn specifies the name of the dump directory to be initialized.

IPCSDDIR uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the SYSDSCAN command to display the titles of the dumps in the SYS1.DUMPnn data sets. The date and time when each dump was produced is included in the display.

---

```
SYSDSCAN   [ xx [ : yy ] ]
```

---

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.

SYSDSCAN uses the following return codes:

0 - Successful completion.

other - Either the nonzero return code from IKJPARS or the nonzero return code from dynamic allocation.

## Data Description Operands

The data description operands describe the location, length, type, and organization of the data and the way an IPCS subcommand is to process it.

The data description operands are both positional and keyword and, to save space, are described here rather than with each subcommand on which they can be specified. Replace the value "data-descr" where it appears in IPCS subcommands with the data description operands that suit your purpose.

The data description operands and their descriptions are:

---

**Address and length operands**

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

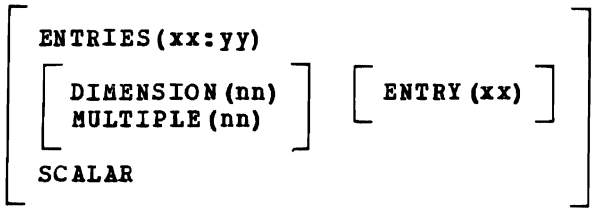
**Address processing keywords**

```
┌                                                    ┐
│ ABSOLUTE                                           │
│                                                    │
│ HEADER                                             │
│                                                    │
│ ┌                ┐                                 │
│ │ ASID(asid)     │  [ CPU(cpu) ]                   │
│ │ REAL           │                                 │
│ │ STATUS         │                                 │
│ └                ┘                                 │
└                                                    ┘
```

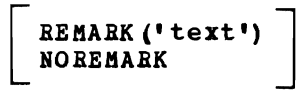
**Attribute keywords**

```
┌                                                    ┐
│ AREA [ (name) ]                                    │
│                                                    │
│ ┌                ┐                                 │
│ │ B              │                                 │
│ │ BIT            │                                 │
│ │ X              │                                 │
│ │ HEXADECIMAL    │                                 │
│ └                ┘                                 │
│                                                    │
│ ┌                ┐                                 │
│ │ C              │                                 │
│ │ CHARACTER      │                                 │
│ └                ┘                                 │
│                                                    │
│ ┌                ┐                                 │
│ │ F              │                                 │
│ │ SIGNED         │                                 │
│ └                ┘                                 │
│                                                    │
│ UNSIGNED                                           │
│                                                    │
│ ┌                ┐                                 │
│ │ PTR            │                                 │
│ │ POINTER        │                                 │
│ └                ┘                                 │
│                                                    │
│ MODULE [ (name) ]                                  │
│                                                    │
│ STRUCTURE [ (name) ]                               │
└                                                    ┘
```

Array keywords




Remark keywords




---

Address and length operands

address [ : address ] specifies any one of the following:

- Symbolic address.
- Relative address.
- Literal address.
- General-purpose register.
- Floating-point register.
- Indirect address.
- Address expression.
- Range of addresses.

A symbolic address is a symbol previously defined in the symbol table for the current dump.

A relative address is a maximum of six 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.

A literal address is a maximum of six hexadecimal digits followed by a period.

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

A floating-point register is designated as a decimal integer followed by a D for double precision. The decimal integer can be 0, 2, 4, or 6.

An indirect address is a symbolic, relative, or literal address, or a general-purpose register followed by a maximum of 255 percent signs (%). You can use general-purpose registers in an address expression but you cannot use floating-point registers in an address expression.

Each percent sign indicates one level of indirect addressing. Indirect addressing means treating the contents of the address preceding the percent sign as a pointer. Indirect addresses are evaluated from the leftmost percent sign to the rightmost percent sign.

An address modifier is a maximum of six decimal digits followed by the letter N, or it is a maximum of six hexadecimal digits (not followed by a period). The N may be in uppercase or lowercase. An address modifier must be preceded by a plus (+) or a minus (-) sign. An address modifier 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.

An address expression has the format:

addr[%...]+expression value[%...][+expression value[%...]...]

where

addr - is a symbolic, relative, literal, or indirect address, or a general-purpose register. You cannot use floating-point registers in an address expression.

expression value - is an address modifier. You cannot use general-purpose or floating-point registers in an expression value.

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.

LENGTH(length) specifies the length of the area beginning at the specified address. The length can be specified in decimal or hexadecimal. Decimal digits are specified as such; hexadecimal digits must be specified as X'xxx...'.

If the address is absolute, virtual, or real, the length can range from 1 through 16,777,216.

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 4104.

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

If you omit this keyword, the subcommand uses the default length.

Address processing keywords

ABSOLUTE specifies that the address or address range is absolute. IPCS is not to simulate dynamic address translation or prefixing on this address or address range.

CPU(cpu) specifies that the address or address range is real or virtual. IPCS is to simulate prefixing for the specified CPU and simulate dynamic address translation for the specified ASID. 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 operand and omit ASID, REAL, and STATUS, the subcommand uses the default ASID.

If you omit ABSOLUTE, HEADER, ASID, REAL, STATUS, and CPU, the subcommand uses the default CPU and ASID.

ASID(asid) specifies that the address or address range is virtual. IPCS is to simulate dynamic address translation for the specified address space and simulate prefixing for the specified or default CPU. The ASID can range from 1 through 999. To process summary dump records, use the special ASID 65,530 (X'FFFA'). You can specify the ASID in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

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

If you omit ABSOLUTE, HEADER, ASID, REAL, STATUS, and CPU, the subcommand uses the default CPU and ASID.

REAL specifies that the address or address range is real. IPCS is to simulate prefixing for the specified or current default CPU. This applies only to stand-alone dumps.

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

If you omit ABSOLUTE, HEADER, ASID, REAL, STATUS, and CPU, the subcommand uses the default CPU and ASID.

STATUS specifies that the address or address range is in one of the CPU status records in the dump. This applies only to stand-alone dumps.

When you use this keyword, the subcommand accesses data in the status records from offset eight. That is, the subcommand processes data in the status record eight bytes beyond the address you specify.

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

If you omit ABSOLUTE, HEADER, ASID, REAL, STATUS, and CPU, the subcommand uses the default CPU and ASID.

HEADER specifies that the address or address range is in the header record for the dump.

When you use this keyword, the subcommand accesses data in the header record from offset zero. That is, the subcommand processes data in the header record at the address you specify.

If you omit ABSOLUTE, HEADER, ASID, REAL, STATUS, and CPU, the subcommand uses the default CPU and ASID.

Attribute keywords

AREA [ (name) ] specifies that the storage at the address or in the address 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 keyword is frequently used when creating a symbol table entry for the storage at the address or in the address range.

If you omit the name, the storage is still given the attribute AREA to distinguish it from MODULE and STRUCTURE. The name can be a maximum of 31 alphameric characters and the first character must be alphabetic.

If you omit all attribute keywords, the default is AREA.

B, BIT, X, or HEXADECIMAL specifies that you interpret the storage at the address or in the address range as hexadecimal data. If you display or print the data, it is shown in hexadecimal format.

C or CHARACTER specifies that you interpret the storage at the address or in the address range as character data. If you display or print the data, it is shown in character format.

F or SIGNED specifies that you interpret the storage at the address or in the address range as 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 keyword, 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.

UNSIGNED specifies that you interpret the storage at the address or in the address range as 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.

PTR or POINTER specifies that you interpret the storage at the address or in the address range as 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 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.

MODULE [ (name) ] specifies that the storage at 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 keyword is frequently used when creating a symbol table entry for the storage at 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 with the attribute MODULE. The name can be a maximum of 31 alphameric characters and the first character must be alphabetic.

STRUCTURE [ (name) ] specifies that the storage at 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 keyword is frequently used when creating a symbol table entry for the storage at the address or in the address range.

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

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

Array keywords

ENTRY(xx) or ENTRIES(xx:yy) specifies that the storage at 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 each be a maximum of 16,777,216 and can be specified in decimal, hexadecimal, or binary. The size of the total array is the length of storage in the specified address range or specified with the LENGTH keyword, multiplied by the number of array elements.

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

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

DIMENSION(nnn) or MULTIPLE(nnn) specifies that the storage at the address or in the address range is an array of dimension nnn. The number nnn can be a maximum of 16,777,216 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 specified with the LENGTH keyword. The total size of the array is the size of an element, multiplied by nnn.

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

SCALAR specifies that the storage at the address or in the address range is a single entity with nonrepeating fields.

If you omit all array and scalar keywords, the default is SCALAR.

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

Remark keywords

REMARK('text') specifies a textual description of the storage at the address or in the address range. This keyword is frequently used when creating a symbol table entry for the storage. The text can be a maximum of 512 characters. The remark is stored in the symbol table.

NOREMARK specifies that the description associated with an existing symbol in the symbol table is not to be copied to the symbol being defined. This keyword is frequently used when equating a new symbol to one previously defined. If you omit NOREMARK, the EQUATE subcommand copies the description of the existing symbol to the new symbol.

## IPCS Subcommands

This section contains descriptions of the IPCS subcommands. The subcommands are in alphabetical order.

## *ADDDSN — 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.

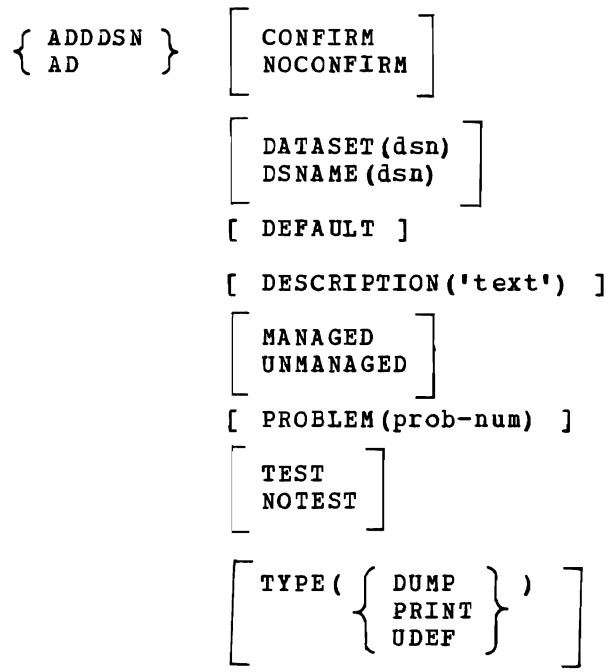
You can specify that the subcommand require your confirmation before proceeding if a data set attribute conflict occurs. These conflicts occur if the data set specified on this subcommand is already associated with a problem and the attributes specified when the data set was associated with the other 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 terminates.

Any user can execute ADDDSN at any time during an IPCS session. This subcommand does not change the current data set unless you specify the DEFAULT keyword.

RELATED SUBCOMMANDS:

- DELDSN
- LISTDSN
- MODDSN




CONFIRM specifies that the subcommand is to request your confirmation before processing a data set attribute conflict.

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 associates the specified data set with the specified problem and leaves the data set attributes as they are.

If you enter N, the subcommand terminates 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 keyword, if specified.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this keyword.

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 keyword.

DATASET(dsn) or DSNAME(dsn) specifies the name of the data set to be associated with the problem. The subcommand ignores a password if you specify one 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 this subcommand is to become the current data set if the subcommand executes successfully.

If you specify a data set name with a password, the data set name and password become the current data set name.

If you omit this keyword, if you enter N to resolve a data set attribute conflict, or if the subcommand fails, the current data set name is not changed.

DESCRIPTION('text') specifies a description of the data set. The description is a maximum of 60 characters of text.

If you omit this keyword, and the data set is not already associated with a problem, there is no description recorded for this data set.

If you omit this keyword and the data set is already associated with a problem, the subcommand uses the description previously provided for this data set.

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. For a complete description of when IPCS scratches a managed data set, see "Attributes of Data Sets" in Chapter 3.

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 keyword, the default is 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 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 keyword, 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 keyword, the subcommand associates the data set with the current problem.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

TYPE(value) specifies the data set type.

If you omit this keyword and the data set is not already associated with a problem, the default is TYPE(DUMP).

If you omit this keyword and the data set is associated with a problem, the subcommand uses the current type attribute for this data set.

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.

ADDDSN uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

## *ADDPROB — 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 issue the subcommand with no operands, it creates a problem and sets default problem attributes as described for each operand.

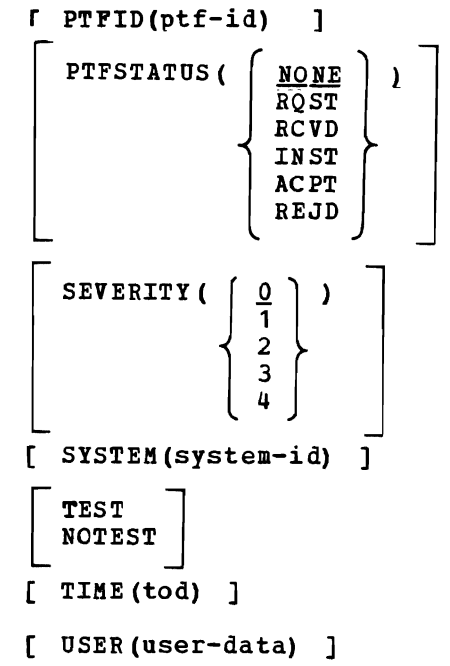
ADDPROB creates a new problem status record and includes this record in the problem directory.

Any user can execute ADDPROB at any time during an IPCS session. This subcommand does not change the default problem unless the DEFAULT keyword is specified.

RELATED SUBCOMMANDS:

- ADDDSN
- DELPROB
- LISTPROB
- MODPROB

```
{ ADDPROB } [ ABSTRACT('text') ]
{ AP      }
          [ 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    }
```




---

ABSTRACT('text') specifies the problem abstract. The abstract is a maximum of 128 characters of text providing a brief description of the problem.

If you omit this keyword, there is no abstract recorded for this problem.

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 keyword, the subcommand sets this field to blanks.

COMPID(component-id) specifies 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 keyword, the subcommand sets this field to blanks.

DATE(date) specifies 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 keyword, the subcommand supplies the current date.

DEFAULT specifies that the problem being added is to become the default problem if the subcommand executes successfully.

If you omit this keyword or if the subcommand fails, the default problem is not changed.

DESCRIPTION('text') specifies the description of the problem. The problem description may contain a maximum of 9,360,000 characters.

If you omit this keyword and if you omit DSDESCRIPTION, there is no description recorded for this problem.

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 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 keyword and if you omit DESCRIPTION, there is no description recorded 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.

If you omit this keyword, the subcommand sets this field to blanks.

FIXSTATUS(value) specifies the status of a local fix, circumvention, or bypass.

If you specify a value for this keyword, the subcommand records the date and time of that specification.

If you omit this keyword, 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 keyword, 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 keyword, the subcommand records the date and time of that specification.

If you omit this keyword, 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 nonacceptable APAR or a duplicate of an APAR which was closed more than ten 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 userid of the person responsible for resolving the problem.

If you omit this keyword, the subcommand sets this field to your TSO userid.

PROBLEM(prob-num) specifies 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 keyword, 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 keyword, 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 keyword, 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 keyword, the subcommand records the date and time of that specification.

If you omit this keyword, 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 whose severity has not been determined. If you specify 0, the subcommand sets this field to blanks.

If you omit this keyword, the default is 0.

The severity codes and their meanings are:

1 - The user is unable to use the program, resulting in a critical impact on his operations.

2 - The user is able to use the program but is severely restricted.

3 - The user is able to use the program with limited functions which are not critical to the overall operations.

4 - The user or the PSR 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 keyword, 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 nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

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 keyword, 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 keyword, the subcommand sets this field to blanks.

ADDPROB uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the ASCBEXIT subcommand to execute a user-supplied routine that is compatible with the AMDPRDMP ASCB exit. The ASCB exit routine must reside in a library allocated to your TSO session.

---

```
{ ASCBEXIT }  pgmname
{ ASCBX    }
              asid
              [ PRINT    ]
              [ NOPRINT  ]
              [ TERMINAL   ]
              [ NOTERMINAL ]
              [ TEST   ]
              [ NOTEST ]
```

---

pgmname specifies the name of the user-supplied exit routine.

asid specifies the address space identifier to be passed to the exit routine. The ASID can range from 1 through 999. To process summary dump records, use the special ASID 65,530 (X'FFFA'). You can specify the ASID in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

PRINT specifies that the subcommand direct the listing to the print data set. However, certain error messages are always directed to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct the listing to the print data set.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct the listing to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct the listing to your terminal. However, certain error messages are always directed to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs. When testing user-written exits, TEST may be used to obtain dumps through normal TSO procedures.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

ASCBEXIT uses the following return codes:

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

If the exit routine puts a return code in register 15, that value is the return code from this subcommand.

Use the ASMCHECK subcommand to analyze control blocks associated with the MVS auxiliary storage manager (ASM) and produce control block information and statistics. When possible, ASMCHECK validates each control block. The subcommand uses the CVT to locate the ASMVT. From there it locates the other paging control blocks, validates them, and makes symbol table and storage map entries for them.

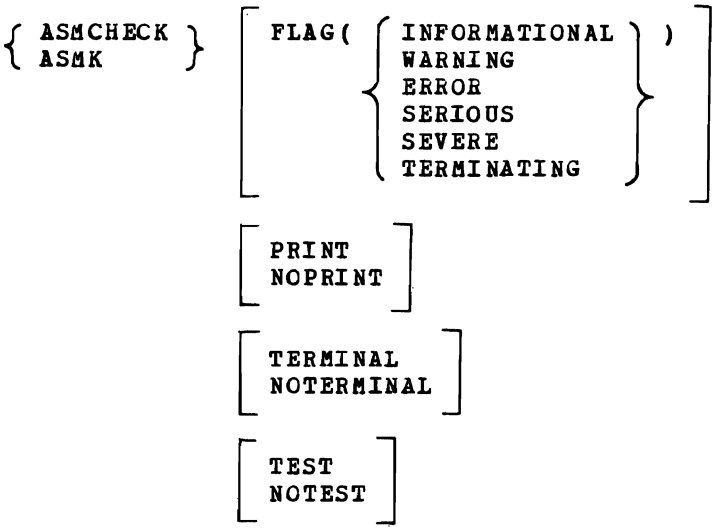
The following control blocks are accessed:

- ASMVT
- IOSBs
- PART
- PARTEs
- IORBs
- UCBs

After analysis, ASMCHECK produces the following statistics:

- Location of the ASMVT.
- Location of the PART.
- Number of I/O requests received by ASM versus the number completed.
- Location of each paging device.

---




---

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

ASMCHECK uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the COMCHECK subcommand to analyze communications task control blocks and produce control block information and statistics. The subcommand uses the CVT to locate the UCM. The UCM points to other control blocks used for queueing messages and managing consoles.

COMCHECK accesses the following control blocks:

- UCM
- WQEs
- OREs
- UCMEs
- UCB
- CQEs

After analysis, COMCHECK produces the following statistics:

- The maximum number of messages that can be queued.
- The number of messages that are queued.
- The number of major WQEs.

If a console UCB has a nonzero status flag, COMCHECK produces the messages

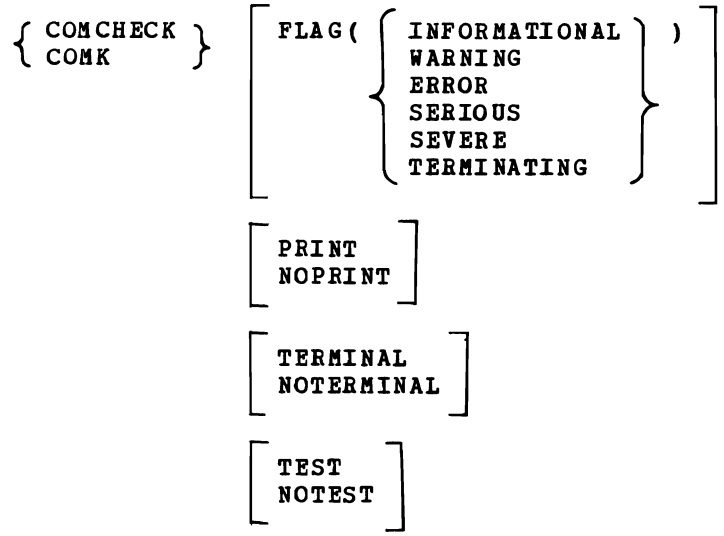
```
BLS18008I  UCMSTS STATUS FLAG IS X'xxx' FOR FOLLOWING CONSOLE

BLS18009I  nnn WQES FOR CONSOLE uuu
```

where nnn is the number of WQEs queued for console uuu.

If there are any outstanding replies, COMCHECK produces the message

```
BLS18010I  OUTSTANDING REPLY nnn
```

where nnn is the reply identifier. This message is followed by the text of the message awaiting the reply.




FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

COMCHECK uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

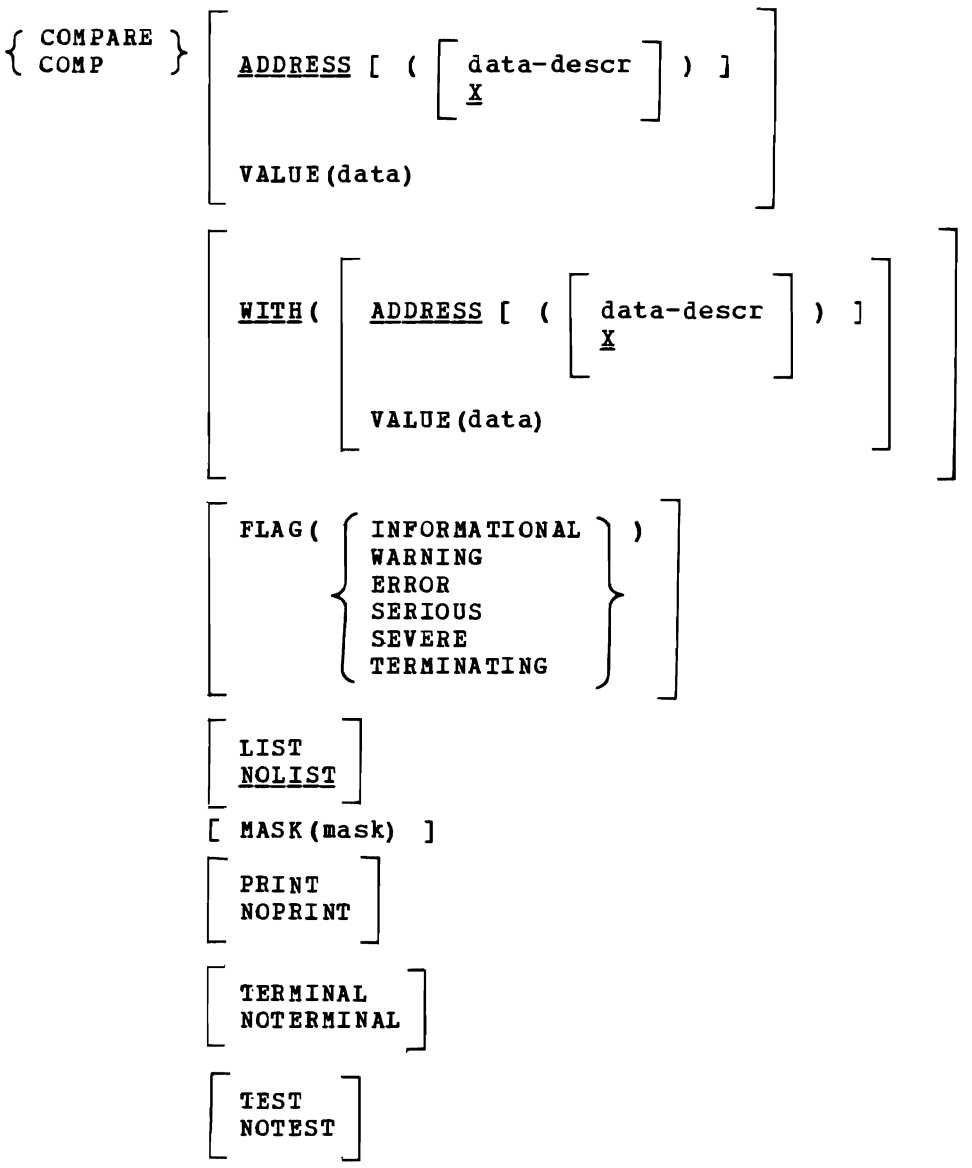
12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the COMPARE subcommand to perform a logical comparison between two data items. COMPARE makes the results of the comparison known to a CLIST 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. If the data items are not of equal length, COMPARE uses the shorter length.

---




---

ADDRESS [ (data-descr) ] specifies the address and attributes of the first operand. The subcommand compares the data described by this keyword to the data specified by the WITH keyword.

The length of the comparison is determined by the length of the data described by this keyword or by the mask, if you specify one. The maximum length is 16,777,216 bytes or, if you use a mask, 8 bytes.

If you omit this keyword, the default is ADDRESS(X).

The address values and their meanings are:

data-descr - any valid combination of data description operands (see "Data Description Operands," above). If you specify a range of address and LENGTH, the length of the range overrides the LENGTH value.

X - the current address.

VALUE(data) specifies the data to be compared. The subcommand compares this data to the data specified by the WITH keyword. The data can be specified in hexadecimal, x'value', or as characters, c'value'.

If you specify this operand as hexadecimal data, the maximum length is 512 hexadecimal digits.

If you specify this operand as character data, the maximum length is 256 bytes.

WITH [ (data) ] specifies the second operand for the comparison.

If you omit this keyword, the default is WITH(ADDRESS(X)).

The WITH values and their meanings are:

ADDRESS(data-descr) - specifies the address and attributes of the second operand. The subcommand compares the first operand to the data described by this operand.

If you specify this operand as an address, the maximum length of the comparison is 16,777,216 bytes.

The ADDRESS values and their meanings are:

data-descr - any valid combination of data description operands (see "Data Description Operands," earlier in this section).

If you specify a range of addresses and LENGTH, the length of the range overrides the LENGTH value.

X specifies the current address.

VALUE(data) - specifies the data to be compared. The subcommand compares the first operand to this data. The data can be specified in hexadecimal, x'value', or as character data, c'value'.

If you specify this operand as hexadecimal data, the maximum length is 512 hexadecimal digits.

If you specify this operand as character data, the maximum length is 256 bytes.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default value for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

LIST specifies that the subcommand display the results of the comparison at your terminal.

NOLIST specifies the subcommand not display the results of the comparison at your terminal.

If you omit this keyword, the default is NOLIST.

MASK(mask) specifies a mask that is logically ANDed with both comparands. COMPARE then compares the results of the AND operations. The mask must be the same size as the two data items being compared. The mask must be specified as hexadecimal digits and can be a maximum of 16 hexadecimal digits. Thus if you use a mask, the data items being compared can be a maximum of 8 bytes.

PRINT specifies that the subcommand direct the message describing the results of the comparison to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct the message describing the results of the comparison to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct the message describing the results of the comparison to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct the message describing the results of the comparison to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

COMPARE uses the following return codes:

0 - The operands are equal.

4 - The first operand is low.

8 - The first operand is high.

12 - The comparison is incomplete.

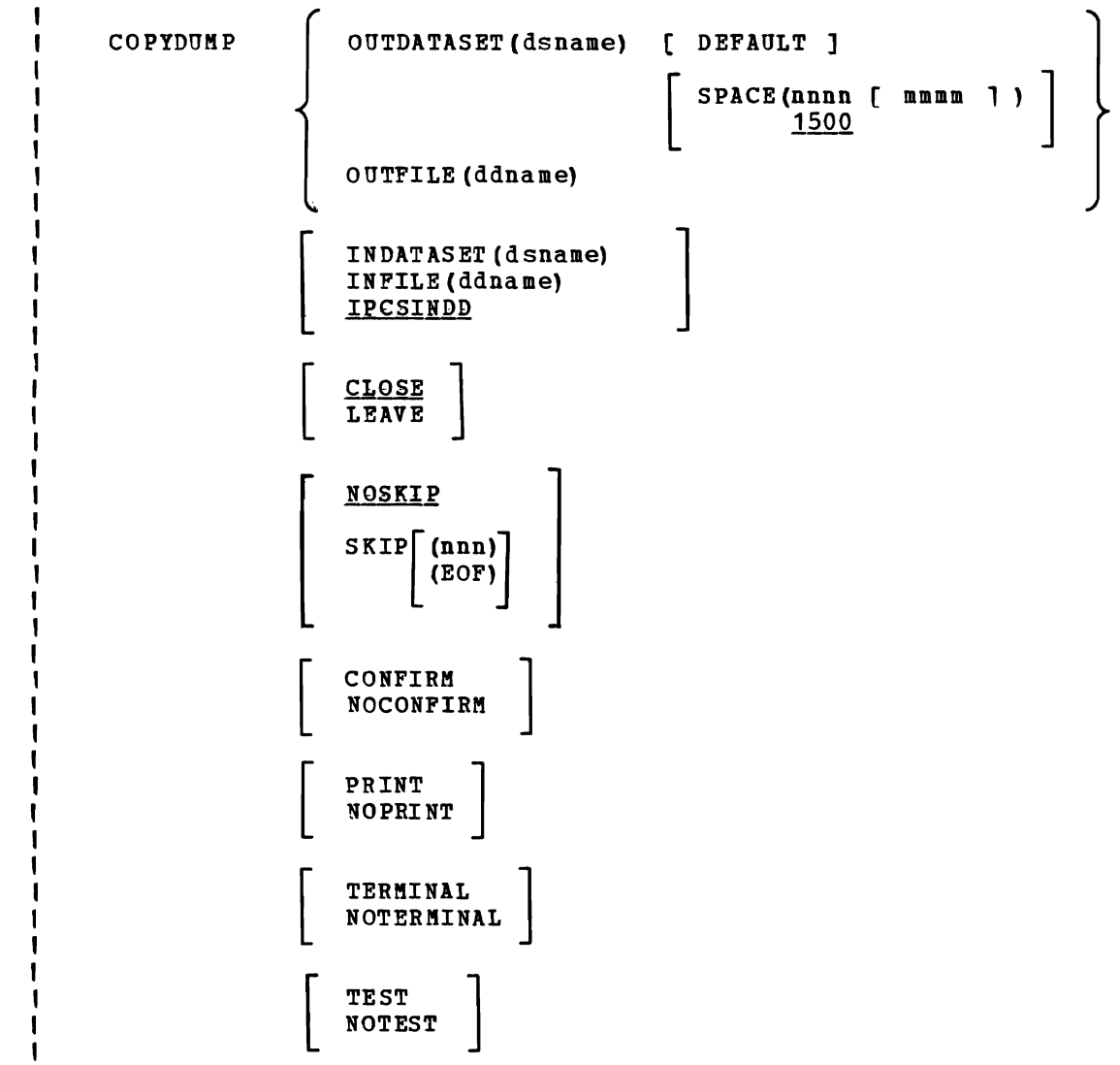
## *COPYDUMP — Copy a Dump*

Use the COPYDUMP subcommand to copy a single unformatted MVS dump from one data set to another. This facility is especially useful for applications (such as IMS) where several SYSMDUMPs may be contiguous in a single data set. Refer to the OS/VS2 MVS JCL manual for information regarding the dispositon of SYSMDUMPs. The COPYDUMP subcommand allows for the extraction of a selected dump or the simple perusal of all the dump titles in the data set.

The input data set may be specified on the command line via ddname or data set name. A default ddname of IPCSINDD is used when no ddname is specified.

The target data set (the data set into which the dump is transcribed) must be specified on the command line via ddname or dsname. If the data set does not exist, COPYDUMP allocates a new data set with the specified name.

The target data set is always closed immediately after the copy function is completed. The input data set is always closed when an end of file is encountered. If COPYDUMP completes without reaching an end of file, a user option determines whether the input data set is closed or remains open. If it remains open, it is positioned for another COPYDUMP subcommand to resume processing of the next dump in the same input data set.




OUTDATASET(dsname) specifies the target data set into which the dump is to be copied. OUTDSNAME and ODS may be used as alternate forms of this keyword. The output, or target, data set is required whenever transcription is performed. However, if SKIP(EOF) is specified, then the output data set is not needed, and if it is specified, it is ignored.

If the designated data set exists, it is dynamically allocated and used by COPYDUMP. It must be cataloged to allow dynamic allocation to locate the data set by name. 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, a new data set with the specified name is allocated. The SPACE keyword may be used to

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

If the target data set is defined using the OUTDATASET keyword, COPYDUMP closes and deallocates it.

DEFAULT specifies that the target data set is to become the current data set if the subcommand executes successfully.

If you specify a data set name with a password, the data set name and password become the current data set name.

If you omit this keyword, or if the subcommand fails, the current data set name is not changed.

NOTE: If the target data set is specified via OUTFILE, then the function of the DEFAULT keyword 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 4104-byte dump records. Excess space is released at the completion of COPYDUMP processing.

If you omit this keyword, both the primary allocation and the secondary allocation are for 1500 records. If only the primary allocation is specified, the secondary allocation defaults to the primary allocation.

OUTFILE(ddname) specifies the target file into which the dump is to be copied. OFILE may be used as an alternate form of this keyword. This file must be allocated by JCL or the TSO ALLOCATE command prior to the use of COPYDUMP. The output, or target, data set is required whenever transcription is performed.

If the target data set is defined using the OUTFILE option, COPYDUMP closes it but does not directly deallocate the OUTFILE. The JCL option FREE=CLOSE may be used to release resources associated with the OUTFILE at the earliest possible moment.

INDATASET(dsname) specifies the input data set from which the dump is copied. INDSNAME and IDS may be used as alternate forms of this keyword.

If a prior COPYDUMP subcommand left this data set open, processing of the data set is resumed where it left off.

If the data set is not open, it is dynamically allocated, opened, and processed beginning with the first record in the data set. The designated data set must exist. It is dynamically allocated and used by COPYDUMP. It must be cataloged to allow dynamic allocation to locate the data set by name. If the data set resides on a volume that is not mounted as RESIDENT or RESERVED, MVS MOUNT authorization is required.

If the input data set is defined using the INDATASET keyword, COPYDUMP closes and deallocates it. Note: INDATASET may not be used to resume processing of a data set initially opened using the INFILE keyword.

INFILE(ddname) specifies the input file from which the dump is copied. IFILE may be used as an alternate form of this keyword. This file must be allocated by JCL or the TSO ALLOCATE command prior to the use of COPYDUMP.

If a prior COPYDUMP subcommand left this file open, processing of the file resumes. If the file is not open, it is allocated and opened and

processing begins with the first record in it.

If the target data set is defined using the INFILE option, COPYDUMP closes it but does not directly deallocate the INFILE. The JCL option FREE=CLOSE may be used to release resources associated with the INFILE at the earliest possible moment.

Note: INFILE may not be used to resume processing of a data set initially opened using the INDATASET keyword.

If you omit both INDATASET and INFILE, and:
- an open data set or file is available, COPYDUMP resumes processing the open data set or file.
- no open data set or file is available, COPYDUMP opens the default input file, IPCSINDD, and processing begins with the first record in it.

  If you omit both INDATASET and INFILE, the default input file is IPCSINDD.

CLOSE causes the input data set to be closed immediately after the dump has been copied.

LEAVE causes 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.

If you omit both LEAVE and CLOSE, the default is CLOSE.

NOSKIP specifies that no dumps are to be skipped before copying begins.

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 omit both SKIP and NOSKIP, the default is NOSKIP.

If you enter SKIP but no number, nnn, is specified, one dump is skipped.

If you specify 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.

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 target data set and drops any existing records in the dump directory associated with the target data set.

If you enter N, the subcommand terminates without copying the dump into the target data set, and ignores the DEFAULT keyword, if specified. The LEAVE/CLOSE keyword determines if the input data set is left open.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default (established via SETDEF) for this keyword.

NOCONFIRM specifies that the subcommand is not to request your confirmation before copying the dump into the target 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 keyword.

PRINT specifies that COPYDUMP direct the message describing the results of the subcommand to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct the message describing the results to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

COPYDUMP uses the following return codes:

0 - End of file reached. The input data set has been closed and a dump has been copied to the output data set.

4 - End of dump reached. The input data set has been left open, positioned immediately after the dump copied by this subcommand.

8 - 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 was terminated 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 execution of COPYDUMP.

- An attention interruption was generated by you prior to any COPYDUMP processing. No input data set was left open by a prior execution of COPYDUMP.

- COPYDUMP read an invalid 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 target data set (if any) has not been altered.

16 - Subcommand processing was terminated after detection of a problem in the IPCS execution environment.

20 - Subcommand processing was terminated as a result of an attention interruption generated by you. The input data set has been closed. The target data set has been loaded with part of a dump.

24 - Subcommand processing was terminated 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 execution of COPYDUMP.

- An attention interruption was generated by you during COPYDUMP skip processing. The LEAVE option was in effect.

- COPYDUMP read an invalid 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 target data set (if any) has not been altered.

Use the DELDSN subcommand to dissociate a data set or a partitioned data set member from a problem.

You can request the subcommand to display the names and attributes of the data sets to be dissociated and request confirmation before continuing with the DELDSN function.

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.

If the data set's name is SYS1.DUMPnn, where nn is a decimal number from 00 through 99, IPCS initializes the data set for reuse rather than scratches it. In order for IPCS to initialize a SYS1.DUMPnn data set, it must be cataloged and be on direct access storage.

DELDSN dissociates a data set from a problem even if it can not 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.

RELATED SUBCOMMANDS:

- ADDDSN
- LISTDSN
- MODDSN

```
{ DELDSN }  [ CONFIRM   ]
{ DD     }  [ NOCONFIRM ]

           [ DATASET(dsn) ]
           [ DSNAME(dsn)  ]
           [ ALL          ]

          [ PROBLEM(prob-num) ]

           [ TEST   ]
           [ NOTEST ]
```

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 terminates without dissociating the data sets from the specified problem.

If you omit both CONFIRM and NOCONFIRM, the subccmmand uses the default for this keyword.

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 keyword.

DATASET(dsn) 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 subçommand dissociates the current data set from the specified

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 keyword, the subcommand dissociates the specified data set or data sets from the current problem.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

DELDSN uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

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.

If the data set's name is SYS1.DUMPnn, where nn is a decimal number from 00 through 99, IPCS initializes the data set for reuse rather than scratches it. In order for IPCS to initialize a SYS1.DUMPnn data set, it must be cataloged and be on direct access storage.

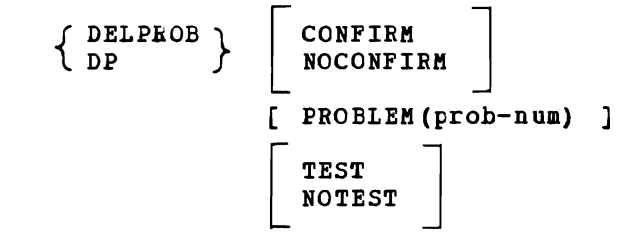
DELPROB dissociates a data set from a problem even if it cannot scratch the data set (for example, if the data set doesn't exist or you can't supply the correct password).

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 execute 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.

RELATED SUBCOMMANDS:

- ADDPROB
- LISTPROB
- MODPROB




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 terminates 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 keyword.

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 keyword.

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 keyword, the subcommand deletes the default problem.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

DELPROB uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the DROPDUMP subcommand to delete all records from your dump directory data set describing a particular dump in order to free space in your dump directory.

RELATED SUBCOMMANDS:

- LISTDUMP

---

```
{ DROPDUMP } [ dsn ]
{ DROPD    }
             [ TEST   ]
             [ NOTEST ]
```

---

dsn specifies the name of the dump data set whose records are to be deleted from your dump directory.

If you omit this operand, the subcommand deletes any records describing the current data set.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

DROPDUMP uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the DROPMAP subcommand to delete records from the storage map describing the current dump.

Delete records from the storage map when you want to free space in the dump directory or when you no longer need a portion of the map.

RELATED SUBCOMMANDS:

- LISTMAP
- SCAN

---

```
{ DROPMAP }  [ ASID(asid) ]
{ DROPM   }
             [ RANGE(addr) ]
             [ TEST   ]
             [ NOTEST ]
```

---

ASID(asid) specifies the address space identifier of map records to be deleted. The ASID can range from 1 through 999. To process summary dump records, use the special ASID 65,530 (X'FFFA'). You can specify the ASID in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

If you specify an address space identifier but do not specify a range, the subcommand deletes all map records for the specified address space.

If you omit both an address space identifier and a range, the subcommand deletes all map records for the current data set.

RANGE(addr) specifies the range of addresses in the dump for which map records are to be deleted. The range can be specified as an address and a length or as a range of addresses (see "Data Description Operands," earlier in this section). If a map record describes an address within the range, the subcommand deletes the map record.

If you specify a range but do not specify an address space identifier, the subcommand deletes all map records, for all ASIDs, in the specified address range.

If you specify the STRUCTURE keyword with a data type, it causes the subcommand to create a map record. This record may be deleted yet it is counted in the resulting statistics.

If you omit both the range and an address space identifier, the subcommand deletes all map records for the current data set.

TEST  nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST  activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

DROPMAP uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the DROPSYM subcommand to delete symbols from the symbol table.

Delete symbols when you want to free space in the dump directory.

RELATED SUBCOMMANDS:

- EQUATE
- LISTSYM

---

```
{ DROPSYM }  [ (symbol list) ]
{ DROPS   }  [ *             ]

             [ PURGE   ]
             [ NOPURGE ]

             [ TEST    ]
             [ NOTEST  ]
```

---

(symbol list) specifies the symbol or symbols to be deleted. You can specify a single symbol, a range of symbols, a list of symbols, or any combination of these. If you specify more than one symbol or range of symbols, enclose them in parentheses. The list can contain a maximum of 31 symbols, ranges of symbols, or both.

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 subcommand 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.

* specifies that the subcommand delete all symbols in the symbol table.

If you omit this operand, the default is *.

PURGE specifies that the NODROP attribute is ignored and all specified symbols are deleted.

NOPURGE specifies that symbols with the NODROP attribute are not to be deleted.

If you omit this keyword, the default is NOPURGE.

TEST  nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST  activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

DROPSYM uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the DSPL3270 subcommand to provide full-screen support for dump viewing and debugging. Other IPCS subcommands display their output on the next available line on the terminal. They will not clear the screen nor format their displays to make full use of the screen.

DSPL3270 makes use of the full 3270 screen and keyboard to display virtual data from the dump and to allow you to easily examine the data. Like all IPCS dump analysis subcommands, DSPL3270 displays data from the current dump data set.

You can manipulate the displayed dump data with several functions provided by DSPL3270 and with the cursor and ENTER key. The standard functions are invoked either by selecting them or pressing a PF key. Additionally you can do implicit scrolling and indirect addressing.

Implicit scrolling consists of selecting the address of a line of data in the data display area. The selected address is made the current address and the data at that address is displayed in the current data display area.

Indirect addressing consists of selecting a word of dump data in the data display area. The selected word is treated as a pointer. The value in the selected word is made the current address and the data at that address is displayed in the current data display area.

The DSPL3270 subcommand:

- Formats the 3270 screen using the default format or the format you defined the last time you used DSPL3270 to process the current dump.
- Defines certain program function (PF) keys.
- Allows you to specify the data to be displayed from the dump.
- Allows you to specify the format of the data (character or hexadecimal).
- Allows you to maintain a stack of addresses from the dump.
- Uses the audible alarm (if available) and messages to notify you of unusual or error conditions.
- Allows you to use both the cursor and ENTER key to select fields and to indicate the next operation to be performed.
- Allows you to execute other IPCS subcommands.

The DSPL3270 subcommand supports the IBM 3275 Model 2 Display Station, the IBM 3277 Model 2 Display Station, the IBM 3276 Control Unit Display Station Models 2 and 12, and the IBM 3278 Display Station Model 2 (attached via the 3276 or 3274). The terminals may be local or remote. They must also be accessible by either TSO/VTAM or TSO/TCAM level 9 or higher.

DSPL3270 initially displays a default screen format if you:

- Specify the RESET keyword.
- Have never processed the current dump with DSPL3270.
- Have processed the current dump with DSPL3270 but have dropped the dump (see the DROPDUMP subcommand) since the last such processing.

The default screen format is defined by the following values:

- ADDR - The address currently associated with the symbol X, if it exists. Otherwise the subcommand uses the address of the CVT.
- ASID - The ASID associated with the symbol X, the current address.
- FMT - X, hexadecimal format.
- AREA - A, the current data display area.
- LINES/AREA - A 19, the current data display area occupies all 19 lines.
- SKIP - 200, X'200' bytes.

NOTE: This subcommand may modify the symbol X, the current address.

---

```
{ DSPL3270 } [ PAUSE   ]
{ DS       } [ NOPAUSE ]

             [ RESET   ]
             [ NORESET ]

             [ TEST    ]
             [ NOTEST  ]
```

---

PAUSE specifies that DSPL3270 inform you before it formats the first screen. The subcommand displays a message, unlocks the keyboard, and waits for the ENTER key to be pressed before clearing the present screen and formatting the DSPL3270 screen.

NOPAUSE specifies that DSPL3270 format the screen without waiting for an attention interrupt.

If you omit this keyword, the default is NOPAUSE.

RESET requests that the default format be used for the initial display.

NORESET requests the initial display use the format defined by you the last time you processed the current dump with DSPL3270 using the current dump directory. The format could have been defined during the current IPCS session or during a previous IPCS session.

If you omit this keyword, the default is NORESET.

| TEST  nullifies IPCS error recovery for ABENDs.

| If you omit both TEST and NOTEST, the subcommand uses the default for
| this keyword.

| NOTEST  activates IPCS error recovery for ABENDs.

| If you omit both TEST and NOTEST, the subcommand uses the default for
| this keyword.

The DSPL3270 default screen format is shown in Figure 5-1.

```
3END  7<-SCROLL+>8  9STACK  10<-SKIP+>11
SK XXXXXX
RF CVT
ADDR XXXXXX ASID NNNN FMT X AREA A LINES/AREA: A 19 B    C    D    SKIP 000200
SUBCMND/CLIST:
```

Figure 5-1. DSPL3270 Default Screen Format

The top line provides the following functions:

- 3END - Terminates the DSPL3270 subcommand and returns to IPCS. This destroys the DSPL3270 screen format and nullifies the program key definitions.

  To terminate DSPL3270, either press PF3 or position the cursor anywhere under END and press ENTER.

- 7<-SCROLL+>8 - Scrolls the data in the current data display area by the number of lines in that display area.

  To scroll backward (toward the beginning of the dump) either press PF7 or position the cursor under the left half of SCROLL and press ENTER.

  To scroll forward (toward the end of the dump), either press PF8 or position the cursor under the right half of SCROLL and press ENTER.

- 9STACK - Saves the current address and its ASID in the address stack (SK). The subcommand displays the address but not its ASID.

  To put the current address in the next element of the stack, either press PF9 or position the cursor under STACK and press ENTER.

- 10<-SKIP+>11 - Subtracts or adds the hexadecimal value after SKIP (on the fourth line) from the current address and displays the data at the new address in the current data display area and makes the new address the current address.

  To skip backward (toward the beginning of the dump), either press PF10 or position the cursor under the left half of SKIP and press ENTER.

  To skip forward (toward the end of the dump), either press PF11 or position the cursor under the right half of SKIP and press ENTER.

The balance of the top line is a message area for DSPL3270.

The second line is an 11-element stack (SK) of addresses from the current dump. The address added to the stack is always the current address (the address after ADDR in line four). This address is explicitly added to the stack if you use the STACK function; it is added to the stack even if it duplicates an address already in the stack. Under other circumstances, the current address is automatically added to the stack when it (the current address) is changed. Automatic stacking depends on how the current address was obtained, not how it is changed.

If the current address was obtained by one of the following means, it is not added to the stack:

- Skipping (SKIP function).
- Explicit scrolling (SCROLL function).
- Implicit scrolling (selecting the address of a line of data in the data display area).
- Selecting an address already in the stack.

If the current address was obtained by any other means (such as typing a new address over it, setting X with another IPCS subcommand or CLIST, using indirect addressing, etc.), it is automatically added to the stack.

On entry to DSPL3270 using the default format the stack is primed with the address of the CVT. Addresses are added to the stack at the right. If the stack is full, DSPL3270 deletes the oldest (leftmost) element, shifts the remaining elements to the left, and adds the new element at the right.

The third line is a note area for the stack elements. You can type a maximum of six characters under each stack element. To delete a stack element, type a slash (/) in the first character of its note and press ENTER. The default format displays "CVT" under the address of the CVT.

The fourth line defines the data being displayed:

- ADDR - This is the current address, the address of the data displayed in the current data display area.

- ASID - This is the address space identifier of the data displayed in the current data display area.

- FMT - This defines the format of the data displayed in the current data display area. C indicates EBCDIC; X indicates hexadecimal.

- AREA - This defines the current data display area. Specify A, B, C, or D to designate the corresponding area. The default format specifies A.

- LINES/AREA: A B C D - This defines the number of lines in each area. Specify the number of lines in each area after the corresponding letter. Each number can be a maximum of two decimal digits and can range from zero through 19.

- SKIP - This specifies the number of bytes to be subtracted from or added to the current address by the SKIP operation. This number can be specified as one to six hexadecimal digits.

DSPL3270 performs validity checks on input on line 4. If it detects errors, the subcommand stops processing the input, intensifies the erroneous field or fields, and displays an error message on the top line of the screen.

The fifth line is an IPCS subcommand, TSO command, or CLIST input area. Whatever you type on this line is executed as an IPCS subcommand, TSO command, or CLIST.

Lines 6 through 24 are the data display area. Each line consists of a six digit hexadecimal address and either eight words in hexadecimal format or 16 words in character format.

DSPL3270 also defines certain keys on the 3270 keyboard, and associates a function for each key or a combination of cursor and a key. You can not define any keys that DSPL3270 does not define.

The keys and their functions are:

- ENTER - Pressing the ENTER key causes DSPL3270 to update the current display. The subcommand processes all fields that have been modified. Their content and the cursor position determines the operations to be performed.

  If you position the cursor under an address at the left margin of the data display area and press ENTER, the selected address becomes the current address and the data at that address is displayed in the current data display area. This is done even if the selected address is not in the current data display area.

  If you position the cursor under a word of dump data displayed in the display area and press ENTER, the selected word is treated as a pointer and the address in that word becomes the current address and the data it points to is displayed in the current data display area. This is done even if the selected word is not in the current data display area.

- CLEAR - Pressing this key causes the display screen to be cleared and the cursor to be positioned at the upper left corner of the screen. The keyboard is then unlocked, allowing you to press the PA2 key, causing the subcommand to restore the screen.

- PA1 - Pressing this key generates an attention interrupt and gives control to IPCS's attention exit.
- PA2 - Pressing this key causes DSPL3270 to rewrite the last valid screen image. Any data entered immediately before you press PA2 is lost.

  The PA2 is also used to refresh the screen after pressing CLEAR or ERASE INPUT.
- PF1 and PF2 - Same as ENTER.
- PF3 - Pressing this key causes DSPL3270 to terminate processing and return to IPCS.
- PF4 - PF6 - Same as ENTER.
- PF7 - Pressing this key scrolls the data in the current data display area backward (toward the beginning of the dump) by the number of lines in that data area.
- PF8 - Pressing this key scrolls the data in the current data display area forward (toward the end of the dump) by the number of lines in that data area.
- PF9 - Pressing this key saves the current address (and its ASID) in the stack.
- PF10 - Pressing this key subtracts the value in the SKIP field (on line 4) from the beginning address in the current data area. The subcommand then displays the data at the new address.
- PF11 - Pressing this key adds the value in the SKIP field (on line 4) to the beginning address in the current data area. The subcommand then displays the data at the new address.
- PF12 - Same as ENTER.

DSPL3270 uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the END subcommand to terminate an IPCS session. Any default values specified with the SETDEF subcommand are lost.

The subcommand closes and deallocates:

- The data set directory.
- The problem directory.
- Any dumps allocated to the user.

The subcommand closes but does not deallocate the:

- Dump directory.
- Print output data set.

RELATED SUBCOMMANDS:

- IPCS
- SETDEF

---

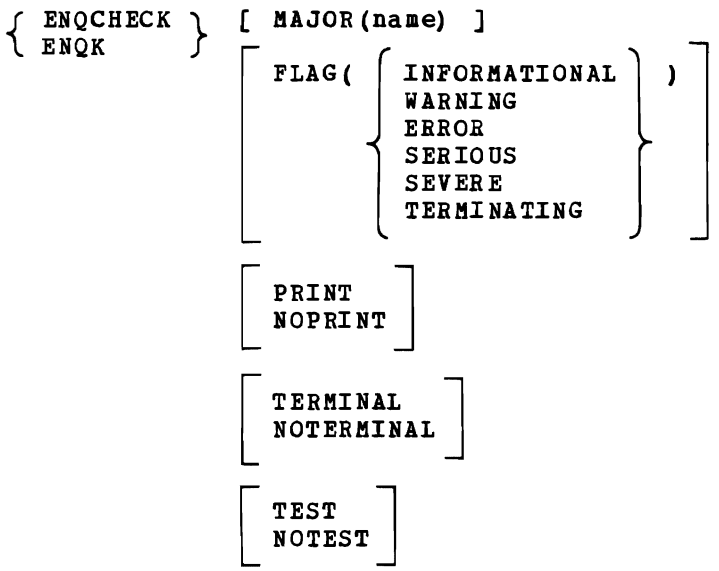
END

---

Use the ENQCHECK subcommand to display the enqueue/dequeue (ENQ/DEQ) facility control blocks. The subcommand uses the CVT to locate the queue of major and minor QCBs and their QELs. The subcommand validates the control blocks and makes storage map entries for them.

The subcommand displays the queue control blocks (QCBs), queue elements (QELs), and the major and minor names associated with the resources represented by those control blocks.

---




---

MAJOR(name) specifies a major queue name whose control blocks are to be displayed. ENQCHECK displays the control blocks for a major queue name that begins with the specified character string.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

ENQCHECK uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the EQUATE subcommand to create a symbol in the symbol table, and associate an address and storage attributes with the symbol.

You specify storage attributes, if any, with the appropriate data description operands.

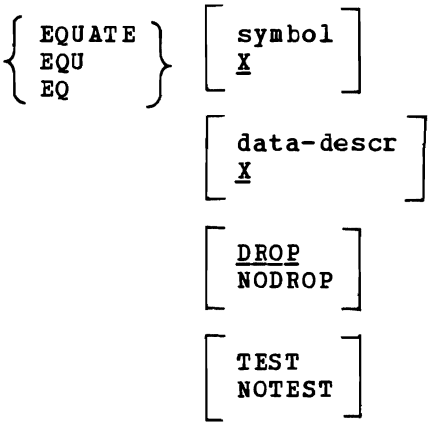
If the specified symbol already exists in the symbol table, the new address and storage attributes overlay the previous address and storage attributes.

NOTE: This subcommand may modify X, the current address.

RELATED SUBCOMMANDS:

- DROPSYM
- LISTSYM

---




---

symbol specifies the symbol being defined. The symbol name can be a maximum of 31 alphameric characters, the first character must be alphabetic.

If you omit this operand, the default is the symbol X.

data-descr specifies the address and attributes to be associated with the symbol being defined.

If you omit this operand, the default is the current address, X.

DROP specifies that the symbol can be deleted from the symbol table by the DROPSYM subcommand without using the PURGE keyword.

If you omit this keyword, the default is DROP.

NODROP specifies that the symbol not be deleted from the symbol table by the DROPSYM subcommand. This can be overridden by the PURGE keyword on the DROPSYM subcommand.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

EQUATE uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the EVALUATE subcommand to retrieve data from a dump and place it in register 15. The subcommand retrieves a maximum of 4 bytes of data, right justifies it in register 15, and zeros the high-order byte of the register. If EVALUATE is used in a CLIST, the next subcommand in the CLIST can obtain the contents of register 15 by using the CLIST variable &LASTCC. Because of this, EVALUATE has little use other than in CLISTs since the retrieved data in &LASTCC is available only to subcommands in the CLIST. The contents of register 15 cannot be obtained directly by any IPCS subcommand not in a CLIST.

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 puts x'0000000C' in &LASTCC. This is, in fact, a return code indicating the failure to retrieve the data, but it could be interpreted as data.

NOTE: This subcommand may modify X, the current address.

---

```
{ EVALUATE } data-descr
{ EVAL     }
           [ TEST   ]
           [ NOTEST ]
```

---

data-descr specifies the address and attributes of the data to be retrieved. If the expressed or implied length of the data is greater than 4, EVALUATE retrieves only the first 4 bytes.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

EVALUATE uses the following return codes:

12 - Severe, requested storage can not 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.

Use the FIND subcommand to locate data in a dump. You can limit the search by specifying the range of addresses to be searched or by specifying a boundary, and a location relative to that boundary.

FIND uses the symbol FINDAREA (recorded in the symbol table) to describe the beginning address and the length of the area being searched. This area is called the search range. If you specify a beginning address for the search but do not specify the end point, FIND sets the length of the search range so the search ends at X'FFFFFF'.

Before the search begins, FIND sets X to the beginning address of the search range. If it finds the search argument, FIND sets X to the address of the search argument. If it does not find the search argument, FIND leaves X set to the beginning of the search range.

Once the subcommand sets the search range (FINDAREA and its length), if you enter another FIND without specifying a new range and if X is outside the current search range, FIND terminates immediately, without modifying X. (X could be outside the current search range only if you modified FINDAREA, X, or both between the two FINDs.)

If you specify no beginning address for the search range but specify a search argument, FIND begins the search at X. If you specify neither a beginning address for the search range nor a search argument, FIND begins the search at X + 1. In either case, the end point of the search range remains the same.

FIND without a search argument attempts to locate the last-entered search argument.

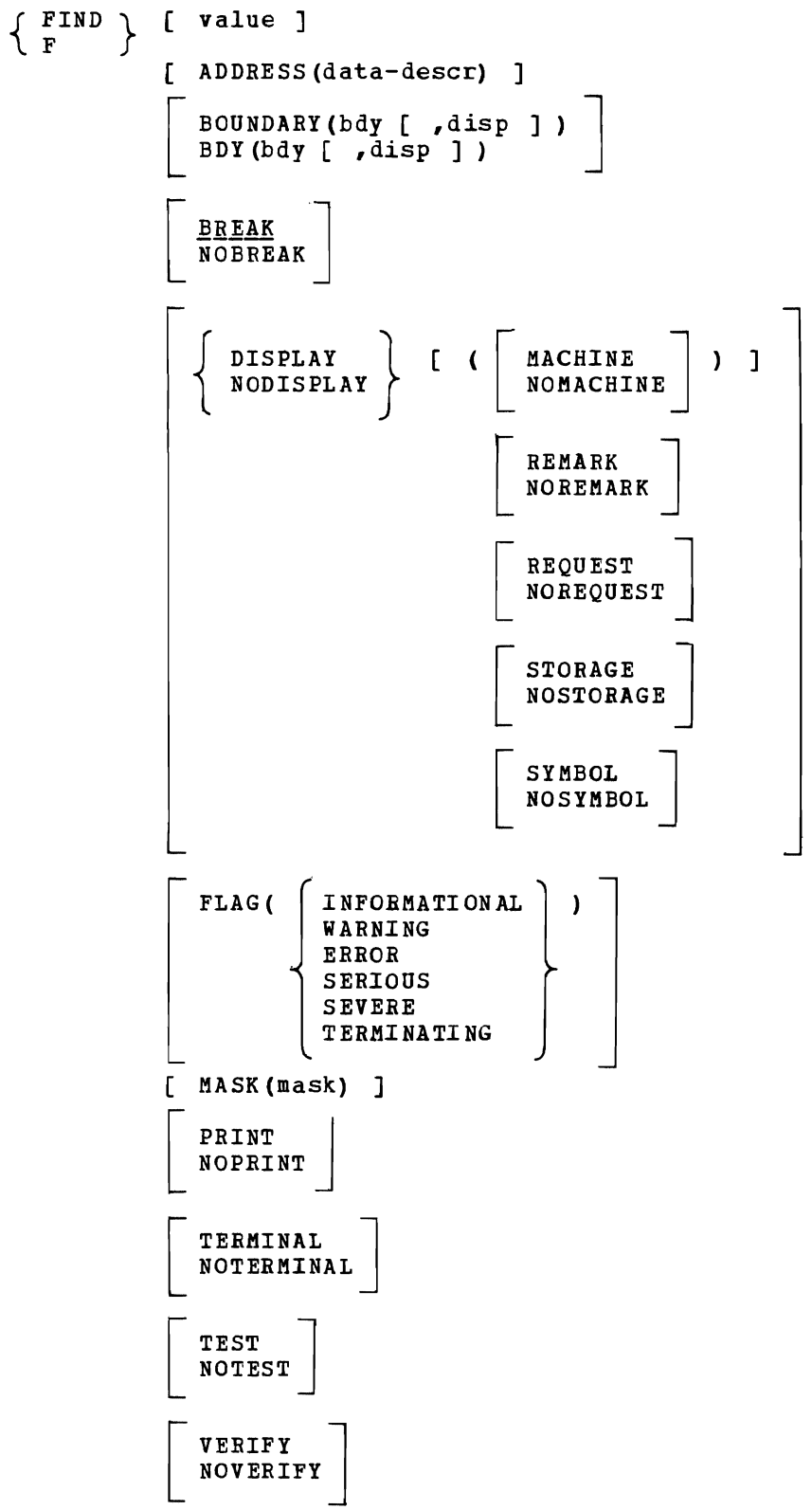
FIND saves FINDAREA and the search argument, boundary, offset, and ASID between uses of the subcommand. It does not save the mask. If you omit a search argument on the next FIND, the subcommand uses the saved values. If you specify a search argument, the subcommand discards the saved values, except FINDAREA, and uses the values you specify in the current search range.

In order to initialize the search argument, you must specify a search argument the first time you use FIND in an IPCS session.

NOTE: This subcommand may modify X, the current address.

RELATED SUBCOMMANDS:

- FINDMOD
- FINDUCB


{ FIND } [ value ]
{ F }
[ ADDRESS(data-descr) ]
[ BOUNDARY(bdy [ ,disp ] ) ]
[ BDY(bdy [ ,disp ] ) ]
[ BREAK ]
[ NOBREAK ]
{ DISPLAY } [ ( [ MACHINE ] ) ]
{ NODISPLAY } [ NOMACHINE ]
[ REMARK ]
[ NOREMARK ]
[ REQUEST ]
[ NOREQUEST ]
[ STORAGE ]
[ NOSTORAGE ]
[ SYMBOL ]
[ NOSYMBOL ]
[ FLAG( { INFORMATIONAL } ) ]
{ WARNING }
{ ERROR }
{ SERIOUS }
{ SEVERE }
{ TERMINATING }
[ MASK(mask) ]
[ PRINT ]
[ NOPRINT ]
[ TERMINAL ]
[ NOTERMINAL ]
[ TEST ]
[ NOTEST ]
[ VERIFY ]
[ NOVERIFY ]

value specifies the search argument. FIND interprets the search argument as literal data. You can specify it in two ways:

Hexadecimal - maximum of 512 hexadecimal digits, specified X'ddd...'.

Character - maximum of 256 bytes, specified, C'ddd...'.

ADDRESS(data-descr) specifies the address and attributes of the search range to be used in locating the search argument. If you specify only one address and do not specify LENGTH, FIND sets the length of the search range so the search ends at X'FFFFFF'.

BOUNDARY(bdy [ ,disp ] ) or BDY(bdy [ ,disp ] ) limits the search to addresses that fall on the specified boundary or to addresses that are at the specified displacement from the specified boundary. Both the boundary and displacement can range from one through 16,777,216 and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

If you omit this keyword, the default for both boundary and displacement is 1.

The boundary and displacement values are:

bdy - an integer specifying the boundary, where:

1 designates a byte boundary.

2 designates a halfword boundary.

3 designates a three-byte boundary.

4 designates a word boundary.

8 designates a doubleword boundary.

10 designates a ten-byte boundary.

etc.

disp - an integer specifying the displacement from the boundary. The first byte beginning at the boundary has a displacement of 1.

BREAK specifies that FIND stop processing if it cannot retrieve storage from the dump to continue the search. This happens if the required storage was not GETMAINed or is not contained in the dump.

If you omit this keyword, the default is BREAK.

NOBREAK specifies that FIND not stop processing if it cannot retrieve storage from the dump. Instead, FIND continues the search with the next available address in the dump.

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by this subcommand is a 3-line header, followed by the data.

To produce this display, specify the VERIFY keyword. DISPLAY is ignored if you specify NOVERIFY.

If VERIFY is specified or defaulted, and the NODISPLAY keyword is also present, a conflict exists. In this case, IPCS generates a minimal display which is equivalent to the DISPLAY(REQUEST) keyword.

The DISPLAY keyword without values is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Since there is no remark associated with the found data, this value is ignored by this subcommand.

NOREMARK - Since there is no remark associated with the found data, this value is ignored.

REQUEST - Displays a model LIST subcommand that could be used to display the information you requested. The LIST subcommand keywords include the data description keywords you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant). If you want to modify the attributes of the displayed data, modify the keywords on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

STORAGE - Displays the storage at the specified or default address, for the specified or default length. The subcommand also displays the storage as in a printed dump: eight bytes in hexadecimal followed by the EBCDIC equivalent.

NOSTORAGE - Suppresses the storage display.

SYMBOL - Since the found data is not a symbol, this value is ignored.

NOSYMBOL - Since the found data is not a symbol, this value is ignored.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

MASK(mask) specifies a mask which the subcommand ANDs with the search argument and with storage before comparing the two. The mask must be specified in hexadecimal digits. The mask must be the same length as the search argument and can be a maximum of 16 hexadecimal digits.

PRINT specifies that FIND direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that FIND not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that FIND direct its output to the user's terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the command uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

VERIFY specifies that the subcommand is to produce output and direct it to the destination or destinations specified by the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

NOVERIFY specifies that the subcommand is not to produce output regardless of the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

FIND uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the FINDMOD subcommand to locate a module in the dump. FINDMOD searches the following areas, in order:

1. Symbol table (searches for specified symbol name with the attribute MODULE).
2. Master CDE chain (modules in the MLPA).
3. Link pack area directory (modules in the link pack area).

If FINDMOD finds the requested module in the symbol table, it creates no new symbols. If it finds the requested module on the CDE chain, it creates the symbols:

- CDEmodulename
- XLmodulename
- modulename

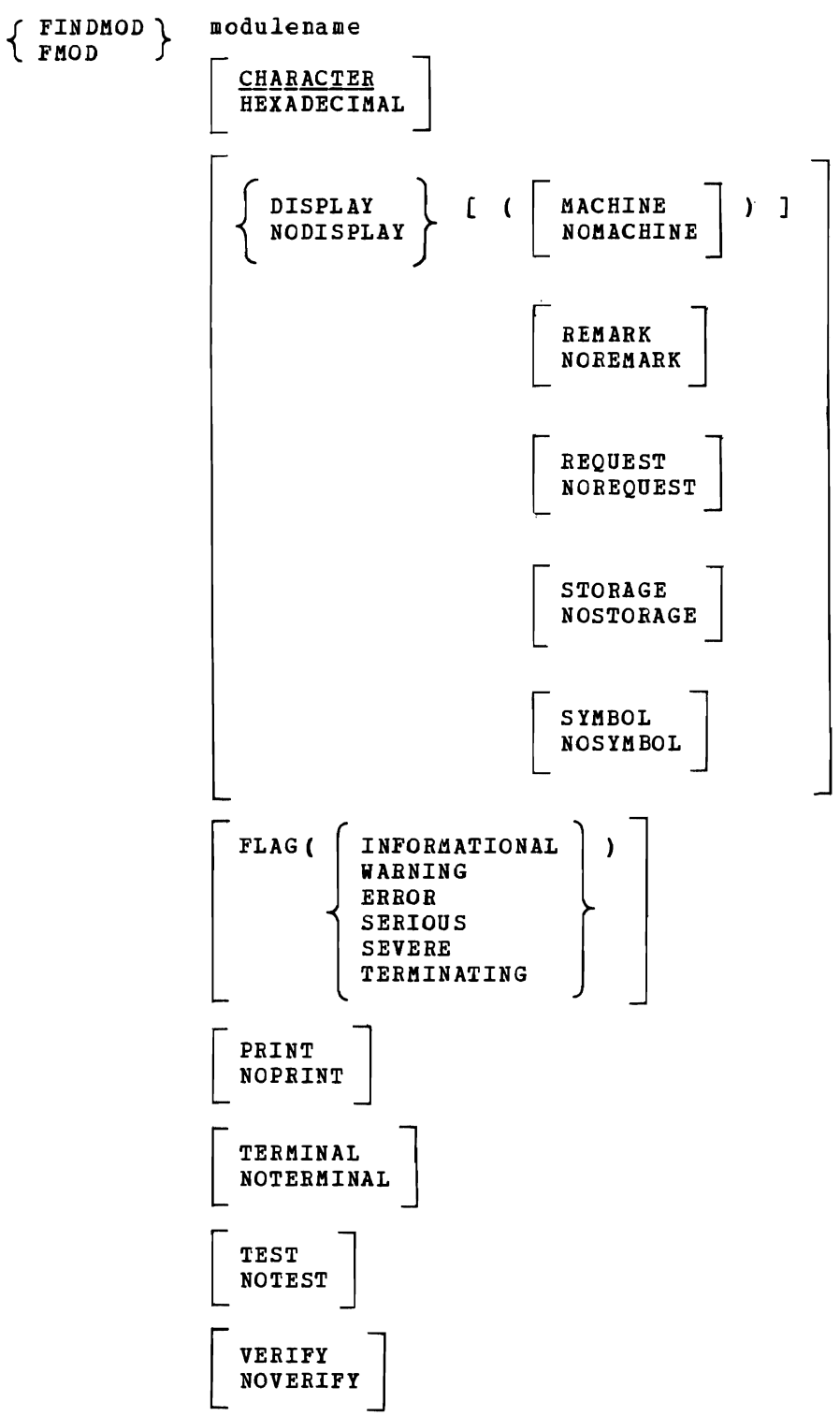
If it finds the requested module on the LPDE chain, it creates the symbols:

- LPDEmodulename
- modulename

NOTE: This subcommand may modify X, the current address.

RELATED SUBCOMMANDS:

- FIND
- FINDUCB


FINDMOD
FMOD
modulename
CHARACTER
HEXADECIMAL
DISPLAY
NODISPLAY
MACHINE
NOMACHINE
REMARK
NOREMARK
REQUEST
NOREQUEST
STORAGE
NOSTORAGE
SYMBOL
NOSYMBOL
FLAG(
INFORMATIONAL
WARNING
ERROR
SERIOUS
SEVERE
TERMINATING
)
PRINT
NOPRINT
TERMINAL
NOTERMINAL
TEST
NOTEST
VERIFY
NOVERIFY

modulename specifies the module name to be located.

CHARACTER specifies that the module name is specified as a string of 1 to 8 EBCDIC characters.

If you omit this keyword, the default is CHARACTER.

HEXADECIMAL specifies that the module name is specified as a string of 2 to 16 hexadecimal digits.

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by this subcommand is a 3-line header, followed by the data.

To produce this display, specify the VERIFY keyword. DISPLAY is ignored if you specify NOVERIFY.

If VERIFY is specified or defaulted, and the NODISPLAY keyword is also present, a conflict exists. In this case, IPCS generates a minimal display which is equivalent to the DISPLAY(REQUEST) keyword.

The DISPLAY keyword without values is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Displays the remark (if any) associated with the module name if it is found in the symbol table.

NOREMARK - Suppresses the display of the remark associated with the module name if it is found in the symbol table.

REQUEST - Displays a model LIST subcommand that could be used to display the information you requested. The LIST subcommand keywords include the data description keywords you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

If you want to modify the attributes of the displayed data, modify the keywords on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

STORAGE - Displays the storage at the specified or default address, for the specified or default length. The subcommand also displays the storage as in a printed dump: eight bytes in hexadecimal followed by the EBCDIC equivalent.

NOSTORAGE - Suppresses the storage display.

SYMBOL - Displays the specified module name if it is successfully located.

NOSYMBOL - Suppresses the specified module name.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand does not change the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

VERIFY specifies that the subcommand is to produce output and direct it to the destination or destinations specified by the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

NOVERIFY specifies that the subcommand is not to produce output regardless of the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

FINDMOD uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the FINDUCB subcommand to locate the unit control block for a specified device. When the subcommand finds the control block, it creates an entry in the symbol table for UCBddd, where ddd is the device address.

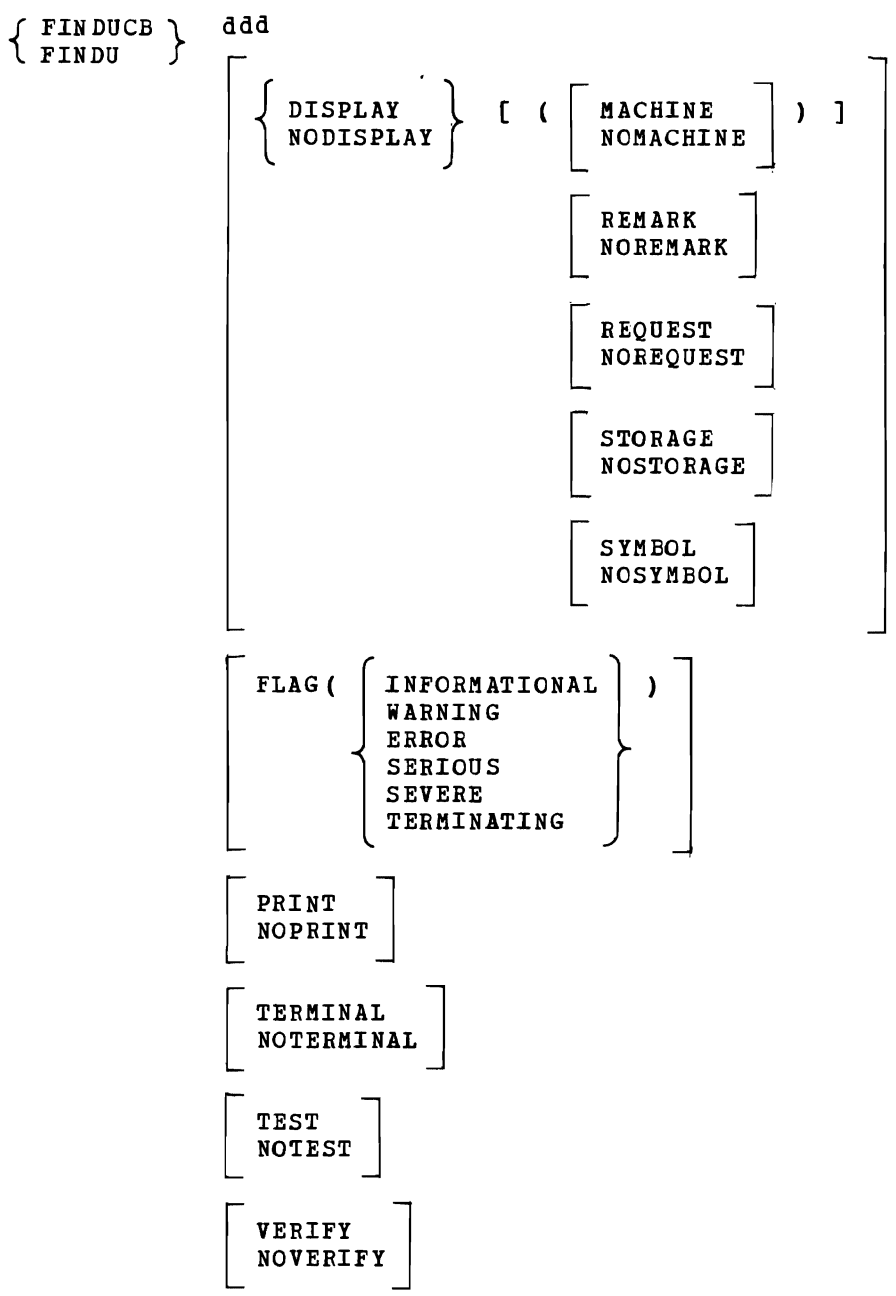
FINDUCB processes the specified device address as follows:

1. Searches the symbol table for the symbol UCBddd. If found, the subcommand displays the storage associated with that symbol.
2. Validates the channel portion of the device address.
3. Verifies that the device was defined during system generation.
4. Uses UCB look-up algorithm to locate the device's UCB.

NOTE: This subcommand may modify X, the current address.

RELATED SUBCOMMANDS:

- FIND
- FINDMOD




ddd specifies the address of the unit whose UCB is to be found. The address is 1 to 3 hexadecimal digits; leading zeros are optional.

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by this subcommand is a 3-line header, followed by the data.

To produce this display, specify the VERIFY keyword. DISPLAY is ignored if you specify NOVERIFY.

If VERIFY is specified or defaulted, and the NODISPLAY keyword is also present, a conflict exists. In this case, IPCS generates a minimal display which is equivalent to the DISPLAY(REQUEST) keyword.

The DISPLAY keyword without values is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Displays the remark (if any) associated with the specified UCB if it is found in the symbol table.

NOREMARK - Suppresses the display of the remark associated with the specified UCB.

REQUEST - Displays a model LIST subcommand that could be used to display the information you requested. The LIST subcommand operands include the data description operands you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

If you want to modify the attributes of the displayed data, modify the operands on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

STORAGE - Displays the storage at the specified or default address, for the specified or default length. The subcommand also displays the storage as in a printed dump: eight bytes in hexadecimal followed by the EBCDIC equivalent.

NOSTORAGE - Suppresses the storage display.

SYMBOL - This value is ignored by this subcommand.

NOSYMBOL - This value is ignored by this subcommand.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

VERIFY specifies that the subcommand is to produce output and direct it to the destination or destinations specified by the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

NOVERIFY specifies that the subcommand is not to produce output regardless of the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

FINDUCB uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

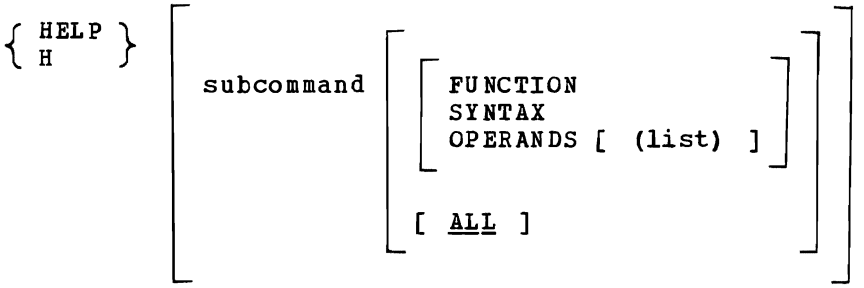
8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the HELP subcommand to obtain information about the function, syntax, and operands of IPCS subcommands. If you enter HELP with no operands you list all the IPCS subcommands.

---




---

subcommand specifies the name of the IPCS subcommand about which you want information.

If you omit this operand, the subcommand displays 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 operands for the specified subcommand. If you specify a list of keyword operands, HELP displays information about those operands. If you specify no operands, HELP displays information about all the operands for the specified subcommand.

ALL specifies that you want all the information available about the specified subcommand.

If you omit the FUNCTION, SYNTAX, and OPERANDS operands, the default is ALL.

Use the IOSCHECK subcommand to analyze control blocks associated with the Input/Output Supervisor component and produce information and statistics about those control blocks. The subccmmand uses the CVT to locate the queue of UCBs and the I/O supervisor's queues for pending and active I/O requests. The subcommand validates the unit control blocks.

IOSCHECK accesses the following control blocks and areas:

- UCBs
- LCH queues
- IOCOM
- IOX
- IOQs
- IOSBs

After control block analysis, IOSCHECK produces the following statistics:

- The number of devices with active I/O.
- The number of teleprocessing devices with active I/O.
- The number of queued requests for each device.

IOSCHECK validates each UCB that is found. If the UCB is not valid, the subcommand produces the message

BLS18034I INVALID UCB AT ADDRESS adr

where adr is the virtual address of the invalid UCB.

IOSCHECK examines flag byte C in each UCB. If the flag byte is nonzero and the device is online or ready, the subcommand produces the message

BLS18033I UCBFLC FIELD FOR UNIT ddd IS X'xxx'

where ddd is the device address and xxx is the contents of the flag byte, in hexadecimal.

When all UCBs are analyzed, the subcommand produces the messages

BLS18035I nnn DEVICES WITH ACTIVE I/O
BLS18036I ttt OF THESE ARE TP DEVICES

where nnn is the number of devices with active I/O and ttt is the number of active teleprocessing devices.

After examining the logical channel queues, IOSCHECK produces the following message

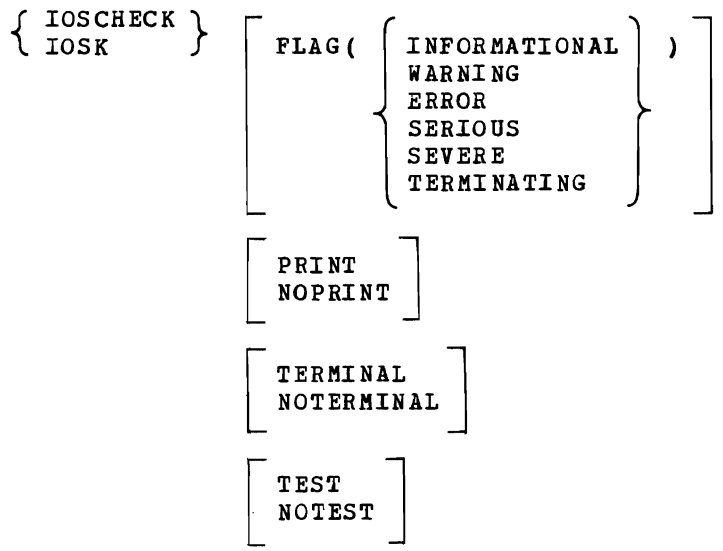
BLS18037I  nnn I/O REQUESTS QUEUED FOR UNIT ddd

where nnn is the number of pending I/O requests for unit ddd.

If no logical channels have pending I/O, IOSCHECK produces the message

BLS18084I  NO DEVICES WITH QUEUED I/O

---




---

FLAG(value)  specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

IOSCHECK uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

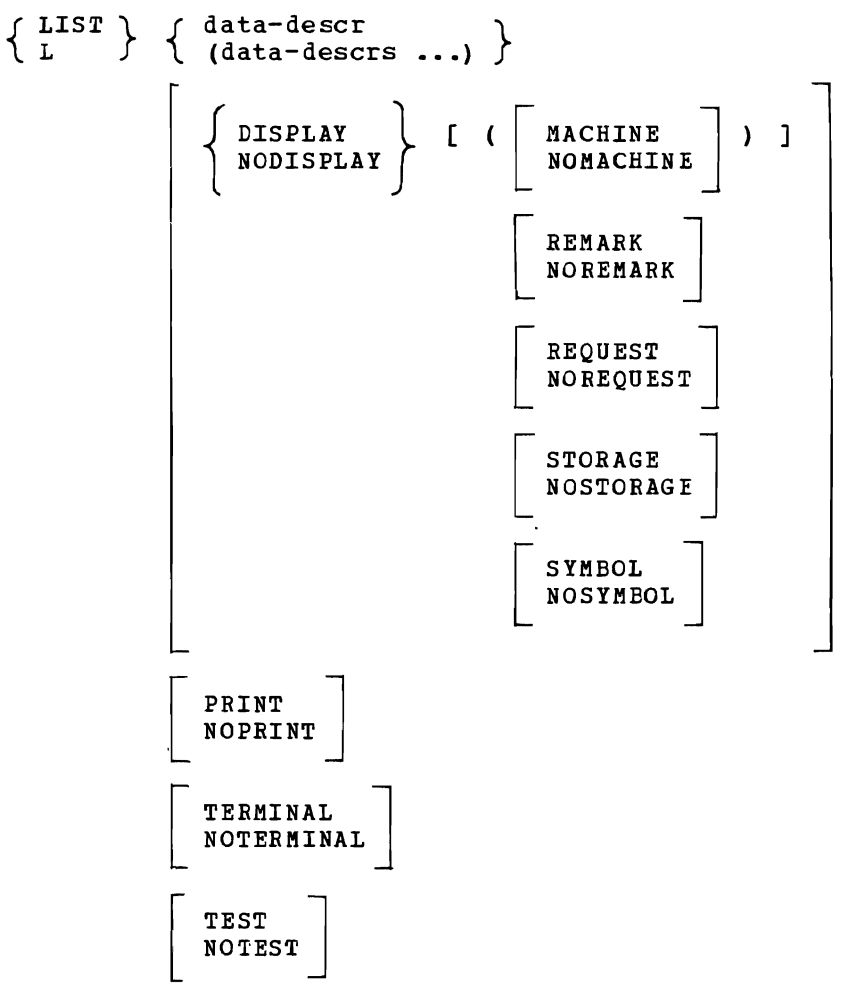
16 - Terminating, an error condition from a called service routine forced an early termination.

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 keywords.

NOTE: This subcommand may modify X, the current address.

RELATED SUBCOMMANDS:

- DSPL3270
- EQUATE
- FIND
- FINDMOD
- FINDUCB
- LISTMAP
- LISTSYM
- STATUS




data-descr or (data descrs ...) specifies a valid address or a list of addresses, and attributes. If you specify a list of addresses, any attributes (CPU, LENGTH, ASID, etc.) apply to all addresses in the list.

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by this subcommand is a 3-line header, followed by the data.

The DISPLAY keyword without values, is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

If the NODISPLAY keyword is present, a conflict exists with the purpose of this subcommand. In this case, IPCS generates a minimal display which is equivalent to the DISPLAY(REQUEST) keyword.

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Displays the remark (if any) associated with the specified SYMBOL.

NOREMARK - Suppresses the display of the remark associated with the specified symbol.

REQUEST - Displays a model LIST subcommand that could be used to display the information you requested. The LIST subcommand operands include the data description operands you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

If you want to modify the attributes of the displayed data, modify the operands on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

STORAGE - Displays the storage at the specified or default address, for the specified or default length. The subcommand also displays the storage as in a printed dump: eight bytes in hexadecimal followed by the EBCDIC equivalent.

NOSTORAGE - Suppresses the storage display.

SYMBOL - Displays the symbol specified with the data description operands, if one was specified.

NOSYMBOL - Suppresses the display of the symbol specified with the data description operands.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

TERMINAL specifies that the subcommand is to direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

LIST uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

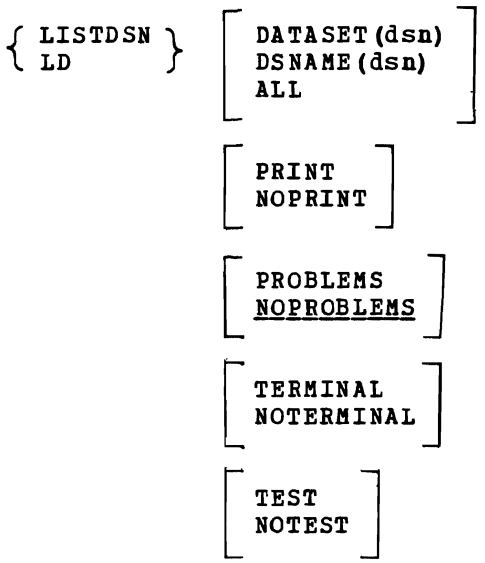
*LISTDSN – List Data Set Attributes*

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 execute LISTDSN any time during an IPCS session.

RELATED SUBCOMMANDS:

- ADDDSN
- DELDSN
- MODDSN

---




---

DATASET(dsn) 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 specifies that the subcommand direct its output to your print data set.

However, certain error messages are always directed to your terminal.

PRINT specifies that the subcommand direct its output to your print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to your print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

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.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

LISTDSN uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

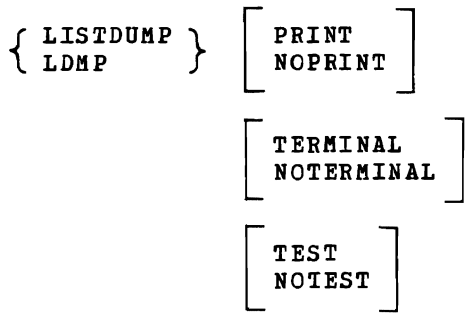
16 - Terminating, an error condition from a called service routine forced an early termination.

Use the LISTDUMP subcommand to display the names of the dump data sets represented in the dump directory in use during this IPCS session.

RELATED SUBCOMMANDS:

- DROPDUMP

---




---

PRINT specifies that LISTDUMP direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to the user's terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST  nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST  activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

LISTDUMP uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

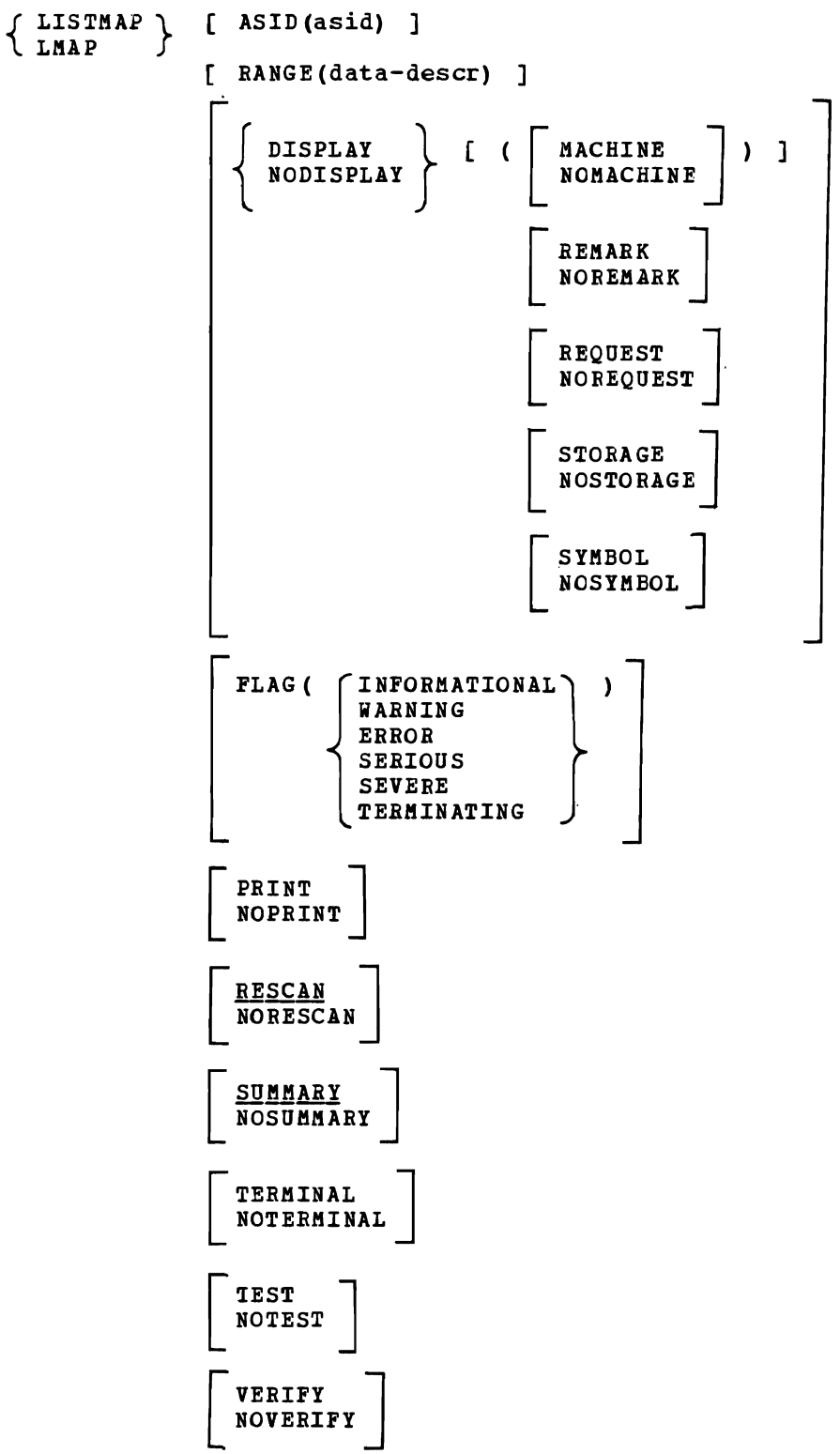
16 - Terminating, an error condition from a called service routine forced an early termination.

Use the LISTMAP subcommand to display the entries in the storage map for the current dump. The storage map describes the arrangement of data in a dump. The map contains entries for AREA(xxx), MODULE(yyy), and STRUCTURE(zzz) areas. Each storage map entry describes one such area and gives its address space, address in the address space, and data type.

For more information about the storage map, see "Using The Storage Map" in Chapter 4.

RELATED SUBCOMMANDS:

- DROPMAP
- SCAN


LISTMAP
LMAP
ASID(asid)
RANGE(data-descr)
DISPLAY
NODISPLAY
MACHINE
NOMACHINE
REMARK
NOREMARK
REQUEST
NOREQUEST
STORAGE
NOSTORAGE
SYMBOL
NOSYMBOL
FLAG(
INFORMATIONAL
WARNING
ERROR
SERIOUS
SEVERE
TERMINATING
PRINT
NOPRINT
RESCAN
NORESCAN
SUMMARY
NOSUMMARY
TERMINAL
NOTERMINAL
TEST
NOTEST
VERIFY
NOVERIFY

ASID(asid) specifies the address space identifier whose storage map entries are to be displayed. The ASID can range from 1 through 999. To process summary dump records, use the special ASID 65,530 (X'FFFA'). You can specify the ASID in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

If you omit this keyword, the subcommand uses the default address space identifier.

RANGE(data-descr) specifies a range of addresses in the specified address space. LISTMAP displays the storage map entries within this range of addresses.

If you specify the STRUCTURE keyword with a data type, it causes the subcommand to create a map record.

If you omit this keyword, the default range is 0.:FFFFFF..

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by this subcommand is a 3-line header, followed by the data.

The DISPLAY keyword without values is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

If VERIFY is specified or defaulted, and the NODISPLAY keyword is also present, a conflict exists. In this case, IPCS generates a minimal display which is equivalent to the DISPLAY(REQUEST) keyword.

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Since there are no remarks in storage map entries, this value is ignored.

NOREMARK - Since there are no remarks in storage map entries, this value is ignored.

REQUEST - Displays a model LIST subcommand that could be used to display the information you requested. The LIST subcommand keywords include the data description keywords you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

If you want to modify the attributes of the displayed data, modify the keywords on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

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: eight bytes in hexadecimal followed by the EBCDIC equivalent.

NOSTORAGE - Suppresses the storage display.

SYMBOL - Since there are no symbols in storage map entries, this value is ignored.

NOSYMBOL - Since there are no symbols in storage map entries, this value is ignored.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that LISTMAP direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that LISTMAP not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

RESCAN requests a reexamination of the storage map entries in the specified address space and address range that have been completely validated by the SCAN subcommand and that have an error condition that meets or exceeds the current default FLAG setting. RESCAN reexecutes the SCAN subcommand on the data area.

NORESCAN suppresses a reexamination of storage map entries that meet or exceed the default FLAG setting.

If you omit this keyword, the default is RESCAN.

SUMMARY specifies that LISTMAP display the information listed below.

If you omit this keyword, the default is SUMMARY.

SUMMARY displays the following information:

- Number of map entries in the specified address range.
- Number of map entries rescanned by this subcommand.
- Number of map entries verified by this subcommand.
- Number of map records for which no validation has been done.

NOSUMMARY suppresses the display of the SUMMARY information.

TERMINAL specifies that LISTMAP direct its output to the user's terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that LISTMAP not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

VERIFY specifies that the subcommand is to produce output and direct it to the destination or destinations specified by the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

NOVERIFY specifies that the subcommand is not to produce output regardless of the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

LISTMAP uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the LISTPROB subcommand to display a list of problems defined to IPCS.

The problems included in the report are specified with the selection keywords: 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 keyword 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 keywords. You can specify several values for each of these attribute keywords. However, to include a problem in the listing, its attributes must match at least one value specified for each attribute keyword.

If you specify no selection keywords, the subcommand lists the default problem.

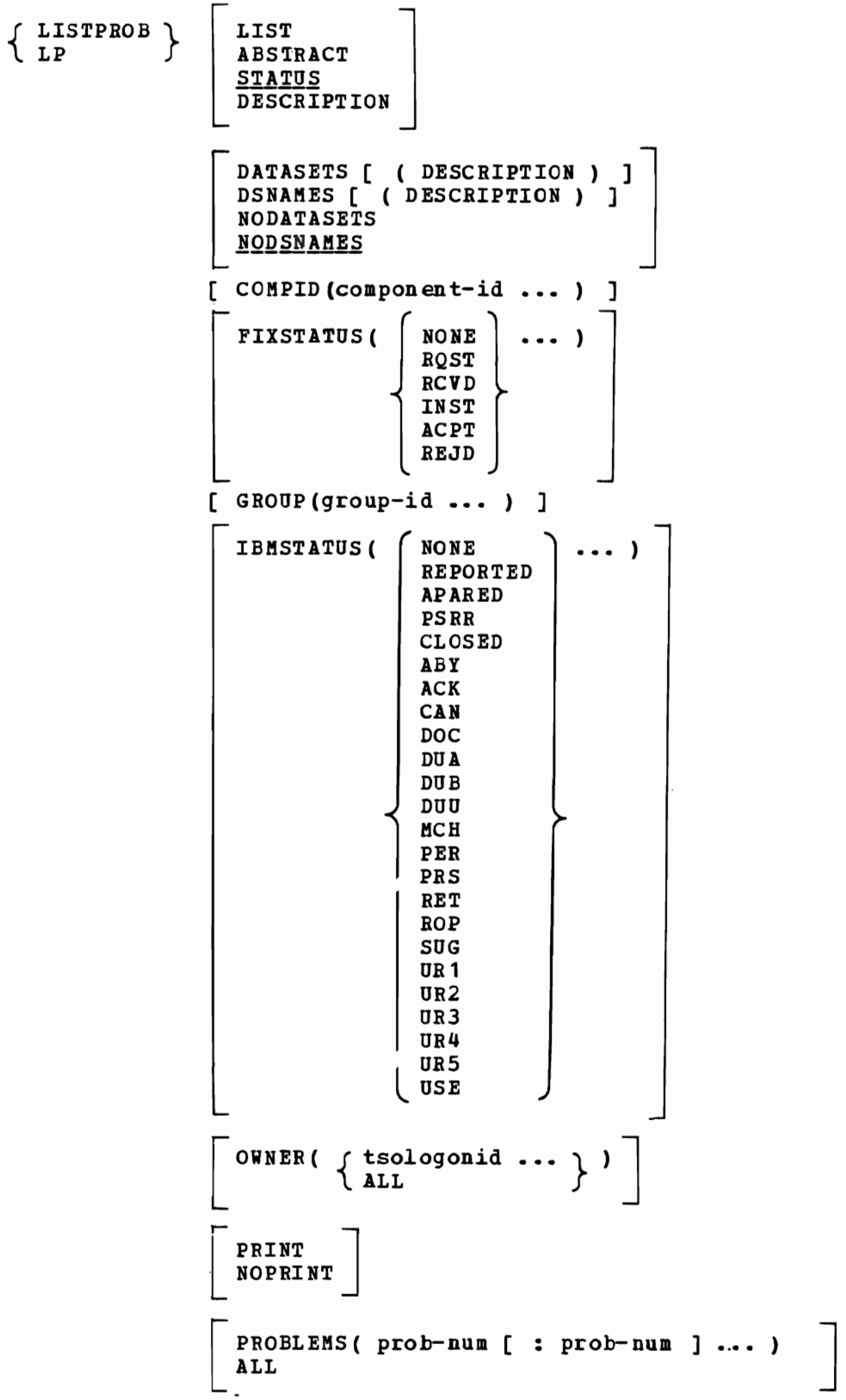
With the exception of OWNER, if you omit an attribute keyword, the subcommand does not consider that attribute when examining the problems. Thus, a problem is eligible for inclusion in the listing regardless of the omitted attribute's value.

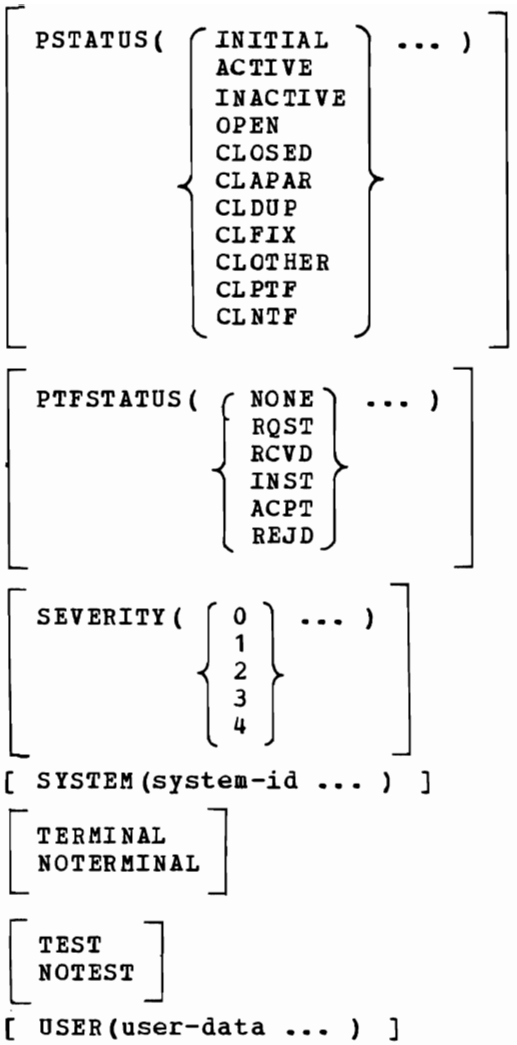
The required type of report is specified by the LIST, ABSTRACT, STATUS, DESCRIPTION, DATASETS, and DSNAMES keywords.

You can execute LISTPROB any time during an IPCS session.

RELATED SUBCOMMANDS:

- ADDPROB
- DELPROB
- MODPROB


LISTPROB
LP
LIST
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
PROBLEMS( prob-num [ : prob-num ] ... )
ALL




---

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 keyword 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 keyword 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 [ ( DESCRIPTION ) ] 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 keywords 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 keyword, 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 keyword, 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 keyword, 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 keyword, 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, nonacceptable APAR or duplicate of an APAR that was closed more than ten 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 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 keyword and all other selection keywords (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 keyword and specify one or more of the other selection keywords, the subcommand lists the problems owned by you that meet the specified selection criteria.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

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 keyword and all other selection keywords (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 keyword and specify one or more of the other selection keywords, 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 keyword and all other selection keywords (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 keyword and specify one or more of the other selection keywords, 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 keyword, 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 keyword, 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 keyword, problems are eligible for inclusion in the report regardless of their severity.

The severity values and their meanings are:

0 - The problem's severity has not been determined.

1 - The user is unable to use the program, resulting in a critical impact on his operations.

2 - The user is able to use the program but is severely restricted.

3 - The user is able to use the program with limited functions which are not critical to the overall operations.

4 - The user or the PSR 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 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 8 alphameric characters.

If you omit this keyword, problems are eligible for inclusion in the report regardless of their system identifier.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

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 8 alphameric characters.

If you omit this keyword, problems are eligible for inclusion in the report regardless of their user data.

LISTPROB uses the following return codes:

0 - Successful completion.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

Use the LISTSYM subcommand to display the entries in the symbol table for the current dump. For each symbol in the table, LISTSYM displays its:

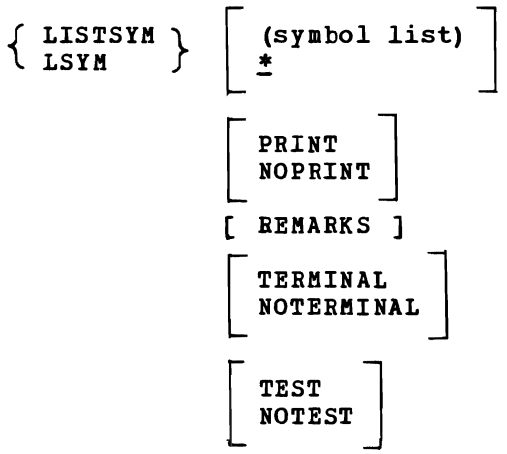
- Name.
- Address space.
- Address in the address space, and offset.
- Length.
- Dimension.
- Data type.

Optionally, the subcommand is used to display any remarks associated with the symbol.

RELATED SUBCOMMANDS:

- EQUATE
- DROPSYM

---




---

(symbol list) specifies the symbol or symbols to be displayed. You can specify a single symbol, a range of symbols, a list of symbols, or any combination of these. If you specify more than one symbol or range of symbols, enclose them in parentheses. The list can contain a maximum of 31 symbols, ranges of symbols, or both.

If you specify a single symbol or a list of symbols, the subcommand lists only the specified symbol or symbols. If you specify a range of

symbols, the subcommand lists 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 lists all the symbols in the range and at both ends of the range.

* specifies that the subcommand will display all symbols in the symbol table.

If you omit this operand, the default is *.

PRINT specifies that LISTSYM direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

REMARKS specifies that the display include any remarks associated with a symbol.

If you omit this keyword, remarks are not displayed.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

LISTSYM uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

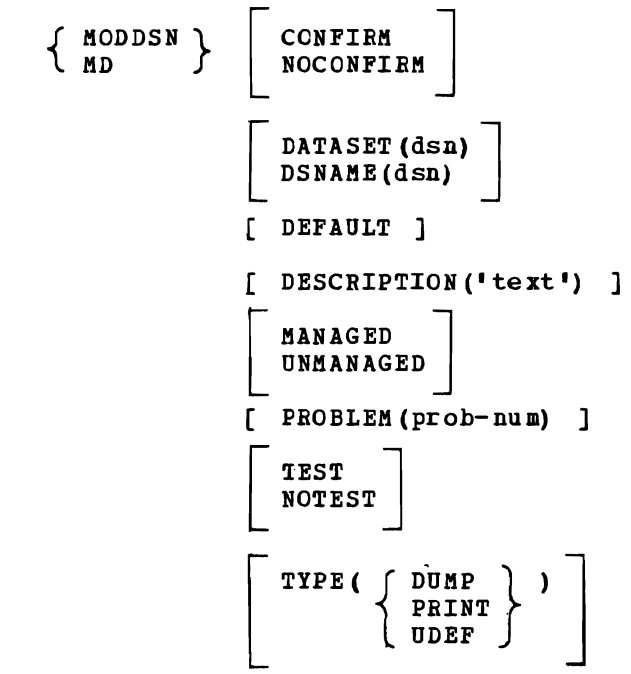
Use the MODDSN subcommand to alter the attributes of a data set or a partitioned data set member. The data set must be associated with the specified or default problem and you must own that problem or have administrative authority. The data set may be associated with other problems that you may or may 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 execute MODDSN at any time during an IPCS session to modify the attributes of data sets associated with problems he owns. The person with administrative authority can execute MODDSN to modify the attributes of any data set defined to IPCS. This subcommand does not change the current data set unless you specify the DEFAULT keyword.

RELATED SUBCOMMANDS:

- ADDDSN
- DELDSN
- LISTDSN




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 terminates without making changes to the data set's attributes and ignores the DEFAULT keyword, if specified.

If you omit both CONFIRM and NOCONFIRM, the subcommand uses the default for this keyword.

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 keyword.

DATASET(dsn) 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 this subcommand is to become the current data set if the subcommand executes successfully.

If you specify a data set name with a password, the data set name and password become the current data set name.

If you omit this keyword, if you enter N to resolve a data set attribute conflict, or if the subcommand fails, the current data set name is not changed.

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 keyword, 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 keyword, 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 nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

TYPE(value) specifies the type of data set.

If you omit this keyword, 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.

MODDSN uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

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 descriptiçn.

This subcommand does not change the default problem unless you specify the DEFAULT keyword.

RELATED SUBCOMMANDS:

- ADDPROB
- DELPROB
- LISTPROB

```
{ 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) ]
```

---

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 keyword, 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 keyword, 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 keyword, 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 for month and day.

If you omit this keyword, the subcommand does not change the date currently recorded for this problem.

DEFAULT specifies that the problem being modified is to become the default problem if the subcommand executes successfully.

If you omit this keyword or if the subcommand fails, the default problem is not changed.

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 keyword and you omit DSDESCRIPTION, the subcommand does not change the current description of the problem.

This keyword 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 keyword can be used only by the problem owner or the person with administrative authority.

If you omit this keyword 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 keyword.

If you omit this keyword, 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 keyword, the subcommand records the date and time of that specification.

If you omit this keyword, 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 keyword, 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 keyword, the subcommand records the date and time of that specification.

If you omit this keyword, 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 nonacceptable APAR or duplicate of an APAR which was closed more than ten 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 logon-id of the person responsible for resolving the problem.

If you omit this keyword, 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 keyword, the subcommand modifies the attributes of the default problem.

PSTATUS(value) specifies the status of the problem.

If you specify a value for this keyword, the subcommand records the date and time of that specification.

If you omit this keyword, 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 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 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 keyword, 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 keyword, the subcommand records the date and time of that specification.

If you omit this keyword, the subcommmand 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 keyword, the subcommand does not change the current severity value.

The severity codes and their meanings are:

1 - The user is unable to use the program, resulting in a critical impact on his operations.

2 - The user is able to use the program but is severely restricted.

3 - The user is able to use the program with limited functions which are not critical to the overall operations.

4 - The user or the PSR 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 8 alphameric characters.

If you omit this keyword, the subcommand does not change the current system identifier.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

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 keyword, 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 keyword, the subcommand does not change the current user-data field.

MODPROB uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

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

The maximum length of the message depends on its destination For terminal display, the message is truncated to 250 characters. For the print output data set, the message is truncated to the data set's logical record length, minus 5. Thus, a message directed to both places may be truncated to a different length for each destination.

You can also assign a severity level to the message 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.

---

```
{ NOTE } [ 'text' ]
{ N    }
          [ CAPS   ]
          [ ASIS   ]

          [ FLAG( { INFORMATICNAL } ) ]
                  { WARNING      }
                  { ERROR        }
                  { SERIOUS      }
                  { SEVERE       }
                  { TERMINATING  }

          [ PAGE   ]
          [ NOPAGE ]

          [ PRINT   ]
          [ NOPRINT ]

          [ SPACE [ (count) ] ]
          [ NOSPACE           ]
          [ OVERTYPE          ]

          [ TERMINAL   ]
          [ NOTERMINAL ]

          [ TEST   ]
          [ NOTEST ]
```

---

'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 for this operand, the subcommand assumes a blank line.

If you omit this operand, no message is transmitted but the subcommand performs the specified spacing or paging relative to the previous line on the terminal or in the print output data set. NOTE directs the message to the IPCSPRNT data set, to a user's terminal, or both, depending on the PRINT and TERMINAL keywords. If you omit the PRINT and TERMINAL keywords, NOTE uses the defaults for these keywords.

CAPS specifies that NOTE translate the message text to uppercase.

If you use this subcommand in a CLIST, the message text is automatically translated to uppercase.

If you omit both CAPS and ASIS, the default is CAPS.

ASIS specifies that NOTE not translate the message text to uppercase but transmit it in its present form.

If you use this subcommand in a CLIST, the message text is automatically translated to uppercase.

If you omit both CAPS and ASIS, the default is CAPS.

FLAG(value) assigns a severity level to the message which determines whether or not the message is sent to its destination.

If you omit this keyword, NOTE sends the message regardless of the current FLAG setting.

The flag values and their meanings are:

INFORMATIONAL - The message is sent to its destination only if the default flag value is INFORMATIONAL.

WARNING - The message is sent to its destination only if the default value for FLAG is INFORMATIONAL or WARNING.

ERROR - The message is sent to its destination only if the current default value for FLAG is INFORMATIONAL, WARNING, or ERROR.

SERIOUS or SEVERE - The message is sent to its destination only if the default value for FLAG is INFORMATIONAL, WARNING, ERROR, or SERIOUS or SEVERE.

TERMINATING - The message is always sent to its destination.

PAGE specifies that the message be printed on 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 keyword for that display.

NOPAGE specifies that a new page not be forced before printing the message.

If you omit both PAGE and NOPAGE, the default is NOPAGE.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

SPACE [ (count) ] specifies the number of blank lines 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) the subcommand uses PAGESIZE - 2. If this keyword 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.

If you omit SPACE, NOSPACE, and OVERTYPE, the default is NOSPACE.

OVERTYPE overlays this message on the previous message. For example, you may use this keyword to underscore all or part of the previous message. The subcommand ignores this keyword if you specify no text or if the output is directed to a terminal.

TERMINAL specifies that the subcommand direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTE uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

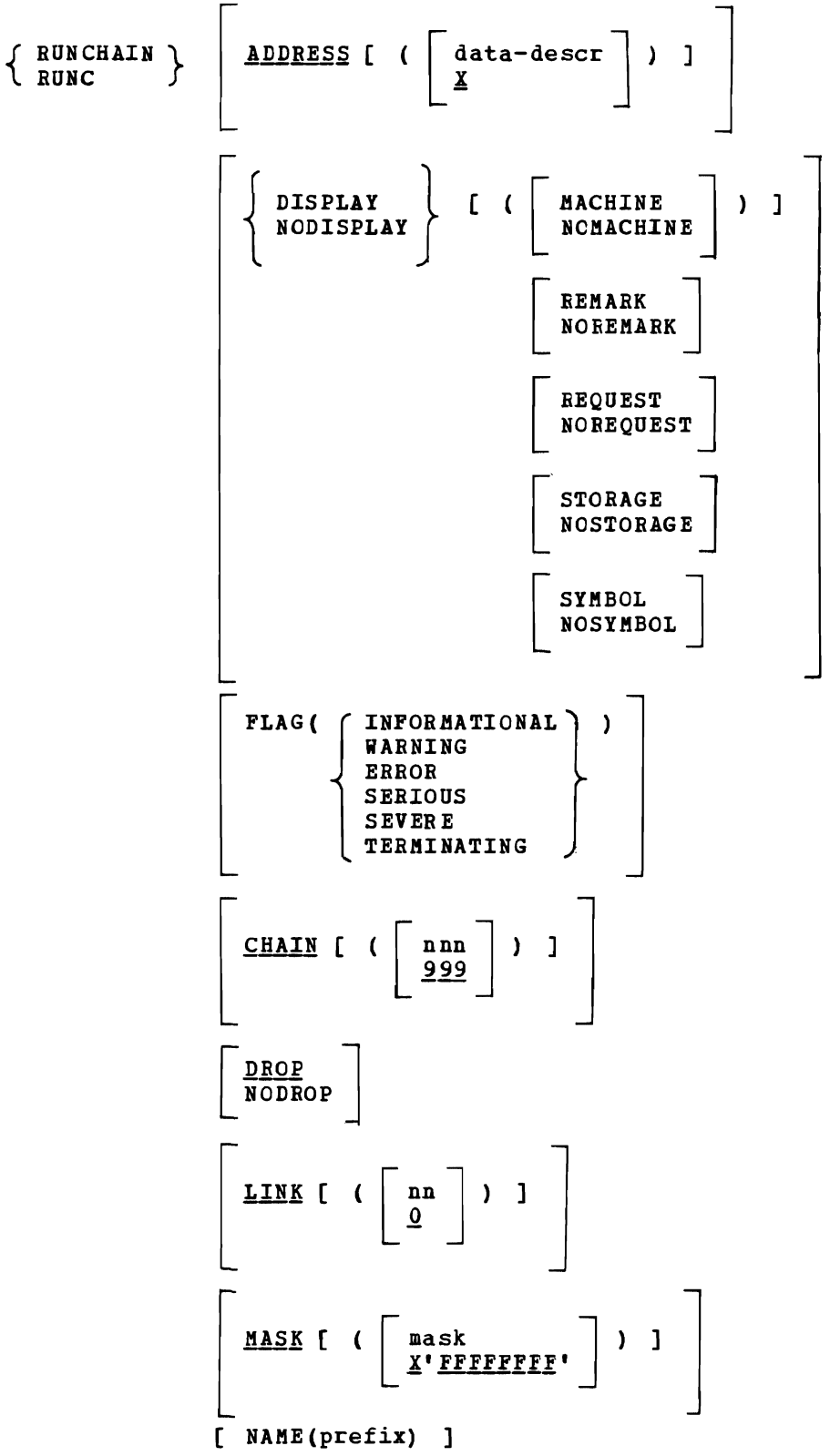
16 - Terminating, an error condition from a called service routine forced an early termination.

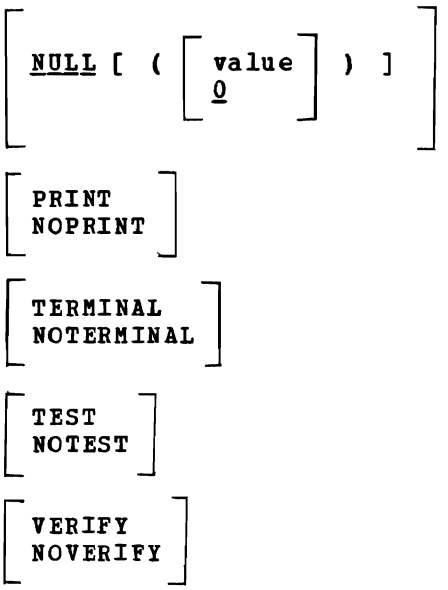
Use the RUNCHAIN subcommand to search a chain of control blocks. Optionally, you can request the subcommand to generate a name for each control block on the chain and enter that name in the symbol table.

NOTE: This subcommand may modify X, the current address.

RELATED SUBCOMMANDS:

- DROPSYM
- EQUATE
- LISTSYM


RUNCHAIN
RUNC
ADDRESS [ ( data-descr ) ]
X
DISPLAY
NODISPLAY
[ ( MACHINE
NOMACHINE ) ]
REMARK
NOREMARK
REQUEST
NOREQUEST
STORAGE
NOSTORAGE
SYMBOL
NOSYMBOL
FLAG( INFORMATIONAL
WARNING
ERROR
SERIOUS
SEVERE
TERMINATING )
CHAIN [ ( nnn ) ]
999
DROP
NODROP
LINK [ ( nn ) ]
0
MASK [ ( mask ) ]
X'FFFFFFFF'
[ NAME(prefix) ]




---

ADDRESS [ ( [ data-descr ] ) ] specifies the address and attributes of the first control block in the chain.

X specifies the current address.

If you omit this keyword, the default is ADDRESS(X).

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by this subcommand is a 3-line header, followed by the data.

To produce this display, specify the VERIFY keyword. DISPLAY is ignored if you specify NOVERIFY.

| If VERIFY is specified or defaulted, and the NODISPLAY keyword is also
| present, a conflict exists. In this case, IPCS generates a minimal
| display which is equivalent to the DISPLAY(REQUEST) keyword.

The DISPLAY keyword without values, is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Displays the remark (if any) associated with each symbol created by this subcommand.

NOREMARK - Suppresses the display of the remark associated with each symbol created by this subcommand.

REQUEST - Displays a model LIST subcommand that could be used to display the information you requested. The LIST subcommand keywords include the data description keywords you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

If you want to modify the attributes of the displayed data, modify the keywords on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

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: eight bytes in hexadecimal followed by the EBCDIC equivalent.

NOSTORAGE - Suppresses the storage display.

SYMBOL - Displays the symbol or symbols created by this subcommand as specified with the NAME keyword.

NOSYMBOL - Suppresses the symbol or symbols created by this subcommand.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

CHAIN [ ( [ nnn ] ) ] 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 keyword, the default is CHAIN(999).

DROP specifies that the names placed in the symbol table will have the DROP attribute which allows the symbols to be deleted from the symbol table with the DROPSYM subcommand.

RUNCHAIN places the names of the control blocks it finds in the symbol table only if you specify the NAME keyword.

NODROP specifies that the names placed in the symbol table will have the NODROP attribute which prevents the symbols from being deleted from the symbol table with the DROPSYM subcommand unless you specify the PURGE keyword on that subcommand.

Names are placed in the symbol table for the control blocks found only when you specify the NAME keyword.

LINK [ ( [ nn ] ) ] specifies the location in each control block of a fullword pointer to the next control block in the chain. The location is an offset relative to the first byte of the control block; the offset of the first byte is 0. 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 keyword, the default is LINK(0).

MASK [ ( [ mask ] ) ] specifies a mask that RUNCHAIN is to AND to the link field before comparing it to the value specified with the NULL keyword.

The length of the mask must be four bytes. If it is less than four bytes, the subcommand right-justifies it and pads it on the left with zeros. If it exceeds four 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 keyword, the default is MASK(X'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 alphabetic.

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 keyword.

The prefix for any control block may not exceed 30 characters.

If you omit this keyword, RUNCHAIN does not generate names for the control blocks it finds.

NULL [ ( [ value ] ) ] specifies the fullword value that indicates the end of the chain.

For each control block on the chain, RUNCHAIN:

- Locates the link field at the offset specified in the LINK keyword.
- 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.

PRINT specifies that RUNCHAIN direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that RUNCHAIN not direct its output to the print output data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that RUNCHAIN direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that RUNCHAIN not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

VERIFY specifies that the subcommand is to produce output and direct it to the destination or destinations specified by the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

NOVERIFY specifies that the subcommand is not to produce output regardless of the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand uses the default for this keyword.

RUNCHAIN uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the SCAN subcommand to validate MVS data areas. You can validate the following data areas:

| | | |
|---|---|---|
| ASCB | LPDEMINOR (3) | RB |
| ASVT | PCCA | SGTE |
| ASXB | PCCAVT | SIRB (2) |
| CDE | PFT | SVRB (2) |
| CDEMAJOR (1) | PFTE | TCB |
| CDEMINOR (1) | PGTE | TIRB (2) |
| CVT | PRB (2) | UCB |
| CVTXTNT2 | PSA | UCBCTC (6) |
| IOCOM | PVT | UCBDA (6) |
| IOX | QCB | UCBGFX (6) |
| IRB (2) | QCBMAJOR (4) | UCBTAPE (6) |
| LCCA | QCBMINOR (4) | UCBTP (6) |
| LCCAVT | QEL | UCBUR (6) |
| LPDE | QELTYPEL (5) | UCB3270 (6) |
| LPDEMAJOR (3) | QELTYPER (5) | XTLST |

Notes:

(1) These control blocks are validated as if they were specified as CDE and the correct structure type is stored in the symbol table and storage map.

(2) These control blocks are validated as if they were specified as RB and the correct structure type is stored in the symbol table and storage map.

(3) These control blocks are validated as if they were specified as LPDE and the correct structure type is stored in the symbol table and storage map.

(4) These control blocks are validated as if they were specified as QCB and the correct structure type is stored in the symbol table and storage map.

(5) These control blocks are validated as if they were specified as QEL and the correct structure type is stored in the symbol table and storage map.

(6) These control blocks are validated as if they were specified as UCB and the correct structure type is stored in the symbol table and storage map.

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.
  - Pointers.
  - Masks or bit maps.
- Offsets of certain fields in the control block.

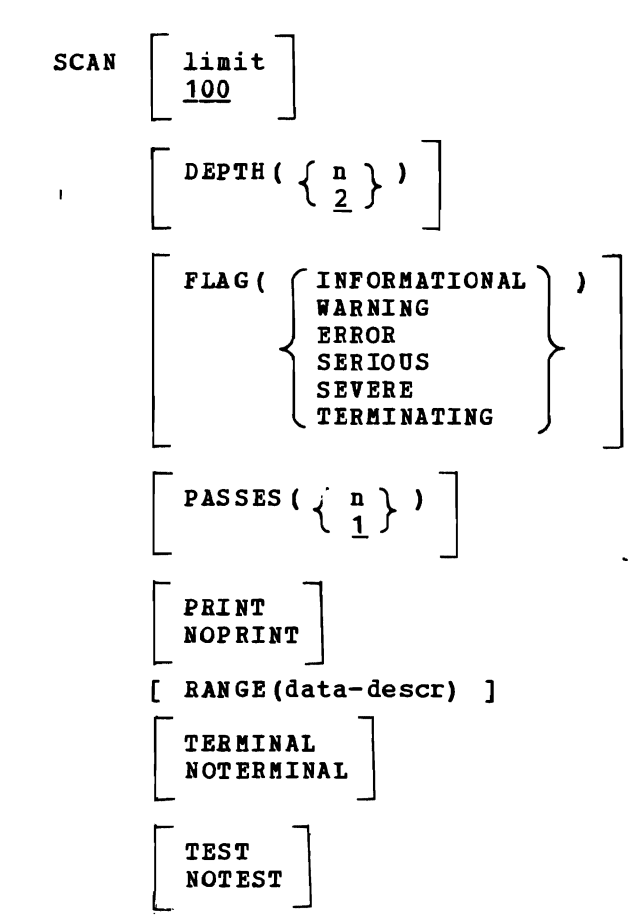
SCAN initiates its processing from your storage map and validates control blocks listed in the storage map that are within the address range you specified. As it validates each control block, SCAN makes new entries in the storage map for the control blocks pointed to from the block being validated. Depending on the DEPTH and PASSES keywords, new entries (control blocks) in the map may or may not be validated; however, if the new control blocks are found to be invalid, 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. Obviously, if you specify a large number for DEPTH, the scan probe of one control block could add many entries to your storage map. If the control block in question is the CVT or an ASCB, one scan probe could map all the AREAs and STRUCTUREs in the dump.

Dump initialization (see "Processing Dumps" in Chapter 2) ensures that there are entries in the storage map for the current dump. SCAN requires at least one entry in the storage map in order to begin its processing.

If a control block does not appear valid, SCAN issues a message to you specifying the control block in question, 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 invalid, the command issues two messages. The first message has a severity level of ERROR which informs you that the original control block contains a bad pointer. The second message has a severity level of SEVERE which informs you that the (alleged) new control block is invalid.




limit specifies the maximum number of scan probes that SCAN is to perform. The limit can range from 1 through 16,777,216 and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

If you omit this operand, the default is 100.

DEPTH(value) 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 limit can range from 1 through 65,535 and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

If you omit this keyword, the default is DEPTH(2).

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PASSES(value) specifies the number of times SCAN processes the storage map entries in the specified address range. The limit can range from 1 through 16,777,216 and can be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

As SCAN reprocesses the storage map, it does not revalidate control blocks previously validated.

If you omit this keyword, the default is PASSES(1).

PRINT specifies that SCAN direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that SCAN not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

RANGE(data-descr) 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 keyword specifies the addresses from which the SCAN probes start.

If you specify the STRUCTURE keyword 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 keyword, SCAN validates all storage map entries not previously validated. A control block may be only partially validated due to previous limits on DEPTH and PASSES on previous scans.

TERMINAL specifies that SCAN direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

SCAN uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the SETDEF subcommand to set, change, and display defaults for various keywords to IPCS subcommands.

SETDEF is a prerequisite for the successful execution of all dump analysis subcommands. You may use SETDEF to specify the dump data set. Optionally, you may use it to set other defaults that subsequent dump analysis subcommands are to use.

After you execute SETDEF to specify the first data set in an IPCS session, or to specify another data set during an IPCS session, the first dump analysis subcommand you execute causes initialization processing for the specified data set (see "Processing Dumps" in Section 2).

The defaults specified with SETDEF can be overridden on many subcommands by specifying the corresponding keyword on that subcommand. The override will not change the value specified with SETDEF.

You can execute SETDEF at any time during an IPCS session. This subcommand will not change a default value unless it is specified on the subcommand. Default values specified with the SETDEF subcommand are not retained between IPCS sessions.

Upon entry to IPCS mode and before issuing a SETDEF subcommand, the following values are in effect:

- ASID - null until you specify a dump data set. When you specify a dump and access it with a dump analysis subcommand, that subcommand sets ASID to the identifier of an address space contained in that dump.

  SETDEF rejects an attempt to set this option before you specify a dump.

- CONFIRM.

- CPU - null until you specify a multiprocessor stand-alone dump. When you specify a multiprocessor stand-alone dump and access it with a dump analysis subcommand, that subcommand sets CPU to the address of a central processing unit specified in that dump.

  SETDEF rejects an attempt to set this option before you specify a dump.

- DISPLAY (NOMACHINE REMARK REQUEST SYMBOL NOSTORAGE).

- NODSNAME.

- FLAG (WARNING).

- LENGTH(4).

- NOPRINT.

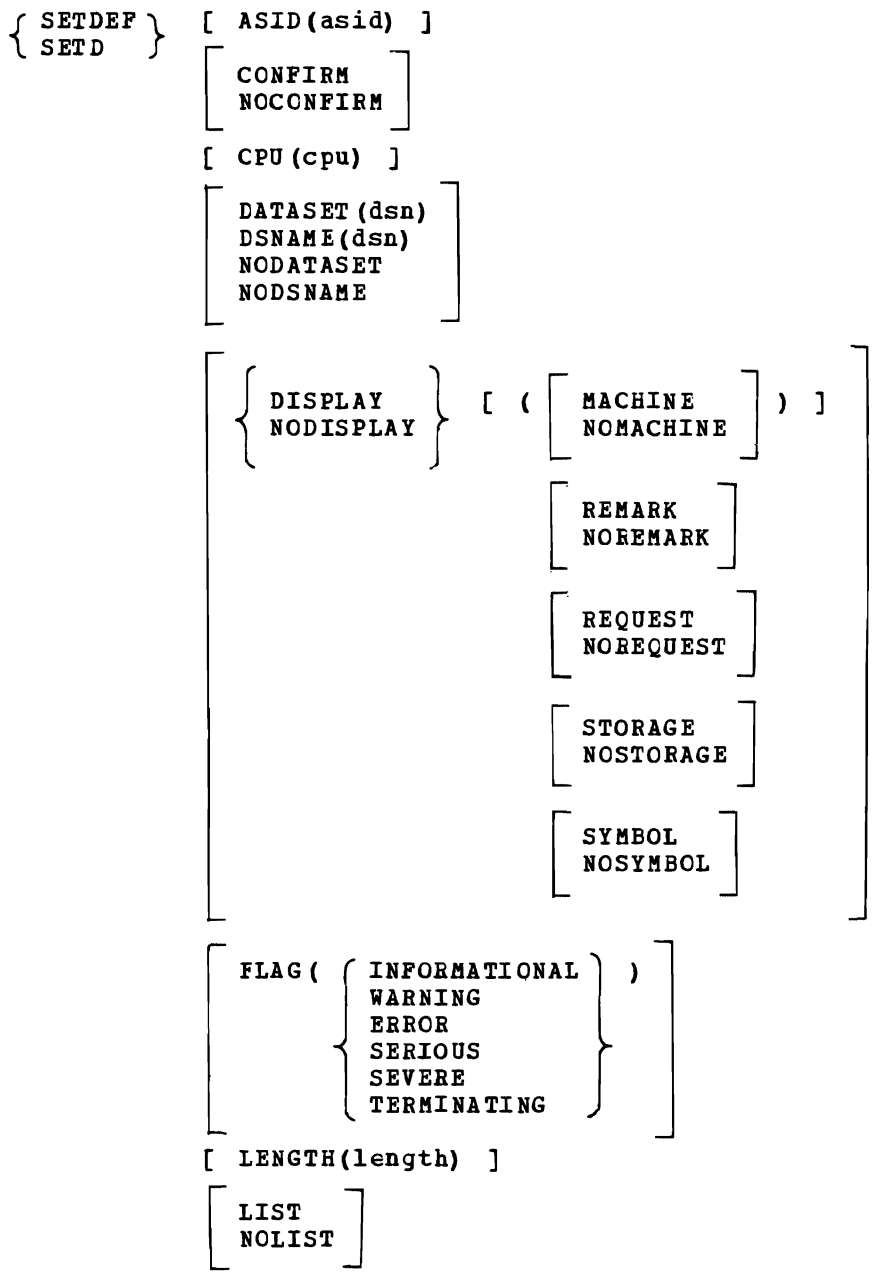
- NOTEST.

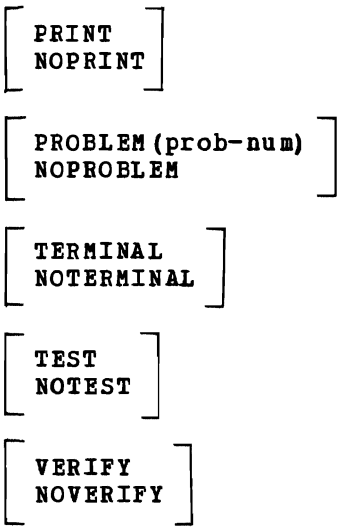
- NOPROBLEM.
- TERMINAL.
- VERIFY.

If all keywords that you specify on a SETDEF subcommand are valid, the subcommand sets the specified values.

If SETDEF rejects any keyword that you specify, the subcommand terminates without changing any values.

---






---

ASID(asid) specifies the address-space identifier used by the dump analysis subcommands. The ASID can range from 1 through 999. To process summary dump records, use the special ASID 65,530 (X'FFFA'). You can specify the ASID in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...').

You cannot specify an ASID before you specify a dump.

If the dump contains multiple address spaces, dump initialization sets ASID to the master scheduler's address space (ASID 1), if it is a stand-alone dump, or to the identifier of the address space that requested the dump, if it is an SDUMP.

If the dump contains only one address space, dump initialization sets ASID to that ASID.

If, while processing a dump, you execute SETDEF to specify a different dump and access the new dump with a dump analysis subcommand, dump initialization resets ASID to the address space identifier in the new dump, under the above conditions.

If you request a list of SETDEF options before a dump analysis subcommand sets ASID, SETDEF will not display the ASID.

If you omit this keyword, the subcommand does not change the current ASID.

CONFIRM specifies that various subcommands are to request confirmation before performing their function.

If you omit both CONFIRM and NOCONFIRM, the subcommand does not change the default for this keyword.

If you specify CONFIRM, the subcommand requires your confirmation before:

- Deleting a problem.
- Dissociating and scratching a data set.
- Modifying a data set's attributes (if it is associated with more than one problem).

The subcommands affected by this keyword are:

- ADDDSN
- DELDSN
- DELPROB
- MODDSN

NOCONFIRM specifies that various subcommands are not to request your confirmation before performing their function.

If you omit both CONFIRM and NOCONFIRM, the subcommand does not change the default for this keyword.

CPU(cpu) specifies the address of the central processing unit to be used by the dump analysis subcommands in a multiprocessor dump to simulate prefixing. The CPU address can range from 0 to 15 and may be specified in decimal, hexadecimal (X'xxx...'), or binary (B'bbb...' ).

You cannot specify a CPU before you specify a dump.

If the dump is a stand-alone dump for a multiprocessor system and you have not set the CPU, the dump analysis subcommand chooses the CPU to be used.

If the dump is either a stand-alone dump for a uniprocessor system or is not a stand-alone dump, the dump analysis subcommand does not set CPU, nor may the user.

If, when processing a dump, you execute SETDEF to specify a different dump and access the new dump with a dump analysis subcommand, the dump analysis subcommand resets CPU to null or to the central processor specified in the new dump, under the above conditions.

If you omit this keyword, the subcommand does not change the default CPU address.

DATASET(dsn) or DSNAME(dsn) specifies the name of the current data set to be used by the IPCS subcommands. You may specify a password, if the data set is password protected, as part of the data set name. If you omit the password and it is required, you are prompted for it.

If you do not specify either a data set name, NODATASET, or NODSNAME, the subcommand does not change the current data set name.

NODATASET or NODSNAME specifies that the subcommand is to set the current data set name to a null value.

If you do not specify either a data set name, NODATASET, or NODSNAME, the subcommand does not change the current data set name.

DISPLAY [ (value ...) ] specifies the content of the dump data displayed by this subcommand. The maximum information displayed by a dump analysis subcommand is a 3-line header, followed by the data.

If you omit both DISPLAY and NODISPLAY, SETDEF does not change the default for this keyword.

If VERIFY is specified or defaulted, and the NODISPLAY keyword is also present, a conflict exists. In this case, IPCS generates a minimal display which is equivalent to the DISPLAY(REQUEST) keyword.

The DISPLAY keyword without values is equivalent to DISPLAY(MACHINE REMARK REQUEST STORAGE SYMBOL).

The NODISPLAY keyword without values is equivalent to DISPLAY(NOMACHINE NOREMARK NOREQUEST NOSTORAGE NOSYMBOL).

The keyword values and their meanings are:

MACHINE - Displays the ASID, virtual address, storage key, and absolute address of the data area being displayed.

NOMACHINE - Suppresses the ASID, virtual address, storage key, and absolute address of the data area being displayed.

REMARK - Displays the remark associated with a symbol requested by the SYMBOL value.

NOREMARK - Suppresses the remark assocaited with a symbol requested by the SYMBOL value.

REQUEST - Displays a model LIST subcommand that is used to display the information you requested. The LIST subcommand keywords include the data description keywords you specify and other relevant default keywords (for example, CPU is relevant only for multiprocessor dumps, REMARK is never relevant).

If you want to modify the attributes of the displayed data, modify the keywords on the model LIST subcommand and execute it.

NOREQUEST - Suppresses the model LIST subcommand.

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.

NOSTORAGE - Suppresses the storage display.

SYMBOL - Displays the symbol (if any) associated with the dump data displayed.

NOSYMBOL - Suppresses the symbol associated with the dump data displayed.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines and by the NOTE subcommand. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump. Any subcommand that accesses the current dump might generate these messages.

If you omit this keyword, the subcommand does not change the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

LENGTH(length) specifies the length of the storage area to be used by dump analysis subcommands. The length may range from 1 to 16,777,216 and may be specified in decimal, hexadecimal, or binary.

If you omit this keyword, the subcommand does not change the current default LENGTH.

LIST specifies that the subcommand is to display all the default values and keywords. This includes values and keywords changed by this execution of SETDEF as well as values and keywords not changed by this execution of SETDEF.

If you specify LIST, the subcommand displays the defaults at your terminal regardless of the current value for the TERMINAL keyword. If the keyword is NOTERMINAL, it suppresses the display of subcommand output after SETDEF lists the defaults.

If you omit LIST and NOLIST but specify any other keyword, the default is NOLIST.

If you issue SETDEF without any keywords, the default is LIST.

NOLIST specifies that the subcommand not display the defaults.

If you specify any keywords but omit LIST and NOLIST, the default is NOLIST.

PRINT specifies that subsequent subcommands direct their output to the print data set. However, certain error messages are always directed to your terminal.

NOPRINT specifies that subsequent subcommands not direct their output to the print data set.

If you omit both PRINT and NOPRINT, the subcommand does not change the default for this keyword.

If you specify both NOPRINT and NOTERMINAL, subsequent subcommands force their output to the terminal.

PROBLEM(prob-num) specifies the default problem number. The problem number is five decimal digits; leading zeros are optional.

If you omit both PROBLEM and NOPROBLEM, the subcommand does not change the default problem number.

NOPROBLEM specifies that the subcommand sets the default problem number to a null value.

If you omit both PROBLEM and NOPROBLEM, the subcommand does not change the default problem number.

TERMINAL specifies that subsequent subcommands direct their output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand does not change the default for this keyword.

If you specify LIST, the subcommand displays the default values at your terminal regardless of the current value for TERMINAL.

NOTERMINAL specifies that subsequent subcommands not direct their output to your terminal. However, certain error messages are always directed to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand does not change the default for this keyword.

If you specify both NOPRINT and NOTERMINAL, subsequent subcommands force their output to the terminal.

TEST nullifies IPCS error recovery for ABENDs. When testing user-written exits, TEST may be used to obtain dumps via normal TSO procedures.

If you omit both TEST and NOTEST, the subcommand does not change the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand does not change the default for this keyword.

VERIFY specifies that subsequent subcommands are to produce output and direct it to the destination or destinations specified by the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand does not change the default for this keyword.

NOVERIFY specifies that subsequent subcommands are not to produce output regardless of the PRINT and TERMINAL keywords.

If you omit both VERIFY and NOVERIFY, the subcommand does not change the default for this keyword.

SETDEF uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

## *STATUS — Describe System Status*

Use the STATUS subcommand to display data usually examined during the initial part of the problem determination process.

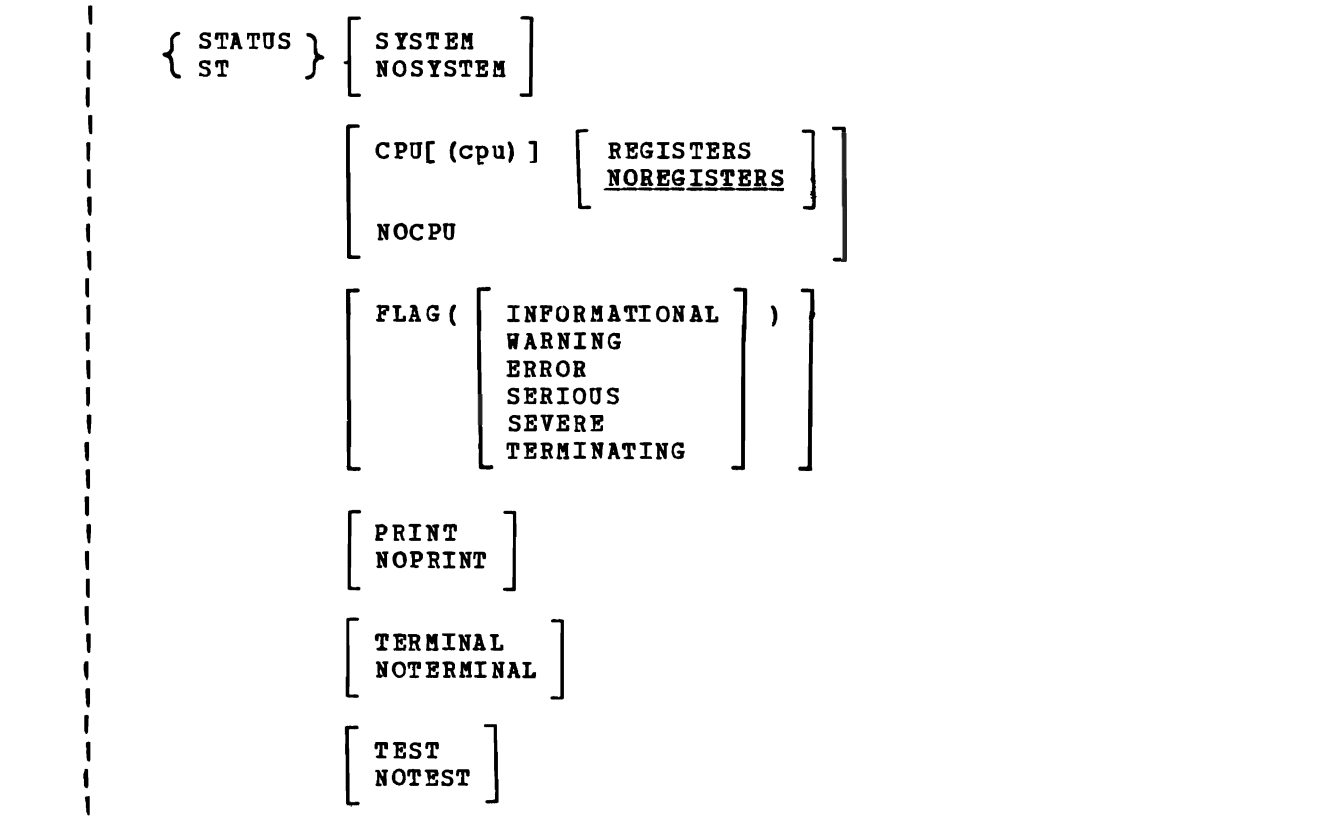
The STATUS subcommand produces two types of information: system status and CPU status. For system status, the subcommand displays the nucleus member name and time-of-day clock. For each CPU, the subcommand displays the PSW, contents of the general purpose registers and control registers, and the current ASCB, ASXB, and TCB. See "Output For STATUS" below for details.

Using the STATUS subcommand operands, you can selectively display which information you want to see.

RELATED SUBCOMMANDS:

- LIST
- SUMMARY

---




---

SYSTEM specifies that system status messages are desired.

NOSYSTEM specifies that system status messages are not desired.

If you omit both SYSTEM and NOSYSTEM, the default is SYSTEM, unless the CPU keyword is used. In that case, the default is NOSYSTEM.

CPU[(cpu)] specifies that CPU status messages are desired and, optionally, limits those messages to ones applicable to a designated CPU address, cpu.

REGISTERS specifies that a display of general purpose and control registers is desired as part of a display of CPU status. REGS may be used as an abbreviation of REGISTERS.

NOREGISTERS specifies that a display of general purpose and control registers is not desired. NOREGS may be used as an abbreviation of NOREGISTERS.

If you omit both REGISTERS and NOREGISTERS, the default is NOREGISTERS.

NOCPU specifies that CPU status messages are not desired.

If you omit both CPU and NOCPU, and
- the SYSTEM keyword is entered, the default is NOCPU.
- the SYSTEM keyword is not entered, the default is CPU.

FLAG(value) specifies the lowest severity level of messages to be produced. The messages are produced by various subroutines and by the NOTE subcommand. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump. Any subcommand that accesses the current dump might generate these messages.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual circumstances that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that the subcommand direct its output to the print data set. However, certain error messages are always directed to your terminal.

NOPRINT specifies that the subcommand not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that the subcommand direct its output to your terminal.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

STATUS uses the following return codes:

0 - Successful completion.

4 - Warning, an unusual (not necessarily an error) condition was detected during subcommand processing.

8 - Error, an error condition was detected during subcommand processing but did not prevent completion of processing.

12 - Severe, subcommand processing was terminated either as a result of an error condition detected in the dump or as a result of an action taken by you.

16 - Terminating, subcommand processing was terminated after detection of a problem in the IPCS execution environment.

## Output for STATUS

The STATUS subcommand generates two groups of messages:

1. One group of messages is concerned with the state of the overall system.

2. The other group of messages is concerned with the status of a CPU and may be repeated for each CPU online, if an absolute storage dump of a multiprocessor system is being examined.

## SYSTEM STATUS MESSAGES

```
SYSTEM STATUS:
NUCLEUS MEMBER NAME: IEANUC0n
TIME OF DAY CLOCK: xxxxxxxx xxxxxxxx  mm/dd/yy hh:mm:ss category
TIME OF DAY CLOCK: xxxxxxxx xxxxxxxx  mm/dd/yy hh:mm:ss category
```

SYSTEM STATUS - introduces the group of messages concerned with the state of the overall system.

NUCLEUS MEMBER NAME: IEANUC0n - identifies the IPLed nucleus. This information is obtained from field CVTNUCLS in the CVT extension addressed by field CVTEXT2.

TIME OF DAY CLOCK: xxxxxxxx xxxxxxxx mm/dd/yy hh:mm:ss category - displays a TOD clock value placed in the dump to indicate when the dump was produced. Two of these messages are produced.

The first message displays the value recorded in the dump data set. If a virtual storage dump is being processed, the category is LOCAL, a representation of the local date and time at the MVS site where the dump was produced. If an absolute storage dump is being processed, the category is GMT, a representation of the date and time in Greenwich, England, if recommended MVS conventions are employed.

The second message displays the time by translating the value in the dump using field CVTTZ in the CVT. If a virtual storage dump is being processed, the category is GMT, a representation of the local date and time at the MVS site where the dump was produced. If an absolute storage dump is being processed, the category is LOCAL, a representation of the date and time in Greenwich, England, if recommended MVS conventions are employed.

Thus, the time-of-day clock is presented in terms of both hardware and software values.

Each message displays the TOD clock value in hexadecimal (as indicated by xxxxxxxx xxxxxxxx), and interprets that value as a date and time of day.

CPU STATUS MESSAGES

```
CPU[(nn)] STATUS:
PSW=xxxxxxxx xxxxxxxx [(WAIT STATE)]
GPR VALUES:
 0R=xxxxxxxx    1R=xxxxxxxx    2R=xxxxxxxx    3R=xxxxxxxx
 4R=xxxxxxxx    5R=xxxxxxxx    6R=xxxxxxxx    7R=xxxxxxxx
 8R=xxxxxxxx    9R=xxxxxxxx   10R=xxxxxxxx   11R=xxxxxxxx
12R=xxxxxxxx   13R=xxxxxxxx   14R=xxxxxxxx   15R=xxxxxxxx

CONTROL REGISTER VALUES:
 0C=xxxxxxxx    1C=xxxxxxxx    2C=xxxxxxxx    3C=xxxxxxxx
 4C=xxxxxxxx    5C=xxxxxxxx    6C=xxxxxxxx    7C=xxxxxxxx
 8C=xxxxxxxx    9C=xxxxxxxx   10C=xxxxxxxx   11C=xxxxxxxx
12C=xxxxxxxx   13C=xxxxxxxx   14C=xxxxxxxx   15C=xxxxxxxx
ASCBnnn AT xxxxxx.[, JOB(cccccccc),] IS CURRENT
ASXBnnn AT xxxxxx. AND TCBnnnaa AT xxxxxx. ARE CURRENT
```

CPU[(nn)] STATUS - This originates a group of messages pertaining to one CPU. The CPU address, nn, is omitted if a virtual dump is being processed.

PSW=xxxxxxxx xxxxxxxx [(WAIT STATE)] - The current program status word is placed in the dump header record for virtual storage dumps. For absolute storage dumps the current program status word is obtained by performing a System/370 store status operation. Note: For dumps generated by AMDSADMP the system operator must remember to perform the store status operation before IPLing AMDSADMP. If the store status operation is forgotten, the PSW will not be accurate.

If the wait bit is on in the PSW, a note is added to call attention to that circumstance.

Register values are displayed for both the general purpose registers (GPR) and the control registers. The registers are displayed in hexadecimal.

ASCBnnn AT xxxxxx.[, JOB(cccccccc),] IS CURRENT - Field PSAAOLD in the PSA for the CPU is used to locate the current ASCB. The ASCB, in turn, is used to determine the current ASID, nnn, and the jobname, cccccccc, associated with the address space.

ASXBnnn AT xxxxxx. AND TCBnnnaa AT xxxxxx. ARE CURRENT - Field PSATOLD in the PSA for the CPU is used to locate the current TCB. The ASCB identified in the preceding message is used to define the current ASID, nnn, and the address of the current ASXB. The ASXB, in turn, is used to locate the TCB priority queue for the address space and to establish the position of the current TCB on the priority queue, aa. If the TCB cannot be located on the priority queue for the current address space, TCBnnnaa appears as TCBnnn??.

Use the SUMMARY subcommand to display key fields of the ASCB, TCBs, and RBs from the specified address space or address spaces. The subcommand uses the CVT to locate the chain of ASCBs. It then scans for the specified ASCBs. For each specified ASCB, it scans its TCB chain and for every TCB, it scans its RB chain. The subcommand generates symbol table and storage map entries for each ASCB, ASXB, and TCB it validates.

| The SUMMARY subcommand can also be used to selectively display TCBs, as
| well as the registers saved in TCBs and RBs processed by SUMMARY.

The TCB fields are displayed as they are encountered on the TCB chain originating in the ASXB. The RB fields are displayed as they are encountered on the RB chain originating in each TCB.

Arrows in the left margin of the display, next to nonzero TCBCMP and TCBRTWA fields, indicate abnormally terminated or failing tasks.

SUMMARY displays all the fields except JOB and CDNM in hexadecimal. JOB and CDNM are displayed in character format. SUMMARY uses the following labels.

Address space control block fields:

- ASID - The address space identifier (from the ASCBASID field).
- ASXB - The address of the address space extension block (from the ASCBASXB field).
- CSCB - The address of the command scheduling control block (from the ASCBCSCB field).
- DSP1 - The nondispatchability flags (from the ASCBDSP1 field).
- FLG2 - A flag byte (from the ASCBFLG2 field).
- FW1 - The flags in the ASCBFW1 field, including the CPU affinity flags.
- JOB - The job name (from the ASCBJBNI or ASCBJBNS fields), in character format. If neither of these fields point to a jobname, SUMMARY uses the INIT jobname.
- LOCK - The local lock (from the ASCBLOCK field).
- SRBS - The number of SRBs dispatched in this address space (from the ASCBSRBS field).
- TSB - The address of the terminal status block (from the ASCBTSB field).

Task control block fields:

- ABR - The ABEND recursion byte (from the TCBABCUR field) and the following three bytes, including the task identifier (from the TCBTID field).
- BITS - Nondispatchability flags (from the TCBBITS field).
- CMP - The task completion code, in hexadecimal (from the TCBCMP field).
- DAR - The damage assessment routine flags (from the TCBDAR field).
- FBTS - Task status flags (from the TCBFBYT1 field).
- JSCB - The address of the job step control block (from the TCBJSCB field).
- NDSP - Nondispatchability flags (from the TCBNDSP field).
- PKFG - A word of flags (from the TCBPKF field, bits 0 - 31).
- PKF2 - A second word of flags (from the TCBPKF field, bits 32 - 63).
- RTWA - The address of the current RTM2 (recovery/termination) work area (from the TCBRTWA field) and the task ID (from the three bytes following the TCBRTWA field).
- STAB - The address of the current STAE control block (from the TCBSTAB field).
- TSFG - Time-sharing flags (from the TCBTSFLG field).

Request block fields:

- CDNM - The module name in character format (from the contents directory entry (CDE) pointed to by the RBCDE field).
- FCDE - The contents of the RBCDE field. This includes flags and the address of the CDE or the LPDE for the module the RB represents.
- OPSW - The program status word that was stored when the error occurred (from the RBOPSW field).
- WLIC - The number of requests waiting at the time of termination, and the interrupt code (from the RBWCSA and RBINTCDA fields).
- WTLK - The wait count and the link address (taken from the RBLINK field).

During processing, SUMMARY creates entries in the symbol table for the ASCBs, TCBs, CDEs, XLs, and examined modules.

Register display by SUMMARY:

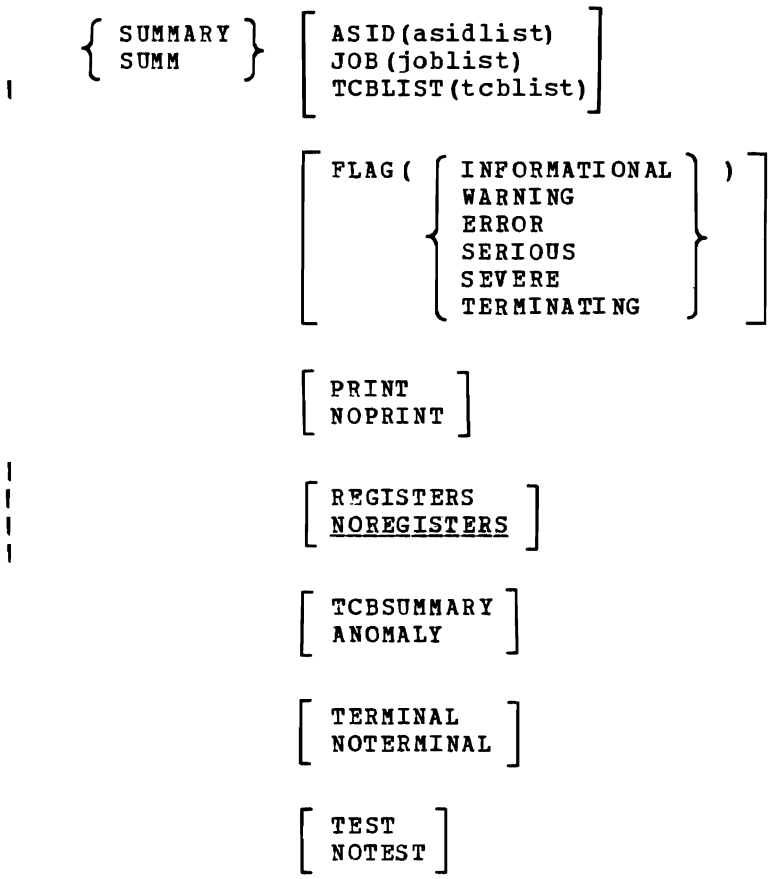
The SUMMARY subcommand produces a display of registers saved in the TCBs and RBs selected:

```
 0R  xxxxxxxx    1R  xxxxxxxx    2R  xxxxxxxx    3R  xxxxxxxx
 4R  xxxxxxxx    5R  xxxxxxxx    6R  xxxxxxxx    7R  xxxxxxxx
 8R  xxxxxxxx    9R  xxxxxxxx   10R  xxxxxxxx   11R  xxxxxxxx
12R  xxxxxxxx   13R  xxxxxxxx   14R  xxxxxxxx   15R  xxxxxxxx
```

xxxxxxxx displays the contents of the registers saved in the data area whose major field(s) have just been displayed. The values are displayed in hexadecimal.

RELATED SUBCOMMANDS:

- LISTSYM
- RUNCHAIN
- SCAN
- STATUS




ASID(asidlist) specifies the list of address space identifiers whose ASCB, TCB, and RB fields are to be displayed. You may specify one or more address space identifiers but not a range of identifiers. The ASID can range from 1 through 999. This subcommand does not process summary dump records (ASID X'FFFA').

If you omit ASID, JOB, and TCBLIST, the SUMMARY subcommand displays the ASCB, TCB, and RB fields for all address spaces in the dump.

JOB(joblist) specifies the list of job names whose ASCB, TCB, and RB fields are to be displayed. You may specify one or more job names but not a range of names. Job names must not be enclosed in apostrophes.

If you omit ASID, JOB, and TCBLIST, the SUMMARY subcommand displays the ASCB, TCB, and RB fields for all address spaces in the dump.

TCBLIST(tcblist) specifies the TCB identifiers (as found in the IPCS symbol table) whose TCB and RB fields are to be displayed. You may specify one or more TCBs but not a range of TCBs.

If you omit ASID, JOB, and TCBLIST, the SUMMARY subcommand displays the ASCB, TCB, and RB fields for all address spaces in the dump.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that SUMMARY direct its output to the print data set. However, certain error messages are always directed to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that SUMMARY not direct its output to the print data set.

If you specify both NOPRINT and NOTERMINAL, for this subcommand only, the subcommand produces no output to your terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

| REGISTERS specifies that the general purpose registers in each TCB/RB
| that is processed are to be displayed. The abbreviation REGS is also
| accepted for this keyword.

| NOREGISTERS specifies that the general purpose registers in the TCB/RB
| are not to be displayed. The abbreviation NOREGS is also accepted for
| this keyword.

| If you omit both REGISTERS and NOREGISTERS, the default is
| NOREGISTERS.

TCBSUMMARY specifies that SUMMARY display the:

- Job name

- ASCB name and address

- TCB name and address
- CMP field
- PKFG field
- PKF2 field
- TSFG field

If the TCBRTWA field is nonzero, the subcommand also displays the:

- DAR field
- RTWA field
- ABR field
- FBTS field

If you omit both TCBSUMMARY and ANOMALY, SUMMARY produces a full display.

ANOMALY specifies that SUMMARY display a subset of the fields requested by TCBSUMMARY. It displays these fields only for TCBs whose TCBCMP and TCBRTWA fields are nonzero.

If the TCBCMP field is nonzero, the subcommand displays the:

- CMP field
- PKFG field
- PKF2 field
- TSFG field

If the TCBRTWA field is nonzero, the subcommand also displays the:

- DAR field
- RTWA field
- ABR field
- FBTS field

If you omit both TCBSUMMARY and ANOMALY, SUMMARY produces a full display.

TERMINAL specifies that SUMMARY direct its output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that the subcommand not direct its output to your terminal. However, certain error messages are always directed to your terminal.

If you specify both NOTERMINAL and NOPRINT, for this subcommand only, the subcommand produces no output to your terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST  nullifies IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST  activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

SUMMARY uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the TCBEXIT subcommand to execute a user-supplied routine that is compatible with the AMDPRDMP TCB exit. The TCB exit routine must reside in a library allocated to your TSO session.

---

```
{TCBEXIT} pgmname
{TCBX   }
          addr
          [ FLAG( { INFORMATIONAL } ) ]
                  { WARNING       }
                  { ERROR         }
                  { SERIOUS       }
                  { SEVERE        }
                  { TERMINATING   }
          [ PRINT   ]
          [ NOPRINT ]
          [ TERMINAL   ]
          [ NOTERMINAL ]
          [ TEST   ]
          [ NOTEST ]
```

---

pgmname specifies the name of a user-supplied exit routine.

The routine name may be a maximum of 8 characters and the first character must be alphabetic. The specified routine must reside in a library allocated to your TSO session.

addr specifies the address of the task control block to be passed to the exit routine.

FLAG(value) specifies the lowest severity level of messages to be displayed at your terminal. The messages are produced by various subroutines. They are used by dump analysis subcommands to notify you of problems and unexpected situations in the current dump.

If you omit this keyword, the subcommand uses the default for this keyword.

The flag values and their meanings are:

INFORMATIONAL - Transmits all messages to your terminal.

WARNING - Suppresses informational messages, but transmits all others. Warning messages describe unusual conditions that are not necessarily wrong but could be errors.

ERROR - Suppresses informational and warning messages, but transmits all others. For example, error messages describe control blocks or data that point to invalid control blocks or data.

SERIOUS or SEVERE - Suppresses informational, warning, and error messages, but transmits all others. For example, serious or severe messages describe control blocks or data that are invalid.

TERMINATING - Transmits only terminating messages and suppresses all others.

PRINT specifies that TCBEXIT direct its output to the print data set. However, certain error messages are always directed to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that TCBEXIT not direct its output to the print data set.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that TCBEXIT direct its output to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that TCBEXIT not direct its output to your terminal. However, certain error messages are always directed to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs. When testing user-written exits, TEST may be used to obtain dumps through normal TSO procedures.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

TCBEXIT uses the following return codes:

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

If the exit routine puts a return code in register 15, that value is the return code from this subcommand.

Use the TSO subcommand to execute TSO commands whose names are identical to IPCS subcommands.

During an IPCS session, you can execute most TSO commands without leaving IPCS. If a TSO command name is different from the IPCS subcommand names, you may enter the TSO command name as if it were an IPCS subcommand. When IPCS determines the name is not an IPCS subcommand, it assumes that it is a TSO command and executes it as such.

However, if the TSO command name is identical to an IPCS subcommand (such as LIST), you must precede the command name with TSO to force IPCS to execute the TSO command rather than the IPCS subcommand.

---

```
TSO  tsocmmd [ operands ]
```

---

tsocmmd specifies the name of the TSO command to be executed.

operands specifies the operands of the TSO command to be executed.

TSO uses the following return codes:

0 - Successful completion.

4 - Warning, subcommand completed with a condition that may be of note to the user.

8 - Error, subcommand encountered an error condition that may be of note to the user.

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

Use the VERBEXIT subcommand to execute a user-supplied exit routine that is compatible with the AMDPRDMP user control statement exit. The verb exit routine must reside in a library allocated to your TSO session.

---

```
{ VERBEXIT }  pgmname
{ VERBX    }
              [ parm ]
              [ PRINT   ]
              [ NOPRINT ]
              [ TERMINAL   ]
              [ NOTERMINAL ]
              [ TEST   ]
              [ NOTEST ]
```

---

pgmname specifies the name of the verb exit routine.

The verb routine name may be a maximum of 8 alphameric characters; the first character must be alphabetic.

The specified routine must reside in a library allocated to your TSO session.

parm specifies a parameter string to be passed to the exit routine. The parameter string must be enclosed in apostrophes; when the subcommand passes the string to the exit routine, it omits the apostrophes.

The parameter string content and its meaning are defined by the verb exit routine.

PRINT specifies that VERBEXIT direct its output to the print data set. However, certain error messages are always directed to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

NOPRINT specifies that VERBEXIT not direct its output to the print data set.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you specify both NOPRINT and NOTERMINAL, the subcommand forces the output to the terminal.

If you omit both PRINT and NOPRINT, the subcommand uses the default for this keyword.

TERMINAL specifies that VERBEXIT direct its output to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

NOTERMINAL specifies that VERBEXIT not direct its output to your terminal. However, certain error messages are always directed to your terminal.

This keyword applies only to the output transmitted via the IPCS simulation of the AMDPRDMP print service routine interface.

If you specify both NOTERMINAL and NOPRINT, the subcommand forces the output to the terminal.

If you omit both TERMINAL and NOTERMINAL, the subcommand uses the default for this keyword.

TEST nullifies IPCS error recovery for ABENDs. When testing user-written exits, TEST may be used to obtain dumps through normal TSO procedures.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

NOTEST activates IPCS error recovery for ABENDs.

If you omit both TEST and NOTEST, the subcommand uses the default for this keyword.

VERBEXIT uses the following return codes:

12 - Severe, an error condition or user request forced early termination of the subcommand.

16 - Terminating, an error condition from a called service routine forced an early termination.

If the exit routine puts a return code in register 15, that value is the return code from this subcommand.

# Chapter 6. Installation Planning and Use

This chapter provides information that you will need to prepare for, install, and use IPCS. More specific information about the actual installation procedure, including sample JCL for the installation, is contained in the program directory, shipped with the IPCS selectable unit from IBM's Program Information Department (PID).

IPCS uses several data sets to record problems, names of data sets associated with problems, accumulated information about dumps associated with problems, and printed output from subcommands. It also requires one or more partitioned data set members for session parameters.

## Data Sets Used by IPCS

| IPCS uses six data sets to contain session parameters and help you track
| and solve problems. By properly allocating the following data sets, as
| described below, IPCS will be ready for an IPCS session. The data sets
| are the:

| • Session parameters member.

| • Data set directory.

| • Problem directory.

| • Dump directory.

| • Print output data set.

| • Dump data set.

| Note: The shared DASD environment is not specifically supported for the
| IPCS VSAM data sets (namely, the problem directory, data set directory,
| and dump directory). Please refer to the VSAM publications listed in
| the preface regarding restrictions applicable to the use of shared DASD.

### *Session Parameters Members*

IPCS session parameters allow an installation to tailor IPCS sessions to its particular requirements. The parameters define the names of various data sets and default values to be used throughout that IPCS session. When you execute the IPCS command, IPCS initialization processes these parameters. Thus, you cannot modify or respecify them during an IPCS session.

The parameters are contained in a member of SYS1.PARMLIB. The member name is IPCSPRnn where nn is a two-digit decimal suffix ranging from 00 to 99. You specify this suffix as an operand to the IPCS command and thereby specify the particular IPCSPRnn member to be used for that session (see the IPCS command for a complete description of this operand). If you specify no suffix on the IPCS command, it defaults to 00. Member IPCSPR00 is distributed with the IPCS installation package.

The installation may have several IPCSPRnn members. You then specify the member that suits your needs for a particular IPCS session. A particular IPCSPRnn member must be added to SYS1.PARMLIB before you can specify that member for an IPCS session.

The parameters that can be specified in an IPCSPRnn member and their meanings are:

DSD(dsn) specifies the data set name of the IPCS data set directory. The data set name must be fully qualified and need not be in apostrophes. IPCS appends nothing to either end of the specified name.

This parameter is required. If not specified, the IPCS command terminates.

PDR(dsn) specifies the data set name of the IPCS problem directory. The data set name must be fully qualified and need not be in apostrophes. IPCS appends nothing to either end of the specified name.

This parameter is required. If not specified, the IPCS command terminates.

PROBIDPREFIX(prefix) specifies a three-character prefix used in creating a problem identifier. This three character value is used whenever an IPCS subcommand displays or prints a problem identifier. You never specify this prefix when specifying a problem to a subcommand. The value must be three alphameric characters. IPCS concatenates a five-digit decimal number to this prefix to form the complete problem identifier.

This parameter is required. If not specified, the IPCS command terminates.

SYSTEM(system-id) specifies the default system identifier. The system identifier can be a maximum of eight alphameric characters. ADDPROB uses this value if you omit the SYSTEM keyword on that subcommand.

This parameter is optional. If not specified, the default system identifier is blank.

GROUP(group-id) specifies the default group identifier. The group identifier can be a maximum of eight alphameric characters. ADDPROB uses this value if you omit the GROUP keyword on the subcommand.

This parameter is optional. If not specified, the default group identifier is blank.

ADMINAUTHORITY(tsologonid) specifies the TSO userid of the person with IPCS administrative authority. The owner of the specified TSO userid has the same authority that a problem owner has, but he has that authority over <u>all</u> problems in the problem directory. Specifying an administrator does not affect the privileges of a problem owner. An owner can still modify and delete problems that he owns.

Specifying a person with administrative authority provides centralized control over all problems defined to IPCS. This administrator can assign and reassign problem owners, access a problem when the owner is unavailable, update problems when their attributes change, correct attributes that were incorrectly specified, etc.

This parameter is optional. If not specified, no one has administrative authority for the installation.

DELETEAUTHORITY(tsologonid) specifies the TSO userid of the person with delete authority. The owner of the specified TSO userid is the only person who can delete problems. Specifying a person with delete authority diminishes the privileges of a problem owner and the person with administrative authority. They can modify problems but can not delete them.

Specifying a person with delete authority provides centralized control over removing problems from IPCS. When a problem is deleted, all the accumulated information about it is lost. By designating one person who can delete problems you can help prevent the inadvertent loss of such information.

This parameter is optional. If specified, only the designated person can use the DELPROB subcommand. Problem owners cannot use the DELPROB subcommand nor can the person designated with the ADMINAUTHORITY parameter (if different from the person specified with this parameter).

If not specified, problem owners and the person specified with the ADMINAUTHORITY parameter can use the DELPROB subcommand.

LINELENGTH(value) specifies the default LRECL for the IPCS print output data set. The length may be a two or three digit decimal number, the minimum is 83 and the maximum is 255.

This parameter is optional. If you do not specify an LRECL for the print output data set and if it is not specified in the session parameters, the default is 137.

PAGESIZE(value) specifies the default number of lines per page for the IPCS print output data set. The value is specified as a decimal number. The minimum is 3, the maximum is $2^{31}-1$.

This parameter is optional. If not specified, the default is 60.

## Processing Session Parameters Members

A session parameters member is processed as follows.

### COMPOSING A LOGICAL LINE

IPCS processes a session parameters member by creating logical lines from card images. A logical line is formed by extracting text from columns 1 through 72 of one or more card images and using the extracted text to compose a single block of text. If column 72 of a card contains neither a plus sign nor a minus sign, then column 72 is included in the block composed. If column 72 of a card does contain a plus or minus sign, column 72 itself is excluded and the next card contributes to the logical line as follows:

- If a plus sign is used to indicate continuation, blanks at the beginning of the next card are omitted when text is extracted. This permits the use of a continuation style similar to that required in System/370 Assembler Language.
- If a minus sign is used to indicate continuation, column 1 of the next card is the first column extracted.

Columns 73 through 80 of all cards are ignored and may be used for sequence numbers.

SYNTAX

A session parameters member is processed by iteratively:

- Composing a logical line from one or more card-images.
- Parsing those operands which appear on a logical line.

After the entire member has been processed through this sequence, IPCS determines whether a usable specification has been provided.

CODING KEYWORDS AND VALUES

All syntax examination occurs after a logical line has been prepared. Multiple logical lines may be used in a session parameters member but any keyword-value pair started on a logical line must be completed on the same logical line. Multiple keyword-value pairs may be specified on a logical line.

It is recommended that you code session parameters member keywords exactly as shown. However, any unambiguous truncation of a keyword is accepted. For example, PROB(XYZ) is processed as if PROBIDPREFIX(XYZ) had been coded.

Blanks and commas are interchangeable and any number may appear:

- Before any keyword.
- Between any keyword and a left parenthesis which follows the keyword.
- Between a left parenthesis and a value.
- Between a value and a right parenthesis.
- Between a right parenthesis and the end of the logical line.
- Between a right parenthesis and the origin of another keyword coded on the same logical line.

A remark may be inserted at any point that a blank or comma may appear. The remark must begin with "/*" and will remain open until:

- A "*/" explicitly ends the remark.
- The end of the logical line is encountered.

A semicolon logically terminates operand processing for a logical line.

### *Data Set Directory*

The IPCS data set directory contains the list of data sets associated with the problems defined in the corresponding problem directory. The IPCS data set directory contains, for each data set associated with a problem, the attributes of the data set and the identifier of the problem or problems the data set is associated with.

The data set directory is a VSAM indexed cluster. It is created with Access Method Services. The data set directory must be created and initialized before the first IPCS session at your installation.

The name of the data set directory must be specified in the IPCS session parameters (see "Session Parameters Member" earlier in this chapter). If the session parameters do not specify the data set directory name, the IPCS command terminates. You do not allocate the data set directory for an IPCS session. IPCS dynamically allocates it during IPCS initialization.

Once allocated, IPCS opens and closes the data set as needed but leaves it allocated for the duration of the session. IPCS allocates the data set directory with a disposition of SHR and serializes access to it.

The characteristics of this VSAM key-sequenced cluster are:

```
INDEXED KEYS(61 0) RECORDSIZE(95 144) SHAREOPTIONS(1)
```

The DASD space required by this cluster is variable. Assuming a control interval size of 4096, approximately 57 data set record groups fit on a 3330 track, assuming each data set is associated with 2 problems.

The following example uses the DEFINE command to create a data set directory cluster. Replace the data set name with a name meaningful to you.

```
DEFINE CLUSTER(NAME('IPCS.DATA.SET.DIRECTRY') VOLUMES(111111)
 INDEXED KEYS(61 0) CYLINDERS(3 3) RECORDSIZE(95 144)
 CONTROLINTERVALSIZE(4096) SHAREOPTIONS(1) )
```

Assuming that 111111 is a 3330 volume, the space allocation in this example is sufficient for approximately 3000 data sets fitting the description above.

The data set directory is initialized by placing a seed record into it. The following procedure uses TSO and Access Method Services during a TSO session for the initialization and assumes that the data set directory has been previously defined and allocated using Access Method Services:

1. Use TSO EDIT to create a new data set and place a single record in it. This is the seed record.

   ```
   edit 'dsdseed' data new nonum
   ```

   Create the following record in character format:

   ```
                                                    5
   1                                                3
   00000000000 ... 00000000000000000000000000000DUMYRECD0000
   <---------------- 53 zeros ----------------->
   ```

2. Save the data set.

3. Allocate the VSAM cluster previously defined as the data set directory:

   ```
   alloc fi(b) ds('IPCS.DATA.SET.DIRECTRY') old
   ```

   where 'IPCS.DATA.SET.DIRECTRY' is the name of the data set directory specified in your session parameters member (see "Session Parameters Member" earlier in this chapter).

4. Use the Access Method Services REPRO command to put the seed record into the data set directory:

```
repro ids('dsdseed') outfile(b)
```

where 'outfile(b)' specifies the data set directory, allocated in step 3.

5. Free the data set directory:

```
free fi(b)
```

You must define multiple data set directories if you define multiple problem directories. Each data set directory must be paired with its own problem directory since records in each data set point to records in the other.

### *Problem Directory*

The IPCS problem directory contains the list of problems defined at your IPCS installation. For each problem defined, it contains the problem attributes and the names of data sets associated with that problem.

The problem directory is a VSAM indexed cluster. It is created with Access Method Services. The problem directory must be created and initialized before the first IPCS session at your installation.

The name of the problem directory must be specified in the IPCS session parameters member (see "Session Parameters Member" earlier in this chapter). If the session parameters do not specify the problem directory data set name, the IPCS command terminates. You do not allocate the problem directory for an IPCS session. IPCS dynamically allocates the directory during IPCS initialization.

Once allocated, IPCS opens and closes the data set as needed but leaves it allocated for the duration of the session. IPCS allocates the problem directory with a disposition of SHR and serializes access to it.

The characteristics of this VSAM key-sequenced cluster are:

```
INDEXED KEYS(14 0) RECORDSIZE(225 952) SHAREOPTIONS(1)
```

The DASD space required by this cluster is variable. Assuming a control interval size of 4096, approximately 6 problem record groups fit on a 3330 track, assuming 5 data sets associated with the problem and 10 lines of description per problem.

The following example uses the DEFINE command to create a problem directory cluster. Replace the data set name with a name meaningful to you.

```
DEFINE CLUSTER(NAME('IPCS.PROBLEM.DIRECTRY') VOLUMES(111111)
 INDEXED KEYS(14 0) CYLINDERS(5 5) RECORDSIZE(225 952)
 CONTROLINTERVALSIZE(4096) SHAREOPTIONS(1) )
```

Assuming that 111111 is a 3330 volume, the space allocation in this example is sufficient for approximately 570 problems fitting the description above.

The problem directory is initialized by placing a seed record into it.

The following procedure uses TSO and Access Method Services during a TSO session for the initialization and assumes that the problem directory has been previously defined and allocated using Access Method Services:

1. Use TSO EDIT to create a new data set and place a single record in it. This is the seed record.

   ```
   edit 'pdseed' data new nonum
   ```

   Create the following record in character format:

   ```
                        1
    1 3                 9
    SV00000000000000000xxxxx
      <-- 17 zeros --->
   ```

   where xxxxx represents a problem number. IPCS starts problem number assignment at xxxxx+1. For a new problem directory the xxxxx field is usually zeros. Thus, the seed record for a new problem directory is:

   ```
                           2
    1 3                    4
    SV0000000000000000000000
      <----- 22 zeros ----->
   ```

2. Save the data set.

3. Allocate the VSAM cluster previously defined as the problem directory:

   ```
   alloc fi(a) ds('IPCS.PROBLEM.DIRECTRY') old
   ```

   where 'IPCS.PROBLEM.DIRECTRY' is the name of the problem directory specified in your session parameters (see "Session Parameters Member" earlier in this chapter).

4. Use the Access Method Services REPRO command to put the seed record into the problem directory:

   ```
   repro ids('pdseed') outfile(a)
   ```

   where 'outfile(a)' specifies the problem directory, allocated in step 3.

5. Free the problem directory:

   ```
   free fi(a)
   ```

You can define multiple problem directories. This allows you to group problems and keep such groups completely separate. If you do define multiple problem directories, it is important that you pair each of them with its own data set directory. You should always use the same problem directory with the same data set directory since records in each data set point to records in the other.

### *Dump Directory*

A dump directory contains the symbol table and storage map records. These records contain, among other things, pointers to the dump records that contain the storage associated with a map entry or a symbol.

You can define a different dump directory for each dump, for groups of dumps, or for each user. Multiple dump directories allow you to use one directory when processing problems in one group and another directory when processing problems in another group. For ease in assigning and reassigning problems, you might want a separate dump directory for each problem. The user's dump directory must be allocated before executing the IPCS command.

The characteristics of this VSAM key-sequenced cluster are:

```
KEYS(128 0) RECORDSIZE(256 3072) SHAREOPTIONS(1)
```

The following example uses the DEFINE command and the IPCSDDIR command to create and initialize a dump directory data set. Replace the data set names with names meaningful to you.

```
DEFINE CLUSTER(NAME('IPCSU1.DUMPDIR.DEBUG') VOLUMES(111111) )
  INDEX(NAME('IPCSU1.DUMPDIR.DEBUGI') TRACKS(1 1) )
  DATA(NAME('IPCSU1.DUMPDIR.DEBUGD') CYLINDERS(1 1) KEYS(128 0)
    BUFFERSPACE(X'10000') CONTROLINTERVALSIZE(X'1000') )

IPCSDDIR 'IPCSU1.DUMPDIR.DEBUG'
```

Specifying too small a bufferspace size causes excessive explicit VSAM I/O operations. Specifying too large a size increases the IPCS working set beyond what your system is tuned to effectively support. You can adjust the value (64K) in the example for optimal performance on your system.

The DASD space required by this cluster is variable. It varies with the size of each dump, the number of dumps represented in it, and the amount of information it contains for each dump.

For most processing, allocate a dump directory with a number of tracks equal to 2-3% of the total number of tracks in the dumps to be represented in that directory. Extensive use of symbol tables and storage maps require increases in the space required by your dump directory.

### *Print Output Data Set*

A print output data set contains the output produced by IPCS subcommands executed with the PRINT operand. It is a sequential data set and is optional. For additional information about this data set, see "IPCS Session Control" in Chapter 2.

### *Dump Data Set*

A dump data set contains a description of a programming environment, usually immediately following the detection of an error. Dump data sets

that may be examined using IPCS are generated by two MVS service aids:

- The high-speed version of AMDSADMP writes dumps (via the SDUMP macro) to provide problem determination information regarding the circumstances at the time of a re-IPL.

- The MVS control program writes dumps to SYS1.DUMP data sets and tape to provide problem determination information regarding authorized portions of the control program. It also writes dumps to data sets associated with SYSMDUMP DDs to provide problem determination information regarding applications which operate in the MVS environment.

Both service aids produce dumps with the same characteristics:

DSORG=PS,RECFM=F,BLKSIZE=4104

Dump data sets must be cataloged and reside in sequential data sets on direct access storage during processing by IPCS.

The low-speed version of AMDSADMP, SYSABEND dumps, and SYSUDUMPs can not be processed by IPCS, but standard TSO facilities may be used.

## Limiting Access to TSO Commands and CLISTs

You may replace dummy IPCS module BLSUGWDM with your own validation routine, which can be used to disable access to TSO commands. BLSUGWDM is called during processing of the IPCS TSO subcommand and all TSO commands invoked under IPCS. Your command name validation routine is passed the command processor parameter list (CPPL) and the command scan output area (CSOA), as filled in by IKJSCAN.

Your routine may change CSOA bits to indicate that a TSO command is invalid (CSOABAD on), or that it is valid only as a CLIST (CSOAEXEC on). If the CSOAEXEC bit is on upon entry to your routine, it should not be turned off.

# Appendix. Control Block Acronyms

The following control block, table, and work area acronyms are used in this book. For more information about these control blocks, tables, and work areas, see the OS/VS2 MVS System Programming Library: Debugging Handbook, Volume 2 and 3.

This list shows each acronym and its meaning:

- ASCB - Address space control block.
- ASMVT - Auxiliary storage manager vector table.
- ASVT - Address space vector table.
- ASXB - Address space extension block.
- CDE - Contents directory entry.
- CSCB - Command scheduling control block.
- CSD - Common system data.
- CQE - Console queue element.
- CVT - Communications vector table.
- CVTXTNT2 - OS/VS1-OS/VS2 Common CVT extension.
- DCB - Data control block.
- GDA - Global data area.
- IOCOM - I/O communication area.
- IOX - I/O communication area extension.
- IRB - Interrupt request block.
- JESCT - Job entry subsystem control table.
- JSCB - Job scheduling control block.
- JQE - Job queue element.
- LCCA - Logical configuration communication area.
- LCCAVT - Logical configuration communication area vector table.
- LDA - Local data area.
- LPDE - Link pack directory entry.
- ORE - Operator reply element.
- OUCB - System resources manager user control block.

- OUSB - System resources manager user swappable block.
- OUXB - System resources manager user extension block.
- PART - Page address resolution table.
- PCB - Paging control block.
- PCCA - Physical configuration communication area.
- PCCAVT - Physical configuration communication area vector table.
- PFT - Page frame table.
- PFTE - Page frame table entry.
- PGT - Page table.
- PGTE - Page table entry.
- PRB - Program request block.
- PSA - Prefix storage area.
- PVT - Paging vector table.
- QCB - Queue control block.
- QEL - Queue element.
- RMCT - Recovery management control table.
- RQE - Request queue element.
- RSMHD - Real storage manager header.
- RTCT - Recovery/termination control table.
- RTM2WA - Recovery/termination manager 2 work area.
- SCVT - Secondary communications vector table.
- SGT - Segment table.
- SGTE - Segment table entry.
- SIRB - System interrupt block.
- SPQE - Subpool queue element.
- SRB - Service request block.
- SVRB - Supervisor call request block.
- TCB - Task control block.
- TIOT - Task I/O table.
- TIRB - Timer interrupt request block.

- TQE - Timer queue element.
- TSB - Terminal status block.
- UCB - Unit control block.
- UCBCTC - Unit control block, channel to channel adapter.
- UCBDA - Unit control block, direct access.
- UCBEXT - Unit control block extension.
- UCBGFX - Unit control block, graphics.
- UCBTAPE - Unit control block, tape.
- UCBTP - Unit control block, communications controller.
- UCBUR - Unit control block, unit record.
- UCB3270 - Unit control block, 3270.
- UCM - Unit control module.
- UCME - Unit control module entry.
- WSAVTC - Work/save area vector table.
- WQE - Write-to-operator queue element.
- WSAVTG - Work/save area vector tables.
- XTLST - Extent list.

&LASTCC 135

% sign (executing CLISTs) 17

A

ABEND 20
abnormal termination 20
ABSOLUTE keyword 66, 69
Access Method Services
  data set directory 254
  dump directory 15, 258
  problem directory 256
acronym list 261
add data sets to a problem 30
add problems to IPCS 11, 25, 79
ADDDSN subcommand 30, 75
ADDPROB subcommand 25, 79
ADDR (DSPL3270) 36, 38
address expression 68
address operands 66, 67
address processing keywords 66, 69
address ranges 68
address space control block 39
administrative authority 11, 25, 252
allocate data sets 15, 251
AMDPRDMP exits
  ASCBEXIT subcommand 87
  TCBEXIT subcommand 243
  VERBEXIT subcommand 249
AMDSADMP dumps 28
analyze ASCBs, TCBs, and RBs 39, 235
analyze auxiliary storage manager 39, 89
analyze communications task 39, 93
analyze dumps
  (see dump analysis)
analyze ENQ/DEQ component 39, 129
analyze I/O supervisor component 39, 157
analyze system components 39
  (see also dump analysis)
APAR closing codes (IBMSTATUS keyword)
  ADDPROB subcommand 79
  LISTPROB subcommand 177
  MODPROB subcommand 195
APARID keyword
  ADDPROB subcommand 79
  MODPROB subcommand 195
AREA keyword 66, 70
array keywords 67, 71
ASCBEXIT subcommand 53, 87
ASID (DSPL3270) 34
ASID keyword 66, 69
ASMCHECK subcommand 39, 89
associate data sets 30, 75
associate dump data sets 20
attention interrupts 20
attribute keywords 66, 70
attributes
  data set 27
    conflict 29, 75, 191
  problem 24
automatic locating of control blocks 44
auxiliary storage manager component 39, 89

B

background terminal monitor program 21
BIT keyword (B) 66, 70

C

chain control blocks 53
CHARACTER keyword (C) 66, 70
check status
  CPU 229
  system 229
check system components 39
CLISTs 17, 53
  DSPL3270 subcommand 38
  invoke during a session
    % sign 17
    examples 18
    TSO subcommand 17
  subcommands used in 53
COMCHECK subcommand 39, 93
command name validation 18, 259
commands, list of 13
communications task component 39, 93
compare data 53
COMPARE subcommand 53, 97
conflicts, data set attribute 29
control a session
  (see session control)
control blocks
  acronyms 261
  locating automatically 44

searching 207
summary of 235
validated 49
copy dump data sets 101
COPYDUMP subcommand 101
CPU keyword 66, 69
CPU status messages 233
create
data sets
dump directory 15
print output data set 15
storage map entries, EQUATE subcommand 48
symbols, EQUATE subcommand 45
current address
DSPL3270 subcommand 36
X symbol 43, 67
current data set 20

## D

data area
(see control blocks)
data description operands 65
resolving defaults 43
specification of 42
syntax 65
use of 40
data display area (DSPL3270) 35
data management 11, 23
data retrieval 135
data set allocation 15, 251
data set attributes
conflict 29, 191
listing 31
managed 27
modifying 31
type 28
data set directory 11, 254
data sets
for IPCS 251
data set directory 254
dump directory 15, 258
print output 15, 258
problem directory 256
session parameters member 251
for problems 28
associating 30
attribute conflicts 29
attributes 27
dissociating 30
listing attributes 31
modifying attributes 31
types
dump 28, 258
print 29
user-defined 29
management 11
modifying attributes 12
data validation 55
default data set 20, 28
default screen format (DSPL3270) 34, 121
defaults, setting of 18, 221
DEFINE command, TSO
data set directory 254
dump directory 15, 258
problem directory 256
DELDSN subcommand 30, 107
delete authority 11, 25, 252
delete data set from problem 30, 107
delete dump directory information 20
delete map records 51, 117
delete problem from IPCS 11, 25, 30, 111
delete symbols 47, 119
DELPROB subcommand 25, 111
DEQ component
(see ENQ/DEQ component)
DIMENSION keyword 67, 71
directories
data set 11, 254
dump 15, 258
problem 11, 256
display control block fields 235
display dump storage 46, 51, 161
display format (DSPL3270) 34, 121
dissociate a data set 107
DELDSN subcommand 30
drop dump directory information 20, 115
drop map records 51, 117
drop symbols 47, 119
DROPDUMP subcommand 20, 115
DROPMAP subcommand 51, 117
DROPSYM subcommand 47, 119
DSPL3270 subcommand 33, 121
termination 39
dummy IPCS validation routine 259
dump analysis 12, 19, 33
bad data analysis 54
checking system components 39
examining dump data 51
executing user-written routines 53
using data description operands 40
using DSPL3270 33, 121
using subcommands in CLISTs 53
using the storage map 48
using the symbol table 43
dump data sets 11, 28, 258
copying 101
dump directory 15, 258
data set 20
delete information 20
initialization 61
list information 20
dumps
processing 19
stand-alone 11, 12, 28

SYSMDUMP 12
SYS1.DUMPnn 28
titles 63

E

EDIT command, TSO
data set directory 254
problem directory 256
end an IPCS session 17
END subcommand 17, 127
ending an IPCS session 127
ENQ/DEQ component 39, 129
ENQCHECK subcommand 39, 129
ENTRIES keyword 67, 71
ENTRY keyword 67, 71
EQUATE subcommand 41, 42, 45, 46, 48, 54, 133
EVALUATE subcommand 54, 135
examine dump data 51
examples
ADDDSN subcommand 30
ADDPROB subcommand 25
ASCBEXIT subcommand 53
ASMCHECK subcommand 39
COMCHECK subcommand 39
COMPARE subcommand 53
DELDSN subcommand 30
DELPROB subcommand 25
DROPMAP subcommand 51
DROPSYM subcommand 47
DSPL3270 subcommand 33
ENQCHECK subcommand 39
EQUATE subcommand 41, 42, 43, 45, 46, 48, 49, 54
EVALUATE subcommand 54
FIND subcommand 42, 52
FINDMOD subcommand 52
FINDUCB subcommand 52
IOSCHECK subcommand 39
LIST subcommand 41, 42, 43, 46, 51
LISTDSN subcommand 31, 32
LISTMAP subcommand 51
LISTPROB subcommand 26
LISTSYM subcommand 47
MODDSN subcommand 31
MODPROB subcommand 26
NOTE subcommand 54
RUNCHAIN subcommand 53
SCAN subcommand 48, 49
SUMMARY subcommand 39
TCBEXIT subcommand 53
VERBEXIT subcommand 53
exit routines
ABEND 20
ASCBEXIT subcommand 53
TCBEXIT subcommand 53
VERBEXIT subcommand 53
explicit scrolling 38, 124

F

F keyword 66, 70
find
data in a dump 52, 137
module in dump 52, 143
UCB in dump 52, 149
FIND subcommand 52, 137
FINDAREA symbol 137
FINDMOD subcommand 52, 143
FINDUCB subcommand 52, 149
FMT (DSPL3270) 37
format data
data description operands 65
DSPL3270 subcommand 37
full screen display (DSPL3270) 34, 121

G

generate attention interrupts 20
generate messages 54, 203

H

HEADER keyword 66, 69
HELP subcommand 21, 155
HEXADECIMAL keyword (X) 66, 70

I

I/O supervisor component 39
IBM status values 182
identifiers 24, 79
implicit scrolling 36, 121
indirect address 68
indirect addressing 36, 121
initialize data sets 251
install IPCS 251
installation planning 251
interrupts, attention 20
invoke CLISTs
% sign 17
DSPL3270 subcommand 38
TSO subcommand 17
IOSCHECK subcommand 39, 157
IPCS command 17, 59
IPCS data sets 251
IPCS default values 221
IPCSDDIR command 61
IPCSDDIR ddname 15
IPCSINDD ddname 101
IPCSPRNT ddname 16
IPCSPRxx data set 59, 251

K

keyboard keys (DSPL3270) 121

L

LENGTH keyword 66, 68
length operands 66, 67
list
  data set attributes 31, 165
  dump data 51
  dump data sets 169
  dump directory information 20
  map entries 51, 171
  problems 26, 177
  symbols 46, 187
LIST subcommand 42, 46, 51, 161
LISTDSN subcommand 31, 165
LISTDUMP subcommand 20, 169
LISTMAP subcommand 51, 171
LISTPROB subcommand 26, 177
LISTSYM subcommand 46, 187
locate
  (see find)
logical compare of data items 53, 97

M

manage data sets 11, 23, 25
manage problems 23, 25
map
  (see storage map)
message generation 54
MODDSN subcommand 31, 191
modify attributes 26, 191, 195
modify authority 26
modify data set attributes 27, 31, 191
modify problems 195
MODPROB subcommand 26, 195
MODULE attribute 71
MODULE keyword 66, 71
MULTIPLE keyword 67, 71

N

naming conventions for symbols 44
NOREMARK keyword 67, 72
NOTE subcommand 54, 203
NOTEST keyword 19
  (see also TEST keyword)

O

operands,data description
  (see data description operands)
output for STATUS subcommand 231

P

PA (program attention) keys 125
parameters
  session 18
    defaults 18
PF (program function) keys 125
POINTER keyword (PTR) 66, 70
PRINT data set type 29
  ADDDSN subcommand 75
  MODDSN subcommand 191
PRINT keyword 15
print output data set 15, 258
problem
  adding 25
  attributes 24
  deleting 25
  identifiers 24, 79
  listing 26
  managing 11, 23
  modification 26
  modifying authority 11
  numbers 24
  owner 11, 24
problem determination, display status 229
problem directory 11, 256
process dumps 19
produce messages 54
program attention (PA) keys 125
program function (PF) keys 125
PTR keyword 66, 70

R

range (of addresses) 68
REAL keyword 66, 69
relative address 67
REMARK keyword 67, 72
remark keywords 67, 72
REPRO command, AMS
  problem directory 257
  data set directory 256
resolving data description operands 43
retrieve data 135
  for a CLIST 54
  for display 51
return codes

(see specific subcommands)
RUNCHAIN subcommand 53, 207

S

SCALAR keyword 67, 72
SCAN subcommand 48, 215
scratch data sets 30, 107, 111
scroll data 36, 123
  explicit 124
  implicit 124
SCROLL function 123
search
  (see find)
search control block chain 53, 207
security 14
seed records 255, 256
select addresses 36, 38
selection keywords 177
session control 15
  ABEND 20
  allocate dump directory 15
  allocate print output data set 15
  attention interrupt 20
  data set allocation 251
  defaults 18, 221
  delete dump directory information 20
  END 17
  HELP subcommand 21
  invoke CLISTs
    % sign 17
    DSPL3270 subcommand 38
  invoke TSO commands 17
    DSPL3270 subcommand 38
  IPCS command 59
  list dump directory information 20
  parameters 18, 251
  process dumps 19
  SETDEF subcommand 221
  start 17
  TEST and NOTEST 19
session parameters
  defaults 18
  member data set 251
  processing 253
  setting 59
set IPCS default values 221
SETDEF subcommand 18, 221
SIGNED keyword (F) 66, 70
size requirements
  (see space requirements)
skip (DSPL3270) 37
SKIP (DSPL3270) 38
space requirements
  dump directory 258
  problem directory 256
specifying data description operands 42
stack addresses 38
stack addresses (DSPL3270) 36
stand-alone dumps 12, 28
start an IPCS session 17
STATUS keyword 66, 69
STATUS subcommand 229
storage map 48
STRUCTURE attribute 71
STRUCTURE keyword 66, 71
structures
  (see control blocks)
subcommands
  getting information about 21
  list of 13
summarize control block fields 39, 235
SUMMARY subcommand 39, 235
symbol table 43
  create symbol 45, 46, 54, 133
  delete symbol 45, 47, 119
  list symbol 45, 46, 187
  naming conventions 44
  naming symbols 207
  validate symbol 55
symbol X 67
symbolic address 67
SYSDSCAN command 63
SYSMDUMP 12
system status messages 232
SYS1.DUMPnn data sets 28, 63
SYS1.PARMLIB 59, 251

T

target data set, for copying 101
task control block 39, 235
TCB exit routine 243
TCBEXIT subcommand 53, 243
terminal monitor program 18
  background 21
terminate
  DSPL3270 39
  IPCS, ABEND 20
terminate an IPCS session 17
termination, abnormal 20
TEST keyword 19
TIME command 20
TSO commands
  for IPCS 57
  list of 13
  use of 17, 247, 259
TSO subcommand 247, 259

U

UDEF data set type 30
UNSIGNED keyword 66, 70
user exits 53, 249
user-defined data set type 30

user-written routines 53, 249

V

validate command names 18, 259
validate control blocks
(see validate structures)
validate structures 49
list of 50
validate system data areas 215
validity check exit routine 259
verb exit routine 249
VERBEXIT subcommand 53, 249
VSAM cluster 15

X

X keyword 66
X symbol 40, 41, 43, 67
(see also specific subcommand)

OS/VS2 MVS Interactive
Problem Control System:
User's Guide and Reference GC34-2006-1

**READER'S COMMENT FORM**

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. It will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

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

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

**Cut or Fold Along Line**

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC34-2006-1

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department Z59, Building 931
P. O. Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along Line

OS/VS2 MVS Interactive Problem Control System: User's Guide and Reference (File No. S370-37)

Printed in U.S.A.

GC34-2006-1

GC34-2006-1

OS/VS2 MVS Interactive Problem Control System: User's Guide and Reference (File No. S370-37)

Printed in U.S.A.

GC34-2006-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601