IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Serviceability Aids Diagnosis Reference

Version 6 Release 1

SC33-6327-00

IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Serviceability Aids Diagnosis Reference

Version 6 Release 1

SC33-6327-00

Note!

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

First Edition (April 1995)

This edition applies to Version 6 Release 1 of VSE/Advanced Functions, which is part of VSE Central Functions, Program Number 5686-066, and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

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

Contents

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Deutschland Informationssysteme GmbH
Department 0215
Pascal Str. 100
70569 Stuttgart
Germany
```

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Programming Interface Information

This publication is intended to help the customer to do diagnosis of VSE/ESA. This publication documents information that is Diagnosis, Modification, or Tuning Information provided by VSE/ESA.

Warning: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in certain countries:

```
IBM
System/370
VSE/ESA
VTAM
```

Preface

This publication is intended primarily for use by IBM personnel responsible for program service. The publication gives an overview of the functions of the serviceability aids, of their design, and their organization; it supplements the program listings.

IBM serviceability aid programs supplement various supervisor and transient functions. The programs provide the means to trace specific events during execution of a program, assist in software debugging, generate dumps, and list error statistics.

Related Publications: Functional descriptions of the serviceability programs in this manual, as well as some not described here, are included in the following publication:

VSE/ESA Diagnosis Tools, SC33-6614.

The reader should also be familiar with:

IBM Enterprise Systems Architecture /370-Principles of Operation, SA22-7085
IBM System /370 Principles of Operation, GA22-7000
VSE/ESA Guide to System Functions, SC33-6611
VSE/ESA Installation, SC33-6604
VSE/ESA System Control Statements, SC33-6613
VSE/ESA Guide for Solving Problems, SC33-6610
VSE/ESA Messages and Codes, SC33-6607

For complete information on VSE/Advanced Functions program logic, consult the following *VSE/Advanced Functions Diagnosis Reference* publications:

Supervisor, SC33-6323.
Error Recovery, SC33-6326
Logical Transients, SC33-6324.
Initial Program Load and Job Control, SC33-6325.
Linkage Editor, SC33-6328.
Librarian, SC33-6330.

Organization of the Manual: The manual consists of the following sections:

- Part 1: SDAID (System Debugging Aid)
- Part 2: Dump and Trace Programs
- Part 3: Dump Utility Program DOSVSDMP
- Part 4: Info/Analysis
- Part 5: Parser (IJPARSER)
- Part 6: LSERV (Label Information Area Display)

Chapter 1. SDAID: Introduction

The SDAID program provides for VSE users, including IBM programming support, the tracing facilities needed for efficient problem analysis. The program allows the user to set up ten different traces per SDAID session (execution) with a variety of trace output being produced.

The following types of events may be traced:

BRanch successfully executed branch instructions

CAncel program (main task) cancel or EOJ

EXTernal external interruptions

INSTruction instruction(s) execution

IO I/O interruptions

MONitorcall MC instruction execution

PGMCheck program check interruptions

PGMLoad phase load request or actual load

SSCH SSCH instruction execution

STorage storage alterations

SVC Supervisor calls

VTAMBU buffer usage in VTAM

VTAMIO VTAM I/O related events

The trace output of SDAID may be written to a line printer, to a tape device, or into a wraparound buffer. If trace data is written into a wraparound buffer then the user may specify an output device (printer or tape) to give out the contents of the wraparound buffer on explicite request.

Any SDAID session consists of two major steps: The first step is the SDAID initialization (SDAID set-up) where the requested traces are prepared. The SDAID initialization is controlled by the commands SDAID, OUTDEV, TRACE, and READY. The second step is the SDAID execution (event tracing) where trace data is collected. The SDAID execution is controlled by the commands STARTSD (formerly STRTSD), STOPSD, and ENDSD.

The following overview describes the function of the SDAID commands:

SDAID invokes the SDAID program. It must be the first command submitted for an SDAID session.

OUTDEV defines the device on which SDAID is to record trace information. The command is mandatory; it must precede the READY command.

TRACE defines the program events whose occurrences are to be traced; the command must precede the READY command.

READY signals to SDAID that all control input from the console is complete. The READY command must be preceded by an OUTDEV command and at least one TRACE command.

STARTSD requests SDAID to start executing the requested trace operation(s).
SDAID still accepts the former command name STRTSD

STOPSD requests SDAID to stop executing the requested trace operation(s).

ENDSD terminates the current SDAID session by freeing all resources that were used by the SDAID program. ENDSD may be given at any time after an SDAID command has been issued. The ENDSD command may be used to flush the setup process or to terminate the SDAID execution.

For more information about the SDAID commands, see *VSE/ESA Diagnosis Tools.*

Chapter 2. SDAID: Program Design and Organization Information

Initialization

The SDAID setup process may be invoked from a user partition (setup via SYSIN) or via the attention supervisor task (setup via attention commands). If the setup via SYSIN is used, the phase SDAID is called (directly or via a JCL procedure) and that phase submits the commands SDAID, OUTDEV, TRACE, and READY to the SDAID adapter phase $$BATTN3. If the SDAID setup is done via the attention task, then the attention commands SDAID, OUTDEV, TRACE, and READY are passed from the LTA phase $$BATTNA to the SDAID adapter phase $$BATTN3.

$$BATTN3 loads the SDAID message library IJSDMSG and the initialization manager IJSDROT into the system GETVIS area if they are not already there.

$$BATTN3 now invokes the initialization manager IJSDROT. This phase determines the entered command, checks for proper command sequence, loads all needed phases and tables into the system GETVIS area and invokes the parser, phase IJPARSER, for command syntax checking.

When the READY command is issued, then the initialization manager IJSDROT determines which execution phases are to be loaded to satisfy all specified trace functions. SDAID loads the execution phases into a V=R area at the end of the supervisor. The default size of this area is 64K bytes. The size of that reserved SDAID area may be increased during IPL using the SDSIZE parameter of the SYS command.

After the STARTSD command is entered, the new PSWs and the control registers are modified so that SDAID is enabled to trace the specified events.

Activation of Partition Traces

A TRACE statement with AREA=syslog-id or JOBNAME=job-name generates a "dormant" trace control block (TCB). SDAID activates and deactivates this TCB dynamically at trace execution time. The TRACE command processor IJSDTRA generates a dormant trace control block for any trace of type "jobname" and for any trace of type "syslog-id". These TCBs contain the user specified parameters, like job name, job number, syslog-id, but they do not contain a partition key nor partition boundaries. If the user specified a Phase as tracing range, IJSDTRA issues a CDLOAD macro to find out, whether the specified phase is contained in the SVA. If the specified phase is already in the SVA, then IJSDTRA moves the phase begin and end address as trace boundaries into the trace control block. If the CDLOAD macro returns a non-zero return code, IJSDTRA generates an implicite PGML trace control block to detect the phase as soon as it is loaded into storage.

At STARTSD time the module IJSDROT activates any dormant trace control block if the pertinent partition or job is already active. IJSDROT scans the queue of trace control blocks. For any TCB of type "jobname" IJSDROT scans the table of partition control blocks for a matching job name. If the jobname in the PCB matches the jobname in the TCB then IJSDROT activates the dormant TCB. IJSDROT fills the PIK and the trace boundaries into the TCB. For any TCB of type "syslog-id" IJSDROT issues a GETFLD macro to retrieve the partition information key (PIK). If GETFLD returns a zero return code, then IJSDROT activates the dormant TCB. IJSDROT fills the PIK and the trace boundaries into the TCB. If at least one trace control block of type "JOBNAME" or "syslog-id" is specified, IJSDROT moves the address of IJSDPPR into the system communications region (SYSCOM).

At STOPSD time IJSDROT scans the queue of trace control blocks and deactivates any TCB of type "jobname" and any TCB of type "syslog-id". IJSDROT resets the address pointer to IJSDPPR in the SYSCOM to zero.

The module IJSDPPR activates and deactivates trace control blocks dynamically at trace execution time. The supervisor calls IJSDPPR during partition preparation and during partition clean-up if the pointer to IJSDPPR in the SYSCOM is different from zero. The supervisor calls IJSDPPR on normal or abnormal termination of a POWER job.

At partition preparation time IJSDPPR scans the queue of trace control blocks for blocks of type "jobname" or of type "syslog-id". If the job name in a dormant TCB matches the job name in the PCB, IJSDPPR activates the TCB and fills the PIK and the trace boundaries into the TCB. If the syslog-id in a dormant TCB matches the syslog-id of a scheduled dynamic partition, IJSDPPR activates the TCB and fills the PIK and the trace boundaries into the TCB.

At partition cleanup time IJSDPPR scans the queue of trace control blocks for blocks of type "jobname" or of type "syslog-id". SDAID deactivates the trace control block if the PIK of the terminating partition matches the PIK of the trace control block.

Trace Output on Printer or Tape

Two modules are used for writing trace data: IJSDXWRP writes trace data to a printer device, IJSDXWRT writes trace data to a tape device.

When SDAID writes trace data to printer or tape, then it enables I/O interruptions for its printer or tape device. The smallest group of devices which can be enabled is an interrupt sub class. VSE/ESA reserves the subclass 7 for the SDAID output device. This extra subclass allows SDAID to perform its output operations without intervening interruptions from other devices. Before SDAID performs an output operation, it disables I/O interruptions for all classes except the SDAID subclass. SDAID issues a MODIFY SUBCHANNEL instruction to assign the SDAID output device to subclass 7. Then it changes the contents of control register 6 to allow only interruptions of class 7. After the SDAID output operation is complete, SDAID resets control register 6 to its previous value.

If the SDAID output device is a printer device and the same printer is used by VSE for non-SDAID output operations, then the SDAID print operations may interfere with a previous non-SDAID print operation. Before SDAID prints a line on the OUTDEV printer it checks via a store subchannel instruction whether the last non-SDAID print operation is complete. If the last non-SDAID operation has not completed, SDAID clears the device by a TEST SUBCHANNEL (TSCH) operation. If the IRB shows that a unit check has occured, then SDAID retrieves the SENSE data, too. IJSDXWRP moves the interrupt information and the sense data (if available) into an interrupt stack. IJSDINT passes this interrupt information at a later time to the I/O supervisor. The interrupt stack in IJSDXWRP contains one or two stack entries as a maximum. If the SDAID output device is a tape device then there is no interrupt stacking.

Supervisor Interface

- SSCH Tracing

  The supervisor issues a monitor call before it starts an SSCH instruction.

  MC $TRSIO1,$MCSDAID

  At the time of the M.C. the registers contain the following values: R1 = CCB address, R2 = cuu, R3 = PUB address. The CAW in low core contains the address of the CCW chain.

  The supervisor issues a monitor call after it has started an SSCH instruction.

  MC $TRSIO2,$MCSDAID

  This monitor call is used to pass the condition code of the SSCH to SDAID

- I/O Interrupt Tracing

  SDAID does not modify the I/O interrupt new PSW for interrupt tracing. The supervisor processes the interruption and performs a task switch to the task which had requested the I/O operation. Then the supervisor transfers control to SDAID via a monitor call.

  MC $TRIO,$MCSDAID

- I/O Simulation

  When the I/O supervisor is ready to accept a simulated I/O interrupt from SDAID, then the supervisor issues a monitor call.

  MC $MCIOS,$MCSDAID

  SDAID moves the I/O interruption code to real locations 184-191 (X'B8'-X'BF') and loads the I/O interrupt new PSW.

- TSCH Hook

  Before the I/O supervisor issues a TSCH instruction it issues a monitor call followed by a TSCH instruction.

  MC $MCIOS1,$MCSDAID
  TSCH

  SDAID compares the I/O interruption code with the code of the SDAID output device. If the codes match and there are data in the SDAID I/O interrupt stack, SDAID moves the IRB and the SENSE data (if present) to low core and returns control to the interrupted location + 4 (_ after the TSCH instruction).

SDAID Initialization Phases

IJSDAID - SYSIN COMMAND PROCESSOR

Function:
Prepares complete SDAID commands and transfers them to
the SDAID adapter phase $$BATTN3.

Phase Name:
SDAID

Module Name:
IJSDAID

Entry point:
IJSDAID

Input:
OUTDEV and TRACE commands from SYSIN or SYSLOG and
procedure parameters from the SVA.

Output:
SDAID commands: SDAID, OUTDEV, TRACE,
READY, ENDSD.
These commands are submitted to $$BATTN3.
For information they are displayed on SYSLOG, too.

Phases called: $$BATTN3
Parameters passed to $$BATTN3:
Register 0 contains the address of an
interface area, called BATTNA.

Macros used:
DTFCN, DTFDI, STXIT

Sequence of Operation
Reads input parameters from SYSIPT or SYSLOG.
If substitution of parameters is required
retrieves SVA parameters via PARMMAC macro.
Substitutes missing parameters
by default values.
Passes SDAID commands to $$BATTN3.
Prepares an OUTDEV command (if required).
Writes information messages to SYSLOG or SYSLST.
On errors issues ENDSD command to
flush the setup process.
Returns to JCL via RETURN macro.

Internal Subroutines

SDOUTD:
Prepares the OUTDEV command
retrieves PRINTER, TAPE, BUFFER parameter.

SDBUTERM:
Processes BUFFOUT parameter.

SDCONS:
Displays SDAID commands
Passes commands to $$BATTN3.

SDENDSD:
Prepares ENDSD command.

SDTRAREA:
Retrieves AREA or JOBNAME parameter.

GETVAL:
Retrieves parameters from SVA (via PARMMAC).

BUFFOUT:
Moves parameter to output buffer.

SDINPUT:
Reads input from SYSLOG or SYSIPT.
Scans input area.
Substitutes placeholders by retrieved
parameters or by default values.

Information Messages:

IJSDAID prepares the messages 4C40I to 4C61I and displays them on SYSLOG via WTO

Normal exit:

to JCL via the RETURN macro.

Error exit:

to JCL via the RETURN macro.

$$BATTN3 - SDAID TO SYSTEM ADAPTER

Function:

This phase has 2 functions:
1. Calls the phase IJSDROT if an SDAID command is entered.
2. Performs I/O operations for SYSLOG and SYSLST.

Entry point:

This phase has 2 entry points:
1. $$BATTN3: entry to process an SDAID command.
2. IORTN: entry to perform the I/O operations.

Input:

For function 1:
The address of the interface area BATTNA is contained
in register 15 (if $$BATTN3 is called from $$BATTNA)
or in register 0 (if $$BATTN3 is called from IJSDAID).
BATTNA contains the SDAID command and flag bytes.

For function 2:
An indicator for the requested function
(READ/WRITE), the address of the I/O area.

Output:

For function 2: The I/O operation required.

Normal exit:

For function 1: returns to $$BATTNA or IJSDAID.
For function 2: returns to calling phase.

Error exit:

Same as normal exit.

Subroutines

SDLOAD: Loads SDAID phases IJSDMSG and IJSDROT.
ENDSD: Releases phases IJSDMSG and IJSDROT.
IORTN: Performs I/O for SYSLST and SYSLOG.

Flow of Control

Gets supervisor state and key zero.
Loads phases IJSDMSG and IJSDROT.
Calls IJSDROT to process SDAID command.
Releases supervisor state and key zero.
Returns to caller.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDROT - SDAID INITIALIZATION MANAGER

Function:

The SDAID initialization manager loads all data areas,
control blocks, and phases needed to process an SDAID command.
IJSDROT processes the SDAID commands,
builds the executable SDAID trace program, and
starts|stops the trace execution.

Entry point:

Same as module name.

Input:

Command name and specified operands.

Output:

Updated information in control blocks.

Normal exit:

Returns to phase $$BATTN3.

Error exit:

Same as normal exit.

Error Messages:

4C01I SDAID ERROR
4C04I SDAID SETUP IN PROGRESS BY ANOTHER TASK
4C05I PROCESSING OF cmd SUCCESSFUL|FAILED
4C06I cmd OUT OF SEQUENCE
4C07I function FOR PHASE FAILED
4C14I BUFFER SIZE HAS BEEN REDUCED
4C26I SDAID AREA IS TOO SMALL
4C28I THE INTERACTIVE TRACE PROGRAM IS ACTIVE
4C29I LOCK MACRO FAILED
4C36I SDAID SETS OFF THE PSEUDO PAGE FAULT PORTION
4C37I SDAID SETS ON THE PSEUDO PAGE FAULT PORTION
4C48I AN UNKNOWN TRACE PROGRAM IS ACTIVE

Printer Messages:

START OF SDAID TRACE (printer headline)
END OF SDAID OUTPUT (final line on printer)

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDROT:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

IJSDROT                                Main entry for all SDAID commands
  |
  |
  |-->ROTOUTD                          Processes the OUTDEV command.
  |
  |     |
  |     |-------------------->IJSDCT2  Sets up OUTDEV cmd directory (17).
  |     |-------------------->IJSDPIF  Checks command syntax (24).
  |
  |-->ROTTRA                           Processes the TRACE command.
  |
  |     |
  |     |-------------------->IJSDCT3  Sets up TRACE command directory (18).
  |     |
  |     |-------------------->IJSDPIF  Checks command syntax (24).
  |
  |-->ROTRDY                           Processes the READY command.
  |
  |     |
  |     |-->ROTFRAL                    Releases initialization phases.
  |     |
  |     |-->ROTGETS                    Computes size of all needed SDAID phases.
  |     |
  |     |-->ROTGETR                    Gets real storage from page pool.
  |     |
  |     |-->ROTGETV                    Gets storage from SVA for SDAID phases.
  |     |
  |     |-->ROTLOAD                    Loads SDAID phases into SVA.
  |     |-------------------->IJSDINT  Establishes new PSWs in EDTCB (27).
  |     +-------------------->IJSDRDY  Processes the READY command (26).
  |
  |-->ROTSDAID                         Processes the SDAID command.
  |
  |     |
  |     |-------------------->IJSDEDT  Allocates execution ctl blocks (14).
  |     |-------------------->IJSDIDT  Builds initialization table (13).
  |     |-------------------->IJSDTCB  Manages TCB queue (16).
  |     |-->ROTGETS                    Loads the PARSER program.
  |     |-->ROTGETS                    Loads IDSDPIF.
  |     |-->ROTGETS                    Loads IJSDTCB.
```

```
|
|-->ROTSTOP                       Processes the STOPSD command.
|                                 Restores all control registers and
|                                 NEWPSWS, enables Interrupts.
|
|-->ROTSTRT                       Processes the STARTSD command.
|                                 Sets up control registers and PSWs.
|
|-->ROTENDSD                      Processes the ENDSD command.
|
|-->ROTSTOP                       Processes the STOPSD command
|                                 if not yet done.
|
|-->ROTFREV                       Releases SDAID storage in SVA.
|
+-->ROTFRAL                       Releases initialization phases.
```

IJSDIDT - SDAID INITIALIZATION DATA TABLE

Function:

This phase allocates all data areas needed during SDAID initialization. The data areas are cleared and anchored in the IDTCB.

Input:

Parameters: none.

Output:

Formatted Initialization Data Table (IDTCB).

Normal exit:

Returns to calling phase.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDEDT - SDAID EXECUTION DATA TABLE

Function:

This phase allocates all data areas needed during SDAID execution. The data areas are cleared and anchored in the EDTCB.

Entry point:

Same as module name.

Input:

None.

Output:

Initialized control blocks.

Normal exit:

Returns to the calling module IJSDROT.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDMSG - MESSAGE LIBRARY AND MESSAGE WRITER

Function:

The SDAID message library contains all SDAID messages. It also contains the message manager, which prepares the messages and writes them to the system console.

Entry point:

Same as module name.

Input:

A parameter list is submitted in register 1. This list contains the message number to be processed, possibly the address and length of text to be inserted in predefined slots, and an identifier for the end of the list.

Output:

The requested message written on SYSLOG.

Exit:

Returns to the calling routine.

Flow of Control:

Search the message library, call subroutine MSGUPD to insert text in message slots, display the message on master console via WTO.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDTCB - TRACE CONTROL BLOCK MANAGER

Function:

The SDAID trace control block manager has the following functions:

- Initializes trace table (TRTCB), trace control block queue (TCBCBQ), and trace control block extension queue (TCECBQ).
- Enqueues a TCB in its appropriate chain.
- Appends a TCB to a master TCB and enqueues it in its appropriate TCB chain.
- Appends a TCE to a master TCB and enqueues it in its appropriate TCE chain.

Entry point:

Same as module name.

Input:

Parameters: interface established in EDTCB.

Output:

Updated trace table (TRTCB) and trace control block queue.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDTCB:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

IJSDTCB                                Manages TCB queue (16).
  |
  |
  |-->TCBAPP                           Appends a TCB.
  |
  |-->TCBENQ                           Enqueues a TCB.
  |
  |     |
  |     |--------------------->IJSDMSG  Writes an error message (15).
  |     +-->TCBAPP                     Appends a TCB.
  |
  +-->TCBINI                           Initializes TCB-QUEUE.
```

IJSDCT2 - OUTDEV COMMAND TABLE

Function:

The SDAID OUTDEV command table contains the definitions of nodes and parameters for the OUTDEV command. (See "Parser" chapter). It also contains a routine to establish the command directory in the initialization data table (IDTCB).

Entry point:

Same as module name.

Input:

IDTPTR contains the address of the initialization data table IDTCB.

Output:

The command directory established in IDTCB.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDCT3 - TRACE COMMAND TABLE

Function:

The SDAID TRACE command table contains the definitions of the nodes and parameters for the TRACE command. (See "Parser" chapter). It also contains a routine to establish the command directory in the initialization data table (IDTCB).

Entry point:

Same as module name.

Input:

IDTPTR contains the address of the initialization data table IDTCB.

Output:

The command directory established in IDTCB.

Tables used:

The PARSER control table for the TRACE command. The PARSER control table is described on the following two pages.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Parser Control Table for the TRACE Command

```
S01:  TRACE
             -+--SVC= ------       S02 -----  S14
              +--PGMC= -----       S03 -----  S14
              +--INST= -----       S04 -----  S14
              +--BRANCH ----       S14
              +--STOR ------       S07 -----  S14
              +--MON= ------       S08 -----  S14
              +--IO --------       S14
              +--EXT -------       S09 -----  S30
              +--VTAMBU ----       S30
              +--VTAMIO ----       S16 -----  S25
              +--CANCEL ----       S14
              +--BUFFER ----       S37
              +--PGMLOAD ---       S10 -----  S11 -----S14
              +--SSCH ------       S14


S12:  JOBNUM=-+--99999 ------      S17


S14:         -+-- AREA=  ----      S16
             -+-- JOBNAME=---      S15

S15:  JOBNAME=+--AAAAAAAA----      S12 -----  S17

S16:  AREA= --+-- BG|Fn|W2 --      S17
              +-- SUP -------      S17
              +-- ALL -------      S17


S17:        --+-- OFFSET=----      S21 -----  S30
              +-- ADDRESS=---      S21 -----  S30
              +-- PHASE=-----      S18
              +-- LTA   -----      S30


S18:        --+-- PHASE-NAME-      S19


S19:        --+-- OFFSET= ---      S21 -----  S30


S25:      ----+--ALL -----------   S30
              +--CHANNEL= ------   S26 ------ S30
              +--CU= -----------   S27 -----  S30
              +--UNIT= ---------   S28 -----  S30
```

```
S30: OUTPUT=--+-GREG---   -------  S37
              +-FREG---+  -------  S37
              +-CREG---+  -------  S37
              +-COMREG-+  -------  S37
              +-SYSCOM-+  -------  S37
              +-IOTAB--+  -------  S37
              +-LTA----+  -------  S37
              +-PTA----+  -------  S37
              +-CCB----+  -------  S37
              +-CCW----+  -------  S37
              +-TOD----+  -------  S37
              +-LOWCORE+  -------  S37
              +-SUP ---+  -------  S37
              +-CCWD=--+  -------  S29
              +-PTAB---+  -------  S37
              +-TTAB---+  -------  S37
              +-DUMP---+  -------  S31
              +-BUFFER-+  -------  S37


S31:  DUMP=   +--PARTition-----    S33  ------S35 ------S37
              +- PHASE-+  -------  S33  ------S35 ------S37
              +--ADDRESS= -----    S35  ------S37
              +--REGISTER= ----    S36  ------S37
              +--PTR= -+  -----    S34. ------S24 ------S37


S37: OPTION=  +- OCC=--+  -------  S38  ------SEND
             -+--HALT--+  -----    SEND
             -+--TERMINATE-----    SEND
             -+--NOJCL-+  -----    SEND
             -+--SUP --+  -----    SEND

S39:          +- STAREA  ------    S40
             -+--STJNAM   -----    S41
             -+--STDSPN   -----    S43

S40: STAREA=  +- SYSLOG-id   --    S17

S41: STJNAM=  +- xxxxx   ------    S42

S42: STJNUM=  +- xxxxx   ------    S17

S43: STDSPN=  +- xxxxx   ------    S17

SEND:  END OF PROCESSING
```

IJSDOUT - SDAID OUTDEV COMMAND PROCESSOR

Function:

The SDAID OUTDEV command processor checks the syntax and the semantics for the OUTDEV command.
It saves the specified parameters in the execution data table.
It reserves the output tape for exclusive usage by SDAID,
checks for the file protect ring,
and whether the tape is on the load point.

Entry point:

Same as module name.

Input:

CPCB: IJPARSER control block containing the SIF for the
OUTDEV command.

Output:

Updated control block EDTCB according the specified parameters
of the OUTDEV command.

Normal exit:

Returns to calling routine.

Error exit:

Same as exit normal.

Error Messages:

4C09I The specified tape is in use by another partition
4C10I Specified address cuu is invalid for printer/tape
4C11I The OUTDEV tape is file protected
4C12I Buffer size is greater than 32k bytes
4C13I No GETVIS space is available
4C15I No free programmer logical unit is available
4C27I The OUTDEV tape is not on load point

Macros used:

ASSGN	Assignes a logical unit to OUTDEV
EXTRACT	Retrieves the OUTDEV PUB entry
GETFLD	Retrieves PUB table address
GETFLD	Retrieves tape owner
GETFLD	Retrieves device properties
MSAT	Reserves/Releases tape unit
SRCHFLD	Retrieves the OUTDEV PUB index

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDOUT

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

IJSDOUT                                Processes the OUTDEV command (21).
  |
  |
  |-->SDOUT01                          Processes primary output device.
  |
  |-->SDOUT02                          Processes size of wrap around buffer.
  |
  |-->SDOUT03                          Processes secondary output device.
  |
  |-->SDOUT04A                         find OUTDEV cuu
  |
  |-->SDOUT04C                         assign tape unit
  |
  |-->SDOUT04D                         get subchannel id
```

IJSDTRA - SDAID TRACE COMMAND PROCESSOR

Function:

The SDAID TRACE command processor builds for each trace command a TRACE control block (TCBCB) and saves all specified parameters and values in it.

Entry point:

Same as module name.

Input:

CPCB: IJPARSER control block containing the SIF for the TRACE command.

Output:

Updated control blocks EDTCB and TCBCB according the specified parameters of the TRACE command.

Normal exit:

Returns to the calling routine.

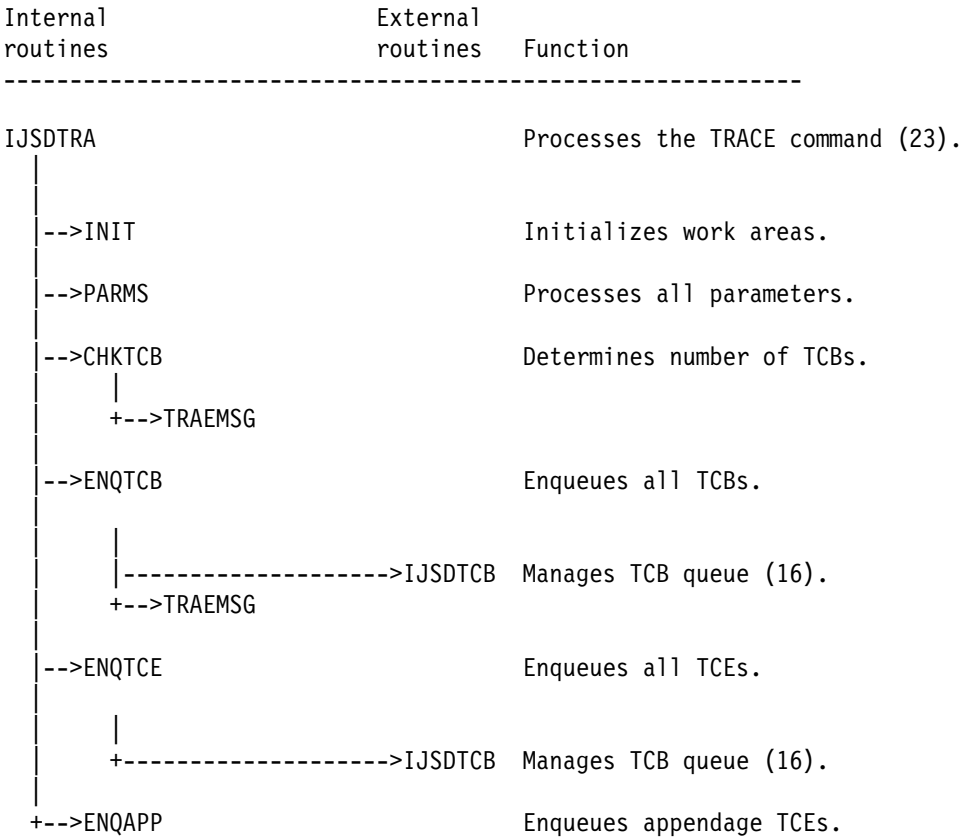
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDTRA



IJSDPIF - SDAID-TO-IJPARSER INTERFACE

Function:

This phase is the only communication medium from the PARSER to the SDAID program.

Entry points:

IJSDPIF: called by IJSDROT to transfer control to the PARSER for syntax checking.

PIFCHK: called by the PARSER to transfer control to the processing routine for the actual command.

PIFIOF: called by the PARSER for I/O processing.

The entry points PIFCHK and PIFIOF are set in the Parser Control Block CPCB and only used by the PARSER.

Input:

For entry PIFCHK: the address of the SIF entry for which a semantic processing is to be done.

For entry PIFIOF: for write request- the text to be written in IJPARSER I/O area in IDTCB.

Normal exit:

For entry IJSDPIF: as provided by the PLS/II return statement.

For entry PIFCHK: to IJPARSER.

For entry PIFIOF: to IJPARSER.

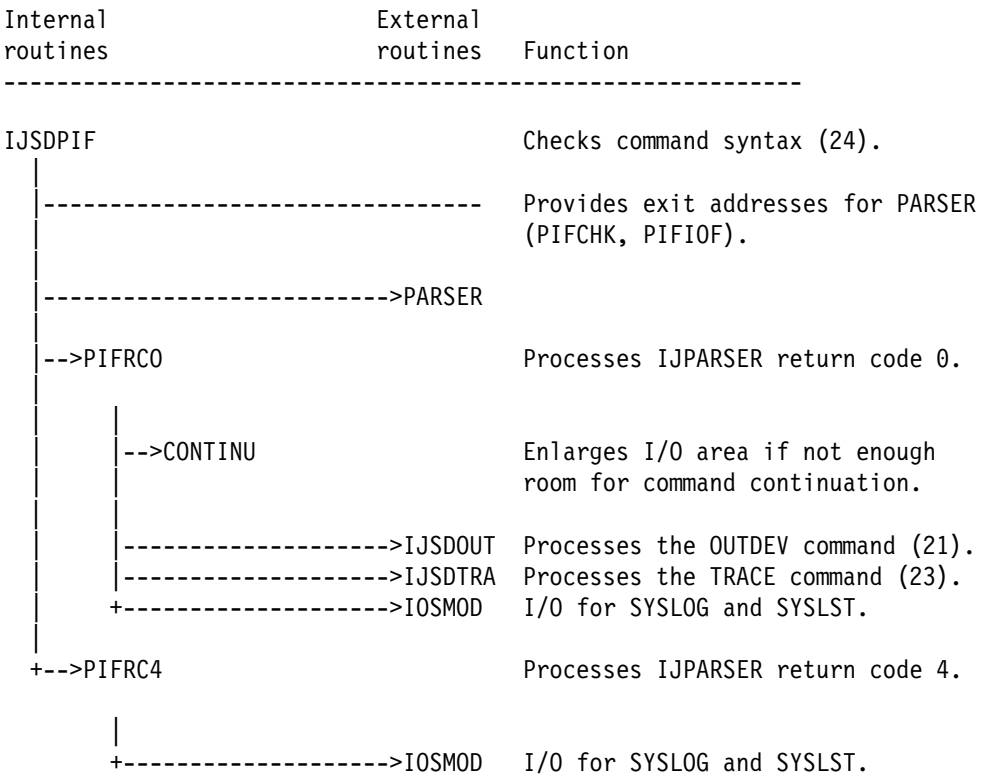
Error exit:

Same as exit normal.

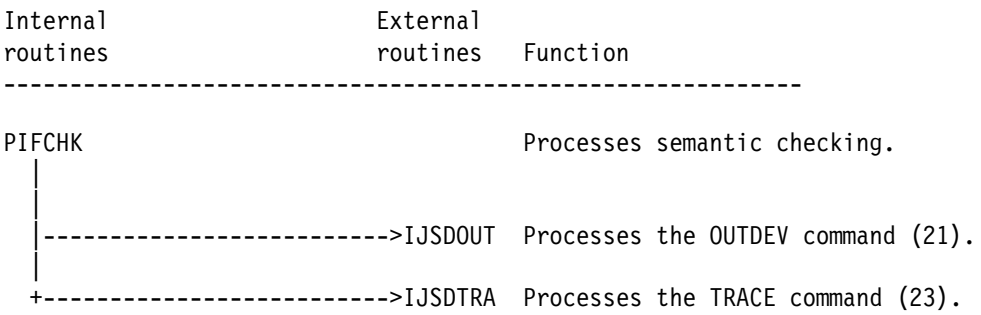
External references:

See "Diagnostic Aids" section of this chapter.

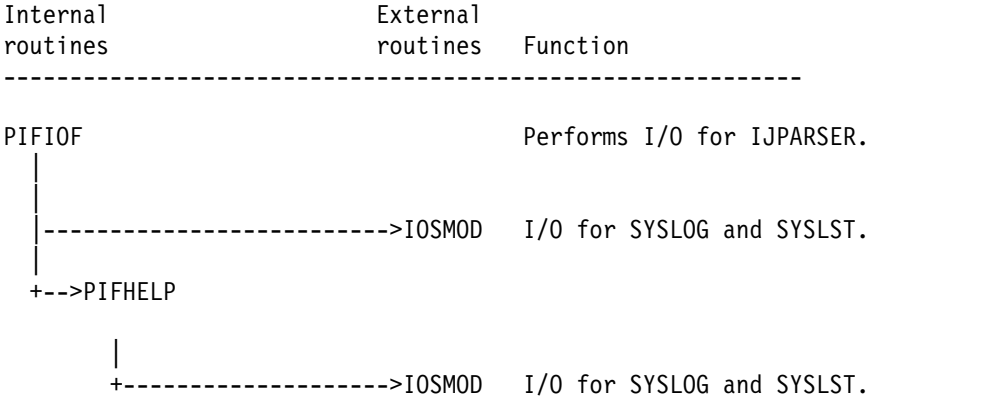
Flow of Control for Entry Point: IJSDPIF:



Flow of Control for Entry Point: PIFCHK



Flow of Control for Entry Point: PIFIOF:



IJSDRDY - READY COMMAND PROCESSOR

Function:
Scans the Trace Control Block queues (TCB queues).
and sets the required flags into the EDTCB.

Entry point:
Same as module name.

Input:
None.

Output:
Updated information in control blocks EDTCB and GDTCB.

Messages issued:
4C16, 4c17, 4c38

Normal exit:
Returns to the calling routine IJSDROT.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

Event Recording Routines

IJSDINT - INTERRUPT PROCESSOR

Function:
The SDAID interrupt processor controls the processing of those program events which are specified to be traced. It invokes the pertinent event processor in order to generate the trace output.

Entry points:
IJSDINT: Establishes interrupt addresses
(called from IJSDROT).
EXTENT: entry for EXT interrupts
PGMENT: entry for PGM interrupts
SVCENT: entry for SVC interrupts

Input:
The interrupt information in fixed storage locations.

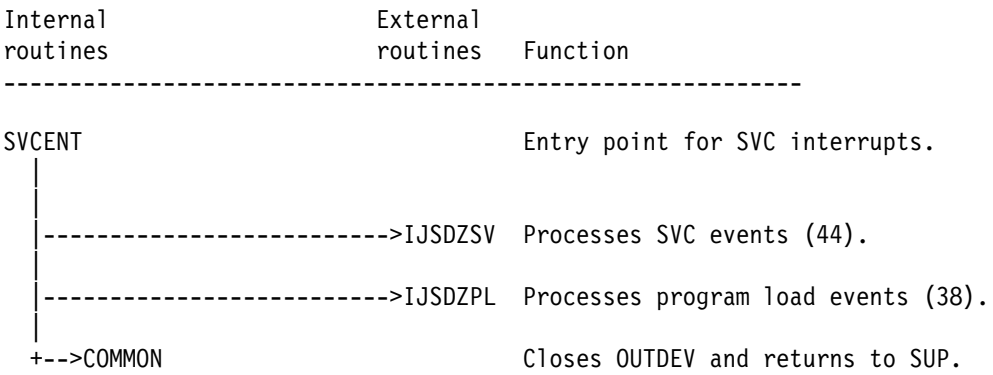
Output:
It is done by the invoked procedures.

Normal exit:
Returns to the interrupted program via LPSW-instruction.

Error exit:
Not applicable.

External references:
See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: SVCENT:



Flow of Control for Entry Point: EXTENT:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

EXTENT                                 Entry point for EXT interrupts.
  |
  |
  |-------------------------->IJSDZEX  Processes EXTERNAL interrupts (33).
  |
  |
  +-->COMMON                           Closes OUTDEV and returns to SUP.
```

Flow of Control for Entry Point: PGMENT:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

PGMENT                                 Entry point for PGM interrupts.
  |
  |
  |-------------------------->IJSDZBR  Generates Branch trace record (30).
  |
  |-------------------------->IJSDZBU  Proc. BUFFER OVERFLOW events (31).
  |
  |-------------------------->IJSDZCA  Processes CANCEL events (32).
  |
  |-------------------------->IJSDZIO  Proc. I/O interruptions     (35).
  |
  |-------------------------->IJSDZIN  Proc. instruction fetch events (34).
  |
  |-------------------------->IJSDZMC  Processes monitor call events (36).
  |
  |-------------------------->IJSDZPC  Processes program check event (37).
  |
  |-------------------------->IJSDZPL  Processes program load events (38).
  |
  |-------------------------->IJSDZSA  Proc. storage alter events (40).
  |
  |-------------------------->IJSDZSI  Processes SSCH events (42).
  |
  |-------------------------->IJSDZVT  Processes VTAMIO events (46).
  |
  +-->COMMON                           Closes OUTDEV and returns to SUP.
```

IJSDSTP - SDAID STOP-ON-EVENT PROCESSOR

Function:

Stops the system after an event has occurred and is processed by SDAID.
The system is put into a wait state with '00EEEE' is the address part of the wait PSW. To go out of the wait state, the operator must press the external interrupt key.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table EDTCB.

Output:

None.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDZBR - SDAID BRANCH TRACE PROCESSOR

Function:
The SDAID branch trace processor traces all or selected successfully executed BRANCH instructions in a specified program area.
It generates a branch trace record and writes it on the specified output device.

Entry point:
Same as module name.

Input:
EDTPTR contains the address of the execution data table EDTCB.
TRTPTR contains the address of the trace table (TRT) entry for branch traces.
TCBPTR contains the address of the first block in the TCB-queue for branch traces.

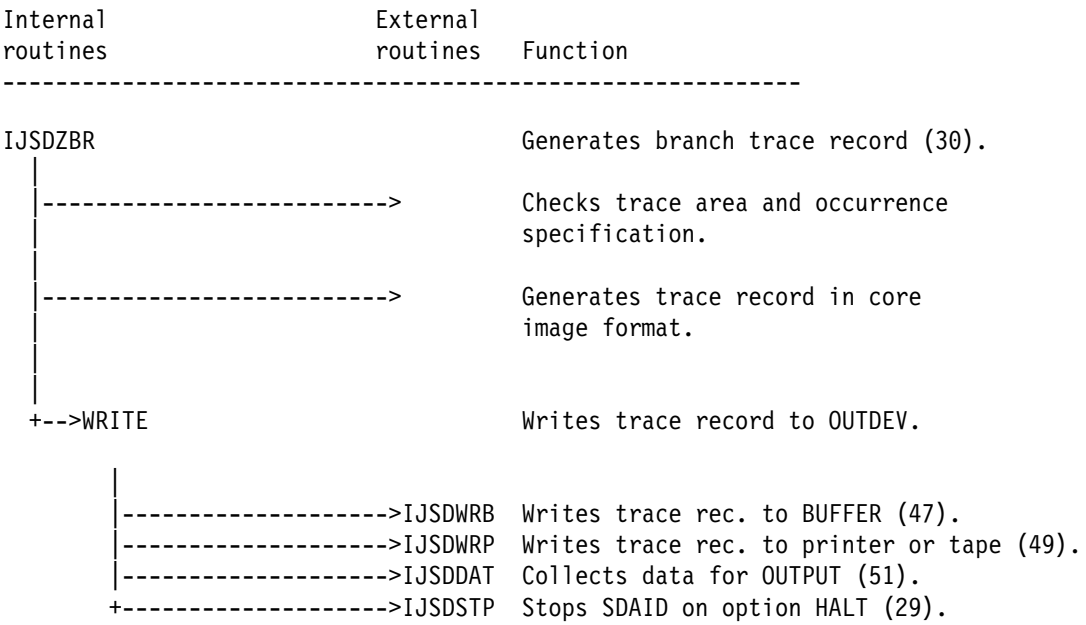
Output:
The branch trace record written on the specified OUTPUT device.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZBR



IJSDZBU - SDAID BUFFER OVERFLOW PROCESSOR

Function:

The SDAID buffer overflow trace processor traces all buffer overflow conditions.
It writes the contents of the SDAID wraparound buffer on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the Execution Data Table (EDTCB).

TRTPTR contains the address of the Trace Table (TRT) entry for buffer overflow traces.

TCBPTR contains the address of the first block in the TCB-queue for buffer overflow traces.

Output:

The wraparound buffer written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

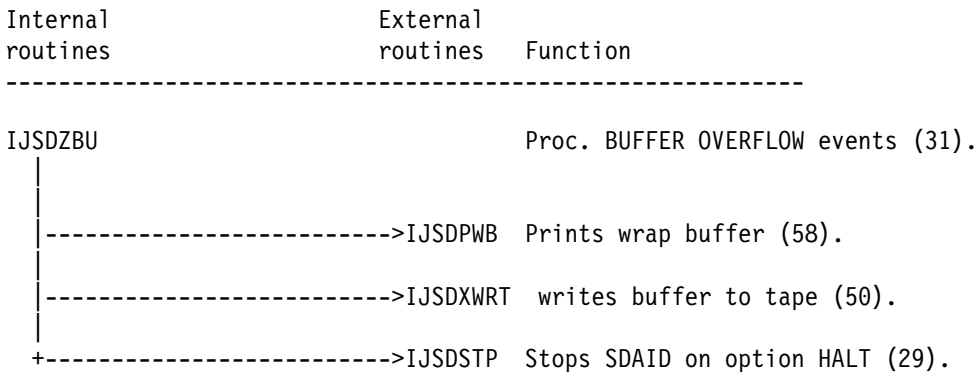
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZBU



IJSDZCA - SDAID CANCEL TRACE PROCESSOR

Function:
The SDAID cancel trace processor traces all program termination conditions which occurred in a specified partition. It generates a cancel trace record and writes it on the specified output device.

Entry point:
Same as module name.

Input:
EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for cancel traces.
TCBPTR contains the address of the first block in the TCB-queue for cancel traces.

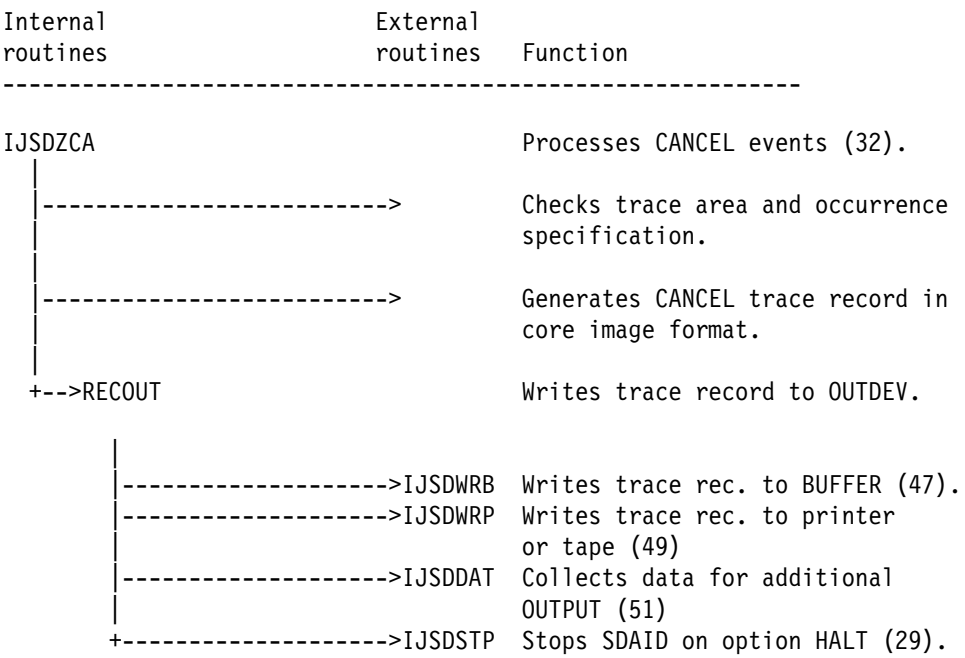
Output:
The cancel trace record written on the specified OUTPUT device.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZCA



IJSDZEX - SDAID EXTERNAL INTERRUPT PROCESSOR

Function:

The SDAID external interrupt processor traces all external interrupts.
It generates an external interrupt trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TRTPTR contains the address of the trace table (TRT) entry for EXT traces.

TCBPTR contains the address of the first block in the TCB-queue for EXT traces.

Output:

The external interrupt trace record written on the specified output device.

Normal exit:

Returns to the interrupt program via LPSW.

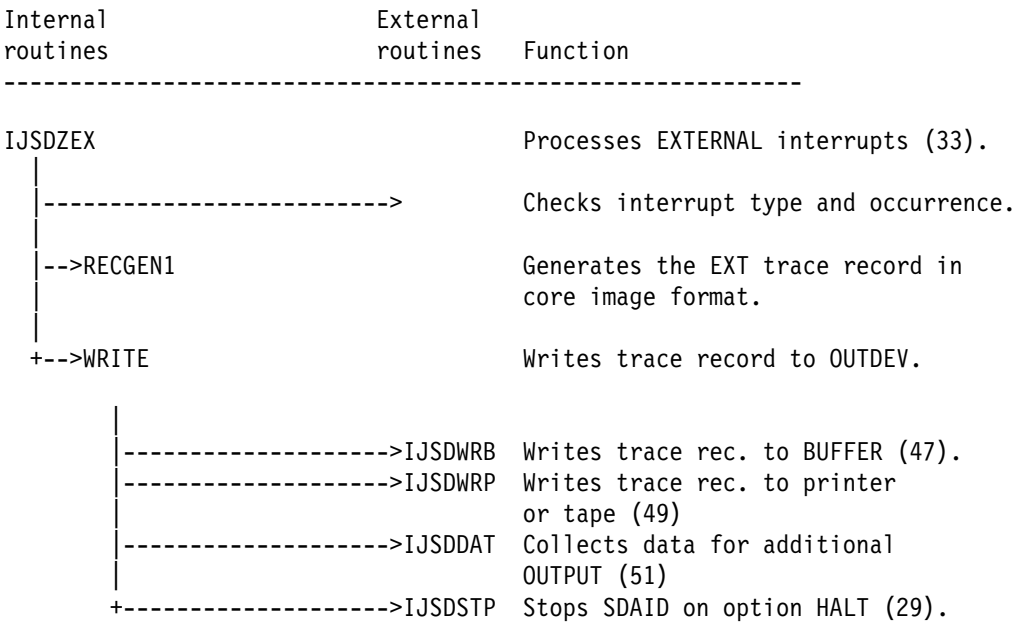
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZEX



IJSDZIN - SDAID INSTRUCTION TRACE PROCESSOR

Function:
The SDAID instruction trace processor traces all or selected instructions in a specified program area.
It generates an instruction trace and writes it on the specified output device.

Entry point:
Same as module name.

Input:
EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for INSTRUCTION traces.
TCBPTR contains the address of the first block in the TCB-queue for INSTRUCTION traces.

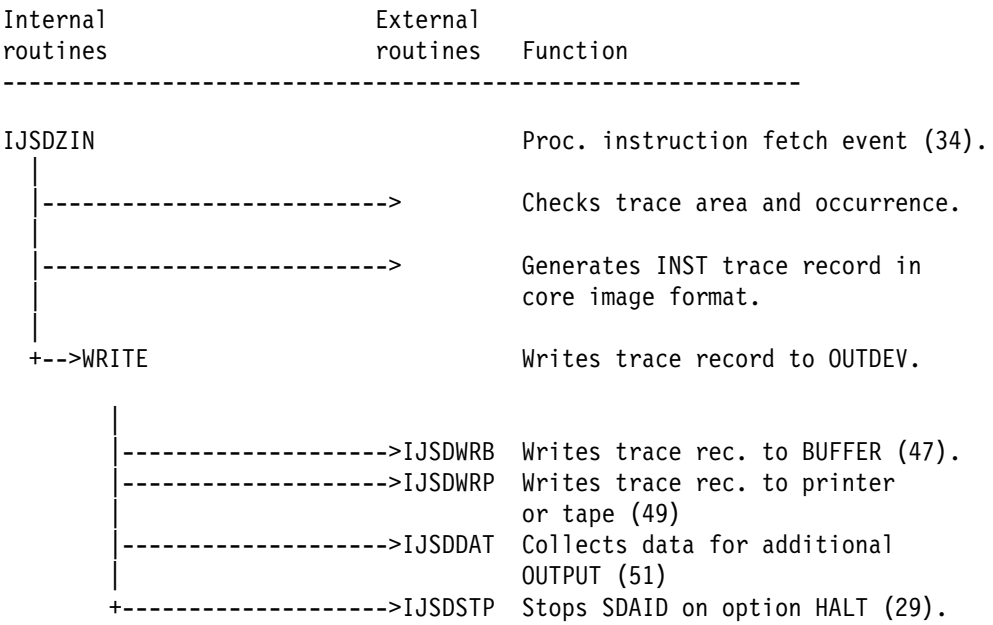
Output:
The instruction trace record written on the specified OUTPUT device.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZIN



IJSDZIO - SDAID I/O TRACE PROCESSOR

Function:

The SDAID I/O interrupt trace processor traces the interrupts from all or selected I/O units, control units or channels. It generates an I/O interrupt trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TRTPTR contains the address of the trace table (TRT) entry for I/O traces.

TCBPTR contains the address of the first block in the TCB-queue for I/O traces.

Output:

The I/O interrupt trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

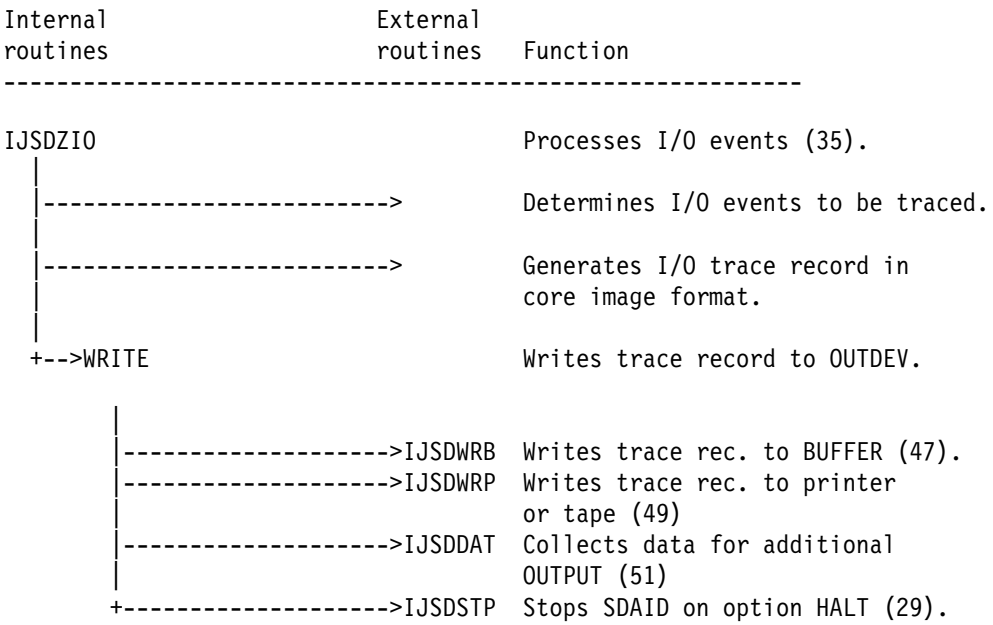
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZIO



IJSDZMC - SDAID MONITOR CALL TRACE PROCESSOR

Function:
The SDAID monitor call event processor traces all or selected monitor call instructions in a specified program area. It generates a monitor call trace record and writes it on the specified output device.

Entry point:
Same as module name.

Input:
EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for monitor call traces.
TCBPTR contains the address of the first block in the TCB-queue for monitor call traces.

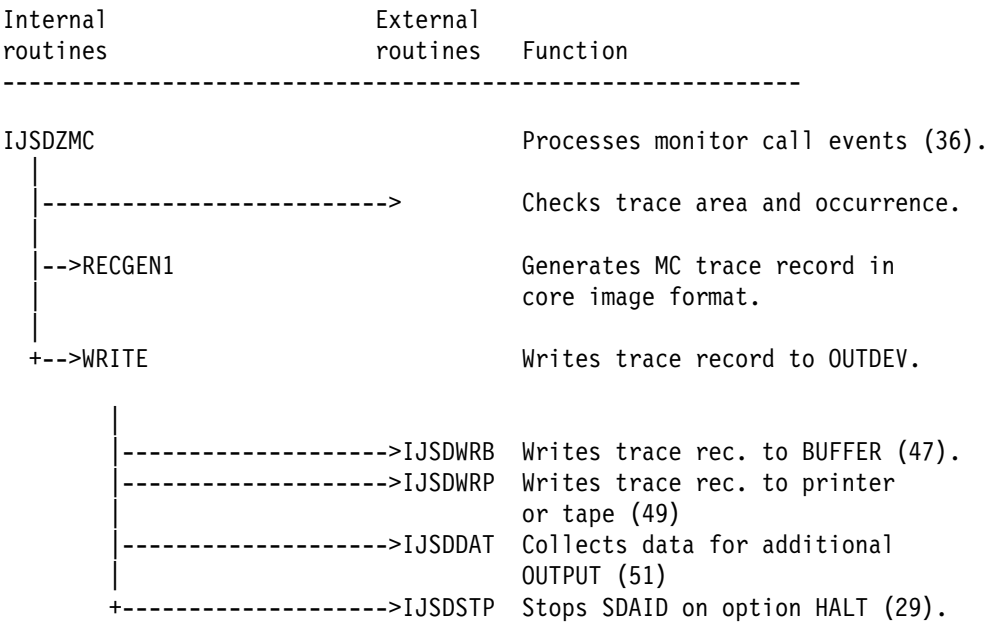
Output:
The monitor call trace record written on the specified OUTPUT device.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZMC



IJSDZPC - SDAID PROGRAM CHECK EVENT PROCESSOR

Function:

The SDAID program check processor traces all or selected program interrupts in a specified program area. It generates a PGMCHECK trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TRTPTR contains the address of the trace table (TRT) entry for program check traces.

TCBPTR contains the address of the first block in the TCB-queue for program check traces.

Output:

The PGMCHECK trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

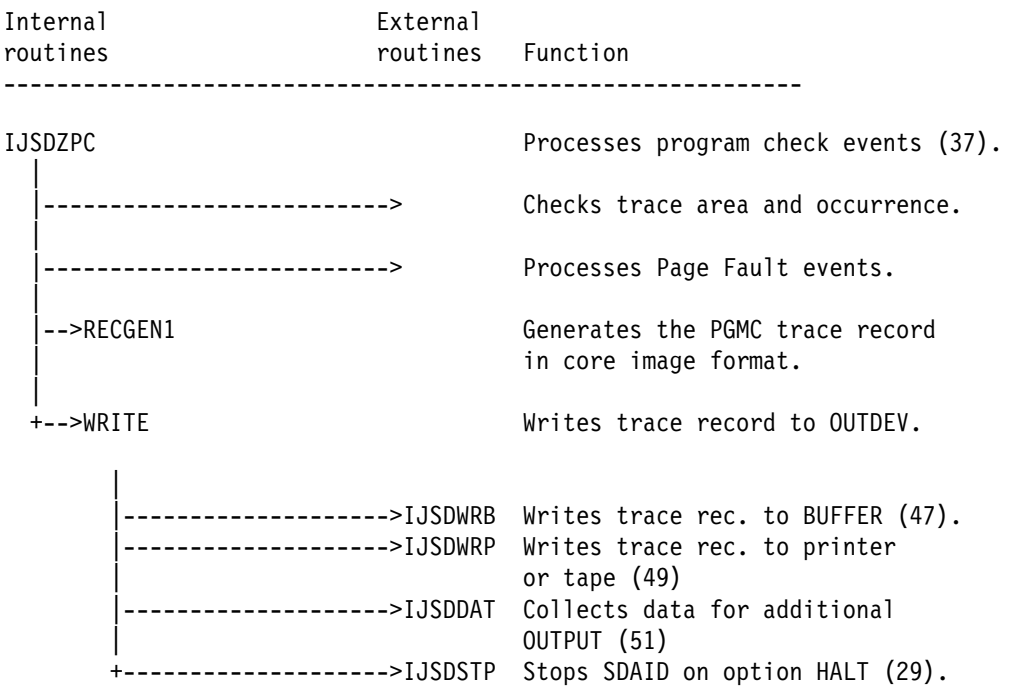
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZPC



IJSDZPL - SDAID PROGRAM LOAD TRACE PROCESSOR

Function:

The SDAID program load event processor traces selected SVC-interrupts and monitor call interrupts used for the program load function, generates a program load trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TRTPTR contains the address of the trace table (TRT) entry for SVC traces in case of an SVC-interrupt, for PGMLOAD traces in case of a monitor call event.

TCBPTR contains the address of the first block in the TCB-queue for SVC or PGMLOAD traces.

Output:

The PGMLOAD trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZPL:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

IJSDZPL                                Processes program load events (38).
  |
  |---------------->                   Analyzes program load SVC
  |      |
  |      |-->RECGEN                    write SVC trace record
  |
  |
  |-->PGMLMON                          Analyzes program load monitor call.
  |
  |      |
  |      |-->GETPNM2                   Processes loaded phase.
  |      |-->PLADDR                    Saves load and ends address of phase.
  |      |-->MODDUMP                   Saves load and ends address for DUMP.
  |      |-->RECGEN                    write HDL trace record
  |


  RECGEN (sub routine)                 Writes trace record to OUTDEV.
     |
     |-------------->IJSDWRB           Writes trace rec. to BUFFER (47).
     |-------------->IJSDWRP           Writes trace rec. to printer
     |                                 or tape (49)
     |-------------->IJSDDAT           Collects data for additional
     |                                 OUTPUT (51)
    +--------------->IJSDSTP           Stops SDAID on option HALT (29).
```

IJSDZSA - SDAID STORAGE ALTER TRACE PROCESSOR

Function:

The SDAID storage alter trace processor traces all instructions which alter a specified program area in storage. It generates a storage alter trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR (=reg3) contains the address of the execution data table (EDTCB).

TRTPTR (=reg4) contains the address of the trace table (TRT) entry for storage alter traces.

TRCBPTR (=reg5) contains the address of the first block in the TCB-queue for storage alter traces.

Output:

The storage alter trace record written on the specified OUTPUT device.

Sequence of Operation:

Find name of altered data space
find SCB address of altered address space
find the instruction which altered storage
loop over all trace control blocks
and check event, pattern, and occurrence
generate the trace record and the continuation record
write trace record to printer/tape device

Normal exit:

Returns to the calling procedure.

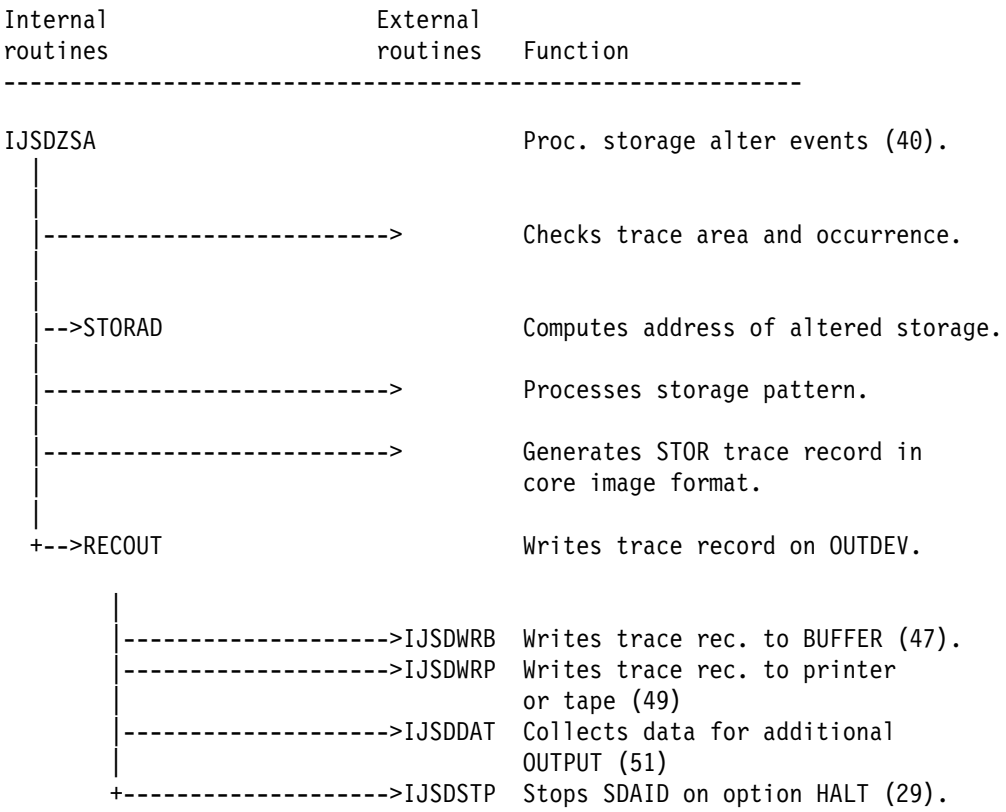
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZSA:



IJSDZSI - SDAID SSCH TRACE PROCESSOR

Function:

The SDAID SSCH trace processor traces the SSCH instruction issued by the supervisor.
It generates an SSCH trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for SSCH traces.
TCBPTR contains the address of the first block in the TCB-queue for SSCH traces.

Output:

The SSCH trace record written on the specified output device.

Normal exit:

Returns to the calling procedure.

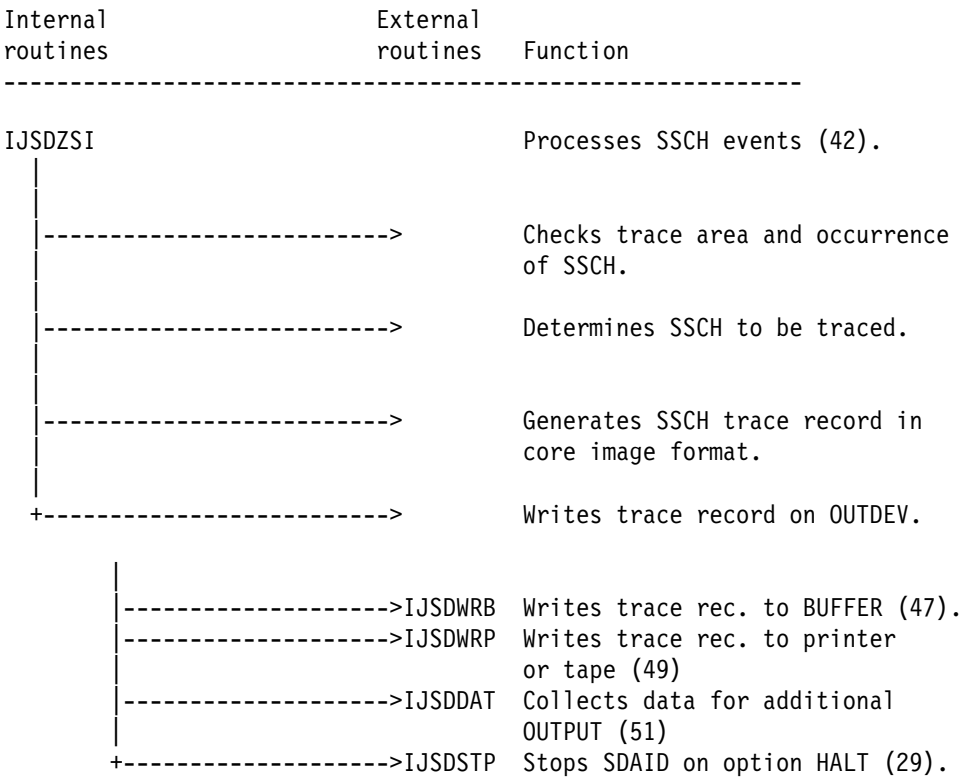
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZSI:



IJSDZSV - SDAID SVC TRACE PROCESSOR

Function:

The SDAID SVC trace processor traces all or selected SVC interrupts in a specified program area.
It generates an SVC trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for SVC traces.
TCBPTR contains the address of the first block in the TCB-queue for SVC traces.

Output:

The SVC trace record written on the specified output device.

Normal exit:

Returns to the calling procedure.

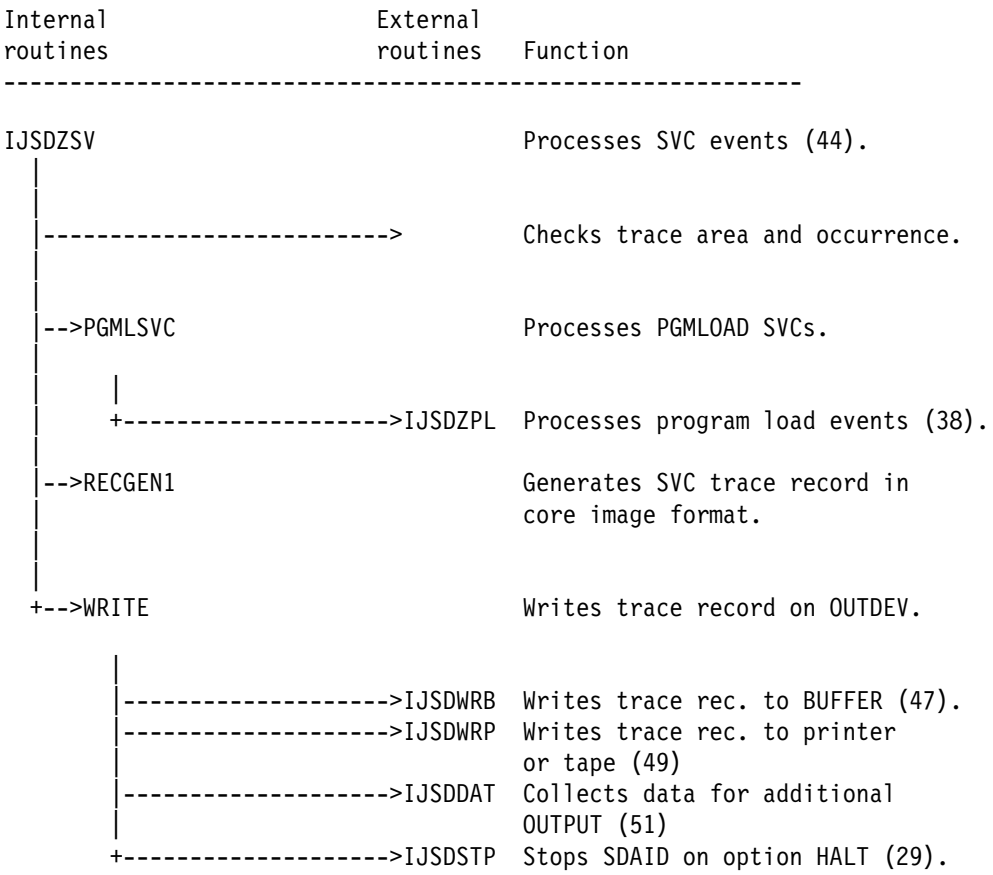
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZSV:



IJSDZVT - SDAID VTAM TRACE PROCESSOR

Function:

The SDAID VTAM trace processor traces all VTAM buffer pool usages.
It generates a VTAM buffer pool trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for VTAM traces.
TCBPTR contains the address of the first block in the TCB-queue for VTAM traces.

Output:

The VTAM trace record written on the specified output device.

Normal exit:

Returns to the calling procedure.

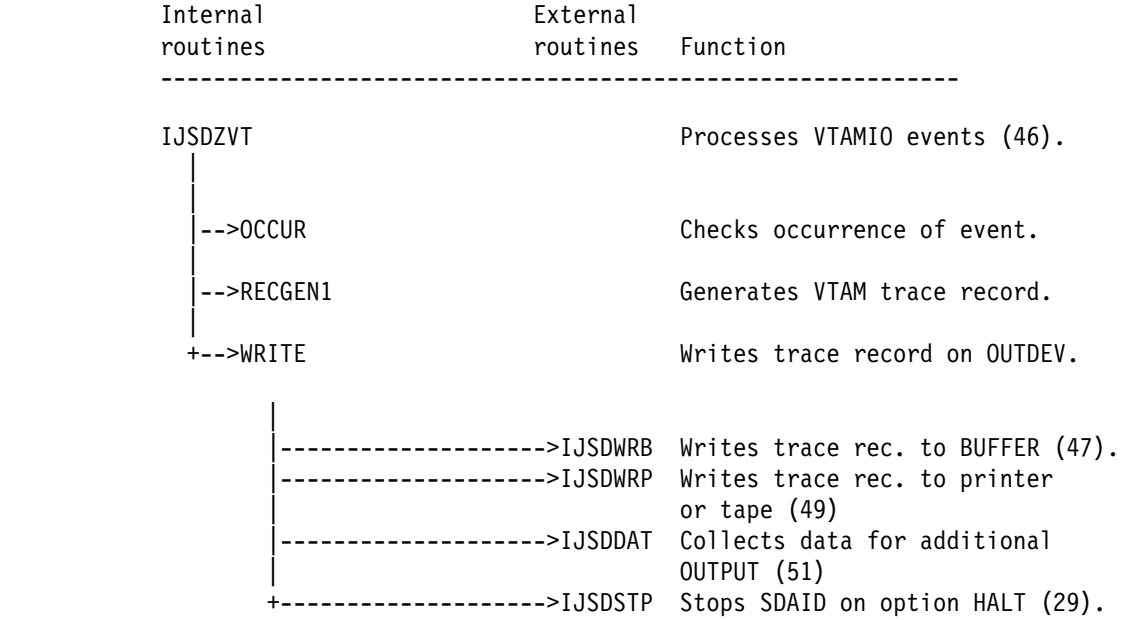
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDZVT



IJSDWRB - SDAID TRACE BUFFER MANAGER

Function:

This routine writes the trace records in an internal buffer in the following modes:

IF OUTDEV B=nnn or OUTDEV B=nnn P=cuu is specified, the buffer is filled in wraparound mode.

IF OUTDEV B=nnn TAPE=cuu or OUTDEV tape=cuu is specified, the buffer is filled and written on tape on overflow.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

Output:

Updated information in control block EDTCB.

Normal exit:

Returns to the calling procdure.

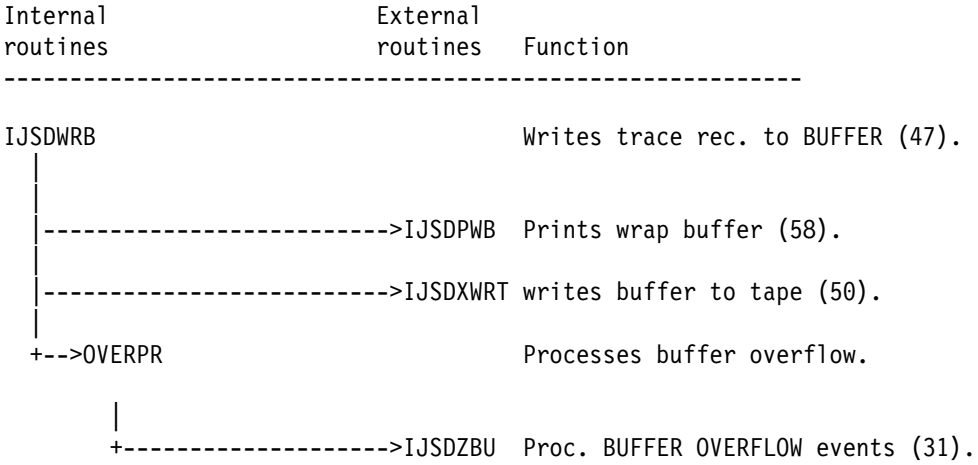
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDWRB



IJSDPPR - ACTIVATE/DEACTIVATE PARTITION TRACES

Function:

The module IJSDPPR activates or deactivates trace control blocks.
The supervisor calls IJSDPPR when a POWER job starts or ends.
IJSDPPR modifies the control registers 9, 10, 11 and
sets/resets the PERACT flag in the PCB.
IJSDPPR activates|deactivates TCBs of the following types:
- TCBFL61 - AREA=syslog-id
- TCBFL65 - JOBNAME=aaaaaaaa
- TCBFL31 - STJBNAM=aaaaaaaa
- TCBFL35 - STAREA=syslog-id

Loaded by:

IJSDROT loads the phase IJSDPPR and moves the load address
into the system communications region (field IJBSDPPR)

Called by:

Partition Preparation Routine (Supervisor) or by
Partition Clean-up Routine (Supervisor)

Input:

None.
IJSDPPR retrieves all required information from the
partition control block PCB.

Output:

None.

Normal exit:

Returns to the calling supervisor routine.

Error exit:

Same as exit normal.

Return Code:

0 no PCB update required
4 set PERACT flag to '1'B
8 set PERACT flag to '0'B

IJSDWRP - OUTPUT PHASE FOR PRINTER OR TAPE

The tracing phases IJSDZxx use the pseudo name IJSDWRP
if they write a trace record to the OUTDEV device.
IJSDWRP therefore stands for IJSDXWRP (if output is to printer),
or for IJSDXWRT (if output is to tape).

IJSDXWRP - WRITE TRACE RECORDS TO OUTDEV PRINTER

Function:

The module IJSDXWRP writes trace records to a printer device.

IJSDXWRP issues a TEST Subchannel Instruction to clear any
pending I/O interruption.
IJSDXWRP stacks the interruption parameter, the IRB, and the
SENSE data. (IJSDINT passes these interruptions at a later time
to the I/O supervisor).
IJSDXWRP modifies control register 6 and the OUTDEV subchannel
to reserve the OUTDEV printer for SDAID.
Then IJSDXWRP writes one or more trace lines to the printer device.
After I/O completion control register 6 and the printer subchannel
are reset to the previous value.

Messages:

4C19A INTERVENTION REQUIRED ON SDAID PRINTER
4C20I I/O ERROR ON SDAID PRINTER

Normal exit:

Returns to the calling SDAID routine.

Error exit:

Same as exit normal.

I/O Errors on OUTDEV Printer:

- Severe I/O errors
  IJSDXWRP sets on SCMSDFL7 to stop tracing;
  If VSE/ESA runs under control of VM/ESA,
  message 4C20I is issued;

- Intervention required on OUTDEV printer
  IJSDXWRP moves the OUTDEV device address to location 0-1;
  IJSDXWRP moves an error code to location 2-3;
  IJSDXWRP moves an error recovery action code to location 5;
  Then IJSDXWRP loads a soft wait PSW of 010E0000 00EEEEEE
  and waits for an external interrupt;
  After external interruption IJSDXWRP retries the I/O operation

IJSDXWRT - WRITE TRACE RECORDS TO OUTDEV TAPE

Function:

The module IJSDXWRT writes trace records to a tape device.
IJSDXWRT modifies control register 6 and the OUTDEV subchannel
to reserve the OUTDEV tape for SDAID.
SDAID keeps the OUTDEV tape reserved till a STOPSD command
is processed.

Note:

The tracing phases IJSDZxx use the descriptive name IJSDWRP
if they write a trace record to the OUTDEV device.
IJSDWRP therefore stands for IJSDXWRP (if output is to printer),
or for IJSDXWRT (if output is to tape).

Messages:

4C18A END OF SDAID TRACE TAPE
4C19A INTERVENTION REQUIRED ON SDAID TAPE
4C20I I/O ERROR ON SDAID TAPE

Normal exit:

Returns to the calling SDAID routine.

Error exit:

Same as exit normal.

I/O Errors on OUTDEV Tape:

- Severe I/O errors
  IJSDXWRT sets on SCMSDFL7 to stop tracing;
  If VSE/ESA runs under control of VM/ESA,
  message 4C20I is issued;

- Intervention required on OUTDEV tape
  IJSDXWRT moves the OUTDEV device address to location 0-1;
  IJSDXWRT moves an error code to location 2-3;
  IJSDXWRT moves an error recovery action code to location 5;
  Then IJSDXWRT loads a soft wait PSW of 010E0000 00EEEEEE
  and waits for an external interrupt;
  After external interruption IJSDXWRT retries the I/O operation

IJSDDAT - SDAID TRACE DATA COLLECTOR

Function:

The SDAID trace data collector produces the header records for the data records according to the specified output parameters of the TRACE command. This header record contains information about the type of data, the address, and the size of data, etc. It is used:

- to write data into a buffer if the output device is tape,
- to write data into the wraparound buffer,
- to convert data to EBCDIC and write it on a printer.

The type of data to be recorded is saved in a so-called data word in the TCBCB for the actual program event during processing of the OUTPUT parameters of the TRACE command. For each active flag bit in the data word, a data record containing the size of the data record, the address of data, and the number of data bytes is generated. If the data area is larger than 2K bytes, a master record and one or more continuation records are generated. The master record keeps a block of data from the beginning of the data area to the next 2K boundary. A continuation record keeps a 2K-byte block beginning at a 2K boundary. The last continuation record keeps the remaining part of the data area beginning at a 2K boundary. But at least a master record is always generated.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TCBPTR contains the address of the first block in the TCB-queue for cancel traces.

Output:

The header record for the data is stored in EDTCB.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDDAT:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

IJSDDAT                                Collects data for OUTPUT (51).
|
|-->PUTREC                             process all record types
     |
     |
     |-->DATGREG                          Collects GREG data.
     |
     |-->DATFREG                          Collects FREG data.
     |
     |-->DATCREG                          Collects CREG data.
     |
     |-->DATCOMR                          Collects COMREG data.
     |
     |-->DATSCOM                          Collects SYSCOM data.
     |
     |-->DATLTA                           Collects LTA data.
     |
     |     |
     |     +-->DUMPCONT                 process block > 2k
     |
     |-->DATPTA                           Collects PTA data.
     |
     |     |
     |     +-->DUMPCONT                 process block > 2k
     |
     |-->DATIRB                           Collects IRB data
     |
     |
     |-->DATCCB                           Collects CCB data.
     |
     |
     |-->DATCCW                           Collects CCW and CCWD data.
     |
     |     |
     |     |-->CCWSCAN                  scan CCW chain to find end of
     |     |                            chain (SSCH trace)
     |     |-->CCWSCAN2                 scan CCW chain to find last
     |     |                            executed CCW (IO trace)
     |     +-->ANALYZE                  analyse CCW command code
     |
     |-->DATTOD                           Collects TOD data.
     |
     |-->DATLOCO                          Collects LOW core data.
     |
     |-->DATSUPV                          Collects SUPV data.
     |
     |     |
     |     +-->DUMPCONT                 process block > 2k
     |
```

```
      |
      |-->DATERRBL                          Collects ERRBL data.
      |
      |     |
      |     +-->DUMPCONT                   process block > 2k
      |
      |-->DATCHQ                            Collects channel queue data.
      |
      |     |
      |     +-->DUMPCONT                   process block > 2k
      |
      |-->DATPUB                            Collects PUB data.
      |
      |-->DATLUB                            Collects LUB data.
      |
      |-->DATPTAB                           Collects PTAB data.
      |
      |-->DATTTAB                           Collects TTAB data.
      |
      |-->DATVT1                            Collects data for VTAM trace.
      |
      |-->DATSNS                            Collects SENSE data
      |
      |-->DATVDMP                           Collects DUMP data.
      |
      |     |
      |     +-->DUMPCONT                   process block > 2k
      |
      |
      |-->DATBUFF                           Collects BUFF data.
          |
          |-------------------->IJSDPWB  Prints wrap buffer and tape (58).
          +-------------------->IJSDWRB  Writes trace rec. to BUFFER (47).
|
|------------------------->IJSDCVT  Converts OUTPUT to EBCDIC (54).
+------------------------->IJSDWRB  Writes trace rec. to BUFFER (47).
```

IJSDCVT - CONVERT OUTPUT TO EBCDIC

Function:

This module converts the data specified by the OUTPUT parameter of the TRACE command to EBCDIC and writes it to the printer.

Entry point:

Same as module name.

Input:

An argument list is passed in register 1. It contains
THEN address and number of bytes of data to be converted.

Output:

The header record for the data is stored in EDTCB.

Normal exit:

Returns to the calling procedure.

Error exit:

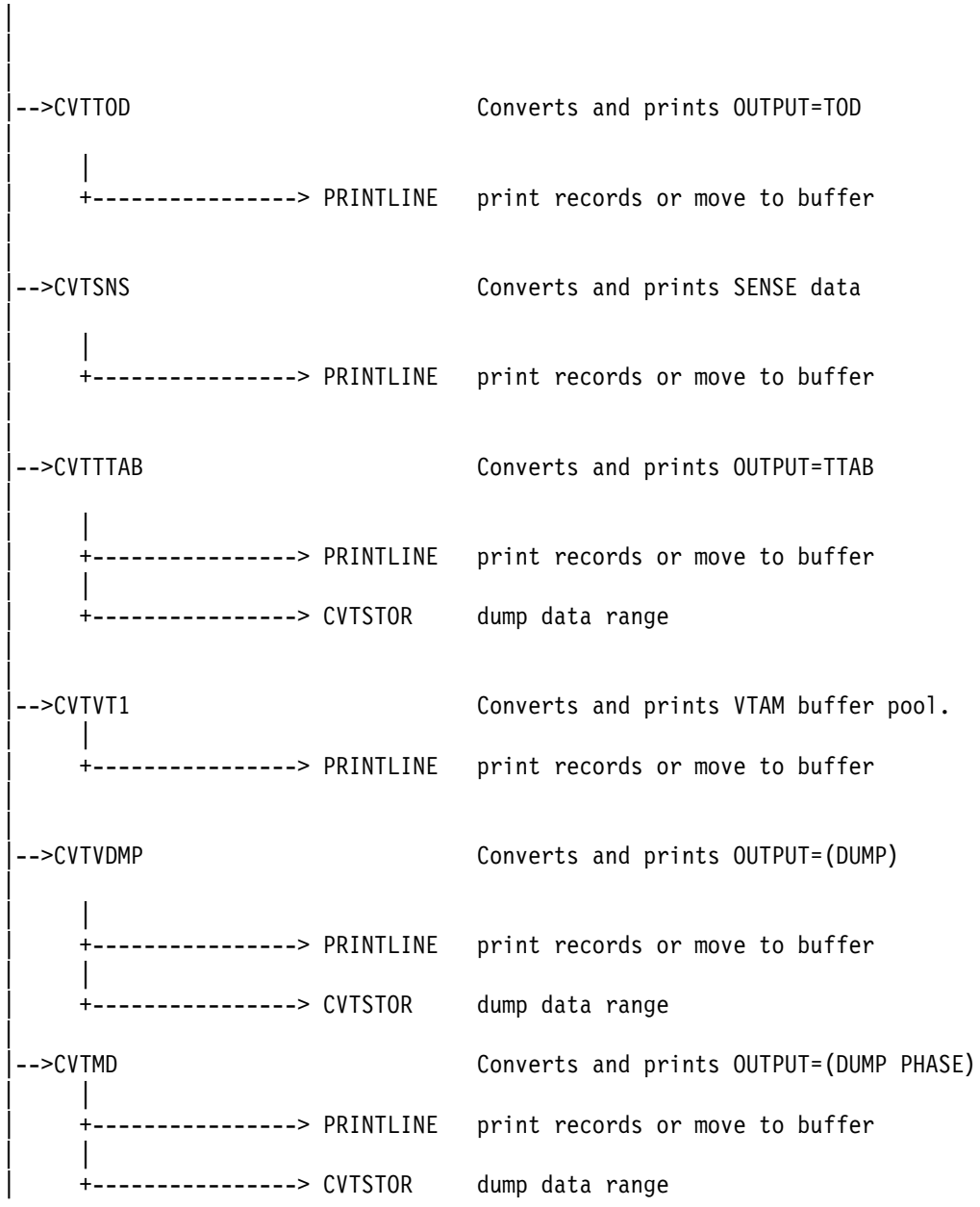
Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDCVT:

```
Internal                    External
routines                    routines   Function
-------------------------------------------------------------

IJSDCVT                                Converts OUTPUT to EBCDIC (54).

  |
  |
  |-->CVTGREG                          Converts and prints
  |                                    OUTP=(GREG,AREAG,CREG,FREG)
  |
  |     |
  |     +----------------> PRINTLINE   print records or move to buffer
  |
  |
  |-->CVTCOMR                          Converts and prints OUTPUT=
  |                                    (COMREG,SYSCOM,LOWCORE,IRB)
  |
  |     |
  |     +----------------> PRINTLINE   print records or move to buffer
  |     |
  |     +----------------> CVTSTOR     dump data range
  |
  |
  |-->CVTLTA                           Converts and prints OUTPUT=
  |                                    (LTA,PTA,PUB,LUB,ERBL,CHQ,SUP)
  |
  |     |
  |     +----------------> PRINTLINE   print records or move to buffer
  |     |
  |     +----------------> CVTSTOR     dump data range
  |
  |-->CVTCCB                           Converts and prints OUTPUT=CCB
  |
  |     |
  |     +----------------> PRINTLINE   print records or move to buffer
  |
  |-->CVTIORB                          Converts and prints OUTPUT=IORB
  |
  |     |
  |     +----------------> PRINTLINE   print records or move to buffer
  |
  |-->CVTCCW                           Converts and prints OUTPUT=(CCW|CCWD)
  |
  |     |
  |     +----------------> PRINTLINE   print records or move to buffer
  |     |
  |     +----------------> CVTSTOR     dump data range
  |
  |
```


-->CVTTOD Converts and prints OUTPUT=TOD
+----------> PRINTLINE print records or move to buffer
-->CVTSNS Converts and prints SENSE data
+----------> PRINTLINE print records or move to buffer
-->CVTTTAB Converts and prints OUTPUT=TTAB
+----------> PRINTLINE print records or move to buffer
+----------> CVTSTOR dump data range
-->CVTVT1 Converts and prints VTAM buffer pool.
+----------> PRINTLINE print records or move to buffer
-->CVTVDMP Converts and prints OUTPUT=(DUMP)
+----------> PRINTLINE print records or move to buffer
+----------> CVTSTOR dump data range
-->CVTMD Converts and prints OUTPUT=(DUMP PHASE)
+----------> PRINTLINE print records or move to buffer
+----------> CVTSTOR dump data range

IJSDNEM - OP CODE CONVERSION

Function:
Converts a hexadecimal operation code into its mnemonic representation.

Phase Name:
IJSDNEM

Module Name:
IJSDNEM

Entry point:
IJSDNEM

Input:
R0 - bits 16-31: First two bytes of operation code.
R1 Address where mnemonic is to be stored.

Output:
Mnemonic operation code

Phases called:
None

Macros used:
None

Sequence of Operation
process code x'01' (PR, UPT)
process code x'07' (branch register instruction)
process code x'47' (branch instruction)
process code x'B2xx' (table B2XESA)
process code x'E5xx' (LASP, TPROT, MVCSK, MVCDK)
Otherwise:
Use the first byte of the op code to index into the mnemonic table (OPCODTAB), then store the mnemonic, and exit.

Information Messages:
None

Normal exit:
Returns to caller.

Error exit:
Returns to caller.

IJSDPWB - PRINT WRAP BUFFER AND TAPE

Function:

The print wraparound buffer processor gets control when an overflow of this buffer has occurred. It prints the content of the wraparound buffer.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table EDTCB.

Output

The content of the wraparound buffer is printed on the specified output device.

Normal exit:

Returns to the calling procedure.

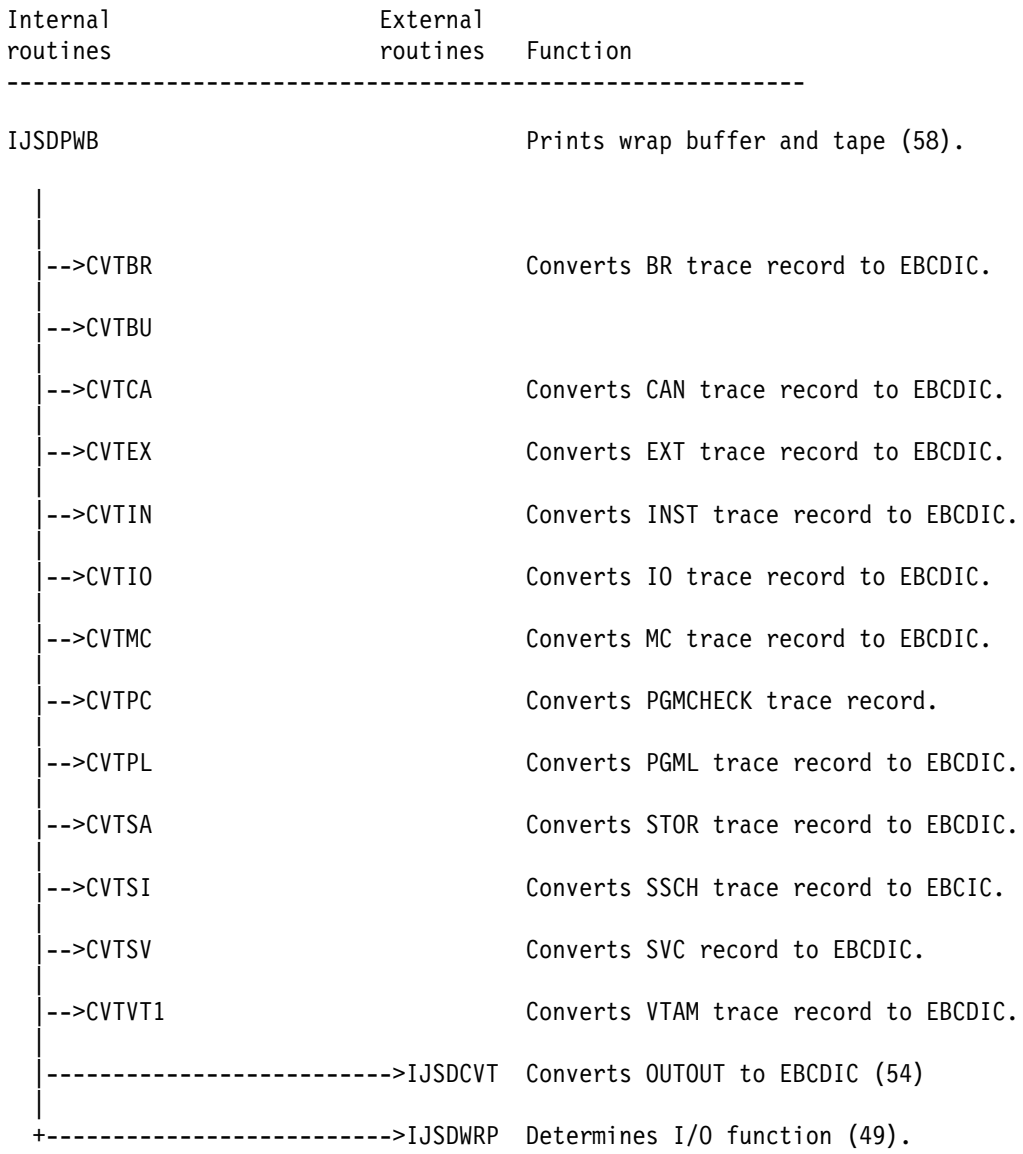
Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

Flow of Control for Entry Point: IJSDPWB:



Chapter 3. SDAID: Data Area Information

This section is provided to give an overview of the structures of the SDAID control blocks.

The following control blocks are described:

1. The Global Data Table GDTCB.
2. The Initialization Data Table IDTCB.
3. The Execution Data Table EDTCB.
4. The Trace Table TRTCB.
5. The Trace Table Entry TRTCB1
6. The Trace Control Block TCBCB.

Figure 1 on page 62 gives you an overview of the control block structure for SDAID.

The remainder of this section shows the layout and explains all fields of the data areas used by SDAID.

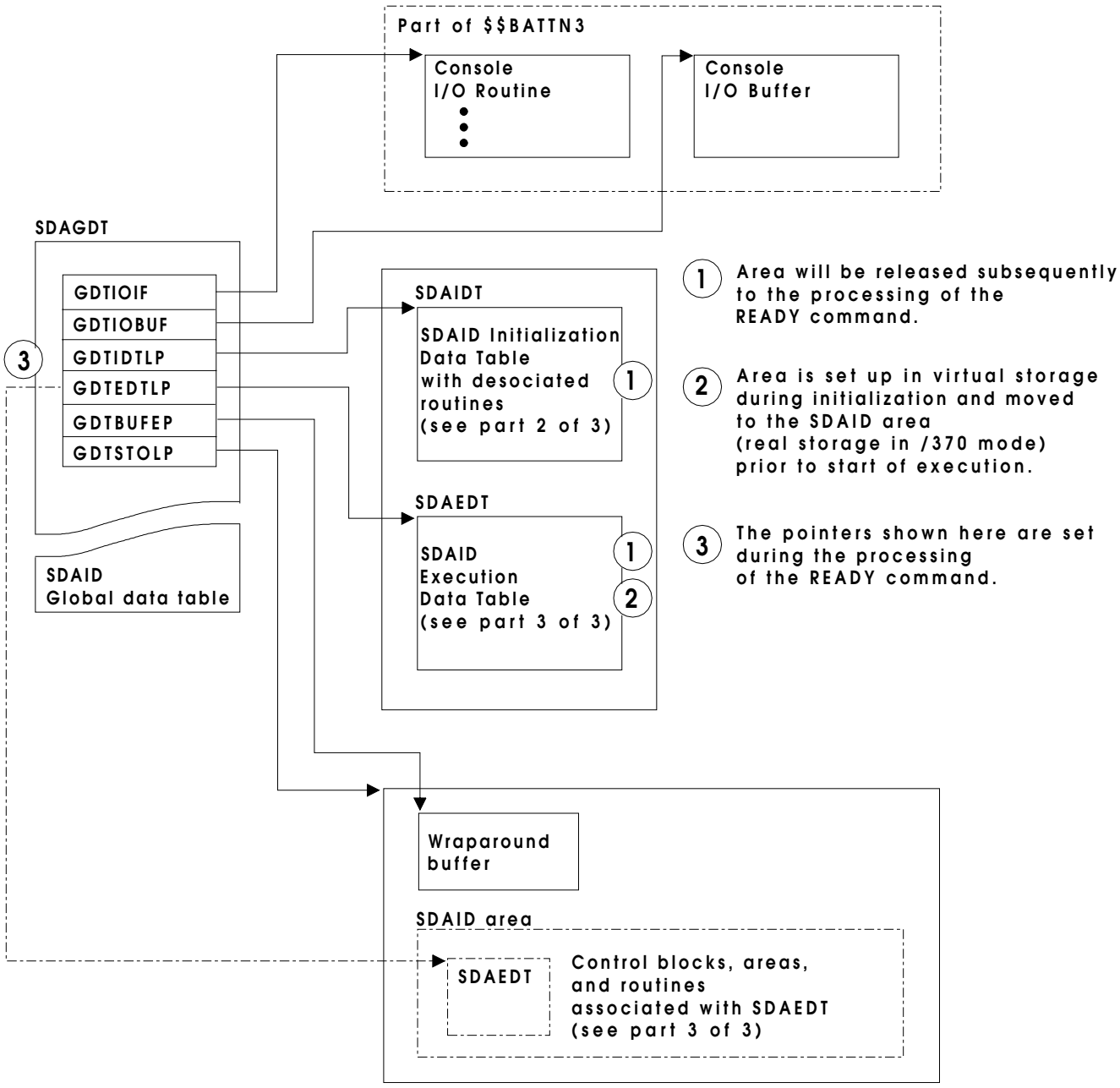
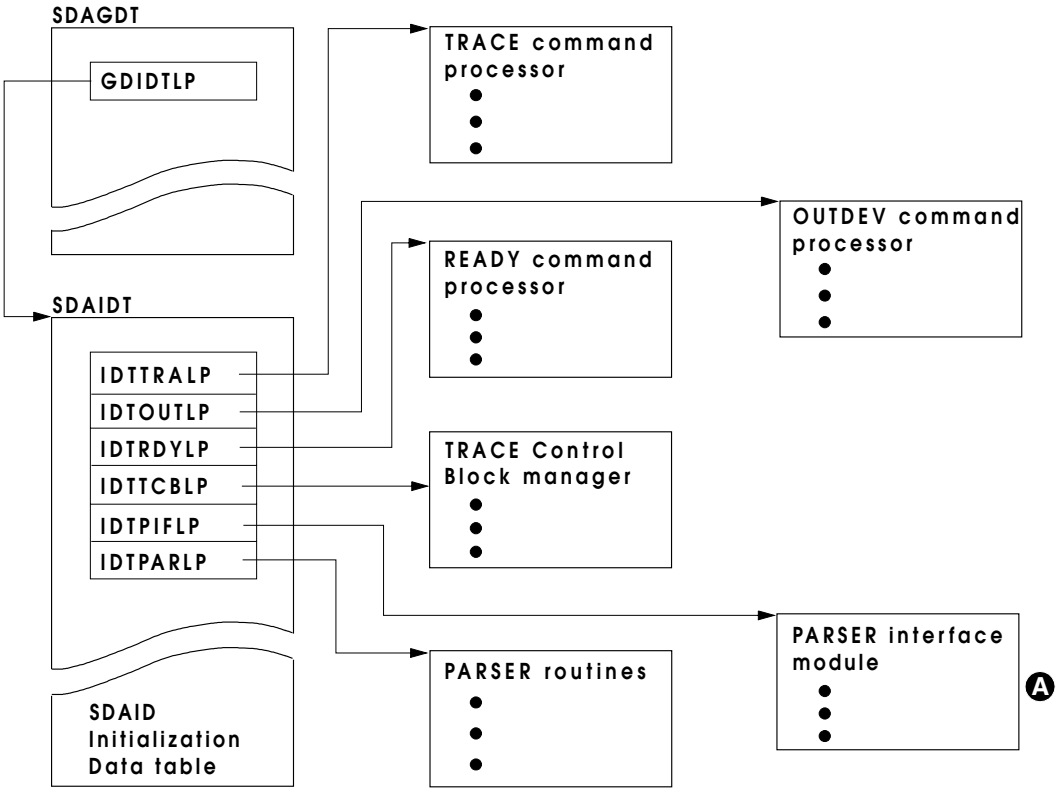


Figure 1 (Part 1 of 3). Structure of SDAID Control Blocks and Data Areas



A Controls required communication between the SDAID initialization routines and the PARSER routines, which are needed during SDAID initialization.

Figure 1 (Part 2 of 3). Structure of SDAID Control Blocks and Data Areas

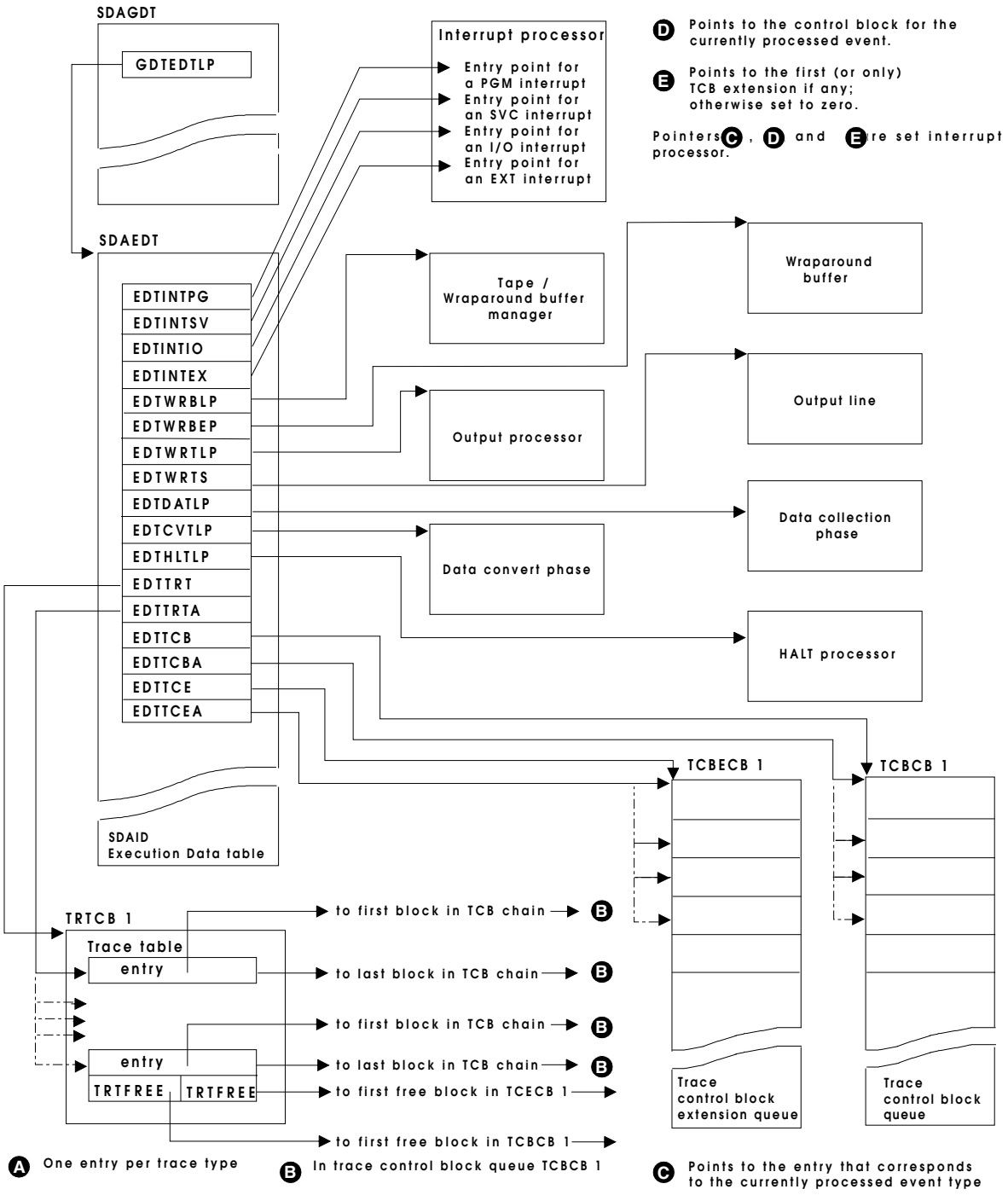


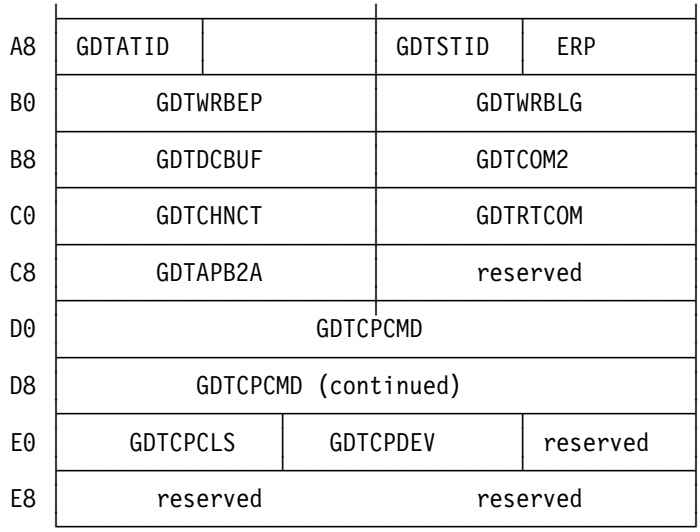
Figure 1 (Part 3 of 3). Structure of SDAID Control Blocks and Data Areas

GDTCB - SDAID Global Data Table

The GDTCB contains information needed when the SDAID program is active. The field IJBSAVSD in the SYSCOM contains a pointer to the GDTCB. F*n indicates flag bytes defined in the following lists.

Created by	Modified by	Used by
IJSDROT	IJSDROT, IJSDRDY	all SDAID routines

0	GDTNAME						
8	F*1	F*2	F*3	F*4	GDTPERUS		
10	GDTROTLP				GDTROTLG		
18	GDTMSGLP				GDTMSGLG		
20	F*5	GDTIORTN			GDTIOBUF		
28	GDTIDTLP				GDTIDTLG		
30	GDTIDTEP						
38	GDTEDTLP				GDTEDTLG		
40	GDTEDTEP						
48	GDTSTOLP				GDTSTOLG		
50	GDTPGMN						
58	GDTSVCN						
60	GDTION						
68	GDTEXTN						
70	GDTXR08				GDTXR09		
78	GDTXR10				GDTXR11		
80	GDTPWBLP				GDTPWBLG		
88	GDTCCWT1				GDTCCWT2		
90	GDTSENSE				GDTOUT		
98	GDTTIBA				GDTINTID	GDTOUTX	
A0							


A8
GDTATID
GDTSTID
ERP
B0
GDTWRBEP
GDTWRBLG
B8
GDTDCBUF
GDTCOM2
C0
GDTCHNCT
GDTRTCOM
C8
GDTAPB2A
reserved
D0
GDTCPCMD
D8
GDTCPCMD (continued)
E0
GDTCPCLS
GDTCPDEV
reserved
E8
reserved
reserved

OFFSET (HEX)	FIELD NAME	LENGTH AND BIT PATTERN	DESCRIPTION
0	GDTCB	240 (dec)	Global Data Table GDTCB
0	GDTNAME	8	Control block name ('GDTCB')
8	GDTFL1	1	F*1 flag byte 1
			Bits defined in GDTFL1
8	GDTFL12	X'40'	OUTDEV command specified
8	GDTFL13	X'20'	TRACE command specified
8	GDTFL18	X'01'	READY command specified
9	GDTFL2	1	F*2 flag byte 2
			Bits defined in GDTFL2
9	GDTFL21	X'80'	SDAID command specified
9	GDTFL22	X'40'	STOPSD command specified
9	GDTFL23	X'20'	STARTSD command specified
9	GDTFL24	X'10'	ENDSD command specified
A	GDTFL3	1	F*3 flag byte 3
			Bits defined in GDTFL3
A	GDTFL31	X'80'	OUTDEV tape is reserved
A	GDTFL32	X'40'	OUTDEV command in progress
A	GDTFL33	X'20'	TRACE command in progress
A	GDTFL37	X'02'	READY command in progress
A	GDTFL38	X'01'	READY command was processed
B	GDTFL4	1	F*4 flag byte 4
			Bits defined in GDTFL4
B	GDTFL41	X'80'	SDAID command in progress
B	GDTFL42	X'40'	STOPSD command in progress
B	GDTFL43	X'20'	STARTSD command in progress
B	GDTFL44	X'10'	ENDSD command in progress
C	GDTPERUS	2	# of I/A traces (not SDAID)
E	*	2	Reserved
10	GDTROTLP	4	Address of IJSDROT or 0
			Bits defined in GDTROTLP
10	GDTROTF1	X'80'	Release IJSDROT
14	GDTROTLG	4	Phase size in hex or 0
	Anchor for message library and manager IJSDMSG		
18	GDTMSGLP	4	Address of IJSDMSG or 0
1C	GDTMSGLG	4	Phase size in hex or 0

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Anchor for I/O routine in $$BATTN3		
20	GDTIOFL	1	F*5 flag byte
			Bits defined in GDTIOFL
20	GDTIOFLW	X'80'	Indicates WRITE request
20	GDTIOFLR	X'40'	Indicates READ request
20	GDTIOFLP	X'20'	Writes on SYSLST
21	GDTIORTN	3	Address of I/O routine in BATTN3
24	GDTIOBUF	4	Address of console I/O buffer
	Anchor for Initialization Data Table IJSDIDT		
28	GDTIDTLP	4	Address of IJSDIDT or 0
2C	GDTIDTLG	4	Phase size in hex or 0
30	GDTIDTEP	4	Address of Initialization Data Table IDTCB
	Anchor for Execution Data Table IJSDEDT		
38	GDTEDTLP	4	Address of IJSDEDT or 0
3C	GDTEDTLG	4	Phase size in hex or 0
40	GDTEDTEP	4	Address of Execution Data Table EDTCB
	Anchor for SDAID execution storage		
48	GDTSTOLP	4	Start address of execution area
4C	GDTSTOLG	4	Size of area in hex or 0
	Save areas for NEW PSWs		
50	GDTPGMN	8	Save area for PGM NEW PSW
54	GDTPGMA	4	Entry point for PGM interrupts
58	GDTSVCN	8	Save area for SVC NEW PSW
5C	GDTSVCA	4	Entry point for SVC interrupts
60	GDTION	8	Save area for I/O NEW PSW
64	GDTIOA	4	Entry point for I/O interrupts
68	GDTEXTN	8	Save area for EXT NEW PSW
6C	GDTEXTA	4	Entry point for EXT interrupts
	Save area for control registers		
70	GDTXR	16	Save area for control registers
70	GDTXR08	4	Control register 8
70	GDTXR08C	4	
74	GDTXR09	4	Control register 9
74	GDTXR09C	4	
78	GDTXR10	4	Control register 10
78	GDTXR10C	4	
7C	GDTXR11	4	Control register 11
7C	GDTXR11C	4	

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Anchor for print buffer phase IJSDPWB		
80	GDTPWBLP	4	Address of IJSDPWB or 0
84	GDTPWBLG	4	Phase size in hex or 0
	Pointers to data areas in DOS/VSE supervisor		
88	GDTCCWT1	4	Address of Device Translation Table (DEVTRTAB)
8C	GDTCCWT2	4	Address of Device Specific Table (DEVLIST)
90	GDTSENSE	4	Address of sense buffer
94	GDTOUT	4	Outdevice characteristics
94	GDTOUTCH	2	Outdevice channel number
96	GDTOUTDT	1	Outdevice device type
97	*	1	reserved
98	GDTTIBA	4	Address of TIB area
9C	GDTINTID	2	TID of set-up partition
9E	GDTOUTX	2	OUTDEV tape PUB index
A0	*	4	reserved
A4	*	4	reserved
A8	GDTATID	2	TID of highest syst task
AA	*	2	reserved
AC	GDTSTID	2	TID of highest sub-task
AE	*	2	reserved
	Anchor for wraparound buffer		
B0	GDTWRBEP	4	Start address of buffer or 0
B4	GDTWRBLG	4	Size of buffer in hex or 0
	Fields used by VSE dump component		
B8	*	8	reserved
C0	GDTCHNCT	4	Channel control table
C4	GDTRTCOM	4	Address of interface area to
C8	GDTAPB2A	4	PUB2AREA
CC	*	4	reserved
	CP command work area (IJSDROT)		
D0	GDTCPCMD	16	CP command work area
E0	GDTCPCLS	3	CP close command: CL
E3	GDTCPDEV	3	CP close command: cuu

IDTCB - SDAID Initialization Data Table

The module IJSDROT loads the IDTCB at SDAID initialization time. The table IDTCB is contained in the module IJSDIDT. The IDTCB contains an entry for each of the initialization routines, the I/O areas and the command tables with its directories. The IDTCB is anchored in the Global Data Table (GDTCB).

0	IDTCBNAM	
8	IDTCMDLP	IDTCMDLG
10	IDTCMDT	
18	IDTTRALP	IDTTRALG
20	IDTOUTLP	IDTOUTLG
28	IDTRDYLP	IDTRDYLG
30	IDTTCBLP	IDTTCBLG
38	IDTTCBF	IDTTCBP
40	IDTTCBM1	IDTTCBM2
48	IDTTCBA1	IDTTCBA2
50	IDTPIFLP	IDTPIFLG
58	IDTPARLP	IDTPARLG
60	IDTPARIP	IDTPARRP
68	IDTPARFL	IDTCPCBP
70	IDTIOAD	IDTIOLG
78	IDTIONXT	reserved
80	IDTCT2LP	IDTCT2LG
88	IDTCT3LP	IDTCT3LG
90	IDTCDOD (28 BYTES)	
AC	IDTCDTR (28 BYTES)	
C8	IDTPARIO (80 BYTES)	
118	IDTPARER (80 BYTES)	

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	IDTCB	360 (dec)	Initialization data table IDTCB
0	IDTCBNAM	8	Control block name ('IDTCB')
8	IDTCMD	16	Anchor for command table
8	IDTCMDLP	4	Address of phase or 0
C	IDTCMDLG	4	Phase size in hex or 0
10	IDTCMDT	4	Address of actual command table
	Anchor for TRACE command processor IJSDTRA		
18	IDTTRALP	4	Address of phase IJSDTRA or 0
1C	IDTTRALG	4	Phase size in hex or 0
	Anchor for OUTDEV command processor IJSDOUT		
20	IDTOUTLP	4	Address of phase IJSDOUT or 0
24	IDTOUTLG	4	Phase size in hex or 0
	Anchor for READY command processor		
28	IDTRDYLP	4	Address of phase IJSDRDY or 0
2C	IDTRDYLG	4	Phase size in hex or 0
	Anchor and interface to trace control block manager IJSDTCB		
30	IDTTCBLP	4	Address of phase IJSDTCB or 0
34	IDTTCBLG	4	Phase size in hex or 0
38	IDTTCBX	24	Interface to IJSDTCB
38	IDTTCBF	4	Function to be performed
3C	IDTTCBP	4	Position of enqueued element
40	IDTTCBM1	4	ID of master TCB
44	IDTTCBM2	4	Position of master TCB
48	IDTTCBA1	4	ID of appendage TCB
4C	IDTTCBA2	4	Position of appended TCB
	Anchor for parser interface module IJSDPIF		
50	IDTPIFLP	4	Address of phase IJSDPIF or 0
54	IDTPIFLG	4	Phase size in hex or 0

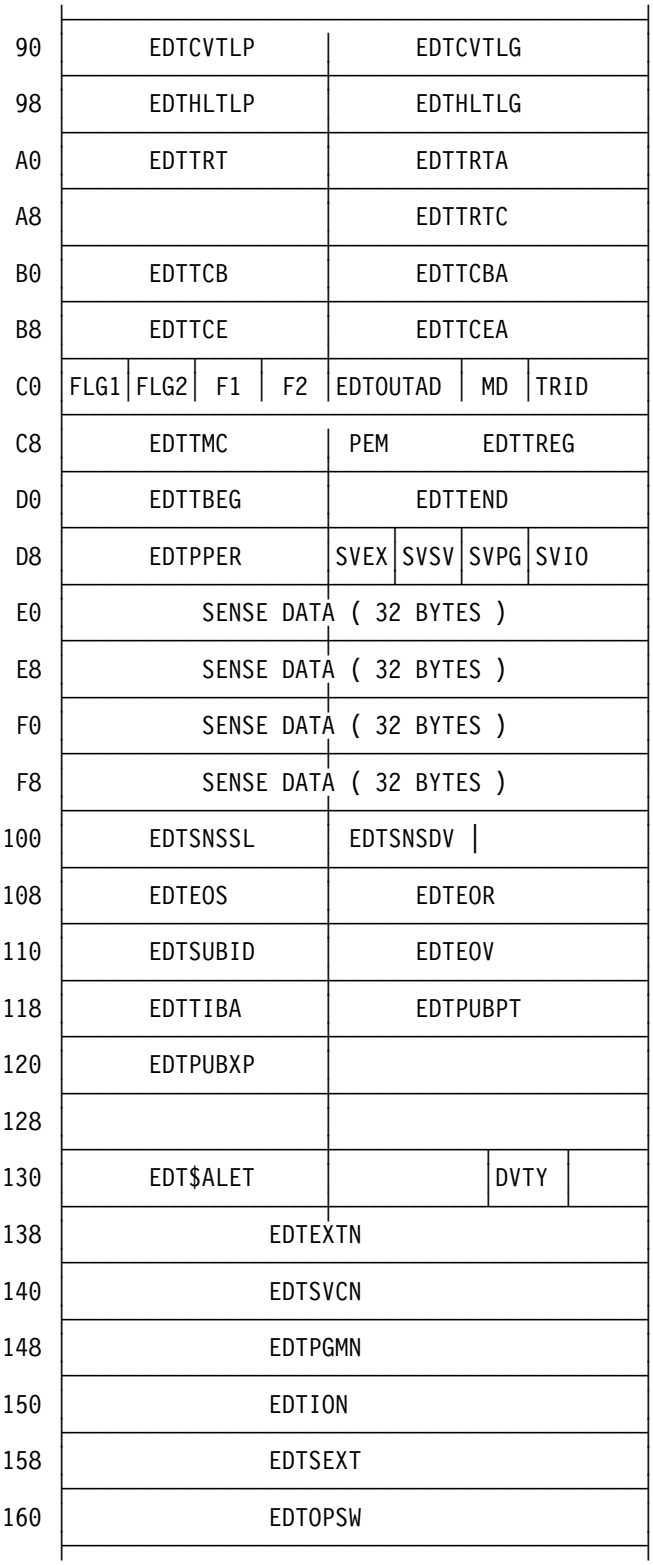
OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Anchor and interface to PARSER		
58	IDTPARLP	4	Address of phase IJPARS or 0
5C	IDTPARLG	4	Phase size in hex
60	IDTPARIP	4	Address of PARSER input area
64	IDTPARRP	4	Address of PARSER reply area
68	IDTPARFL	4	PARSER flag byte
			Bits defined in IDTPARFL
68	IDTPARF1	X'80'	Check exit active
68	IDTPARF2	X'40'	Check semantic error
68	IDTPARF5	X'08'	
6C	IDTCPCBP	4	Address of PARSER control block CPCB
	Anchor for command input area		
70	IDTIO	16	Anchor for I/O area
70	IDTIOAD	4	Address of I/O area 0
74	IDTIOLG	4	Size of I/O area or 0
78	IDTIONXT	4	Address of continuation
	Anchor for command tables		
80	IDTCT	32	Anchor for command tables
80	IDTCT2	8	Anchor for OUTDEV cmd table
80	IDTCT2LP	4	Address of command table or 0
84	IDTCT2LG	4	Size of table in hex or 0
88	IDTCT3	8	Anchor for TRACE command table
88	IDTCT3LP	4	Address of command table or 0
8C	IDTCT3LG	4	Size of table in hex or 0
	Command directory for the SDAID commands		
90	IDTCDOD	28	Directory for OUTDEV command
AC	IDTCDTR	28	Directory for TRACE command
	I/O areas		
C8	IDTPARIO	80	PARSER I/O area 1
118	IDTPARER	80	PARSER I/O area 2

EDTCB - SDAID Execution Data Table

The execution data table contains information needed during SDAID's tracing, and is anchored in the global data table (GDTCB).

Created by	Modified by	Used by
IJSDEDT	all SDAID phases	

0	EDTCBNAM					
8	FL1	FL2	FL3	FL4	EDTGDTPT	
10	EDTPPRLP				EDTPPRLG	
18	EDTSCMPT				EDTPCMPT	
20	EDTINTLP				EDTINTLG	
28	EDTINTPG				EDTINTSV	
30	EDTINTIO				EDTINTEX	
38	EDTWRBLP				EDTWRBLG	
40	EDTWRBEP				EDTWRBSZ	
48	EDTWRTLP				EDTWRTLG	
50	EDTWRTF				EDTWRTS	
58	EDTWRTL					
60	EDTIOSLP				EDTIOSLG	
68	EDTIOSCT				EDTIOSNO	
70	EDTIOSAR				EDTIOSSZ	
78						
80	EDTDATLP				EDTDATLG	
88	EDTDATF					


90 EDTCVTLP EDTCVTLG
98 EDTHLTLP EDTHLTLG
A0 EDTTRT EDTTRTA
A8 EDTTRTC
B0 EDTTCB EDTTCBA
B8 EDTTCE EDTTCEA
C0 FLG1 FLG2 F1 F2 EDTOUTAD MD TRID
C8 EDTTMC PEM EDTTREG
D0 EDTTBEG EDTTEND
D8 EDTPPER SVEX SVSV SVPG SVIO
E0 SENSE DATA (32 BYTES)
E8 SENSE DATA (32 BYTES)
F0 SENSE DATA (32 BYTES)
F8 SENSE DATA (32 BYTES)
100 EDTSNSSL EDTSNSDV
108 EDTEOS EDTEOR
110 EDTSUBID EDTEOV
118 EDTTIBA EDTPUBPT
120 EDTPUBXP
128
130 EDT$ALET DVTY
138 EDTEXTN
140 EDTSVCN
148 EDTPGMN
150 EDTION
158 EDTSEXT
160 EDTOPSW

168	EDTPID	EDTPID1	EDTIOAD	EDTSID
170	EDTOEV1		EDTSEV1	
178	EDTBLKLP		EDTBLKLG	
180	EDTCAW		EDTCCB	
188	EDTCSW			
190	EDTCLOCK			
198	EDTSAV13			
1A0	EDTSAV1 (72 BYTES)			
1E8	EDTSAV2 (64 BYTES)			
228	EDTSAV3 (64 BYTES)			
268	EDTTREC (96 BYTES)			
2C8	EDTTREC1 (120 BYTES)			
340				
348	EDTWORK (80 BYTES)			
398	EDTPGOLD			
3A0	EDTMCSAV		EDTPGSAV	
3A8	EDTIOMAD		EDTIOPAD	
3B0	EDTSTCKM			
3B8	EDTERRCD			
3C0	EDTNEMLP		EDTNEMLG	
3C8			EDTTREX	
3D0	EDTIDALP		EDTIDALG	

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	EDTCB	1048 (dec)	Structure for Execution Data Table EDTCB
0	EDTCBNAM	8	Control block name ('EDTCB')
8	EDTFL1	1	tape: set mode op-code
9	EDTFL2	1	Flag byte 2
			Bits defined in EDTFL2
9	EDTFL21	X'80'	PER event mixed with other
9	EDTFL22	X'40'	PER event mixed with MC
9	EDTFL23	X'20'	PER event mixed with PGM
A	EDTFL3	1	Flag byte 3
			Bits defined in EDTFL3
A	EDTFL31	X'80'	user in 31-bit mode
A	EDTFL34	X'10'	Access register mode
A	EDTFL35	X'08'	IJSDPPR to be loaded
A	EDTFL37	X'02'	OUTP=CCW requires DAT off
A	EDTFL38	X'01'	Virtual addressing (DAT on)
B	EDTWRBFL	1	Flag byte 4
			Bits defined in EDTWRBFL
B	EDTWRBPC	X'80'	record buffer if pgmc
B	EDTWRBOF	X'40'	record buffer if overflow
B	EDTWRBEN	X'20'	record buffer if endsd
B	EDTWRBBU	X'10'	record buffer on event
B	EDTWRBBT	X'02'	write buffer on tape
B	EDTWRBOV	X'01'	buffer overflow occured
C	EDTGDTPT	4	Address of Global Data Table (GDTCB).
10	EDTPPRLP	4	Address of IJSDPPR
14	EDTPPRLG	4	Length of IJSDPPR
18	EDTSCMPT	4	Address of DOS SYSCOM
1C	EDTPCMPT	4	Address of DOS COMRG
20	EDTINTLP	4	Address of phase IJSDINT or 0
24	EDTINTLG	4	Phase size in hex or 0
28	EDTINTPG	4	Entry point for PGM interrupts
2C	EDTINTSV	4	Entry point for SVC interrupts
30	EDTINTIO	4	Entry point for I/O interrupts
34	EDTINTEX	4	Entry point for EXT interrupts
	Anchor and interface to wrap buffer manager IJSDWRB		
38	EDTWRBLP	4	Address of phase IJSDWRB or 0
3C	EDTWRBLG	4	Phase size in hex or 0
40	EDTWRBEP	4	Address of wrap buffer
44	EDTWRBSZ	4	Size of wrap buffer or 0

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Anchor and interface for output processor		
48	EDTWRTLP	4	Address of phase or 0
4C	EDTWRTLG	4	Phase size in hex
50	EDTWRTX	C	Interface definition
50	EDTWRTF	4	Function to be performed
54	EDTWRTS	4	Address of line to be written
58	EDTWRTL	4	Number of bytes to be written
5C	*	20	reserved
	Address of I/O stack		
70	EDTIOSAR	4	Address of I/O stack
74	EDTIOSSZ	4	Size of I/O stack
78	EDTACCR	4	ALET used in PTR dump
	Anchor and interface for data collector IJSDDAT		
80	EDTDATLP	4	Address of phase IJSDDAT or 0
84	EDTDATLG	4	Phase size in hex
88	EDTDATF	4	Function bytes
	Anchor and interface for data converter IJSDCVT		
90	EDTCVTLP	4	Address of phase IJSDCVT or 0
94	EDTCVTLG	4	Phase size in hex or 0
	Anchor for HALT processor IJSDSTP		
98	EDTHLTLP	4	Address of phase IJSDSTP or 0
9C	EDTHLTLG	4	Phase size in hex or 0
	Anchor for trace tables and control blocks		
A0	EDTTRT	4	Address of trace table TRTCB
A4	EDTTRTA	4	Address of actual TRTCB entry
AC	EDTTRTC	4	Number of traces of same type
B0	EDTTCB	4	Address of Trace Control Block Queue TCBCBQ
B4	EDTTCBA	4	Address of active TCBCB
B8	EDTTCE	4	Address of TCBCB extension
BC	EDTTCEA	4	Address of active TCECB
	Output device information Status		
C0	EDTDEV	2	
C0	EDTDFLG1	1	Outdev flag byte 1
	Bits defined in EDTDFLG1		
C0	EDTDVOPN	X'80'	Outdevice is open
C1	EDTDFLG2	1	OUTDEV flag byte 2
	Bits defined in EDTDFLG2		
C1	EDTOUTFT	X'80'	First time switch
C1	EDTISTCK	X'20'	I/O stack not empty
C1	EDTIOSIM	X'10'	I/O interr simulation reqd
C1	EDTOUTDV	X'08'	Unit check from OUTDEV
C1	EDTPBUF	X'04'	Print buffer not empty
C1	EDTDWTCE	X'02'	Wait for channel end
C1	EDTDWTDE	X'01'	Wait for device end

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Device Information		
C2	EDTOUT	6	
C2	EDTOUTF1	1	V*1 flags
			Bits defined in EDTOUTF1
C2	EDTOUTPR	X'80'	Printer is outdevice
C2	EDTOUTMT	X'40'	Tape is outdevice
C2	EDTDEVTR	X'10'	OUTDEV traced by SIO
C2	EDTOUTBU	X'08'	Buffer is outdevice
C2	EDTOUTTD	X'01'	Tape-dump active
C3	EDTOUTF2	1	V*2 flags
			Bits defined in EDTOUTF2
C3	EDTT3480	X'80'	Tape type 3480
C3	EDTT8809	X'40'	Tape type 8809
C3	EDTT9346	X'20'	Tape type 9346
C3	EDTT4248	X'10'	printer 4248
C3	EDTBUFFD	X'08'	buffered printer
C4	EDTOUTAD	2	Physical address of OUTDEV
C4	EDTOUTCH	1	Channel addr of OUTDEV
C6	EDTOUTMD	1	Tape mode
C7	EDTTRID	1	ID of trace in progress
	Preparation area for NEW PSWs and control registers		
C8	EDTTCR8	4	Control register 8
C8	EDTTMC	4	Monitor call mask
CC	EDTTCR9	4	Control register 9
CC	EDTTPER	4	PER control register
CC	EDTTPEM	1	PER event masks
CE	EDTTREG	2	Reg.alter mask
D0	EDTTCR10	4	Control register 10
D0	EDTTBEG	4	PER start address
D4	EDTTCR11	4	Control register 11
D4	EDTTEND	4	PER end address
D8	*	4	reserved
DC	EDTSVEX	1	Saved system mask Ext new PSW
DC	EDTSVEXF	1	
DD	EDTSVSV	1	Saved system mask SVC new PSW
DD	EDTSVSVF	1	
DE	EDTSVPG	1	Saved system mask PGM new PSW
DE	EDTSVPGF	1	
DF	EDTSVIO	1	Saved system mask I/O new PSW
DF	EDTSVIOF	1	

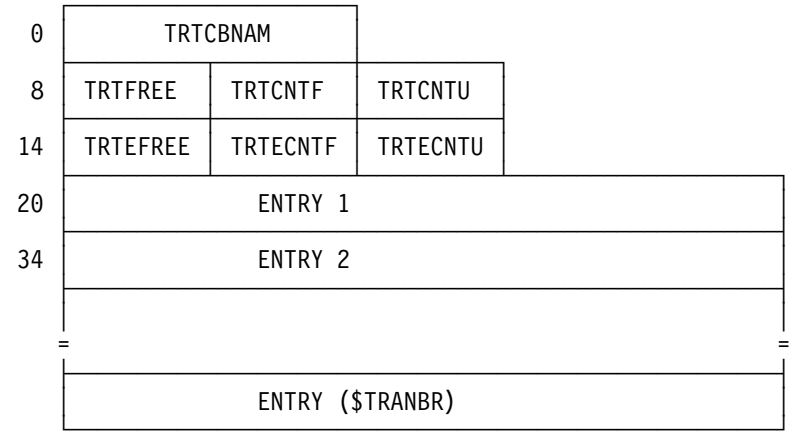
OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Miscellaneous addresses		
E0	EDTSNSD	32	SENSE data
100	EDTSNSl	4	SENSE datalength
104	EDTSNSDV	2	Device address
108	EDTEOS	4	End of supervisor
10C	EDTEOR	4	End of real storage
110	EDTSUBID	4	subsystem-id mask
114	EDTEOV	4	End of virtual storage
118	EDTTIBA	4	Address of TIB area
11C	EDTPUBPT	4	PUB Table address
120	EDTPUBXP	4	PUB Extension PTR of OUTDEV
124	*	C	reserved
130	EDT$ALET	4	ADDR($IJBALET)
134	*	2	reserved
136	EDTDEVTY	1	PUB device type
137	*	1	reserved
			Save area for modified NEW PSWs
138	EDTEXTN	8	EXT NEW PSW save area
140	EDTSVCN	8	SVC NEW PSW save area
148	EDTPGMN	8	PGM NEW PSW save area
150	EDTION	8	I/O NEW PSW save area
158	EDTSEX	8	SDAID EXT wait PSW
15D	EDTSEXA	3	EXT NEW PSW address
	Saved old PSW after interrupt		
160	EDTOPSW	8	Saved old PSW
160	EDTOPSM	1	System mask
161	EDTOPFL1	1	
			Bits defined in EDTOPFL1
161	EDTOPID	xxxx....	Protection key
161	EDTOPCMW	xxxx	CMWP bits
162	EDTOPFL2	1	
162	EDTOPCC	..xx....	Condition code
162	EDTOMSK	xxxx	Program mask
165	EDTOPADD	3	Instruction address
168	EDTPID	2	ID of interrupted task
16A	EDTPID1	2	SYSLOG ID
16C	EDTIOADD	2	I/O address at interrupt
16C	EDTIOAD	2	I/O address at interrupt
16C	EDTIOCH	1	I/O channel address
16D	EDTIOCU	1	I/O cu and device address
16E	EDTSID	2	ID of current space
170	EDTPCB	4	PCB of active partition

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Specified program events		
174	EDTSEV1	1	Code for specified events
			Bits defined in EDTSEV1
174	EDTSEVSV	X'80'	SVC interrupt
174	EDTSEVIO	X'40'	I/O interrupt
174	EDTSEVEX	X'20'	EXT interrupt
174	EDTSEVPG	X'10'	PGM interrupt
174	EDTSEVVT	X'08'	VTAM event
175	*	1	reserved
176	EDTSEV3	1	Code for specified PER event
			Bits defined in EDTSEV3
176	EDTSEVBR	X'80'	Successful branch event
176	EDTSEVIN	X'40'	Instruction fetch event
176	EDTSEVSA	X'20'	Storage alter event
177	EDTSEV4	1	reserved
178	EDTBLKLP	4	Addr of SDAID execution area
17C	EDTBLKLG	4	Size of SDAID area
	CAW save area		
180	EDTCAW	4	CAW save area
184	EDTCCB	4	CCB for IO and SSCH trace
188	EDTCSW	8	CSW save area
189	EDTCSCMD	3	Address of last CCW + 8
18C	EDTCSUST	1	Unit status
18E	EDTCSCNT	2	Residual count
190	EDTCLOCK	8	TOD clock
	Work and save areas		
198	EDTSAV13	4	Register 13 save area
1A0	EDTSAV1	72	Intermodule save area
1E8	EDTSAV2	64	Save area for registers, saved at interrupt time
228	EDTSAV3	64	Save area for control registers

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Preparation area for trace records		
268	EDTTREC	60	Trace record core image
2C8	EDTTREC1	78	Trace record EBCDIC format
340	*	8	reserved
348	EDTWORK	80	Workarea
	Save area for OLD PSWs if a PGM or MC event mixed with		
	A PER event has occurred		
398	EDTPGOLD	8	Save area for PGM OLD PSW
3A0	EDTMCSAV	4	Save area for MC codes
3A4	EDTPGSAV	4	Save area for PGM codes
	Interface to I/O stack		
3A8	EDTIOMAD	4	I/O entry point for MCs
3AC	EDTIOPAD	4	I/O interr call entry
3B0	EDTSTCKM	4	I/O interr addr in PSW
3B4	*	4	reserved
3B8	EDTERRCD	4	OUTDEV I/O error
3BC	*	4	reserved x
3C0	EDTNEMLP	4	Address of phase or 0
3C4	EDTNEMLG	4	Phase size in hex
3C8	*	4	reserved
3CC	EDTTREX	4	Page exception address
3D0	EDTIDALP	4	Printer buffer address
3D4	EDTIDALG	4	Length of print buffer
3D8	EDTSAV4	64	Access registers
417			last byte of EDTCB

TRTCB - SDAID Trace Table

The trace table contains information as to where the trace control blocks to a specific type of event to be traced are located and how many events of this type are to be traced. The trace table consists of a number of entries which is equal to the number of event types that can be traced. Each entry in the trace table is 28 bytes (7 times 4) long and keeps the pointer to the first and last element in the appropriate trace control block queue, the address of the event processor, the number of specified events and the number of occurred events.



OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	TRTCB1	480 (dec)	Trace table structure
0	TRTCBNAM	8	Control block name (TRTCBQ)
8	TRTSTAT	C	Statistic information
8	TRTFREE	4	Address of next free element in TCB-QUEUE
C	TRTCNTF	4	Number of free TCBS in queue
10	TRTCNTU	4	Number of TCBS in use
14	TRTEXT	C	Extension for storage DUMP
14	TRTEFREE	4	Address of next free element in TCE queue.
18	TRTECNTF	4	Number of free elements in TCE queue.
1C	TRTECNTU	4	Number of TCES in use
20	TRTBLK	1C0	Begin of trace table
20	TRTBLK1	1C0	Number of entries depends on variable $TRANBR, use structure TRTCB for mapping the entries
1E0	TRTEND	0	End of trace table

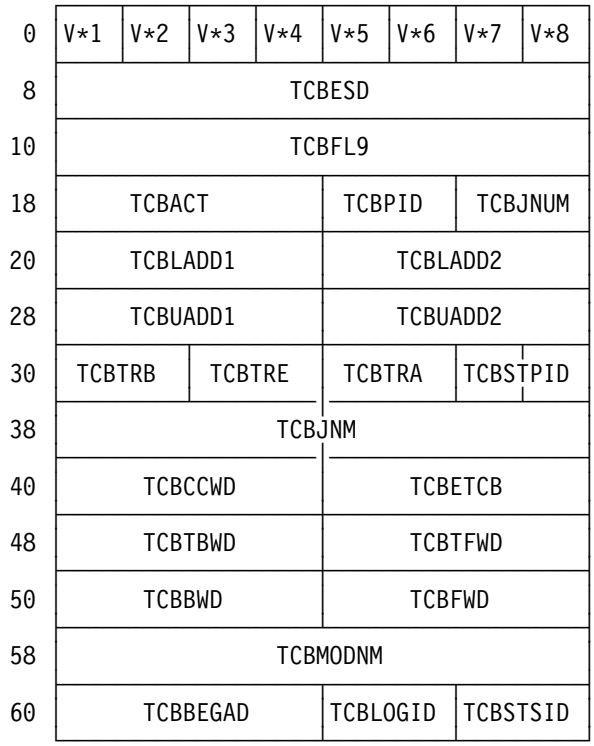
TRTCB1 - SDAID Trace Table Entry

The based structure TRTCB1 describes the format and organization of an entry in the trace table.

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	TRTCB	28 (dec)	Structure of one entry in the trace table TRTCB
0	TRTFXX	4	Address of 1st TCB in chain
4	TRTLXX	4	Address of last TCB in chain
8	TRTCXX	4	Number of TCBS in chain
C	TRTELP	4	Load address of event processor
10	TRTELG	4	Phase size in hex or 0
14	TRTCLP	4	Number of occurred events
18	TRTCLG	4	

TCBCB - SDAID Trace Control Block

The trace control block queue keeps information specified by the TRACE command(s). The trace control block queue consists of a number of entries equal to the number of events that can be specified. The based structure TCBCB describes the format and the organization of an element in the trace control block queue. V*n indicates flag bytes defined in the following lists.



OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	TCBCB		Structure of a trace control block TCBCB
0	TCBTYP	1	V*1 type of TCB
1	TCBFL2	1	V*2 flag byte 2
			Bits defined in TCBFL2
1	TCBFL21	X'80'	TCB is waiting for trigger
1	TCBFL22	X'40'	Trigger event has occurred
1	TCBFL23	X'20'	TCB is implicit specified
1	TCBFL24	X'10'	Stop when event is recorded
1	TCBFL25	X'08'	Occurence counter overflow
1	TCBFL26	X'04'	Occurrence specified
1	TCBFL27	X'02'	TCB is active
1	TCBFL28	X'01'	PER type trace
2	TCBFL3	1	V*3 flag byte 3
			Bits defined in TCBFL3
2	TCBFL31	X'80'	STORJNAM
2	TCBFL32	X'40'	STORJNUM
2	TCBFL33	X'20'	STORDSPN
2	TCBFL34	X'10'	STORAREA=ALL
2	TCBFL35	X'08'	STORAREA=syslog-id
2	TCBFL36	X'04'	STORJNAM is active
2	TCBFL37	X'02'	STORJNAM by default
2	TCBACTMO	X'01'	dump of a phase
3	TCBFL4	1	V*4 flag byte 4
			Bits defined in TCBFL4
3	TCBFL41	1	Number of specified elements in list, for STOR trace the length of the specifid pattern
4	TCBFL5	1	v*5 flag byte 5
			Bits defined in TCBFL5
4	TCBFL51	X'80'	Trace all events of the specified trace type
4	TCBFL52	X'40'	Trace specific I/O units, SVCs, instructions, program phases, monitor calls
4	TCBFL53	X'20'	Trace-specific control units, 1 pattern specified for STOR trace, instruction trace with branches only
4	TCBFL54	X'10'	Trace-specific channels,
4	TCBFL55	X'08'	Option 'ALL' specified for pgml trace

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
4	TCBFL56	X'04'	Option 'REQ' specified for pgml trace
4	TCBFL57	X'02'	Option 'HDL' specified for pgml trace
	Trace area		
5	TCBFL6	1	V*6 flag byte 6
			Bits defined in TCBFL6
5	TCBFL61	X'80'	Trace in user partition
5	TCBFL62	X'40'	Trace in SUPERVISOR
5	TCBFL63	X'20'	Trace in a PHASE
5	TCBFL64	X'10'	Trace in any area ('ALL')
5	TCBFL65	X'08'	Trace job-name
5	TCBFL66	X'04'	Trace job-number
5	TCBFL67	X'02'	jobname tcb is active
5	TCBFL68	X'01'	PER trace with option SUP
6	TCBFL7	1	V*7 flag byte 7
			Bits defined in TCBFL7
6	TCBFL71	X'80'	'ADDRESS' specification
6	TCBFL72	X'40'	'OFFSET' specification
6	TCBUFL11	X'20'	upper offset value is '*'
6	TCBFL75	X'08'	Suppress trace in JCL
6	TCBFLB1	X'08'	Terminate SDAID on event
6	TCBFL77	X'02'	Trace in SVA phase
6	TCBFL78	X'01'	Specified phase is active
	TCB subtype		
7	TCBFL8	1	V*8 flag byte 8
			Bits defined in TCBFL8
7	TCBFL81	X'80'	SVC-TCB for PGMLoad, PGM-TCB for paging trace
7	TCBFL82	X'40'	SVC-TCB for vtamio trace
7	TCBFL83	X'20'	SSCH-TCB for vtamio trace
7	TCBFL84	X'10'	IO-TCB for vtamio trace
	Event specific data		
8	TCBESD	8	multi function field
10	TCBFL9	8	multi function field

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Output specifications		
18	TCBACT	4	Output specifications
18	TCBACT1	1	Flag byte 1
			Bits defined in TCBACT1
18	TCBACTBU	X'80'	Dump wrap buffer
18	TCBACTDP	X'01'	DUMP
19	TCBACT2	1	Flag byte 2
			Bits defined in TCBACT2
19	TCBACTVT	X'80'	Dump VTAM usage buffers
19	TCBACTTB	X'20'	Dump Task Control Blocks
19	TCBACTPB	X'10'	Dump Partition Control Blocks
19	TCBACTLU	X'08'	Dump LUB
19	TCBACTPU	X'04'	Dump PUB
19	TCBACTCQ	X'02'	Dump Channel Queue
19	TCBACTEB	X'01'	Dump Error Block
1A	TCBACT3	1	Flag byte 3
			Bits defined in TCBACT3
1A	TCBACTIR	X'80'	Dump IRB
1A	TCBACTSV	X'40'	Dump SUPERVISOR
1A	TCBACTLO	X'20'	Dump low core
1A	TCBACTCX	X'10'	Dump full CCW data (CCWD)
1A	TCBACTCW	X'08'	Dump Channel program
1A	TCBACTCB	X'04'	Dump CCB
1A	TCBACTSN	X'04'	Dump sense data
1B	TCBACT4	1	Flag byte 4
			Bits defined in TCBACT4
1B	TCBACTPT	X'80'	Dump PTA
1B	TCBACTLT	X'40'	Dump LTA
1B	TCBACTTO	X'20'	Dump TOD
1B	TCBACTSC	X'10'	Dump SYSCOM
1B	TCBACTPC	X'08'	Dump COMRG
1B	TCBACTCR	X'04'	Dump control registers
1B	TCBACTFR	X'02'	Dump floating point registers
1B	TCBACTGR	X'01'	Dump general registers

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Trace area specifications		
1C	TCBPID	2	ID of specified partition
1E	TCBJNUM	2	Job number
20	TCBLADD1	4	Specified lower limit
24	TCBLADD2	4	True lower limit
28	TCBUADD1	4	Specified upper limit
2C	TCBUADD2	4	True upper limit
	Occurrence specifications		
30	TCBTRB	2	Start value
32	TCBTRE	2	Stop value
34	TCBTRA	2	Actual occurrence of event
36	TCBSTPID	2	Stor trace: target PID
	Mixed fields		
38	TCBJNM	8	POWER job name
40	TCBCCWD	4	Value of CCWD
	Chain pointers		
44	TCBETCB	4	Offset to DUMP chain
48	TCBTBWD	4	BWD offset to appended TCB
4C	TCBTFWD	4	FWD offset tp appended TCB
50	TCBBWD	4	BWD offset to master TCB
54	TCBFWD	4	FWD offset to master TCB
58	TCBMODNM	8	phase name
60	TCBBEGAD	4	part begin address
64	TCBLOGID	2	syslog-id
66	TCBSTSID	2	Stor trace: target SID

Chapter 4. SDAID: Diagnostic Aids

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
$$BATTN3	IJSDROT	$$BATTNA IJSDAID	GDTCB SDAID M GDTCB R	MAC LOAD MAC GETVIS MAC LOAD MAC FREEVIS MAC EXCP MAC WAIT MAC EXCP MAC WAIT MSG 4C07I
IJSDROT	IJSDPIF IJSDCT2 IJSDCT3 IJSDPIF IJSDEDT IJSDIOS IJSDRDY IJSDIDT IJSDEDT IJSDTCB IJSDMSG IJSDWRB IJSDMSG	$$BATTN3	GDTCB R IDTCB M IDTCB R SYSCOM M	MAC MESSAGE MAC GETVIS MAC FREEVIS MAC MESSAGE SVC 100 SVC 55 SVC 54 SVC 4 MSG 4C01I MSG 4C04I MSG 4C05I MSG 4C06I MSG 4C07I MSG 4C14I MSG 4C26I MSG 4C28I MSG 4C29I MSG 4C36I MSG 4C37I MSG 4C48I
IJSDAID		JCL		4C40 - 4C61
IJSDIDT		IJSDROT	SYSCOM * R IDTCB M	
IJSDEDT		IJSDROT	GDTCB R	

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDMSG	IOSMOD	IJSDROT IJSDTCB IJSDOUT IJSDTRA IJSDPIF IJSDRDY		
IJSDTCB	IJSDMSG	IJSDROT IJSDTRA	IDTCB R	
IJSDCT2		IJSDROT	IDTCB M	
IJSDCT3		IJSDROT	IDTCB M	
IJSDOUT		IJSDPIF IJSDPIF	CPCB R EDTCB M PUBOWNER * R IORB * M GDTCB M	MAC WAIT MAC MESSAGE SVC 0 MSG 4C09I MSG 4C10I MSG 4C11I MSG 4C12I MSG 4C13I MSG 4C15I MSG 4C27I
IJSDTRA	IJSDTCB IJSDMSG	IJSDPIF IJSDPIF	CPCB R	MSG 4C21I MSG 4C22I MSG 4C23I MSG 4C24I MSG 4C25I MSG 4C30I MSG 4C31I MSG 4C32I MSG 4C33I MSG 4C34I MSG 4C39I

Phase Name	Calls to	Called by	Control Blocks and Data Areas		Macros, SVCs Messages
IJSDPIF	PARSER IJSDOUT IJSDTRA IOSMOD IJSDMSG IJSDOUT IJSDTRA	IJSDROT IJSDROT	CPCB IDTCB	R R	MAC GETVIS MAC MESSAGE MAC FREEVIS MSG 4C02A MSG 4C07I MSG 4C03I
IJSDRDY	IJSDMSG	IJSDROT	TRTCB TCBCB EDTCB	R R R	MSG 4C38I MSG 4C16I MSG 4C17I
IJSDINT	IJSDZBR IJSDZBU IJSDZCA IJSDZEX IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT	IJSDROT	GDTCB EDTCB LOCORE TRTCB TCBCB	R M R R R	
IJSDZBR	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB TRTCB TCBCB	M R R	MAC SDAOCC1 MAC RECOUT
IJSDZBU	IJSDPWB IJSDWRP IJSDSTP	IJSDINT IJSDWRB			MAC SDAOCC1 MAC SDCALL

Phase Name	Calls to	Called by	Control Blocks and Data Areas		Macros, SVCs Messages
IJSDZCA	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB TRTCB TCBCB	M R R	 MAC SDAOCC MAC RECOUT1
IJSDZEX	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB TRTCB TCBCB	M R R	 MAC RECOUT
IJSDZIN	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB TRTCB TCBCB	M R R	MAC SDAEVENT MAC SDAOCC1 MAC RECOUT
IJSDZIO	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT			MAC SDAOCC1 MAC RECOUT
IJSDZMC	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB TRTCB TCBCB	M R R	MAC SDAEVENT MAC SDAOCC1 MAC RECOUT
IJSDZPC	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB TRTCB TCBCB	M R R	MAC SDAEVENT MAC SDAOCC1 MAC RECOUT
IJSDZPL	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT IJSDZSV	FCHWORK * TRTCB TCBCB EDTCB	R M R M	MAC SDAVALID

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDZSA	IJSDDAT IJSDSTP IJSDWRB IJSDWRP IJSDNEM	IJSDINT	EDTCB R TCBCB R TRTCB M	MAC ALETMAC MAC SDAOCC1
IJSDZSI	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TCBCB R TRTCB M	MAC SDAOCC1
IJSDZSV	IJSDZPL IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT		MAC SDAOCC1 MAC RECOUT
IJSDZVT	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT		MAC SDAOCC1 MAC RECOUT
IJSDWRB	IJSDXWRT IJSDPWB IJSDXWRT IJSDZBU	IJSDROT IJSDZBR IJSDZCA IJSDZEX IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT IJSDDAT	EDTCB R TRTCB R	

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDXWRT	-	IJSDZBR IJSDZCA IJSDZEX IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT IJSDCVT IJSDPWB		
IJSDXWRP	-	IJSDZBR IJSDZCA IJSDZEX IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT IJSDCVT IJSDPWB		
IJSDSTP	-	IJSDZBR IJSDZCA IJSDZEX IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT		

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDNEM		IJSDZBR IJSDZIN IJSDZPC IJSDZSA		
IJSDPPR	-	POWER Init & POWER Term.		
IJSDDAT	IJSDWRB IJSDCVT IJSDPWB IJSDWRB	IJSDZBR IJSDZCA IJSDZEX IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT	TIB M COMREG SYSCOM PUBTAB CHQ PUB CCB DEVTRTAB DEVLIST PIB1TAB PIB2TAB TCB LUB PCB TIBATAB	
IJSDCVT	IJSDWRP	IJSDDAT IJSDPWB IJSDPWB		
IJSDPWB	IJSDCVT IJSDWRP IJSDCVT	IJSDZBU IJSDWRB IJSDDAT		
IJSDDEB	IJSDPWB	IJBXSDA IJBXDM5		

Phase Directory: The phase directory is organized by symbolic phase name. It describes the contents of each phase with its subroutine(s) by segment name and function, allowing you to quickly find any desired code.

In the following tables, the phase name appears in the first (leftmost) column. The second column contains an entry-point label, the label of an internal procedure (subroutine or segment). The third column differentiates between entry point (EP), procedures (PR) and segments (SE). The fifth column contains an abstract of the function of the procedure.

Phase Name	Segment Name	Type		Function
IJSDAID	IJSDAID	EP		Builds and submits SDAID commands
$$BATTN3	$$BATTN3	EP		Controls processing of an SDAID command
	SDLOAD	PR		Loads phases IJSDMSG and IJSDROT
	ENDSD	PR		Releases IJSDMSG and IJSDROT
	IOSMOD			
IJSDROT	IJSDROT	EP		Main entry all SDAID commands
	ROTOUTD	PR		Processes the OUTDEV command
	ROTTRA	PR		Processes the TRACE command
	ROTRDY	PR		Processes the READY command
	ROTSDAID	PR		Processes the SDAID command
	ROTSTOP	PR		Processes the STOPSD command
	ROTSTRT	PR		Processes the STARTSD command
	ROTENDSD	PR		Processes the ENDSD command
	ROTFRAL	PR		Releases initialization phases
	ROTGETS	PR		Compute size of a phase
	ROTLOAD	PR		Load a phase into the SVA
	ROTGETV	PR		Gets storage from SVA
	ROTFREV	PR		Releases storage in SVA
	ROTGETR	PR		Gets real storage from page pool
	ROTFRER	PR		Releases storage in page pool
IJSDIDT	IJSDIDT	EP		Builds initialization table
IJSDEDT	IJSDEDT	EP		Allocates execution ctl blocks
IJSDMSG	IJSDMSG	EP		Writes an error message
	MSGUPD	PR		Inserts text in message slots
	MSGLIB	SE		SDAID message library

Phase Name	Segment Name	Type		Function
IJSDTCB	IJSDTCB	EP		Manages TCB queue
	TCBINI	PR		Initializes TCB-QUEUE
	TCBENQ	PR		Enqueues a TCB
	TCBAPP	PR		Appends a TCB
	DAREA			
IJSDCT2	IJSDCT2	EP		Sets up OUTDEV cmd directory
	DOUTDEV	SE		OUTDEV command definition
IJSDCT3	IJSDCT3	EP		Sets up TRACE command directory
	DTRACE	SE		TRACE command definition
IJSDOUT	IJSDOUT	EP		Processes the OUTDEV command
	SDOUT01	PR		Processes primary output device
	SDOUT02	PR		Processes size of wrap around buffer
	SDOUT03	PR		Processes secondary output device
	SDOUT04A	PR		Processes PRINTER|TAPE device address
	SDOUT04C	PR		Check tape device
	SDOUT04D	PR		Retrieve sub channel id
IJSDTRA	IJSDTRA	EP		Processes the TRACE command
	INIT	SE		Initializes work areas
	PARMS	SE		Processes all parameters
	CHKEX	SE		
	CHKTCB			Determines number of TCBs
	ENQTCB	SE		Enqueues all TCBs
	ENQTCE	SE		Enqueues all TCEs
	TRAEMSD	PR		Processes error messages
IJSDPIF	IJSDPIF	EP		Checks command syntax
	PIFRC0	SE		Processes IJPARSER return code 0
	PIFRC4	SE		Processes IJPARSER return code 4
	CONTINU	SE		Enlarges I/O area
	PIFGETV	PR		Gets storage from SVA
	PIFFREV	PR		Releases storage in SVA
	PIFCHK			
	PIFIOF			
	PIFHELP	SE		Processes the HELP function
IJSDRDY	IJSDRDY	EP		Processes the READY command

Phase Name	Segment Name	Type		Function
IJSDINT	IJSDINT	EP		Establishes new PSWs in EDTCB
	SVCENT	EP		Process SVC interruptions
	EXTENT	EP		Process external interruptions
	PGMENT	EP		Process programming interruptions
	COMMON	SE		Closes OUTDEV and returns to SUP
	SDINT	PR		
IJSDZBR	IJSDZBR	EP		Generates Branch trace record
	WRITE	SE		Writes trace record to OUTDEV
	REGGEN2	SE		Converts record to EBCDIC
IJSDZBU	IJSDZBU	EP		Proc. BUFFER OVERFLOW events
IJSDZCA	IJSDZCA	EP		Processes CANCEL events
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZEX	IJSDZEX	EP		Processes EXTERNAL interrupts
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZIN	IJSDZIN	EP		Proc instruction fetch events
	WRITE	SE		Writes trace record to OUTDEV
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZIO	IJSDZIO	EP		Processes I/O events
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZMC	IJSDZMC	EP		Processes monitor call events
	REGGEN1	SE		Generates MC trace record
	WRITE	SE		Writes trace record to OUTDEV
	REGGEN2	SE		Converts trace record to EBCDIC

Phase Name	Segment Name	Type		Function
IJSDZPC	IJSDZPC	EP		Processes program check event
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZPL	IJSDZPL	EP		Processes program load events
	PGMLSVC	SE		Analyzes program load SVC
	PGMLMON	SE		Analyzes program load monitor call
	GETPNM2	SE		Processes loaded phase
	PLADDR	SE		Saves load and end address of phase
	MODDUMP	SE		Saves load and end address for DUMP
	REGGEN	PR		Generates PGML trace record
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZSA	IJSDZSA	EP		Proc. storage alter events
	REGGEN2	PR		Converts trace record to EBCDIC
IJSDZSI	IJSDZSI	EP		Processes SSCH events
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZSV	IJSDZSV	EP		Processes SVC events
	WRITE	SE		Writes trace record to OUTDEV
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDZVT	IJSDZVT	EP		Processes VTAMIO events
	REGGEN1	SE		Generates VTAM trace record
	WRITE	SE		Writes trace record to OUTDEV
	REGGEN2	SE		Converts trace record to EBCDIC
IJSDWRB	IJSDWRB	EP		Writes trace rec. to BUFFER
	OVERPR	SE		Processes buffer overflow
IJSDXWRP	IJSDXWRP	EP		Determines I/O function
	FPUT	PR		Print contents of buffer
	WRCOPEN	PR		OPEN printer device
	WRCLOSE	PR		Closes the printer device
	WRCPUT	PR		Initiates OUTPUT operation
	WRCSSCH	PR		Issues SSCH
	WRCRETRY	PR		Retry failing CCW
	WRCREXT	PR		Wait for external interrupt

Phase Name	Segment Name	Type		Function
IJSDXWRT	IJSDXWRT	EP		Determines I/O function
	WRCOPEN	PR		OPEN tape for SDAID
	WRCPUT	PR		Initiates OUTPUT operation
	WRCSSCH	PR		Issues SSCH
	WRCREXT	PR		Wait for external interrupt
IJSDDAT	IJSDDAT	EP		Collects data for OUTPUT
	DATGREG	PR		Collects GREG data
	DATFREG	PR		Collects FREG data
	DATCREG	PR		Collects CREG data
	DATCOMR	PR		Collects COMREG data
	DATSCOM	PR		Collects SYSCOM data
	DATLTA	PR		Collects LTA data
	DATPTA	PR		Collects PTA data
	DATIRB	PR		Collects IRB data
	DATCCB	PR		Collects CCB data
	DATCCW	PR		Collects CCW and CCWD data
	DATSCAN2	PR		Scan CCW chain for I/O
	DATSCAN	PR		Scan CCW chain for SSCH
	ANALYSE	PR		Analyse CCW chain
	DATTOD	PR		Collects TOD data
	DATLOCO	PR		Collects LOCO data
	DATSUPV	PR		Collects SUPV data
	DATERRBL	PR		Collects ERRBL data
	DATCHQ	PR		Collects CHQ data
	DATPUB	PR		Collects PUB data
	DATLUB	PR		Collects LUB data
	DATPTAB	PR		Collects PTAB data
	DATTTAB	PR		Collects TTAB data
	DATVT1	PR		Collects data for VTAM trace
	DATSNS	PR		Collects SENSE data
	DATVDMP	PR		Collects DUMP data
	DATBUFF	PR		Collects BUFF data
	VALADDR	PR		Checks addressability of data
	DUMPCONT	PR		Process continuation records
	PUTREC	PR		Write one record to printer or buffer

Phase Name	Segment Name	Type		Function
IJSDCVT	IJSDCVT	EP		Converts OUTPUT to EBCDIC
	CVTGREG	SE		processes GREG,AREG,CREG
	CVTFREG	SE		Converts and prints OUTPUT=FREG
	CVTCOMR	SE		Processes blocks < 2k COMREG, SYSCOM, LOWCORE, IRB
	CVTLTA	SE		Processes blocks > 2k LTA,PTA,PUB,LUB,ERBL,CHQ,SUP
	CVTCCB	SE		Converts and prints CCB
	CVTIORB	SE		Converts and prints IORB
	CVTCCW	SE		Converts and prints OUTPUT=(CCW|CCWD)
	CVTTOD	SE		Converts and prints OUTPUT=TOD
	CVTSNS	SE		Converts and prints SENSE data
	CVTTTAB	SE		Converts and prints OUTPUT=TTAB
	CVTVT1	SE		Converts and prints VTAM buffer pool
	CVTVDMP	SE		Converts and prints OUTPUT=DUMP
	CVTMD	SE		Converts and prints OUTPUT=(DUMP PHASE)
	CVTSTOR	PR		Converts and prints data
IJSDPWB	IJSDPWB	EP		Prints wrap buffer and tape
	CVTSV	SE		Converts SVC record to EBCDIC
	CVTPC	SE		Converts PGMCHECK trace record
	CVTIN	SE		Converts INST trace record to EBCDIC
	CVTBR	SE		Converts BR trace record to EBCDIC
	CVTSA	SE		Converts STOR trace record to EBCDIC
	CVTPL	SE		Converts PGML trace record to EBCDIC
	CVTMC	SE		Converts MC trace record to EBCDIC
	CVTIO	SE		Converts IO trace record to EBCDIC
	CVTSI	SE		Converts SSCH trace record to EBCDC
	CVTEX	SE		Converts EXT trace record to EBCDIC
	CVTVT1	SE		Converts VTAM trace records to EBCDIC
	CVTCA	SE		Converts CAN trace record to EBCDIC

Message-to-Phase Cross-Reference List:

MESSAGE NUMBER	PHASE NAME	SEGMENT NAME
4C01I	IJSDROT	ROTSTOP
4C02A	IJSDPIF	PIFRC0
	IJSDAID	IJSDAID
4C03I	IJSDPIF	PIFHELP
4C04I	IJSDROT	IJSDROT
4C05I	IJSDROT	IJSDROT
4C06I	IJSDROT	IJSDROT
4C07I	$$BATTN3	$$BATTN3
	IJSDROT	ROTFREV
		ROTGETS
		ROTGETV
	IJSDPIF	PIFFREV
		PIFGETV
4C08I	IJSDCT3	IJSDCT3
4C09I	IJSDOUT	SDOUT04
4C10I	IJSDOUT	SDOUT04
4C11I	IJSDOUT	SDOUT04
4C12I	IJSDOUT	SDOUT03
4C13I	IJSDOUT	SDOUT04
4C14I	IJSDROT	ROTRDY
4C15I	IJSDOUT	SDOUT04
4C16I	IJSDRDY	IJSDRDY
4C17I	IJSDRDY	IJSDRDY
4C18A	IJSDXWRT	-
4C19I	IJSDXWRP	-
	IJSDXWRT	-
4C20I	IJSDXWRP	-
	IJSDXWRT	-

MESSAGE NUMBER	PHASE NAME	SEGMENT NAME
4C21I	IJSDTRA	SDTRA31
4C22I	IJSDTRA	SDTRA08
4C23I	IJSDTRA	SDTRA16 SDTRA40
4C24I	IJSDTRA	SDTRA17
4C25I	IJSDTRA	SDTRA21
4C26I	IJSDROT	ROTRDY
4C27I	IJSDOUT	SDOUT04
4C28I	IJSDROT	ROTSTRT
4C29I	IJSDROT	ROTSTRT
4C30I	IJSDTRA	SDTRA31
4C31I	IJSDTRA	CHKTCB
4C32I	IJSDTRA	CHKTCB
4C33I	IJSDTRA	CHKTCB
4C34I	IJSDTRA	SDTRA21 SDTRA35 SDTRA38
4C35I	IJSDTRA	-
4C36I	IJSDROT	ROTSTRT
4C37I	IJSDROT	ROTSTOP
4C38I	IJSDRDY	IJSDRDY
4C39I	IJSDTRA	SDTRA35
4C40I	IJSDAID	SDCONS
4C41I	IJSDAID	IJSDAID

MESSAGE NUMBER	PHASE NAME	SEGMENT NAME
4C42I	IJSDAID	IJSDAID
4C44I	IJSDAID	IJSDAID
4C45I	IJSDAID	IJSDAID
4C46I	IJSDAID	IJSDAID
4C47I	IJSDAID	IJSDAID
4C48I	IJSDROT	ROTSTRT
4C49I	IJSDROT	ROTSTRT
4C51I	IJSDAID	BUFFOUT
4C52I	IJSDAID	SDOUTD
4C53I	IJSDAID	SDOUTD
4C54I	IJSDAID	SDOUTD
4C56I	IJSDAID	SDCONS
4C57I	IJSDAID	SDCONS
4C6oI	IJSDAID	IJSDAID
4C61I	IJSDAID	IJSDAID

Chapter 5. Dump and Trace Routines

Introduction

This chapter describes the phases $IJBDCMD, $IJBHDUP, $IJBSDMP, $IJBSINA, and $IJBTRAC. These phases control the system dump, the interactive trace program, and the attention DUMP command. The mentioned phases are SVA-eligible. The IPL program loads them into the Shared Virtual Area.

The phases $IJBSDMP, $IJBHDUP, $IJBTRAC, and $IJBSINA are reentrant. They process the system dump and trace operations for all partitions in parallel.

The phase $IJBDCMD is a serially-reusable resource. It processes the attention DUMP command.

Figure 2 on page 106 shows the relationship between the different dump and trace phases. This general overview is followed by a detailed description of the dump and trace phases and their modules. The third part of this chapter lists the PLX structures, that define the interface areas passed between the different dump and trace modules.

Control Flow between Dump and Trace Phases

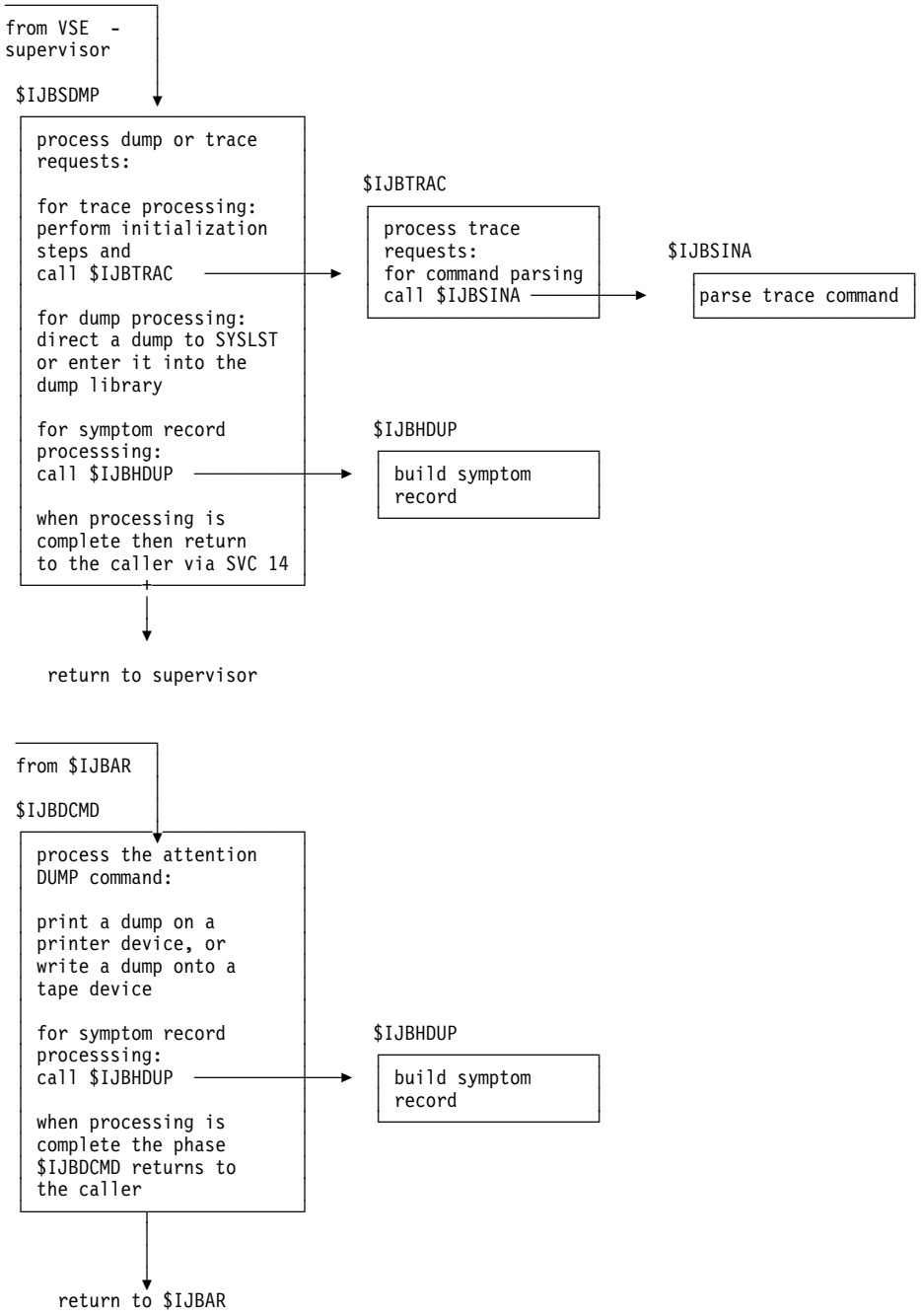


Figure 2. Processing a Dump or Trace Request -- Control-Flow Overview

Phase Descriptions

This chapter describes the modules of the dump and trace phases $IJBSDMP, $IJBHDUP, $IJBTRAC, $IJBSINA, and $IJBDCMD.

Phase $IJBSDMP

The phase $IJBSDMP is the starting point for all system dump operations and all trace operations.

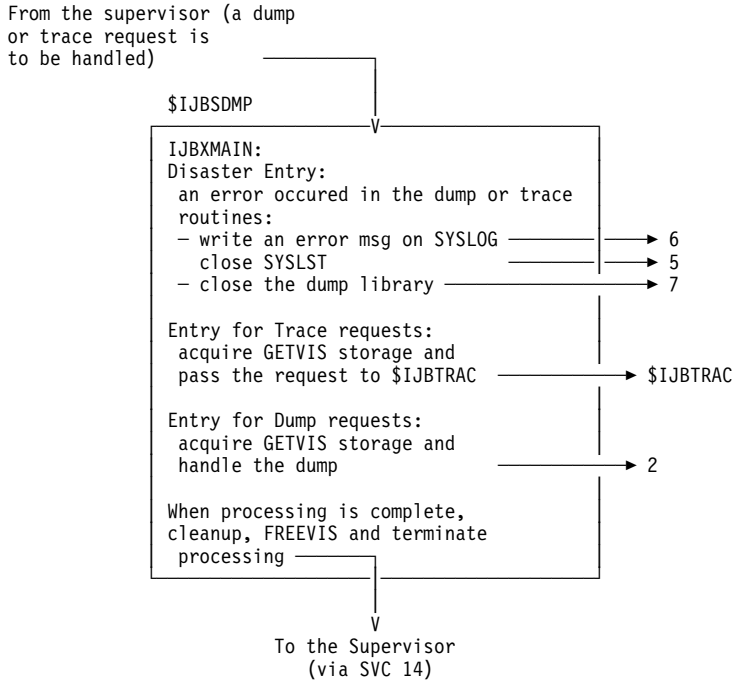
The VSE supervisor passes control to the phase $IJBSDMP if:

- A program issues a dump request macro (DUMP, IDUMP, JDUMP, PDUMP, SDUMP, or SDUMPX).
- A program (or task) ends abnormally, that is it ends before reaching normal end of job (task).
- Job Control processes an EXEC statement with the TRACE option (initialize the trace program)
- A PER interruption occurs (perform tracing)
- A program ends normally or abnormally and the trace program is still active for that partition (terminate tracing)
- An ABEND condition occurs in the dump or tracing routines.

Dump requests: The phase $IJBSDMP processes system dump requests. It writes a cancel message on SYSLOG and generates a storage dump. The amount of dump data for ABEND dumps is controlled by Job Control options. If the supervisor routines calls the phase $IJBSDMP for the processing of an IDUMP, SDUMP, or SDUMPX macro, then the amount of dump output is controlled by submitted input parameters. Details about the dump macros IDUMP, SDUMP, and SDUMPX are given in Chapter 6, “Dump Macros IDUMP, SDUMP, SDUMPX” on page 147. The storage dump is directed to SYSLST or it is entered into the dump library. The phase $IJBSDMP calls the phase $IJBHDUP for symptom record processing.

Trace requests: If the supervisor passes a tracing request to $IJBSDMP, then the phase $IJBSDMP initializes the tracing environment, acquires the required GETVIS space, and passes the tracing request to the phase $IJBTRAC. After the request has been handled, $IJBTRAC returns control to $IJBSDMP.

Abnormal Termination of Dump or Trace Processing: The supervisor calls $IJBSDMP too, if a cancel condition occurs in the dump or trace routines. The phase $IJBSDMP displays a cancel message on SYSLOG, and closes SYSLST and the dump library.



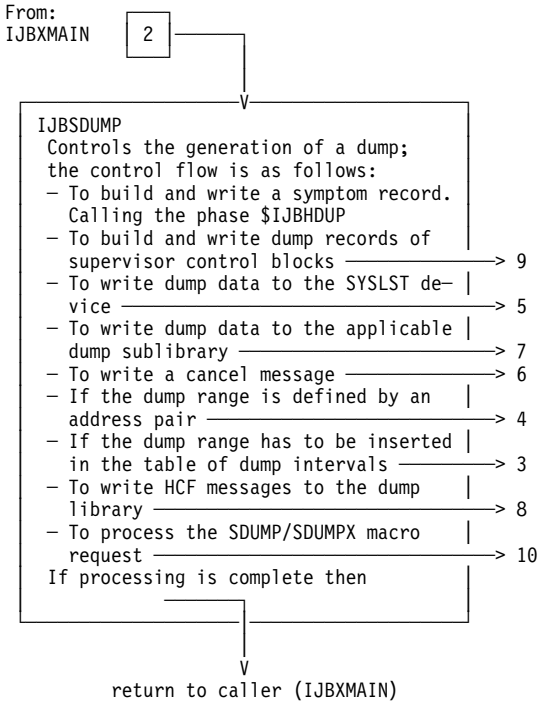


Figure 3 (Part 1 of 5). Control-Flow of phase $IJBSDMP

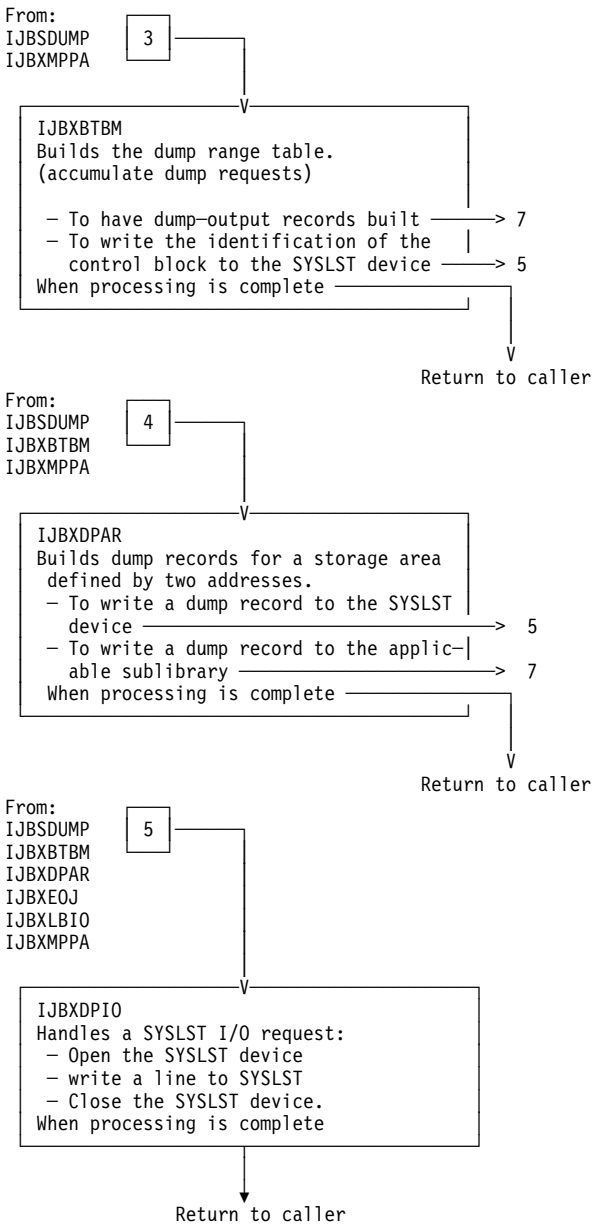


Figure 3 (Part 2 of 5). Control-Flow of phase $IJBSDMP

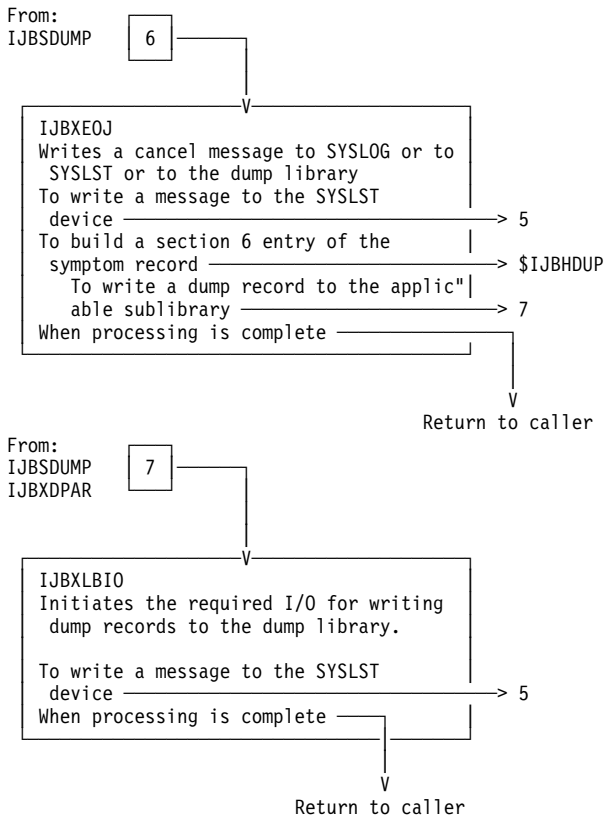


Figure 3 (Part 3 of 5). Control-Flow of phase $IJBSDMP

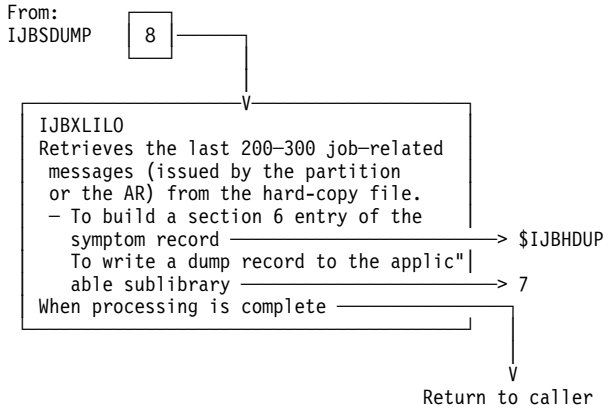


Figure 3 (Part 4 of 5). Control-Flow of phase $IJBSDMP

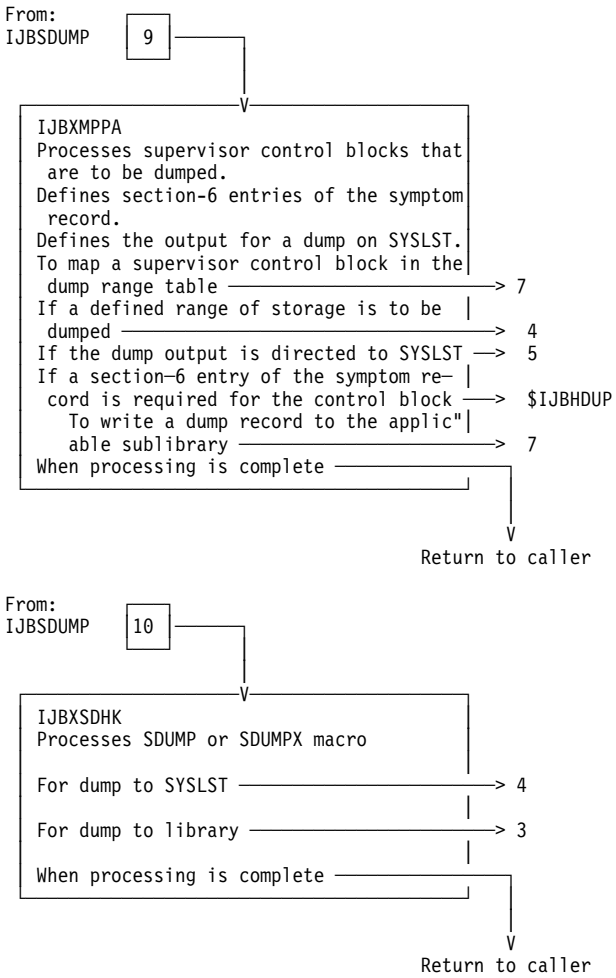


Figure 3 (Part 5 of 5). Control-Flow of phase $IJBSDMP

Phase $IJBTRAC

The phase $IJBTRAC controls the interactive trace program. The interactive trace program (partition debugger) traces user programs interactively on SYSLOG. The phase $IJBTRAC is called from $IJBSDMP and it returns to the caller in all cases. $IJBTRAC initializes and terminates the tracing environment and it handles the PER events.

Trace Initialization: Trace initialization is performed when the JCL routines process an EXEC statement with the TRACE parameter. $IJBTRAC builds the trace table in partition getvis storage, defines an initial set of traces, and updates the tracing fields in the Partition Control Block PCB.

Tracing Requests: The phase $IJBTRAC scans the trace table to find out whether the PER event is to be monitored. It builds a tracing line and displays it on SYSLOG via the WTOR macro. For command parsing $IJBTRAC passes the WTOR response to the phase $IJBSINA to have it analysed. The phase $IJBTRAC handles the DISPLAY, ALTER, and TRACE requests. $IJBTRAC remains in interaction with the console operator until the operator terminates the interaction via a GO command.

Trace Termination: The supervisor sends a trace termination request to $IJBTRAC (via $IJBSDMP), if a main task terminates normally or abnormally, and the trace program is still active. (The trace table pointer PCBATRAX in the partition control block is different from zero).

Trace Triggering: The interactive trace program uses the Program Event Recording (PER) feature of the /390 ESA Architecture to control the execution of selected instructions. When the dispatcher routine selects a traced user program for processing, it posts the PER bit in the PSW and modifies the control registers. The Partition Control Block PCB contains the indication that a partition is to be traced and it contains the values to be entered into the control registers. Control register 9 defines the trace type and control registers 10 and 11 contain the trace start and end address. The supervisor routines and all non-traced programs are dispatched with the PSW PER bit off. This implementation ensures that tracing in one partition has no impact, or only a slight impact, on the execution of programs running in other partitions.

It is not possible to run the interactive trace program concurrent with another program which uses the PER feature. The field IJBPERUS in the system communications region controls the usage of the PER feature. Program using the PER feature should only start execution if the field IJBPERUS contains binary zeros.

The interactive trace program sets the character 'I' into the SYSCOM field before it starts tracing in a partition. It resets the SYSCOM field to binary zeros when the last partition stops tracing. The SDAID program sets the value 'S' into IJBPERUS when it starts an SDAID session with an instruction trace, a branch trace, or a storage alteration trace. SDAID resets IJBPERUS to binary zeros during a STOPSD or an ENDSD command. The interactive trace program and SDAID use the LOCK and UNLOCK macro to avoid concurrent update of IJBPERUS. The DTL has the parameters NAME=IJBXTRAC, LOCKOPT=1, CONTROL=E.

Common Usage of the PER Feature: Any vendor debugging or performance measurement tools which uses the PER feature should join the above agreement. It is desirable that an identifying character, different from 'I' and 'S', be assigned to any performance measurement tool which uses the PER feature. This identifying character should be communicated to the VSE community. The initialization procedure of these programs should test the status of the IJBPERUS field and stop the initialization process if the contents of this field is different from zero. If the field is zero, then the initialization routine enters its identifying character into IJBPERUS before it modifies . the control registers 9 till 11. It is required to reset IJBPERUS to binary zero after the control registers have been reset to their initial value. Any modifications of IJBPERUS should be done under control of the LOCK and UNLOCK macro. The

DTL used for the LOCK and UNLOCK macro has the parameters NAME=IJBXTRAC, LOCKOPT=1, CONTROL=E.

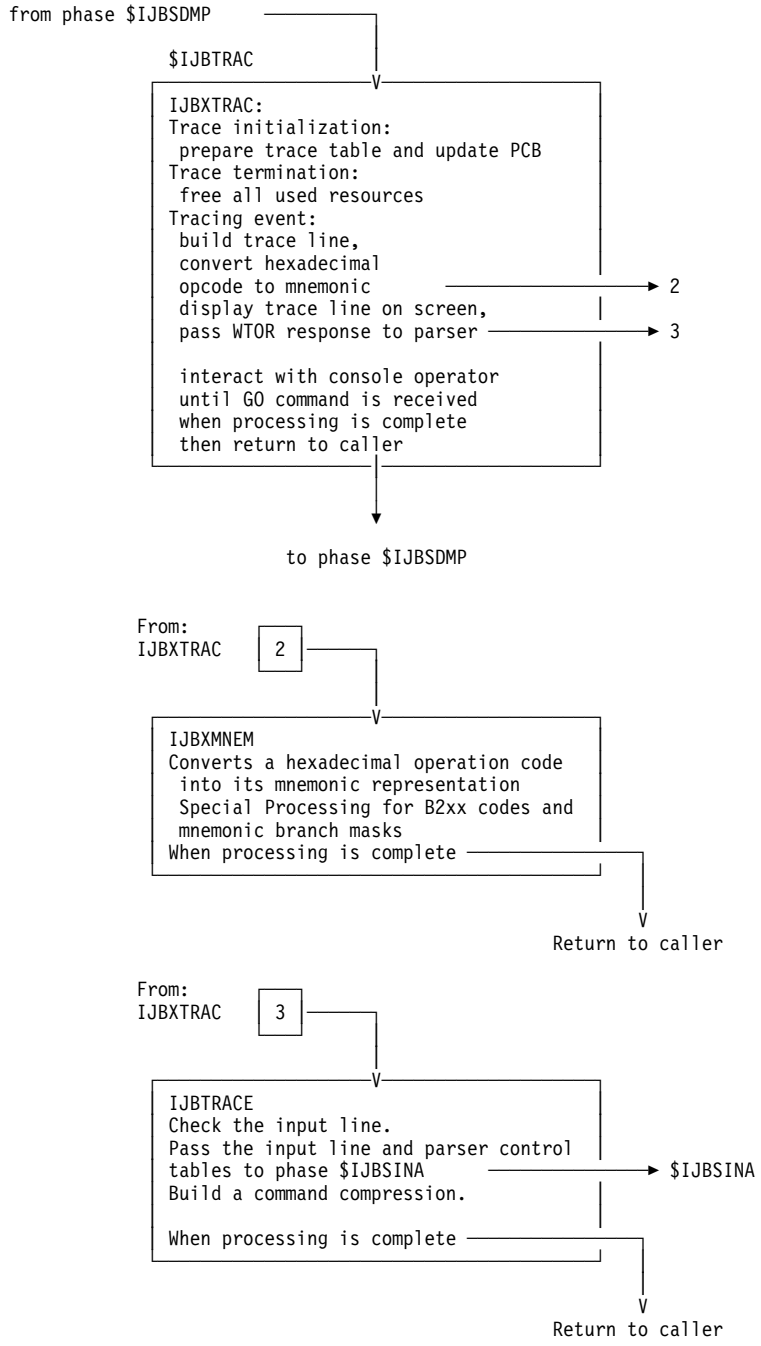


Figure 4. Control-Flow of phase $IJBTRAC

Phase $IJBDCMD

The phase $IJBDCMD processes the attention DUMP command. It writes a storage dump on a physical printer device or on a tape device.

The Attention Routine $IJBAR processes the operator commands. If a DUMP command is to be processed, $IJBAR calls the phase $IJBDCMD. The phase $IJBDCMD processes the DUMP command and returns in all cases to the calling module $IJBAR. The phase $IJBDCMD calls the phase $IJBHDUP for symptom record processing.

For more detailed information of the DUMP command refer to *System Control Statements* manual.

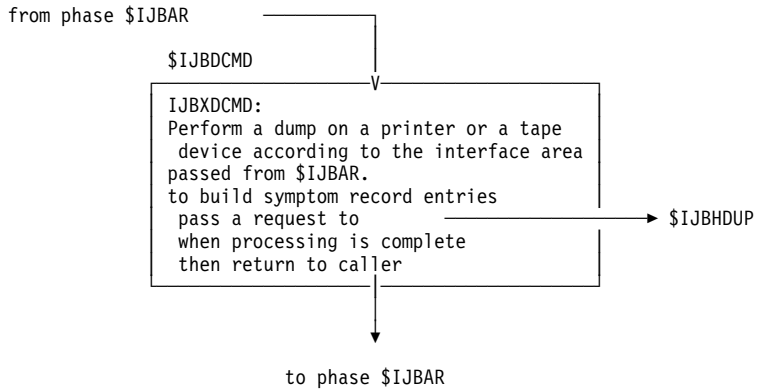


Figure 5. Control-Flow of phase $IJBDCMD

Phase $IJBHDUP

The phase $IJBHDUP prepares the symptom record. The phase $IJBSDMP and the phase $IJBDCMD call $IJBHDUP, if a symptom record is to be built.

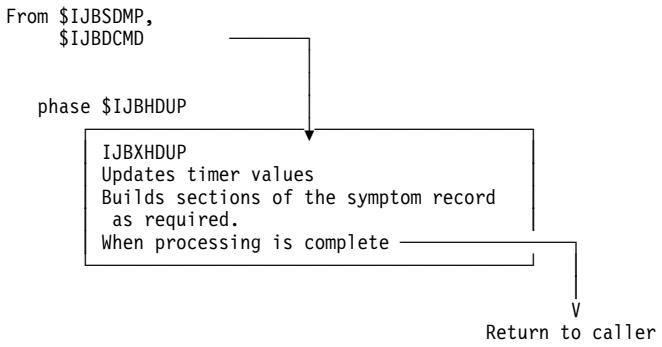


Figure 6. Control-Flow of phase $IJBHDUP

Phase $IJBSINA

The phase $IJBSINA parses the interactive trace commands. It prepares a compression of the trace command and passes this compression back to the calling module IJBXTRAC.

The phase $IJBSINA consists of one module, called IJBXSINA. This module is a building block (reused code) which has been developped for command parsing.

Module Descriptions.

Phase $IJBSDMP – Dump and Trace Processing Routines

The phase $IJBSDMP consists of the modules IJBXMAIN, IJBSDUMP, IJBXBTBM, IJBXDPAR, IJBXDPIO, IJBXEOJ, IJBXLBIO, IJBXLILO, IJBXMPPA, IJBXSDHK, IJBVTSPR, IJBVTSDL. The name of the link book which generates the phase $IJBSDMP is IJBXDMPT.

Following is a description of the modules that make up phase $IJBSDMP. The modules IJBXSDHK, IJBVTSDL, IJBVTSPR may be found in Chapter 6, “Dump Macros IDUMP, SDUMP, SDUMPX” on page 147

Module IJBXMAIN – Dump and Trace Initialization

Entry Point: IJBXMAIN

Function: Initiates dump and trace processing

Called By

- The supervisor's system dump initialization routine.

Modules Called

- IJBXDPIO to close SYSLST (on disaster errors).
- IJBXLBIO to close the dump library (on disaster errors).
- IJBXEOJ to write an ABEND message on SYSLOG.
- IJBSDUMP to process dump requests.

External Routines Called

- Phase $IJBTRAC to process trace requests

Subroutines Used

PREPARE Retrieve VSE fields and call $IJBXEOJ.
CONVBYTS: Converts hex data to printable form.

Data Areas Used

- IJBXSCT6, the IDUMP parameter list IDUMPAR
- IJBXCA, the dump-routine communication area
- MAPCOMR, partition communication area
- MAPPCB, partition control block
- MAPPIB, the program information block (in the supervisor).
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPSAVAR, VSE save area mapping
- MAPTCB, task control block
- SDAGDT, SDAID general data table
- SGLOWC, low core mapping
- SYSCOM, system communication area

Messages Caused: None.

Messages Issued

```
0S09I AN IDUMP MACRO WAS ISSUED
0S10I GETVIS FAILURE IN DUMP ROUTINE. FUNCT.=3
4I02I TRACE INITIALIZATION FAILED
4I11I CDLOAD FOR $IJBTRAC FAILED. RC=
4I15I GETVIS FAILED
4I15I GETVIS FAILED. RC=
4I18I DUMP ERROR. REASON=11
4I18I DUMP ERROR. REASON=12
4I18I TRACE ERROR. REASON=1
4I20I TRACING TERMINATED
```

Input

- Parameter list (MAPSAACM) passed by the supervisor
- Dump start and end address (PDUMP, IDUMP)
- IDUMP parameter list and symptom record (IDUMP)
- List of start and end addresses (SDUMP, SDUMPX)

For a detailed description of the interface areas, refer to this chapter's "Data Area Information" section.

Output: None

Exit: Return control to the supervisor routine via SVC 14

Register Use

0 - 7	Work registers.
8	Base for module data.
9	Base for module code.
12	Pointer to area IJBXCA.
13 - 15	Module-call link registers.

Sequence of Operation

IJBXMAIN (Disaster entry): Call IJBXEOJ to issue cancel message on SYSLOG; Close dump library, close SYSLST; FREEVIS and return to caller

SAATSTRT: Issue GETVIS, load $IJBTRAC and transfers control to it.

SAATPER: Call $IJBTRAC to process tracing request

SAATSTOP: Call $IJBTRAC, Issue FREEVIS.

SAAMCB (No GETVIS available for dump): cancel message on SYSLOG, no dump on SYSLST or Library

Normal Dump entry: Initialize IJBXCA, give cancel message on SYSLOG, call IJBSDUMP for dump processing.

Module IJBSDUMP – Dump Main-Line Module

Entry Point: IJBSDUMP

Function: Controls the generation of a dump.

Called By

- The module IJBXMAIN.

Modules Called

- IJBXBTBM to build and scan the dump range table.
- IJBXDPAR to build storage-dump records.
- IJBXDPIO to write dump output to SYSLST.
- IJBXEOJ to write an ABEND message.
- IJBXLBIO to write dump records to dump sublibrary.
- IJBXLILO to retrieve and dump messages from the Hardcopy file to the Dump Library
- IJBXMPPA to initiate a dump of supervisor control blocks.
- IJBXSDHK to generate the SDUMP or SDUMPX macro.

External Routines Called

- Phase $IJBHDUP to build and write the various sections of the symptom record.

Subroutines Used

DSPNAM: Finds the data space name. ALETMAC macro interface.
S6ENTRY: Builds a section 6 entry.
CONVERT: Converts data to printable form.
PREGS: Dumps registers.
CHECKSVA: Finds the failing phase in the SVA.
PUTPRINT: Calling IJBXDPIO.
INDUMP15: Processes library errors. Input: XCALIBRC.
LIBFINMB: Finishes the dump member.

Data Areas Used

- ADSSR, the dump symptom record mapping macro
- IDUMPAR, the parameter list passed by IDUMP
- IJBXCA, the dump-routine communication area
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP
- IJBXRC, dump record format
- IJBXSCT6, IDUMP mapping macro
- IORB, I/O request block
- SUPAVT, Supervisor address vector table
- SVASVDL. Supervisor SVA subdirectory list
- SYSCOM, system communication area
- MAPCOMR, partition communication area (COMREG)
- MAPPIB, the program information block (in the supervisor).
- MAPSAVAR, VSE save area mapping
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPTCB, task control block
- MAPSVAHD, SVA directory

Messages Caused: None.

Messages Issued

```
0S23I DUMP ROUTINE CANCELED. CANCEL CODE = nn
0S24I AN SDUMP OR SDUMPX MACRO WAS ISSUED
1I51I DUMP COMPLETE
1I58I PHASE $IJBHDUP NOT FOUND
```

Input

- Parameter list (MAPSAACM)
- The parameter list if the dump request was made via the macros.

Output

- The Dump output to the dump sublibrary or to SYSLST.

Exit: Return of control to the calling module (IJBXMAIN).

Register Use

0 - 7 Work registers.
8 Base for module data.
9 - 11 Base for module code.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

IJBSDUMP: Establishes addressability; sets up area IJBXCA; load the address of the phase $IJBHDUP in field XCAHDUP of IJBXCA.

Sets up the controls for dump generation in response to a dump-request macro.

PDUMPROT: Generates the PDUMP macro.

IDUMPROT: Generates the IDUMP macro.

Calling IJBXSDHK Generates the SDUMP or SDUMPX macro.

ABDMPROT: Processes the Abend dump.

SRPROC: Builds the symptom record. Phase $IJBHDUP interface.

CBPROC: Processes the control blocks for Abend dump and IDUMP macro. IJBXMPPA interface.

DSPROC: Processes the data spaces.

EOJROUT: Ends the processing for dump generation.

Module IJBXBTBM – Dump-Range-Table Build and Scan

Entry Point: IJBXBTBM.

Function: Either ensures that all requested pages are dumped in the dump library or scans through section 6 of the IDUMP parameter list.

Called By

- Module IJBSDUMP
- Module IJBXMPPA

Modules Called

- IJBXDPAR to build storage-dump records.
- IJBXDPIO to write the identification of a control block to the SYSLST device.

Subroutines Used: None.

Data Areas Used

- ADSLBD, symptom record section 6
- ADSLBDA, symptom record section 6
- ADSLBDF, symptom record section 6
- ADSLBDL, symptom record section 6

- IJBXCA, the dump-routine communication area
- IJBXRC, dump record format
- IJBXSCT6, IDUMP mapping macro
- MAPCOMR, partition communication area (COMREG)

Messages Caused: None.

Messages Issued: None.

Input

- IJBXCA, the dump communication area (in module IJBSDUMP).
- Request code in field XCABTBM of area IJBXCA.
- Begin address of dump range in field XCABEGAR.
- End address of dump range in field XCAENDAR.

Output

- A list of the dump intervals which to be dumped into the dump library.

Exit: Return of control to the calling module.

Register Use

```
 0 - 3  Work registers.
 4      Base for module data.
 5 - 8  Work registers.
 9 - 10 Base for module code.
11      Work register.
12      Pointer to area IJBXCA.
13 - 15 Module-call link registers.
```

Sequence of Operation

IJBXBTBM: Depending on the function byte XCABTBM in IJBXCA do:

XCABTBM Contains	Name of Routine
XCABTS6	S6LOOP
XCABTAR	SETBT1
XCABTDP	BMTSCAN1

S6LOOP: Scans through section 6 of the symptom record and collects all data ranges. If the dump is directed to SYSLST, writes a title for each area via IJBXDPIO and dumps it via IJBXDPAR.

SETBT1: Inserts the dump requests into a table of dump intervals. A interval always begins on a page boundary. Since the interval length is a multiple of a page an intervall ends 1 bit before a page. The insert routine checks for overlapping dump requests. It is ensured that all requested pages are dumped in ascending order and that no storage page is dumped twice.

BMTSCAN1: Scans the table and sets up the defined address range in XCABEGAR and XCAENDAR for further processing by module IJBXDPAR.

Module IJBXDPAR – Build Storage-Dump Records

Entry Point: IJBXDPAR

Function: Builds storage-dump records for dumping a storage area that is defined by two addresses.

Called By

- Module IJBSDUMP
- Module IJBXBTBM
- Module IJBXMPPA

Modules Called

- IJBXDPIO to write dump output to SYSLST.
- IJBXLBIO to write dump records to dump sublibrary.

Subroutines Used: None.

Data Areas Used

- IJBXCA, the dump-communication area (in module IJBSDUMP).
- IJBXTRTB, character translation table
- MAPCOMR, partition communication area
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- SYSCOM, system communication area

Messages Issued: None.

Input

- Pointers to the start and end addresses in XCABEGAR and XCAENDAR, respectively, of the storage area that is to be dumped.

Output

- A dump of the storage area as defined by the contents of XCABEGAR and XCAENDAR as follows:
  - If the dump is to be written to SYSLST: line by line (this dump output includes page headings as required).
  - If the dump is to be written onto tape or into the applicable dump sublibrary: in blocks of 4K bytes.

Exit Normal: Return of control to the calling module.

Register Use

0 - 9	Work registers.
10	Base for module code.
11	Base for module data.
12	Pointer to area IJBXCA.
13 - 15	Module-call link registers.

Sequence of Operation

IJBXDPAR: Establishes addressability.

Builds dump records for output in one of the following ways:

- Checks the addresses for validity.
- For output to SYSLST:

  The contents of the affected storage area in lines of eight 4-byte words followed by the same 32 bytes either as blanks if their contents are non-printable or as the applicable characters. For identical words and lines, the first line contains the identical word followed by the word --same--; subsequent identical lines are omitted.

- For output into a dump sublibrary:

  The contents of the affected storage area as 4K dump records, each of them containing a page of virtual storage. The specified addresses are adjusted to page boundaries: to the nearest lower for the start address and the nearest higher for the end address.

Module IJBXDPIO – Write Dump Output to SYSLST

Entry point: IJBXDPIO.

Function: Handles a SYSLST I/O request: Opens the device, sets up the channel program, writes a line, closes the device

Called By

- Module IJBSDUMP
- Module IJBXBTBM
- Module IJBXDPAR
- Module IJBXEOJ
- Module IJBXLBIO
- Module IJBXMPPA

Modules Called: None.

Subroutines Used: None.

Data Areas Used

- IJBXCA, the dump-communication area (in module IJBSDUMP).
- SYSCOM, system communication area
- MAPCOMR, partition communication area
- MAPDEVTY, device type codes
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPSAVAR, save area mapping
- IORB, IORB/CCB mapping

Messages Caused: None.

Messages Issued

```
1I51I   DUMP COMPLETE
0S23I   DUMP ROUTINE CANCELED. CANCEL CODE = nn
1I56I   END-OF-VOLUME WHILE DUMPING TO SYSLST
```

Input

- Request code in area IJBXCA (at IOREQ).
- Line to be written as contained in the output buffer.
- Pointer to IJBXCA, the dump communication area.

Output

- The required page header when a new page is to be started
- The line as contained in the buffer when the module receives control
- The flag bits XCALSTOK and XCALSTOP in IJBXCA are set, when the device is open and is ok. During termination of printing the flag bit XCALSTOK is set off.

Exit: Return of control to the calling module.

Register Use

0 - 9 Work registers.
10 Base for module code.
11 Base for module data.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

IJBXDPIO: Determines the operation to be performed by testing byte IOREQ.

OPENROUT: Receives control if IOREQ indicates open the device. Initializes the I/O part of IJBXCA and sets up the correct channel program for performing SYSLST I/O operations.

PRINROUT: Writes one line onto SYSLST, which can be assigned to a printer, to tape, or to FBA disk. Performs I/O at EXCP level.

If SYSLST is assigned to a 3800 printer, sets up the printer for standard output before writing the dump to the device; resets the printer to the original setting on completion of the dump output.

CLOSROUT: Restores SYSLST for normal output.

Module IJBXEOJ – Write-Message Module

Entry Point: IJBXEOJ.

Function: Writes all cancel messages to SYSLOG or to SYSLST or to both.

Called By

- Module IJBSDUMP

Modules Called

- IJBXDPIO to write a message to SYSLST.
- IJBXLBIO to write a symptom record to dump sublibrary.

External Routines Called

- Phase $IJBHDUP to build and write the various sections of the symptom record.

Subroutines Used

LOGGER: To write a message to SYSLOG, to SYSLST, or into the dump library. If the dump is entered into the dump library, a message LBD (Locator Block Definition) is built and included into section 6 of the symptom record.

Data Areas Used

- IJBXCA, the dump-routine communication area
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP
- IJBXRC, dump record format
- IORB, I/O request block
- SYSCOM, system communication area
- MAPCOMR, partition communication area (COMREG)
- MAPDMPIF. additional information for cancel messages
- MAPSAVAR, VSE save area mapping
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPTCB, task control block

Messages Caused: None.

Messages Issued

0P70I	0P80I	0S02I	0S19I	0V08I	0V95I
0P71I	0P81I	0S03I	0S20I	0V09I	0V96I
0P72I	0P82I	0S04I	0S21I	0V10I	
0P73I	0P84I	0S05I	0S22I	0V11I	
0P74I	0P85I	0S06I	0S27I	0V12I	
		0S07I	0S35I		
0P76I	0P86I	0S08I	0V02I	0V13I	
0P77I	0P88I	0S11I	0V03I	0V14I	
0P78I	0P92I	0S12I	0V04I	0V15I	
0P79I	0P93I	0S13I	0V06I	0V16I	
	0P94I	0S14I	0V07I	0V17I	
	0P95I	0S15I		0V18I	
	0S00I	0S16I		0V19I	
	0S01I	0S17I			
		0S18I			

Input

- IJBXCA, the dump communication area.
- Area MAPSAACM (see "Data Areas Used" above).
- The PSW as stored in the partition's save area or, if the cancelation is caused by code residing in the LTA, as stored in the LTA's save area.

Output

- Contents of all registers, unchanged.
- The applicable cancel message (see "Messages Issued" above).

Exit: Return of control to the calling module.

Register Use

0 - 7 Work registers.
8 Base for module data.
9 - 10 Base for module base.
11 Work register.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

IJBXEOJ: Establishes addressability.

Uses the cancel code in field XCARELCC of area IJBXCA to find, via TABLE, the cancel message that is to be written to the SYSLST device.

If the cancel code is one of the following, the message is retrieved from the cancel-code table but is postprocessed. The codes are:

```
X'20':  Processes program-check error-messages in routine PROGCHK.
X'21':  Processes invalid SVC error-messages.
X'22':  Processes phase not found error-messages.
X'26':  Processes devices which are not assigned.
X'27':  Processes devices which are not defined.
X'28':  Processes error messages indicating that the phase to be loaded
        does not fit into the LTA or the applicable partition.
X'29':  Processes invalid library structure.
X'2B':  Processes I/O error during fetch.
X'0B':  Access Violation.
X'3D':  Processes library errors.
X'44':  Processes security error.
X'45':  Processes execution mode violation.
X'46':  Processes invalid DSPSERV or ALESERV macro.
X'47':  ABEND OCCURED. REASON=,ID=
X'48':  MVS MACRO FAILED. ABEND CODE=, REASON=
X'nn':  Processes unknown cancel codes.
```

Module IJBXLBIO – Write Dump Records to Library

Entry Point: IJBXLBIO.

Function: Initiates the I/O operations for a dump that is directed to a dump sublibrary.

Called By

- Module IJBSDUMP
- Module IJBXDPAR

Modules Called

- Module IJBXDPIO to write a message to SYSLST.

Subroutines Used: None.

Data Areas Used

- IJBXCA, the dump-routine communication area
- IJBXRC, dump record format
- IORB, I/O request block
- SYSCOM, system communication area
- MAPCOMR, partition communication area (COMREG)
- MAPSAVAR, VSE save area mapping
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPDEVTY, device types
- SPLIST, SETPRT parameter list
- Librarian control blocks

Messages Caused: None.

Messages Issued

```
0S10I   NO GETVIS FOR LIBRARIAN FUNCTION
0S30I   DUMP STARTED. MEMBER=XXXXXXXX. DUMP IN SUBLIB=xxxxxxx.xxxxxxxx
0S31I   THE LIBRARY DUMP HAS BEEN CANCELED. CANCEL CODE = nn
0S32I   THE LIBRARY DUMP TERMINATED ABNORMALLY
0S33I   LIBDEF STATEMENT IS MISSING FOR THE DUMP LIBRARY
0S34I   DUMP LIBRARY ERROR. FUNCTION = nn
1I49I   DUMP LIBRARY FULL
1I51I   DUMP COMPLETE
```

Input

- Request code in field IOREQ of area IJBXCA.
- A 4K block of dump data (Pointer in field XCABUFAD).

Output

- Dump member name (XCAMEMNM in IJBXCA)
- Return code (XCALIBRC in IJBXCA)

Exit: Return of control to the calling module.

Register Use

```
 0 - 8  Work registers.
 9 - 10 Base for module code.
11      Base for module data.
12      Pointer to area IJBXCA.
13 - 15 Module-call link registers.
```

Sequence of Operation

IJBXLBIO: Establishes addressability within the module and sets up required librarian control blocks.

Gives control to the proper routine in accordance with the function code (IOREQ) passed to this module:

Code	Name of Routine	Code	Name of Routine
XCAOPLIB	OPENLIBR	XCACSMEM	CLOSMEMB
XCANXTNM	NEXTNAME	XCACSLIB	CLOSLIBR
XCAOPMEM	OPENMEMB	XCADELET	PURGMEM
XCAPUTXK	PUTRECRD		

OPENLIBR: The dump library is opened with the following librarian control blocks:

- INLMLAMB (library access method control block)
- LBRACCCB (high level library-access control block)
- LBRACCES (high level library-control-table access control-block)
- INLMLRPL (library-access request parameter list)

For a detailed description of these control blocks, refer to *Diagnosis Reference: Librarian*.

NEXTNAME: Provides the next unique dump member name. Creates the housekeeping member in the dump sublibrary, if this does not exist already, or updates this member.

OPENMEMB: Connects to the member whose name is contained in field XCAMEMNM.

PUTRECRD: Puts a 4K dump record into the connected dump sublibrary.

CLOSMEMB: Disconnects the dump member.

CLOSLIBR: Disconnects the dump sublibrary.

PURGMEM: Deletes a dump member from the directory.

Module IJBXLILO – Hardcopyfile-Message Handling

Entry Point: IJBXLILO

Function: Retrieves job-related Hardcopyfile-Messages (messages that had been issued by the terminating partition and by the AR); if more than 300 messages are available, only the last 200 messages are dumped; otherwise all available messages are dumped; defines section-6 entries (LBDs and Text Extensions) of the symptom record for dumping the Hardcopyfile-Messages into the Dump Library.

Called by

- Module IJBSDUMP.

Modules Called

- IJBXLBIO to write a symptom record to dump sublibrary.

External Routines Called

- Phase $IJBHDUP to build and write the various sections of the symptom record.

Subroutines Used

DUMPIT: Write a HCF Message to the Dump Library (build LBD by calling the phase $IJBHDUP).

Data Areas Used

- CRTGEN (CRTTAB),
- IJBXCA, the dump-routine communication area
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP
- SYSCOM, system communication area
- MAPCOMR, partition communication area (COMREG)
- MAPHCFRC, Hard Copy File record

Messages Caused or Issued: None.

Input: IJBXCA, the dump communication area

Output

- Parameter list IJBXPARM when calling phase $IJBHDUP.
- Hardcopyfile-Messages as section-6 entries.

Exit: Return to the calling module.

Register Use

0 - 9	Work registers.
10	Base for module code.
11	Base for module data.
12	Pointer to area IJBXCA.
13 - 15	Module-call link registers.

Sequence of Operation

IJBXLILO: Establishes addressability; initializes control data; checks the Hardcopyfile environment.
Counter Loop: Counts the HCF Messages from the logical end of the HCF towards the begin.
Print Loop: Prints the Messages (build LBD entries for each message) according to the counting results.
NORMEXIT: Processes end of module matters.

Error Handler: Handles errors occurring during processing (add an error message to the LBD).

Module IJBXMPPA – Supervisor Control-Block Dump Initiation

Entry Point: IJBXMPPA

Function: Processes supervisor control blocks that are to be dumped; defines section-6 entries (link fields) of the symptom record for a dump of storage into the dump sublibrary; defines the output for a dump on SYSLST.

Called by

- Module IJBSDUMP.

Modules Called

- IJBXBTBM to fill the dump-range table.
- IJBXDPAR to build storage-dump records.
- IJBXDPIO to write dump lines to SYSLST.
- IJBXLBIO to write a symptom record to dump sublibrary.

External Routines Called

- Phase $IJBHDUP to build and write the various sections of the symptom record.

Subroutines Used: None.

Data Areas Used

For information retrieval:
- IJBXCA, the dump-routine communication area
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP
- IJBXRC, dump record format
- INLCLPT, the Library pointer table (in the librarian).
- INLCLOT, the Library offset table (in the librarian).
- MAPCOMR, the partition communication region (in the supervisor).
- MAPDIBF, the FBA disk information block (in the supervisor).
- MAPPCB, the partition control block (in the supervisor).
- MAPPIB, the program information block (in the supervisor).
- MAPPUB, the physical unit block table (in the supervisor).
- MAPRFTAB, the Recorder-file table (in the supervisor).
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPSCB, space control block
- MAPDSCB, data space control block
- MAPTCB, task control block
- MAPTIB, the task information block (in the supervisor).
- SDAGDT, SDAID global data table.
- SGLOWC, the low core area
- SYSCOM, the system communication region (in the supervisor).

Modified by this module:
- IJBXCA, the dump-routine communication area
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP

Messages Caused or Issued: None.

Input: Pointers in the dump-routine communication area IJBXCA (see "Register Use" below).

Output

- Pointers passed to the module on input, unchanged.
- Parameter list IJBXPARM if the dump is to be written to the dump sublibrary. The list contains the name of the control block to be dumped, its address, and its length.
- Begin and end address of control block to be processed by IJBXBTBM.

Exit: Return to the calling module.

Register Use

0 - 3 Work registers.
4 Base for module data.
5 - 6 Work registers.
7 Base for module code.
8 Work registers.
9 Base for module code.
10 Work register.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

IJBXMPPA: Establishes addressability; initializes control data; processes all supervisor control blocks whose contents have to be dumped.

CBPROC: Sets up the parameter list for a call of phase $IJBHDUP to build section 6 entries of the symptom record.

PRHEAD: Builds the header of the control blocks for a dump output to SYSLST.

PRHDDMP: Builds the header for a dump output to SYSLST only when OPTION DUMP.

DUMPCB: If the dump is directed to library the module IJBXBTBM is called to fill the dump-range table. If the dump is directed to SYSLST the module IJBXDPAR is called to dump the control block.

Modules IJBXSDHK, IJBVTSPR, IJBVTSDL

A detailed description of these modules may be found in chapter Chapter 6, “Dump Macros IDUMP, SDUMP, SDUMPX” on page 147

Phase $IJBTRAC – Trace Processing Routines

The phase $IJBTRAC consists of three modules, called IJBXTRAC, IJBXMNEM, and IJBTRACE and some syntax tables. The name of the link book which generates the phase $IJBTRAC is IJBLNKTR.

Following is a description of the modules that make up phase $IJBTRAC.

Module IJBXTRAC – Trace Routine

Entry Point: IJBXTRAC

Function:

- initialize the tracing environment
- process a PER event .
- terminate the tracing environment

Called By

- The phase $IJBSDMP (module IJBXMAIN).

Modules Called

- IJBXMNEM to convert an opcode into its mnemonic representation
- IJBTRACE to parse the operator response (to the WTOR macro)

External Routines Called: None.

Subroutines Used

TRACROUT Build a trace line
TARGETRT: Evaluate address part of instruction
CONVERT: Convert words from hex into EBCDIC
CONVBYTS: Converts single bytes into EBCDIC
WTOROUT: Display one line on SYSLOG via WTO

Data Areas Used

- IJBXCA, the dump-routine communication area
- IORB, Input/Output Request Block
- MAPCOMR, partition communication area (COMREG)
- MAPSAVAR, VSE save area mapping
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPPCB, Partition Control Block
- SDAGDT, SDAID General Data Table
- SGLOWC, Low Core Mapping
- SYSCOM, system communication area

Messages Caused

- 4I01I - 4I19I
- 4I41D - 4I49I

Input

- Parameter list (MAPSAACM) passed by the supervisor

For a detailed description of the interface area, refer to this chapter's "Data Area Information" section.

Output:

- Trace lines on SYSLOG or SYSLST
- Storage data and register contents on SYSLOG

Exit: Return of control to the calling module ($IJBSDMP).

Register Use

0 - 7 Work registers.
8- 10 Base for module code.
11 Base for module data.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

When (SAATSTRT) Prepare trace table, update PCB and return

When (SAATSTOP) Reset tracing environment and return

When (SAATPER) Check trace table whether event is to be traced, Prepare trace line, write it to operator, handle responses, and return to caller

Module IJBXMNEM – Operation Code Conversion

Entry Point: IJBXMNEM

Function:

- convert the /390 operation code from the hexadecimal form into its mnemonic representation.

Called By

- module IJBXTRAC

Modules Called: None

External Routines Called: None.

Data Areas Used

- OPCODTAB table of operation codes,
- B2XESA table of B2xx sub codes,
- BCCODE mnemonics for Branch instructions
- BRCODE mnemonics for BCR instructions

Messages Caused: None

Input

- address of op code field
- address where mnemonic is to be stored

Output:

- A 5-byte mnemonic code
- a flag byte describing the instruction type

Exit: Return of control to the calling module (IJBXTRAC).

Syntax Tables: Ten syntax tables, named PTRACE, DEFINE, LEFINE, DELETE, DISPLAY, XISPLAY, LISPLAY, ALTER, XLTER, GO define the syntax of the user entered trace commands. The module IJBTRACE passes the address of these tables and the address of the WTOR response to the parser phase $IJBSINA. The parser phase uses the syntax tables to check the validity of the user entered commands.

Module IJBTRACE – Trace Command Parser Routine

Entry Point: IJBTRACE

Function:

- parse the trace command by calling the BBX parser

Called By

- module IJBXTRAC

Modules Called: $IJBSINA

External Routines Called: None.

Subroutines Used

DoParsing	Call the BBX Parser to analyze the command
ContHdlr	Continuation handler - interface to the BBX parser
ProcessDefine	Process the DEFINE command
ProcessLefine	Process the LEFINE command (Define with address length)
ProcessDelete	Process the TRACE END command
ProcessDisplay	Handle the DISPLAY command
ProcessXisplay	Handle the XISPLAY command
ProcessLisplay	Handle the LISPLAY command
ProcessAlter	Handle the ALTER command
ProcessXlter	Handle the XLTER command
PackHex	Pack a Hex string
ProcessGo	Handle the GO command
Message	Call WTO to send a message

Data Areas Used

- IJBXCA, the dump routine communication area
- Syntax tables

  Symbols and Assembly Tables generated by BPQPARS:
  - PTRACE command table
  - DEFINE operand table
  - LEFINE operand table
  - DELETE operand table
  - DISPLAY operand table
  - XISPLAY operand table
  - LISPLAY operand table
  - ALTER operand table
  - XLTER operand table
  - GO operand table
- Feedback Table, interface block to IJBXTRAC

Messages Caused

- 4I22I - 4I36I

Input: The user's command input The command inputs recognized by this module are:

-
```
TRace {BRANCH|INST|STOR|ABEND} ADDRess={address | address.length
                                        address1:address2}
```

  This command is used to DEFINE a trace. The name of this command is DEFINE. The related control blocks and subroutines contain the word DEFINE or DEF as prefix.

  Because the building block parser BBX PARSER has difficulties to recognize all kinds of operands. This module will change the user command input to a BBX PARSER appropriate command. The modified command name is called LEFINE (define with an address length.) The related control blocks and subroutines contain the word LEFINE or LEF as prefix.
-
```
TRace END {n|ALL}
```

  This command is used to DELETE a trace or all traces. The name of this command is DELETE.

  The related control blocks and subroutines contain the word DELETE or DEL.
-
```
DISPlay {address | address.length | address1:address2}
DISPlay {GR | Gn |FR}
DISPlay PSW
```

  This command is used to DISPLAY the addresses, registers or PSW

  DISPLAY is the command name to display the registers and PSW. The related control blocks and subroutines contain the word DISPLAY or DISP.

  The modified command name is called XISPLAY (display an address and/or address range.)

  The related control blocks and subroutines contain the word XISPLAY or XISP as prefix.

  The modified command name is called LISPLAY (display with an address length.)

  The related control blocks and subroutines contain the word LISPLAY or LISP as prefix.
-
```
ALTer address DATA=data
ALTer  Gn  DATA=data
```

  This command is used to ALTER an address or register content.

  ALTER is the command name to alter the register.

  The related control blocks and subroutines contain the word ALTER or ALT.

  The modified command name is called BLTER (alter an address content.)

  The related control blocks and subroutines contain the word XLTER or XLT as prefix.
-
```
GO  address OUTPut=SYSLST OPTion={DUMP|PARTDUMP|NODUMP}
```

  This command is used to reactivates the stoped user program. The name of this command is GO.

  The related control blocks and subroutines contain the word GO.
-
```
QUERY
```

The QUERY command is part of the PTRACE command table and does not exist therefore separately as an operand table.

Output:

- The Partition Trace Information control block (also know as Feedback Information Area which is based in the IJBXCA)

Exit: Return of control to the calling module (IJBXTRAC).

Register Use

0 - 7 Work registers.
8- 10 Base for module code.
11 Base for module data.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

- Initialize the BBX (Building Block Language Extension) environment.
- Intercept the trace command and pass the command input to the BBX parser.
- Analyze the output, perform additional semantic checking, update the Partition Trace Infrmation control block (also know as Feedback Information Area) and pass the control back to the caller (IJBXTRAC).

Phase $IJBDCMD - DUMP Command Processing

The phase $IJBDCMD is called from the attention routine $IJBAR. The phase $IJBDCMD is made up from one module, called IJBXDCMD.

Module IJBXDCMD

Entry Point: IJBXDCMD

Function: Initiates the generation of a dump in accordance with the DUMP command parameter (IJBXDMPC) passed to this module.

Called By

- The operator command processor $IJBAR.

Modules Called: None.

External Routines Called

- Phase $IJBHDUP to build and write the various sections of the symptom record.

Data Areas Used

- ADSSR, the dump symptom record mapping macro
- IJBXCA, the dump-routine communication area
- IJBXDMPC, the parameter list passed by $IJBAR
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP
- IJBXRC, dump record format
- IJBXTRTB, character translation table
- IORB, I/O request block
- MAPCOMR, partition communication area (COMREG)
- MAPDNTRY, CIL directory entry
- MAPPIB, the program information block (in the supervisor).
- MAPSAVAR, VSE save area mapping
- MAPTCB, task control block
- MAPSVAHD, SVA directory
- SDAGDT, SDAID global data table.
- SUPAVT, Supervisor address vector table
- SVASVDL. Supervisor SVA subdirectory list
- SYSCOM, system communication area

Messages Caused: None.

Messages Issued

```
1I38I SPACE NOT ACTIVE
1I43I SDAID BUFFER NOT AVAILABLE
1I44I SYSLOG-ID OR SPACE-ID XX NOT AVAILABLE
1I46I INVALID DUMP DEVICE
1I51I DUMP COMPLETE
1I52I DUMP COMMAND CANCELED BY OPERATOR
1I53D CUU ASSIGNED TO XX. TO USE TAPE REPLY YES
1I54I END-OF-VOLUME ON DUMP TAPE
1I58I PHASE $IJBHDUP NOT FOUND
1I59D ENTER PHASE NAME, SVA24, GETVIS24, SVA31, GETVIS31 OR ALL
1I60I SPACE-ID MISSING FOR DUMP INTERVAL IN PRIVATE SPACE
1I61I PHASE NOT FOUND IN SVA
1I62I INVALID DUMP INTERVAL
1I63I DATA SPACE NOT FOUND
1I64I SPECIFIED AREA NOT AVAILABLE
```

Input

- Pointer of parameter list (IJBXDMPC) in register 1 passed by the phase $IJBAR.

Output

- The Dump output to a tape or a printer.

Exit: Return of control to the calling module (Attention routine).

Register Use

```
 0      Work register.
 1      Pointer of parameter list IJBXDMPC
 2 - 7  Work registers.
 8 - 11 Base for module code.
12      Pointer to area IJBXCA.
13 - 15 Module-call link registers.
```

Sequence of Operation

IJBXDCMD:

- Establishes addressability;
- Sets up area IJBXCA;
- Checks for the correctness of the parameters;
- Loads the address of the phase $IJBHDUP in field XCAHDUP of IJBXCA;
- Sets up the control for dump generation in response to the parameter list of the DUMP command;
- Initializes the printer or tape;
- Processes the symptom record;
- Dumping of the area which is specified;

Subroutines

S6ENTRY: Builds a section 6 entry (Calling of the phase $IJBHDUP).
CONVERT: Converts hex data into printable characters.
CONVBYTE: Converts single bytes in printable characters.
PREGS: Dumps registers.
PUTTAPE: Puts a 4K record to the tape.
PUTPRINT: Prints a line on a printer device.
PUTAREA: Processes a block of pages (PAGESTAT).
CVTSTOR: Writes a block of pages on printer.
BLOCKPUT: Writes a block of pages on tape.

Phase $IJBHDUP – Symptom-Record Processing

The phase $IJBHDUP is called by the phase $IJBSDMP and the phase $IJBDCMD which processes the Attention DUMP command. The phase $IJBHDUP is made up from one module, called IJBXHDUP.

Module IJBXHDUP – Symptom Record Processing

Entry Point: IJBXHDUP.

Function: Builds sections of the symptom record as required.

Called By

- Module IJBSDUMP
- Module IJBXEOJ
- Module IJBXLILO
- Module IJBXMPPA
- Phase $IJBDCMD

Modules Called: None.

Subroutines Used: None.

Data Areas Used

- ADSLBD, symptom record section 6
- ADSLBXA, symptom record section 6
- ADSLBXX, symptom record section 6
- ADSLBXT, symptom record section 6
- ADSSR, the dump symptom record mapping macro
- IJBXCA, the dump-routine communication area
- IJBXPARM, the parameter list to be passed to phase $IJBHDUP
- IJBXRC, dump record format
- IJBXSCT6, the parameter list passed by IDUMP
- MAPCOMR, partition communication area (COMREG)
- MAPSAVAR, VSE save area mapping
- MAPSAACM, the interface area between Supervisor and $IJBSDMP
- MAPTCB, task control block
- SUPAVT, Supervisor address vector table
- SVASVDL. Supervisor SVA subdirectory list
- SYSCOM, system communication area

Messages Caused: None.

Messages Issued: None.

Input

- IJBXCA, the dump communication area.
- IJBXPARM, a parameter list.

Output

- Contents of all registers, unchanged.
- Symptom record sections 1 through 3 and 6.
- Timer fields in IJBXCA

Exit: Return of control to the calling module.

Register Use

0 - 8 Work registers.
9 Base for module data.
10 - 11 Base for module code.
12 Pointer to area IJBXCA.
13 - 15 Module-call link registers.

Sequence of Operation

IJBXHDUP: Establishes addressability of the symptom record and initializes control areas. Passes control to the proper routine in accordance with the function request contained in field CAHDREQ of area IJBXCA:

CAHDREQ Contains	Name of Routine	Contains	Name of Routine
XCATIMES	TIMEROUT	XCAIN5	S5FILLER
XCAIN1C	ADDRCH	XCAS6	LBDFILL
XCAIN1	S1FILLER	XCAWRO	BUFWRITE
XCAIN3	S3PROC		

TIMEROUT: Fill the timer fields in IJBXCA for the heading line.

ADDRCH: Checks the validity of the IDUMP-macro provided addresses in section 2 of the symptom record.

S1FILLER: Completes section 1 of the symptom record.

S3PROC: Establishes the addressability of section 3 of the symptom record. Updates section 2 of this record.

REGDIFF: Calculates the symptom REGS/DIFF.
PROGCH: Provides the symptom information for a program check.
INVALPHS: Provides the symptom information for a phase-not-found condition.
INVALSVC: Provides the symptom information for an invalid SVC.
INVALSYS: Provides the symptom information for a logical unit not assigned.
RIDSSYM: Builds the symptom RIDS/AAAAAAAA.
ADDRSYM: Builds the symptom ADDR/XXXXXXXX.
OFFSSYM: Builds the symptom OFFS/XXXXXXXX.

S5FILLER: Builds section 5 and establishes the addressability of section 5 of the symptom record. Updates section 2 of this record.

LBDFILL: Establishes addressability for section 6 entries of the symptom record. Passes control to the proper routine in accordance with the content of field CBACTION in area IJBXPARM:

CBACTION Contains	Name of Routine	Contains	Name of Routine
CBLBD	LBDBODY	CBTEXT	TXTENTRY
CBLBTXT	LBDBODY	CBREGS	DMPREGS
CBLBXA	LBDBODY		

LBDBODY: Builds the main LBD entry.
TXTENTRY: Builds the LBD entry for a text extension of an existing LBD entry.
TXTOUT: Handles a buffer-full condition for text extensions.
DMPREGS: Collects LBD entries with hex data extensions for the registers and the PSW.
HEXPROC: Builds LBD entries with hex data extensions.

BUFWRITE: Writes a 4K buffer-record.

Program Organization Information

This section describes the dump and tracing modules. The list provides, for each module, a brief statement of function, it indicates from where a module may receive control (the Called-By column) and which other module it may call or exit to (the Calls/Exits To column).

Phase $IJBSDMP – System Dump Generation

Module	Called By	Function	Calls
IJBXMAIN	Supervisor routine	Initializes the dump and trace routines. It acquires the GETVIS space and handles internal programming errors.	IJBSDUMP to perform a system dump, phase $IJBTRAC to handle tracing requests, IJBXDPIO to close the SYSLST device, IJBXLBIO to close the dump library, IJBXEOJ to write a Cancel msg on SYSLOG
IJBSDUMP	IJBXMAIN	Initiates the generation of a dump if one of the following occurs: - The system encounters a program-cancel condition. - A main- or a subtask of a program issues a dump-request macro.	IJBXBTBM to fill the dump range table according to the dump range limits and to scan through the table for dumping. IJBXDPAR to dump a range of storage. IJBXDPIO to write a dump line on SYSLST. IJBXEOJ to write a cancel message. $IJBHDUP to build a symptom record section. IJBXLBIO to initiate I/O required for dump output to dump sublibrary. IJBXLILO to write HCF messages to Dump Lib. IJBXMPPA to process the system control blocks.
IJBXBTBM	IJBSDUMP IJBXMPPA	Builds the dump range table. Passes storage–dump range limits to the dump generating modules called by this module.	IJBXDPAR to dump a defined range of storage. IJBXDPIO to write a line to SYSLST. To the calling module when finished.
IJBXDPAR	IJBSDUMP IJBXBTBM	Builds storage-dump records for dumping an area that is	IJBXDPIO to write dump output to SYSLST. IJBXLBIO to write dump records into the dump sublibrary.
IJBXDPIO	IJBSDUMP IJBXBTBM IJBXDPAR IJBXEOJ IJBXLBIO IJBXMPPA	Writes dump records or messages to SYSLST	

Phase $IJBSDMP -- Dump Generation (continued)

Module	Called by	Function	Calls
IJBXEOJ	IJBSDUMP	Writes a cancel message to SYSLOG, to SYSLST or to the dump sublibrary.	$IJBHDUP to process a section of the symptom record. IJBXDPIO to write a message to SYSLST.
IJBXLBIO	IJBXDPAR IJBSDUMP	Initiates the necessary I/O for writing a dump output to the accessible dump sublibrary.	IJBXDPIO to write a message to SYSLST. The librarian program
IJBXLILO	IJBSDUMP	Retrieves the last 200-300 job-related messages from the Hardcopyfile and dumps them in the symptom record into the dump sublibrary.	$IJBHDUP to process section 6 of a symptom record.
IJBXMPPA	IJBSDUMP	Processes system control blocks that are to be dumped. Defines section-6 entries of the symptom record for a dump output into a dump sublibrary. Builds header line for a dump on SYSLST. When OPTION DUMP some control blocks are dumped on SYSLST.	IJBXBTBM to fill the bit map table according to dump range limits. IJBXDPAR to dump a defined range of storage. IJBXDPIO to write dump output to SYSLST. $IJBHDUP to process a section of a symptom record.
IJBXSDHK	IJBSDUMP	hooks the SDUMP/X function (SVC 123) to the VSE/ESA Dump Services	depends on the function code passed by IJBXMAIN it calls then IJBVTSPR, or IJBVTSDL correspondently.
IJBVTSPR	IJBXSDHK	has the following functions: 1. copy the SDUMP/X parmlist, ID, HDR text, and the storage list 2. parse the LISTD, SUMLIST/ SUMLSTL option.	
IJBVTSDL	IJBXSDHK	builds the address range table for the requested storage or data space ranges.	IJBXBTBM is called to dump the storage whose addresses appear in address range table.

Phase $IJBTRAC – Trace Processing

Module	Called By	Function	Calls
IJBXTRAC	IJBXMAIN	Build and maintain the trace table, update the PCB, process the PER interruptions build the trace lines, and control the dialog with the console operator	IJBXMNEM to convert a hexadecimal op code into its mnemonic representation IJBTRACE to parse the user responses retrieved via WTOR
IJBXMNEM	IJBXTRAC	Converts a hexadecimal operation code, like D2, into its mnemonic representation, like MVC	
IJBTRACE	IJBXTRAC	Analyses the trace commands entered from the user console	IJBXSINA to check and parse the interactive trace commands

Phase $IJBDCMD – Attention DUMP Command

Module	Called By	Function	Calls
IJBXDCMD	$IJBAR	Process the attention DUMP command; print a dump on printer device or write a dump on a tape device	phase $IJBHDUP to build a symptom record section

Phase $IJBHDUP – Symptom Record Processing Routine

Module	Called By	Function	Calls/ Returns To
IJBXHDUP	IJBSDUMP IJBXEOJ IJBXLILO IJBXMPPA $IJBDCMD	Builds sections of the symptom record as required.	To the calling phase when finished.

Data Area Information

This sub chapter describes the key data areas used by the dump and trace routines to pass information from one module (or phase) to other modules (or phases). Each summary gives the displacements of the area's fields in decimal notation.

IJBXCA – Dump-Routine Communication Area

This area is used by all dump and trace modules for communication. It contains the variables that need to be passed between these modules and phases.

The area is located in and set up by module IJBXMAIN, which stores its address into register 12. All other modules of the dump and trace routines access the area via a based PLX mapping structure.

If IJBXMAIN is called for dump creation, it builds a 'big' interface area IJBXCA (called IJBXCA1). It contains all fields required for dump and trace processing.

If IJBXMAIN is called for trace initialization, it builds a 'small' interface area IJBXCA (called IJBXCA2). It contains only the fields required for trace processing.

IJBXCA is just an internal interface and often subject to change. Thus, it is not described in this manual. Please refer to the program listing of one of the dump or trace modules for closer information.

IJBXDMPC – DUMP Command Communication Area

This DSECT is the interface between the Attention Routine IJBAR and the Dump phase to pass all information of the DUMP command. Its address is passed in register 1 to the dump phase. The layout and contents of the list are:

Displ.

0-3 Start Address of Dump.

4-7 End Address of Dump.

8-9 Area Identifier.

10-11 Dump Device Address.

12 DUMP Command Switch 1 (DPCSW1):

X'80' (DPCSVA) Request dump of SVA.
X'40' (DPCGETV) Request dump of GETVIS area.
X'20' (DPCSUP) Request dump of Supervisor.
X'10' (DPCBUF) Request dump of SDAID Buffer.
X'08' (DPCPART) Indicate Address Area Information supplied.
X'04' (DPCADDR) Indicate Address Range Information supplied.
X'02' (DPCDSP) Request dump of data space.
X'01' (DPCSYSDS) Request dump of system data space (not used).

13-58 Dump Command Text.

59-66 Data space name.

IJBXPARM – Parameter List for Calling Phase $IJBHDUP

When module IJBXHDUP gets control to generate a section-6 entry of the symptom record, the module expects this list at the location pointed to by the address in field XCACBPTR of the area IJBXCA. The layout and contents of the list are:

Displ.

0-7	Name of the control block to be dumped.
8-15	Component identifier.
16-19	Length of the control block.
20-23	Address of the control block.
24	Identifier for the different types of section 6 entries (LBDs).
25	LBD special-handling flag X'80' Do not print the area defined by this LBD. X'40' Do not process this LBD.
26-31	Reserved.
32-35	Number of array elements in the block.
36-39	Address of text to be supplied.
40-43	Length of this text.

IJBXRC – DUMP Record Format

The VSE/AF system dumps, the stand alone dumps, and the dumps written on tape via the DUMP command have a common format. The record length is 4112 bytes. The first full word of any dump record contains an identification field. The identification field may have the following contents:

Displ.

0-3 Dump Record Header.

- `X'00000001'` Symptom record or section 6 record
- `X'00000002'` Dump data record.
- `X'00000004'` End of File Record (SA Dump only).
- `X'00000008'` End of Dump Record (SA Dump only).

Symptom record

0-3 id-field (x'00000001')

4-4099 Symptom Record (first two characters: 'SR')

4100-4111 Reserved.

Section 6 Extension record

0-3 id-field (x'00000001')

4-15 Chaining field (first four characters: 'SCT6')

16-4099 Section 6 records (LBD, LBXT, LBXX)

4100-4111 Reserved.

Dump data records:

0-3 id-field (x'00000002')
4-7 Virtual Address of Data.
8-11 Real Address of Data (if available).
12 Storage Key (if available)
13 Control Flags:
X'10' Storage Key present.
X'08' Real Address present.
X'04' Page without Address Translation.
X'02' Page from Processor Storage.
X'01' Page from Page Data Set.
14-15 Reserved.
16-4111 Dump Data.

Chapter 6. Dump Macros IDUMP, SDUMP, SDUMPX

The IDUMP Macro

The VSE system programs and the subsystems may use the IDUMP macro (internal dump macro) to request a dump and to communicate the symptom string to the VSE dump routines. The IDUMP macro writes the symptom string and the dump into a dump library (if OPTION SYSDUMP is specified) or to SYSLST (if OPTION NOSYSDUMP is specified).

Format of the IDUMP Macro

The format of the IDUMP macro is as follows:

```
IDUMP {addr1|(r1)}
      ,{addr2|(r2)}
      ,{addr3|(r3)}
      [,MFG={param|(S,param)|(r4)}]
```

addr1 This parameter specifies the start address of the area in the issuer's partition that is to be dumped. If the address is lower than the partition start address, it is adjusted to partition start. If it is higher than the partition end address, no area is dumped.

addr2 This parameter specifies the end address of the area to be dumped. If it is higher than the partition end address, it is adjusted to partition end. If it is lower or equal to addr1, no area is dumped.

addr3 The address specified by this parameter points to a parameter list. This parameter list contains dump control information and an address pointer to the symptom record. The parameter list and the symptom record may be located in the issuer's partition or in the system GETVIS area. If the address contains an invalid value, the IDUMP routine builds a dummy symptom record without user supplied data.

MFG This parameter is optional and can be used to provide a 12-byte area where the other three parameters are stored to make the issuing program reentrant.

The IDUMP macro changes the contents of general registers zero and one.

Format of the Parameter List

The macro IJBXSCT6 defines the structure of the parameter list (IDMPAREA).

Offset	Type	Length	Description
0	char	4	control block id (IDMP)
4	ptr	4	address of save area
8	hex	2	length of parameter list (X'0028')
10	hex	1	control flags
11	hex	1	dump flags
12	hex	4	reserved (X'00000000')
16	char	8	library member name
24	hex	10	reserved (hex zeros)
34	hex	2	length of symptom record
36	ptr	4	address of symptom record

- IDMPTYP - Type of IDUMP

  This field is used as a control block identification. It contains the character string 'IDMP' to distinguish the IDUMP macro from the IDUMP macro of former releases.

- IDMPSAV - Address of Save Area

  The user may supply here the address of a save area which is relevant for the analysis of the problem. The mapping macro MAPSAVAR describes the structure of the save areas.

  The specified save area can have one of the following formats:

  1. user save area (SVEARA)
     The user save area contains an 8-byte program name, the 8-byte PSW, a 64-byte field containing the general registers 9 to 8, 8 byte of control information, and a 32-byte area containing the floating point registers.
  2. exit routine save area (SVUARA)
     The exit routine save area contains the 8-byte BC-mode PSW, a 64-byte field containing the general registers 0 to F, the actual PSW of interruption, 72 bytes error information, and a 64-byte field containing the access registers 0 to F.

  The control flag bit IDMPUARA (see below) controls the processing of the save area specified via IDMPSAV.

  If the save area address is zero, then the task's save area is dumped. If the save area address is not equal to zero and the flag bit IDMPUARA is off, then the specified save area is formatted according to the mapping structure SVEARA (as user save area). If the save area address is not equal to zero and the flag bit IDMPUARA is on, then the user task's save area is dumped and an extra control block with the name EXSAVAR is added to the dump file.

- IDMPLENP - Length of this IDUMP parameter list

  Specify a halfword containing the value of 40.

- IDMPFLG1 - Control Flags

  – IDMPNOPT (X'80') - ignore JCL dump option
    If this flag bit is on then the IDUMP routine ignores the JCL dump options DUMP, PARTDUMP, or NODUMP.

  – IDMPRRET (X'40') - request return code
    If this flag bit is on then the IDUMP macro passes a return code to the issuer of the IDUMP macro in register 15. If this flag bit is off, then register 15 remains unchanged.

  – IDMPUARA (X'20') - exit routine save area
    This flag bit controls the format of the specified save area. This flag bit should be on, if the specified save area is an exit routine save area. If the flag bit is off, the dump routines format the save area according to the structure SVEARA.

- Dump flags

  – X'80' dump user partition
  – X'40' dump supervisor
  – X'20' dump supervisor control blocks
  – X'10' dump SVA
    --> if 31-bit mode SVA below and above 16 MB are dumped.
  – X'08' dump system GETVIS space
    --> if 31-bit mode system GETVIS space below and above 16 MB are dumped.
  – X'04' dump dynamic space GETVIS space

- IDMPNAME - Library Member Name

  Dump member names are created by the VSE system. The generated dump name is returned to the issuer of the IDUMP macro in this eight-byte name field. Leave this field blank if you issue an IDUMP macro.

- IDMPLENS - Length of symptom record

  This is the length of the symptom record excluding additional section six extension records, if any.

- IDMPADDS - Address of symptom record

  This is a four-byte address pointer to the area where the issuer of the IDUMP has created the symptom record.

IDUMP Return Codes:

The IDUMP macro changes the contents of registers zero and one, and if a return code is requested, the contents of register 15.

If a return code is requested (bit 1 of the control byte in the IDUMP parameter list is on), it is passed to the issuer of the IDUMP macro in register 15. If the mentioned flag bit is off, register 15 remains unchanged.

The following return codes may be encountered:
0 request completed successfully, or the operator canceled the dump
4 dump library full or dump library not defined
8 library error (I/O error or OPEN/CLOSE error)
12 GETVIS error - GETVIS space not sufficient to handle the dump request

In the case of a non-zero return code the dump fragment is purged from the dump library. Therefore the dump library will not contain incomplete dump members. If the user requests a return code, then the return code is passed to the user in register 15. If the user does not request a return code, register 15 remains unchanged. The VSE dump routines try to print the dump on SYSLST.

The Symptom String

The symptom string consists of the symptom record and the section six extension records.

The symptom record consists of six sections. The first two sections are the fixed part of the symptom record. The size of sections one and two is 120 bytes. Sections three to six are the variable part of the symptom record. The size of sections three to five is restricted by the fact, that the total size of the symptom record cannot exceed 4108 bytes. The size of section six is not limited. It is possible to define any number of section six extension records.

The first section is the environment section. It contains the CPU identification, the date and the time, the component identification and other system supplied values. The issuer of the IDUMP macro moves the characters 'SR' to the first two bytes of section one and clears the remainder to binary zeros.

The second section provides offset and length pointers to the variable part of the symptom record (sections 3, 4, 5, and 6). The offset pointers describe the offset of a section from the beginning of the symptom record (from the 'SR' field). If one or more of the variable sections are omitted, the length pointers must be cleared to binary zeros.

The symptoms in section three to five describe the error conditions in free format. A blank character is used as separator character. The symptoms in section six describe control blocks, tables or data areas. The control block locators are defined as LBD entries.

The macro ADSSR defines the structure of sections one and two. The macros ADSLBD (control block descriptor), ADSLBDA (alternate control block descriptor), ADSLBDF (formatting descriptor record), ADSLBXT (text extension record), ADSLBXX (hexadecimal data descriptor record), describe the format of the different section six entries. These LBD blocks may be specified in section 6 of the symptom record or they may be specified in the section six extension record.

If a section six is defined, its first 16 bytes are used as a chaining field to possible extension records. The chaining field may be followed by LBD entries.

Section Six Extension Records:

The symptom header record may be followed by one or more extension records. These extension records are used to add more section 6 entries to the dump. The block size of the header record and the section 6 extension records must not exceed 4108 bytes.

Each section 6 extension record contains a 16-byte chain field followed by the control block locating definitions (LBD entries).

The layout of the 16-byte chain field is as follows:

- 4 bytes characters 'SCT6' to identify the start of section 6.
- 4 bytes of zeros or address of an additional section 6 in storage if 4K were not enough to keep the complete symptom record for this one IDUMP request.
- 4 bytes of zeros or length of next section 6.
- 4 bytes of zeros.

The address and length, if specified, must describe an area in the partition of the issuer of the IDUMP macro or in the system GETVIS area. Otherwise they are treated as zeros.

The area that the address is pointing to must contain a complete section 6. The maximum length of this section 6 is again 4K bytes. The 16 bytes at the beginning of this section 6 must also follow the rules described above. There is no limitation to the number of section 6 chained together for one IDUMP request.

The chaining pointers (address and length of next section six entry are only relevant for the programmer of the IDUMP macro. If the IDUMP output has been entered into the dump library then the chaining fields are meaningless. The program Info/Analysis controls the processing of the symptom record and the section 6 extensions via internal storage maps,

Figure 7 on page 151 shows the format of the symptom record. The fields IDMPADDS and IDMPLENS (contained in the IDUMP parameter list) contain the address and the length of the user specified symptom record.

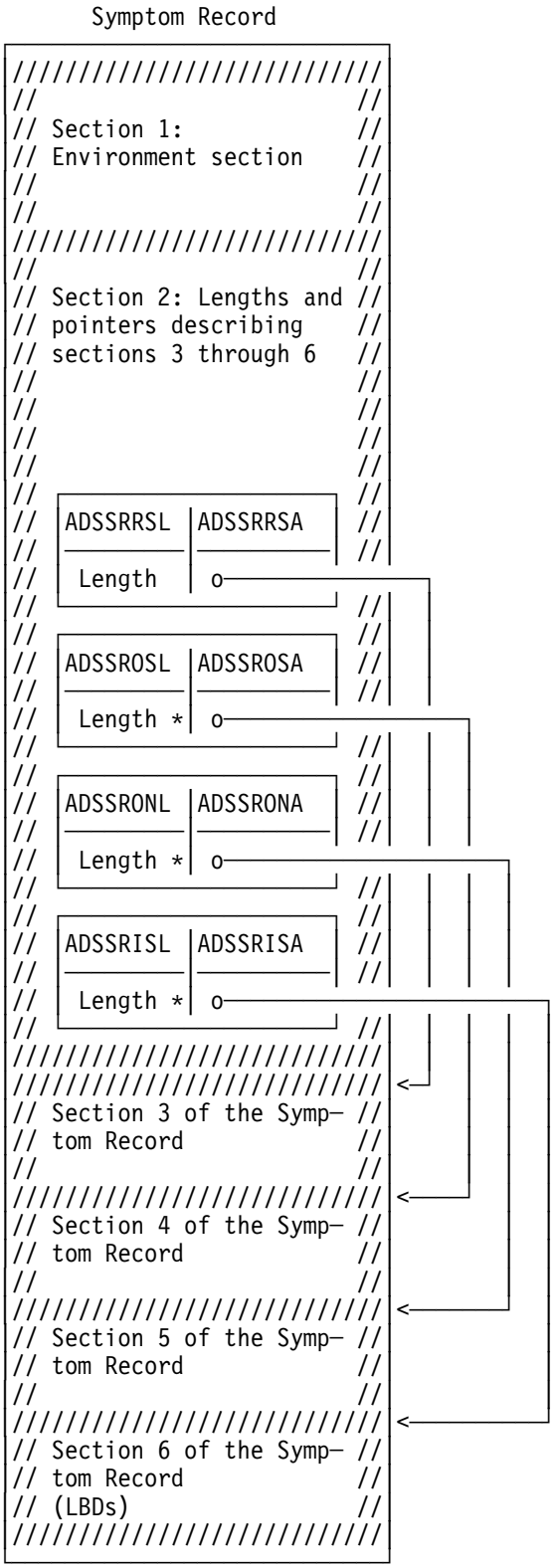


Figure 7. Format of Symptom Record

ADSSR – Symptom-Record Mapping

The module IJBSDUMP and the phase $IJBHDUP use this DSECT to access the sections 1-6 of the symptom record.

The layout of this DSECT is:

```
 Displacement               ADSSR
                  ************************************************************************
                  *      MAP FOR SYMPTOM RECORD                                          *
                  ************************************************************************
 000000           ADSSR    DSECT
 000000                    DS    0F
 000000                    DS    CL116
 000074      00000         ORG   ADSSR
 000000           ADSSRID  DS    CL2          SYMPTOM RECORD ID
 000002           ADSSRCPM DS    CL4          CPU MODEL ID
 000006           ADSSRCPS DS    CL6          CPU SERIAL NUMBER
 00000C           ADSSRDTS DS    CL30         DATE/TIME STAMP DATA AREA
 00002A      0000C         ORG   ADSSRDTS
 00000C           ADSSRTME DS    CL11         LOCAL TIME STAMP
 000017           ADSSRDAT DS    CL8          DATE (YY/MM/DD)
 00001F           ADSSRGMT DS    CL11         GREENWICH MEAN TIME
 00002A      0000C         ORG   ADSSRDTS
             0000C ADSSRSTK EQU   ADSSRDTS     ALTERNATE DATE/TIME STAMP FIELD
 00000C           ADSSRSTT DS    CL8          OUTPUT OF STORE CLOCK INSTRUCTION
 000014      0000C         ORG   ADSSRSTT
 00000C           ADSSRST1 DS    CL4          FIRST WORD OF STORE CLOCK INSTRUCTION
 000010           ADSSRST2 DS    CL4          SECOND WORD OF STORE CLOCK INSTRUCTION
 000014           ADSSRSTU DS    CL12         NOT USED FOR STORE CLOCK DATA
 000020           ADSSRSTC DS    F            LOCAL TIME ZONE CONVERSION FACTOR
 000024           ADSSRSTN DS    CL6          NOT USED FOR STORE CLOCK DATA
 00002A      0002A         ORG   ADSSRDTS+30
 00002A           ADSSRSID DS    CL12         SYSTEM ID
 000036      0002A         ORG   ADSSRSID
 00002A           ADSSRCMP DS    CL9          COMPONENT ID OF BASE OPERATING SYSTEM
 000033           ADSSRREL DS    CL1          RELEASE
 000034           ADSSRFEA DS    CL2          FEATURE
 000036      00036         ORG   ADSSR+54
 000036           ADSSFL01 DS    BL1          FLAG FIELD 1
 000037      00036         ORG   ADSSFL01
 000036           ADSSFL1F DS    BL1          0=SYMPTOM RECORD WITHIN (2K) 1=EXCEEDS 2K
             00036 ADSSFL1S EQU   ADSSFL01+0   0=SR COMES FROM HOST MACHINE 1=GUEST
             00036 ADSSFL1R EQU   ADSSFL01+0   RESERVED BITS
 000037      00037         ORG   ADSSR+55
 000037           ADSSFL02 DS    BL1          FLAG FIELD 2 RESERVED
 000038           ADSSRDTP DS    CL8          TYPE OF DUMP
 000040           ADSSRPRB DS    CL8          PROBLEM NUMBER (PROBLEM MANAGEMENT)
 000048           ADSSRRSL DS    FL2          LENGTH OF SECTION 3
 00004A           ADSSRRSA DS    FL2          OFFSET TO SECTION 3
 00004C           ADSSROSL DS    FL2          LENGTH OF SECTION 4. ZERO IF NON
 00004E           ADSSROSA DS    FL2          OFFSET TO SECTION 4
 000050           ADSSRONL DS    FL2          LENGTH OF SECTION 5. ZERO IF NON
 000052           ADSSRONA DS    FL2          OFFSET TO SECTION 5
 000054           ADSSRISL DS    FL2          LENGTH OF SECTION 6. ZERO IF NON
 000056           ADSSRISA DS    FL2          OFFSET TO SECTION 6
 000058           ADSSRR11 DS    FL4          RESERVED
 00005C           ADSSRASD DS    FL4          MVS - ASID OF COMPONENT DEPENDENT AREA
 000060           ADSSSNAM DS    CL8          DATA SPACE NAME                 @D52VDJO
 000068           ADSSMNAM DS    CL8          LIBRARY MEMBER NAME OF DUMP OF  @D52VDJO
                  *                           PERTINENT PRIMARY SPACE         @D52VDJO
 000070           ADSSRR16 DS    FL4          RESERVED
             00074 ADSSRSSS EQU   *            START OF SYMPTOM STRING SECTION 3,4,5,6
```

```
        ********************************************************************
        *  SECTION 3-4-5-6 DATA IS ADDRESSABLE BY OFFSET FIELDS WITHIN ADSSR  *
        *  WHICH IS AN OFFSET FROM THE START OF ADSSR TO THE SECTION DESIRED. *
        ********************************************************************
00000   ADSSRSS3 EQU   0             SECTION 3 DATA AREA
00000   ADSSRSS4 EQU   0             SECTION 4 DATA AREA
00000   ADSSRSS5 EQU   0             SECTION 5 DATA AREA
00000   ADSSRSS6 EQU   0             SECTION 6 DATA AREA

        ADSSRIDC DC    C'SR'         SYMPTOM RECORD ID
        ADSSTSAD DC    C'SADUMP  '   STANDALONE DUMP
        ADSSTOPR DC    C'OPRREQ  '   ID FOR OPERATOR REQUESTED DUMP VSE
        ADSSTPGM DC    C'PGMREQ  '   ID FOR PROGRAM REQUESTED DUMP VSE
        ADSSTSCP DC    C'SCPREQ  '   ID FOR SCP REQUESTED DUMP VSE
        ADSSTIDU DC    C'IDUMP   '   ID FOR IDUMP   VSE
        ADSSTUDU DC    C'SDUMP   '   ID FOR SDUMP/SDUMPX
0007F   ADSSFCMP EQU   B'01111111'   SYMPTOM RECORD EXISTS WITHIN 2K
00080   ADSSFEXC EQU   B'10000000'   SYMPTOM RECORD EXCEEDS 2K
000BF   ADSSFHST EQU   B'10111111'   SYMPTOM RECORD COMES FROM HOST MACHINE
00040   ADSSFGST EQU   B'01000000'   SYMPTOM RECORD COMES FROM GUEST MACHINE
```

ADSLBD – Control Block Locators

The phase $IJBHDUP and the module IJBXBTBM use this DSECT to access the section 6 of the symptom record.

The layout of this DSECT is:

```
 Displacement                ADSLBD
                ************************************************************************
                *      FIXED PORTION OF SECTION 6 OF DUMP SYMPTOM RECORD               *
                ************************************************************************
 000000         ADSLBD   DSECT
 000000                  DS    0F
 000000                  DS    CL44
 00002C  00000           ORG   ADSLBD
 000000         ADSLBDID DS    CL4          CONTROL BLOCK ID
 000004         ADSLBLEN DS    FL2          CONTROL BLOCK LENGTH
 000006         ADSLBSEQ DS    FL2          LBD SEQUENCE NUMBER
 000008         ADSLBCMP DS    CL8          COMPONENT ID ACRONYM
 000010         ADSLBNAM DS    CL8          NAME OF CONTROL BLOCK/DATA AREA DEFINED
 000018         ADSLBCBL DS    FL4          LENGTH OF DEFINED CONTROL BLOCK/DATA AREA
 00001C         ADSLBSHF DS    FL1          SPECIAL HANDLING FLAG FIELD
        00080   ADSLBSPR EQU   X'80'       DO NOT PRINT WITH OTHER LBDS
        00040   ADSLBSNP EQU   X'40'       DO NOT PROCESS AS NORMAL LBD
        00020   ADSLBZIV EQU   X'20'       ADDRESS OF ZERO IS VALID FOR THIS CTL BLOCK
 00001D         ADSLBRS1 DS    CL1          RESERVED
 00001E         ADSLBMDE DS    CL2          ADDRESSING MODE
 000020         ADSLBQAL DS    FL4          ADDRESSING QUALIFIER
 000024         ADSLBADR DS    AL4          ADDRESS OF CONTROL BLOCK/DATA AREA
 000028         ADSLBCMQ DS    CL4          COMPONENT ID QUALIFIER IF NEEDED
        0002C   ADSLBDSL EQU   *-ADSLBD     LENGTH OF FIXED PORTION
        0002C   ADSLBVAR EQU   *            START OF VARIABLE EXTENTION (OPTIONAL)

                ADSLBDCH DC    C'LBD '      CONTROL BLOCK ID
                ADSLBMDV DC    C'V '       VIRTUAL ADDRESSING MODE
                ADSLBMDR DC    C'R '       REAL ADDRESSING MODE
                ADSLBMDH DC    C'H '       HEADER RECORD ADDRESSING MODE
```

ADSLBDA – Alternate ControlBlock Locator Record

The module IJBXBTBM uses this DSECT to access the section 6 of the symptom record.

The layout of this DSECT is:

```
                  ********************************************************************
                  *      ALTERNATE SECTION 6 FIXED PORTION OF DUMP SYMPTOM RECORD      *
                  ********************************************************************

000000            ADSLBDA  DSECT
000000                     DS    0F
000000                     DS    CL32
000020      00000          ORG   ADSLBDA
000000            ADSLBAID DS    CL4      CONTROL BLOCK ID
000004            ADSLBALN DS    FL2      LENGTH OF THIS LBD
000006            ADSLBASQ DS    FL2      LBD SEQUENCE NUMBER
000008            ADSLBACI DS    CL8      COMPONENT ID ACRONYM
000010            ADSLBAR1 DS    FL4      RESERVED
000014            ADSLBAR2 DS    CL2      RESERVED
000016            ADSLBAMD DS    CL2      ADDRESSING MODE FOR LOCATORS IN ADSLBAVA
000018            ADSLBAQL DS    FL4      ADDRESSING QUALIFIER FOR LOCATORS IN ADSLBAVA
00001C            ADSLBAEL DS    FL4      NUMBER OF ELEMENTS IN ADSLBAVA
          00020   ADSLBAFL EQU   *-ADSLBDA   LENGTH OF FIXED PORTION
          00020   ADSLBAVR EQU   *        LIST OF CONTROL BLOCKS
```

ADSLBDF – Formatting Descriptor Record

The module IJBXBTBM uses this DSECT to access the section 6 of the symptom record. The layout of this DSECT is:

```
                  ********************************************************************
                  *           FORMATTING DESCRIPTOR HEADER                           *
                  ********************************************************************
000000            ADSLBDF  DSECT
000000                     DS    CL12
00000C      00000          ORG   ADSLBDF
000000            ADSDFID  DS    CL4             CONTROL BLOCK ID
000004            ADSDFLEN DS    FL2             CONTROL BLOCK LENGTH
000006            ADSDFCNT DS    FL2             NUMBER OF FORMAT ENTRIES IN BLOCK
000008            ADSDFATR DS    BL1             ATTRIBUTES OF ENTRIES
000009      00008          ORG   ADSDFATR
000008            ADSDFASQ DS    BL1             ENTRIES ARE SEQUENTIAL OFFSETS
          00008   ADSDFASP EQU   ADSDFATR+0      ENTRIES ARE SPECIFIC OFFSETS
          00008   ADSDFRS1 EQU   ADSDFATR+0      RESERVED
000009      00009          ORG   ADSLBDF+9
000009            ADSDFRS2 DS    CL1             RESERVED
00000A            ADSDFANO DS    FL2             OFFSET FOR START
          0000C   ADSDFHLN EQU   *-ADSLBDF       LENGTH OF HEADER
          0000C   ADSDFVAR EQU   *               ARRAY OF FORMAT ENTRIES
```

ADSLBXT – Text Descriptor-Record

The phase $IJBHDUP uses this DSECT to access the section 6 of the symptom record. The layout of this DSECT is:

```
                ************************************************************************
                *           STRUCTURE FOR TEXT EXTENSION                               *
                ************************************************************************

000000            ADSLBXT  DSECT
000000                     DS    0F
000000                     DS    CL12
00000C      00000          ORG   ADSLBXT
000000            ADSXTID  DS    CL4         CONTROL BLOCK ID
000004            ADSXTLEN DS    FL4         LENGTH OF THIS EXTENSION
000008            ADSXTPRP DS    FL1         RELATIVE POSITION OF TEXT IN OUTPUT
            00000 ADSXTPFR EQU   X'00'       TEXT APPEARS BEFORE CONTROL BLOCK DATA
            00080 ADSXTPAF EQU   X'80'       TEXT APPEARS AFTER CONTROL BLOCK DATA
000009            ADSXTRS1 DS    FL1         RESERVED
00000A            ADSXTRS2 DS    FL2         RESERVED
            0000C ADSXTHLN EQU   *-ADSLBXT   LENGTH OF TEXT EXTENSION HEADER
            0000C ADSXTDAT EQU   *           ONE OR MORE TEXT LINE ENTRIES
00000C      0000C          ORG   ADSLBXT+12

                ************************************************************************
                *           STRUCTURE FOR TEXT LINE ENTRY                              *
                ************************************************************************
000000            ADSLBTXT DSECT
000000                     DS    CL1
000001      00000          ORG   ADSLBTXT
000000            ADSTXTLN DS    FL1         LENGTH OF TEXT LINE MAX - 255 BYTES
            00001 ADSXTELN EQU   *-ADSLBTXT  LENGTH OF TEXT LINE ENTRY
            00001 ADSTXTTX EQU   *           TEXT DATA
000001      00001          ORG   ADSLBTXT+1
                  ADSXTCHR DC    C'LBXT'     LBXT CONTROL BLOCK ID
```

ADSLBXX – Hexadecimal Data Record

The phase $IJBHDUP uses this DSECT to access the section 6 of the symptom record. The layout of this DSECT is:

```
                ************************************************************************
                *          STRUCTURE FOR HEX DATA EXTENSION TO SECTION 6 RECORD        *
                ************************************************************************

000000             ADSLBXX  DSECT
000000                      DS    CL12
00000C      00000           ORG   ADSLBXX
000000             ADSXXID  DS    CL4        CONTROL BLOCK ID (LBXX)
000004             ADSXXTYP DS    CL1        TYPE OF HEXADECIMAL DATA
000005             ADSXXRGS DS    FL1        FIRST REGISTER NUMBER STORED
000006             ADSXXRS1 DS    CL1        RESERVED
000007             ADSXXRS2 DS    CL1        RESERVED
000008             ADSXXRS3 DS    CL1        RESERVED
000009             ADSXXRS4 DS    CL1        RESERVED
00000A             ADSXXLEN DS    FL2        LENGTH OF EXTENSION INCLUDING HEX DATA
            0000C  ADSXXHLN EQU   *-ADSLBXX   LENGTH OF HEADER PORTION OF EXTENSION
            0000C  ADSXXDAT EQU   *          HEXADECIMAL DATA
00000C      0000C           ORG   ADSLBXX+12
                   ADSXXCHR DC    C'LBXX'     LBXX CONTROL BLOCK ID
                   ADSXXPSW DC    C'P'        PSW STORED IN EXTENSION
                   ADSXXGPR DC    C'G'        GENERAL REGISTERS IN EXTENSION
                   ADSXXFPR DC    C'F'        FLOATING POINT REGISTERS IN EXTENSION
                   ADSXXCPR DC    C'C'        CONTROL REGISTERS IN EXTENSION
                   ADSXXELS DC    C' '        OTHER TYPES OF DATA IN EXTENSION
```

SDUMP Macro

Overview

The SDUMP and SDUMPX macros are two new macros introduced for the first time in VSE/AF 5.2.0 to provide support in data spaces. The SDUMP and SDUMPX macros use a new calling interface derived from MVS to support the data spaces and 31-bit addressing. The SDUMP and SDUMPX macros support only keyword parameters.

The SDUMP and SDUMPX macros will be resolved in a new SVC 123, which is compatible with the SVC 51 on MVS.

The SDUMP and SDUMPX macros provide a dump of virtual storage which contains user data and system data into a dump library or to SYSLST. It dumps address ranges in the primary address space and it dumps storage ranges in address or data spaces to which addressability via an ALET or via an STOKEN exists.

If the program is in primary ASC mode, SDUMP or SDUMPX can be used. Otherwise, wenn the program runs in access register(AR) mode, SDUMPX must be used. SDUMPX provides all of the function of SDUMP but generates code and address that are appropriate for AR mode.

The SDUMP macro cannot dump data space storage. To dump data space storage, SDUMPX is to be used instead by including either the LISTD or SUMLISTL parameter.

The following topics will be described:

1. the external interfaces, which explains about the invocation and the use of the macro's keyword and the return and reason codes,
2. the code to support the macros, and
3. the internal control blocks and their layout.

Description of External Interfaces

Invocation: VSE/ESA supports the requesting of address space or data space dumps through the SDUMP and SDUMPX macros. These macros may be issued in any program written in High Level Assembler.

SDUMP Macro:
The SDUMP macro expand into an SVC. On entry to the SVC, the parameter list address is loaded into register 1.
The SDUMP macro supports only keyword parameters.

```
[name] SDUMP        [{HDR='dump title' | HDRAD=dump_title_addr}]

                    [,{STORAGE=(start_addr, end_addr)|
                    ,LIST=list_addr}]

                    [,SUMLIST=list_addr]

                    [,MF={(E,ctrl_addr) | L}]
```

Figure 8. SDUMP macro

Note: All parameters of this macro are compatible with the parameters of the MVS SDUMP macro. VSE will ignore the parameters and values which are not significant for VSE.

The supported parameters are described as follows:

name symbolic name

HDR=*'dump title'*

specifies the title to be used for the dump.

HDRAD=*dump_title_addr*

specifies the address of the title to be used for the dump. The dump title field consists of a 1-byte length field followed by the specified title.

The length field specifies the length of the title excluding the length of the length byte itself. The macro excepts a length value of 1 to 100.

STORAGE=*(start_addr, end_addr)*

specifies one or more pairs of starting and ending addresses of areas to be included into the dump.

LIST=*list_addr*

specifies a list of starting and ending addresses.

The list must contain an even number of addresses, and each address must occupy one fullword. In the list, the high order bit of the fullword containing the last ending address of the list must be set to 1, and all other high order bits must be set to 0.

SUMLIST=*list_addr*

VSE/ESA handles and processes this keyword parameter in the same way as LIST.

MF={(E,*ctrl_addr)* **| L}**

If this keyword parameter is not specified, the macro generates the standard form.

The list form is used only to generate the control structure for the macro.

If the execute form is used, the macro will refer to the parameter list specified in the list form.

SDUMPX Macro:

The SDUMPX macro provides a dumping capability for routines that are running in access register mode. This macro is similar to the SDUMP macro, except that it generates code and addresses that are appropriate for access register mode. All parameters on the SDUMP macro are valid for the SDUMPX macro. The SDUMPX macro supports only keyword parameters. The LISTD and SUMLSTL parameters are valid only on the SDUMPX macro, which are now described in this section:

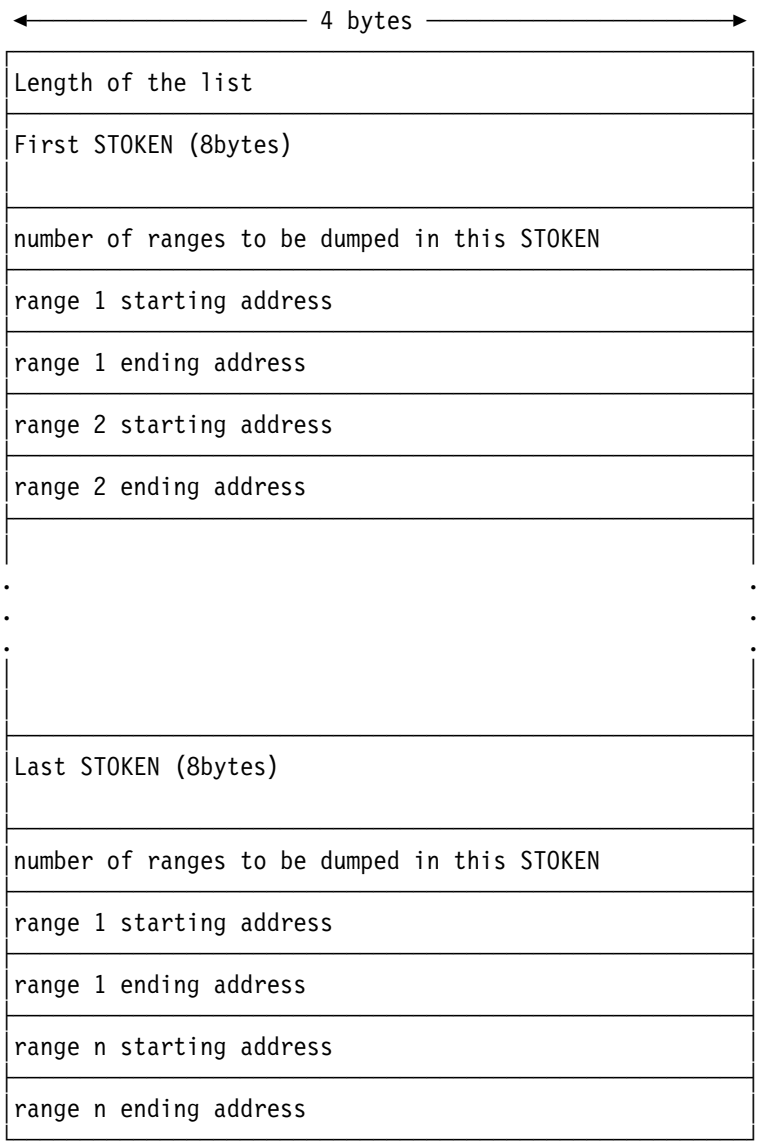
```
[name] SDUMPX       [{HDR='dump title'| HDRAD=dump_title_addr}]

                    [,{STORAGE=(start_addr, end_addr)|
                    ,LIST=list_addr}]

                    [,LISTD=list_addr]

                    [,SUMLIST=list_addr]
                    [,SUMLISTL=list_addr]

                    [,MF={(E,ctrl_addr) | L}]
```

Figure 9. SDUMPX macro

Note: All parameters of this macro are compatible with the parameters of the MVS SDUMP macro. VSE will ignore the parameters and values which are not significant for VSE.

LISTD=*list_addr*

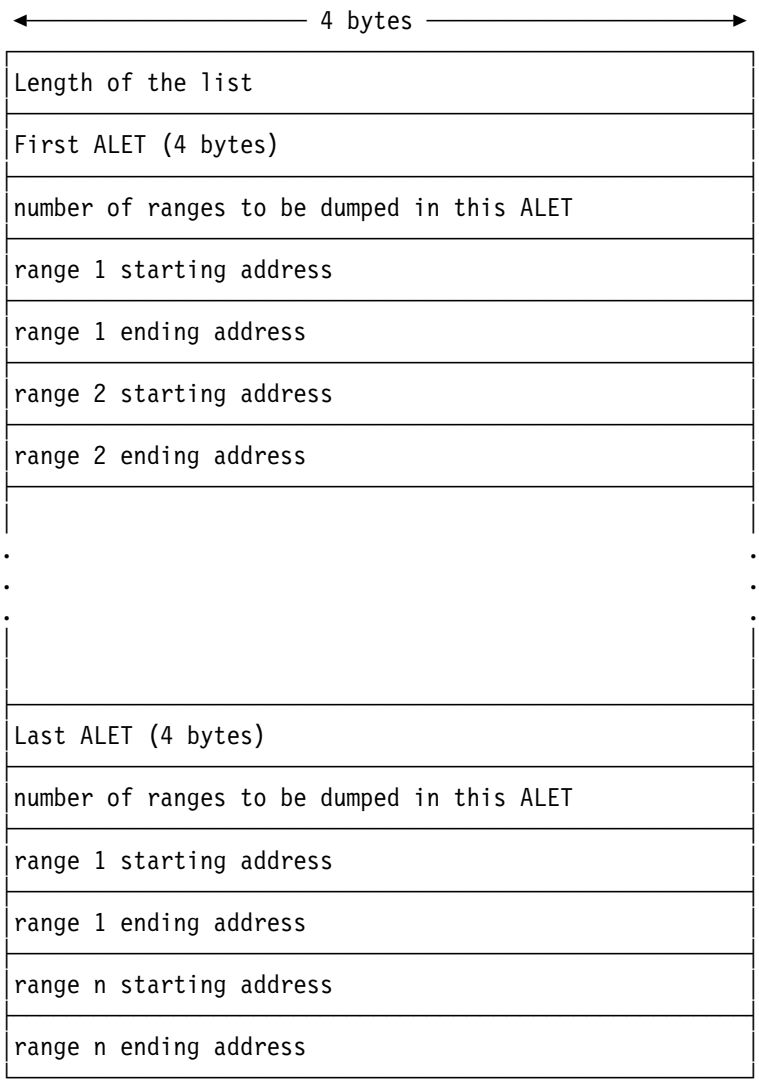
specifies a list of address ranges, qualified by STOKENs, of areas to be included in the dump. Specify the STOKEN's and address ranges as follow:



The first fullword of the list contains the number of bytes (including the first word) in the list. STOKEN refers to any address/data space.

SUMLSTL=*list_addr*

specifies a list of address ranges, qualified by ALETs, of areas to be included in the dump and will be processed like LIST.
Specify the ALETs and address ranges as follow:



The first fullword of the list contains the number of bytes (including the first word) in the list. ALET refers to entries in either a DU-AL or a PASN-AL, and associated with any address/data space that the caller has addressability to.

Programming notes:
Because the control blocks SDWA, ECB, and SRB do not exist in VSE/ESA, only the register 15 is used to store the return code and the reason code when control is returned as if TYPE=FAILRC is specified. The register 15 contains one of the following return codes in bits 24-31

Hexadecimal Code	Meaning
00	A complete dump was taken.
04	A partial dump was taken.
08	The system was unable to take a dump.

When a return code of not equal zero is received, a reason code is supplied in bits 16-23 of the register 15.
The reason code explains why the dump failed.

The reason codes are as follows:

Figure 10. SDUMP Reason codes

Macro Rsn Code	ABEND Rsn Code	Meaning
15	04	The parameter list address is zero
16	08	invalid parm list options - invalid dump type
18	10	invalid storage range specified. Conflicting begin and end addresses
19	14	invalid user data specified - header too long
1B	1c	invalid storage list or LISTD - (not fit in RTSDLTBL)
1E	28	invalid address space or dataspace
2A	58	The caller-supplied storage list is inaccessible.
2B	5c	The user header data is inaccessible.
2E	68	The caller-supplied SUMLIST is inaccessible.
33	7c	out of space limits
34	80	The caller-supplied STOKEN and range list in LISTD is inaccessible.
35	84	The caller-supplied ALET and range list in SUMLSTL is inaccessible.
E0	E0	SYSLST is not available
E1	E1	Dump library is full - part of dump is on SYSLST
E2	E2	Dump library is not defined - dump is on SYSLST
E3	E3	Dump library is in error - dump is on SYSLST

Internally a common mapping macro is used for the parameter list of SDUMP and SDUMPX.

SDUMP Parameter List: The SDUMP parameter list only occupies the first part of the mapping macro.

The valid and significant fields of the parameter list used in the SDUMP macro are shown in the following data structure:

```
 Offset   type        Len     Description
 ------------------------------------------------------------------------

   0      BITSTRING   1       first byte of flag
          ..1. ....           1 = storage list specified 0 = no storage list
          ...1 ....           1 = header data specified  0 = no header data
   1      BITSTRING   1       second byte of flag
          1... ....           1 = SDUMP request
          ..1. ....           always ON
          .... 1...           1 = SUMLIST specified
   2      BITSTRING   2       SDATA option flags
   4      ADDRESS     4       not used by VSE/ESA
   8      ADDRESS     4       Address of storage
  12      ADDRESS     4       Address of header
  16      ADDRESS     4       not used by VSE/ESA (ECB Address)
  20      BITSTRING   2       not used by VSE/ESA (current ASID)
  22      BITSTRING   2       not used by VSE/ESA (other ASID)
  24      ADDRESS     4       not used by VSE/ESA (address of ASID list)
  28      ADDRESS     4       Address of SUMLIST
  32      ADDRESS     4       reserved
  36      ADDRESS     4       storage addresses
  40      BYTE        1       length of header data
  41      BITSTRING   100     header data
```

SDUMPX Parameter List: The significant fields of the parameter list used in the SDUMPX macro are shown in the following data structure:

Offset	type	Len	Description
0	BITSTRING	1	first byte of flag
	..1.		1 = storage list specified 0 = no storage list
	...1		1 = header data specified 0 = no header data
1	BITSTRING	1	second byte of flag
	1...		1 = SDUMP request
	..1.		always ON
	.... 1...		1 = SUMLIST specified
2	BITSTRING	2	SDATA option flags
4	ADDRESS	4	not used by VSE/ESA (address of DCB)
8	ADDRESS	4	Address of STORAGE
12	ADDRESS	4	Address of HDR
16	ADDRESS	4	not used by VSE/ESA (address of ECB/SRB)
20	BITSTRING	2	not used by VSE/ESA (current ASID)
22	BITSTRING	2	not used by VSE/ESA (other ASID)
24	ADDRESS	4	not used by VSE/ESA (address of ASID list)
28	ADDRESS	4	Address of SUMLIST
32	ADDRESS	4	reserved
36	BITSTRING	1	Flag byte
37	BITSTRING	1	control flags
38	BITSTRING	1	type flag byte
39	BITSTRING	1	version
40	BITSTRING	1	not used by VSE/ESA (exit flag byte)
41	BITSTRING	1	reserved
42	BITSTRING	1	SDATA options
43	BITSTRING	1	reserved
44	ADDRESS	4	not used by VSE/ESA (SUBPLST address)
48	ADDRESS	4	not used by VSE/ESA (KEYLIST address)
52	ADDRESS	4	reserved
56	BITSTRING	4	not used by VSE/ESA (ALET of DCB parameter)
60	BITSTRING	4	not used by VSE/ESA (ALET of STORAGE param)
64	BITSTRING	4	not used by VSE/ESA (ALET of HDR parameter)
68	BITSTRING	4	not used by VSE/ESA (ALET of ASIDLST param)
72	BITSTRING	4	not used by VSE/ESA (ALET of SUBPLST param)
76	BITSTRING	4	not used by VSE/ESA (ALET of KEYLIST param)
80	ADDRESS	16	LISTD/SUMLSTL
96	ADDRESS	8	reserved
104	BYTE	1	length of header data
105	BITSTRING	100	header data

Description of Internal Routines: By detecting an SVC for SDUMP or SDUMPX, the dispatcher will load and pass the control to the main dump module IJBSDUMP. The SDUMP parameter list of the user program can be then found in the register 1 specified in the SAACOMM. IJBSDUMP allocates the dynamic space, and reserves a part of this space for the SDUMP routines by saving the starting address of this space at the XCARTCTP field of IJBXCA.

When a SDUMP or SDUMPX request is detected in the SAACOMM, IJBSDUMP will:

1. set the XCASDDIA on and call IJBXSDHK which diagnoses the SDUMP(X) parameter list and collects the information of the requested address space or data spaces needed during the dump process.
2. build the symptom record
3. set the XCASDDMP on and call IJBXSDHK to dump the address space ranges, when XCAMAIN is set.
4. then enter into a loop to process all dump requests for data spaces, by:
   - setting the function code XCASDINF and call IJBXSDHK to query the data space information about the ALET, the ALET name, the STOKEN, and STOKEN name
   - and set the XCASDDMP on to dump the data space ranges.
5. get the return code and reason code and pass the control back to the dispatcher.

IJBXSDHK - SDUMP Hook:
Its main function is to hook the SDUMP function to the VSE/ESA dump services.

IJBXSDHK is the interface module between IJBSDUMP and the SDUMP subroutines. IJBXSDHK intercepts the user's parameter list, identifies the requested function code and routes the control to the requested routine.

1. When IJBXSDHK is called for the first time, i.e., when XCASDDIA is set, the control block chain is built to keep the data needed during the dump process.

   The user 's parameter list is validated for its addressability before IJBVTSPR will be called to analyze the content of the parameter list.
2. When IJBXSDHK is requested to provide the dataspace information, it will call IJBVTSPR to get these data (XCASDINF is set for this purpose.)
3. When IJBXSDHK gets a dump request (XCASDDMP=ON), IJBVTSDL will be called.

The logic of this module is derived from IEAVAD00 of MVS.

This module and the following two are originally MVS source modules, which are mostly simplified and reused by VSE. The MVS error recovery (retry function) is totally removed from the code because such a solution does not exist on VSE.

IJBVTSPR - SDUMP Subroutine for Parameter Analysis:
This module is called for the following functions:

1. Analyze and process the user input
   a. Copy SDUMP parameter list, HDR, and storage list, and merge the dump options with the user requested options (e.g., suppress the dump.)
   b. Parse the LISTD option for SDUMPX
   c. Parse the SUMLIST/SUMLSTL option for SDUMPX
2. Provide the dataspace information

The control flow is shown in Figure 11 on page 168.

IJBVTSPR is called from IJBXSDMP to process one or more of the specified dump options. The subroutine to be called is determined by the function code in the RTSD (RTSDFNCD). The subroutines are:

- **COPYPARM - RTSDFNCD = 1**
  - Copy the callers SDUMP parameter list from the users storage to the RTSD (RTSDSDPL).
  - Copy the user specified dump header to the RTSD (RTSDHDR). Copy the caller specified ID to the RTSD (RTSDCID)
  - Copy the user specified storage list to the RTSD (RTSDLTBL).
  - Merge the change dump options with the user specified dump options and place the merged options in the copied SDUMP parameter list.
- **SDLISTD - RTSDFNCD = 5**

  for each STOKEN in list of STOKENs and ranges:
  - When the STOKEN represents an address space, the ranges are copied into the LISTD table starting at RTSDLSTDB
  - When the STOKEN represents a data space, the data space is checked for SCOPE(ALL) or SCOPE(COMMON) and the ranges are copied into the LISTD table starting at RTSDLSTDB. The entry structure of the LISTD table has the name SDWDDRNG.
- **SDSUML - RTSDFNCD = 6**

  The following areas will be checked for each ALET and ranges specified SUMLIST/SUMLSTL option on the SDUMP macro:
  - Any storage ranges specified with the SUMLIST keyword
  - Any storage ranges specified with the SUMLSTL keyword

    Each ALET in the parameter list is associated with one or more storage ranges.

    The ranges and their dataspace info will be copied to the current entry SDWDDRNG of the LISTD table.
- **SDQINFO - RTSDFNCD = 7**

  This subroutine provides the dataspace related information for the main module IJBSDUMP.

  As input, the current dataspace number XCASDCNR is provided.

  Following dataspace information will be stored in the IJBXCA:
  - XCAACALE - the ALET of the dataspace
  - XCAACSTO - the STOKEN
  - XCAACNAM - the dataspace name

This module is derived and combined the logic from the IEAVTSPR and IEAVTSSE of MVS. The COPYPARM and SDLISTD routines are two of five functions supported by the origal IEAVTSPR of MVS (, that is also the reason that the function codes 2 to 4 are unused.)

The SDSUML subroutine was part of the IEAVTSSE module. The reason to group together the funtions of the two modules in one is that, after leaving the diagnostic phase, IJBSDUMP must know the total number of data spaces or address space to be dump, in order to create the members in the dump library.

The SDQINFOP is VSE related only.

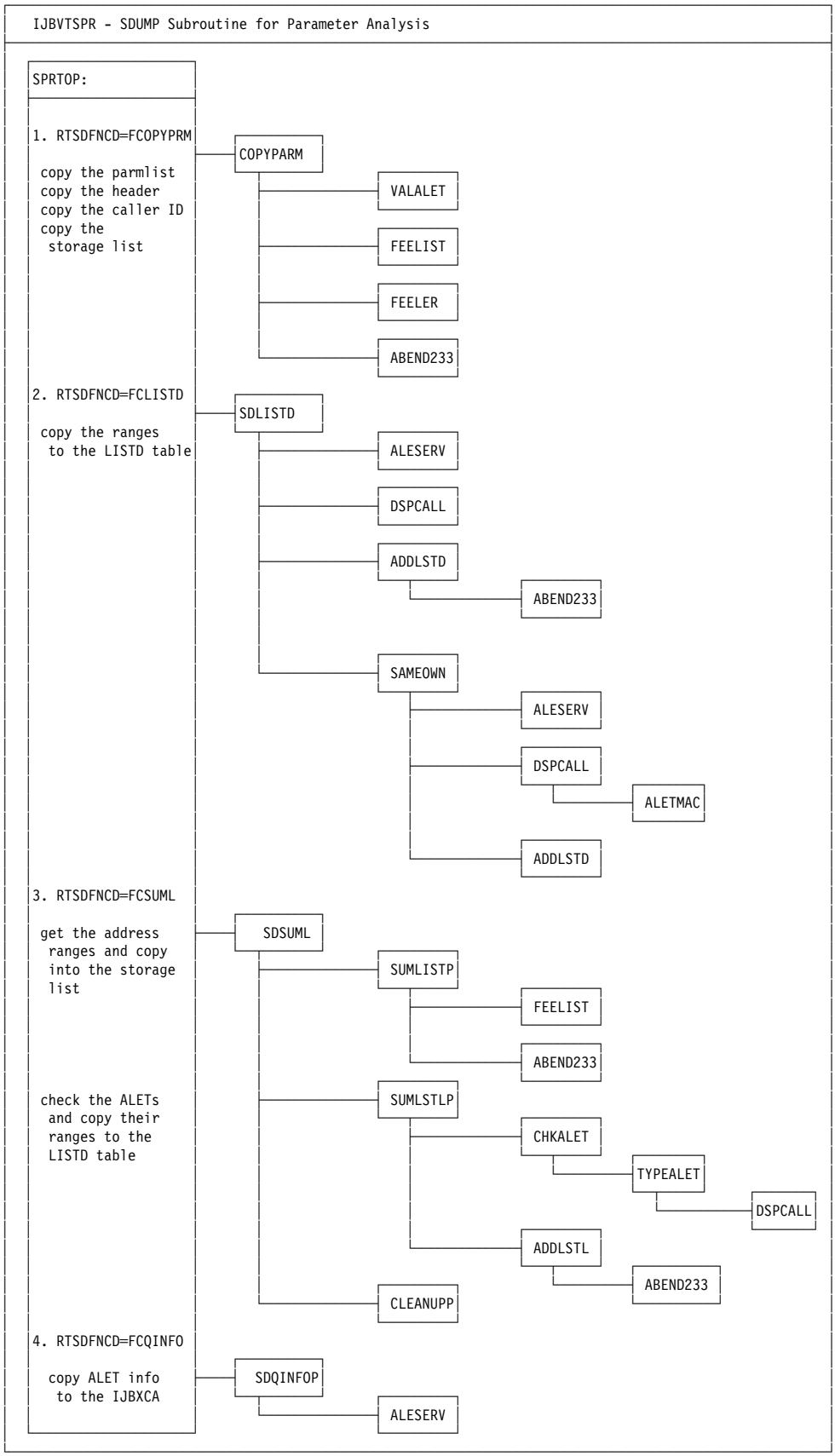


Figure 11. Control flow of IJBVTSPR

IJBVTSDL - Dump Routine for the Address Ranges:
Depend on the virtual storage types, IJBVTSDL builds:

1. the address range table for the requested storage ranges.
2. or the range tables for the requested data spaces

and pass the dump ranges contained in the local table to the ADDRANGE subroutine.

ADDRANGE is the interface routine to the VSE I/O dump modules. This segment will pass the address ranges of the address range table to IJBXBTBM or IJBXDPAR.

- As input, SPRNGPTR contains address of first address range in the part of table to be arranged, last range in the part of of the table is indicated by having a STARTADR of ENDPART and an ENDADDR of null with LASTRNGE flag ON.

- On the output side, the contents of the table is passed to IJBXBTBM or DPAR SPRNGPTR contains address of last range in this last of the address range table (ENPART address range). SPRTFUL=ON if table full else SPRTFUL=OFF.

The control flow is shown in Figure 12 on page 170.

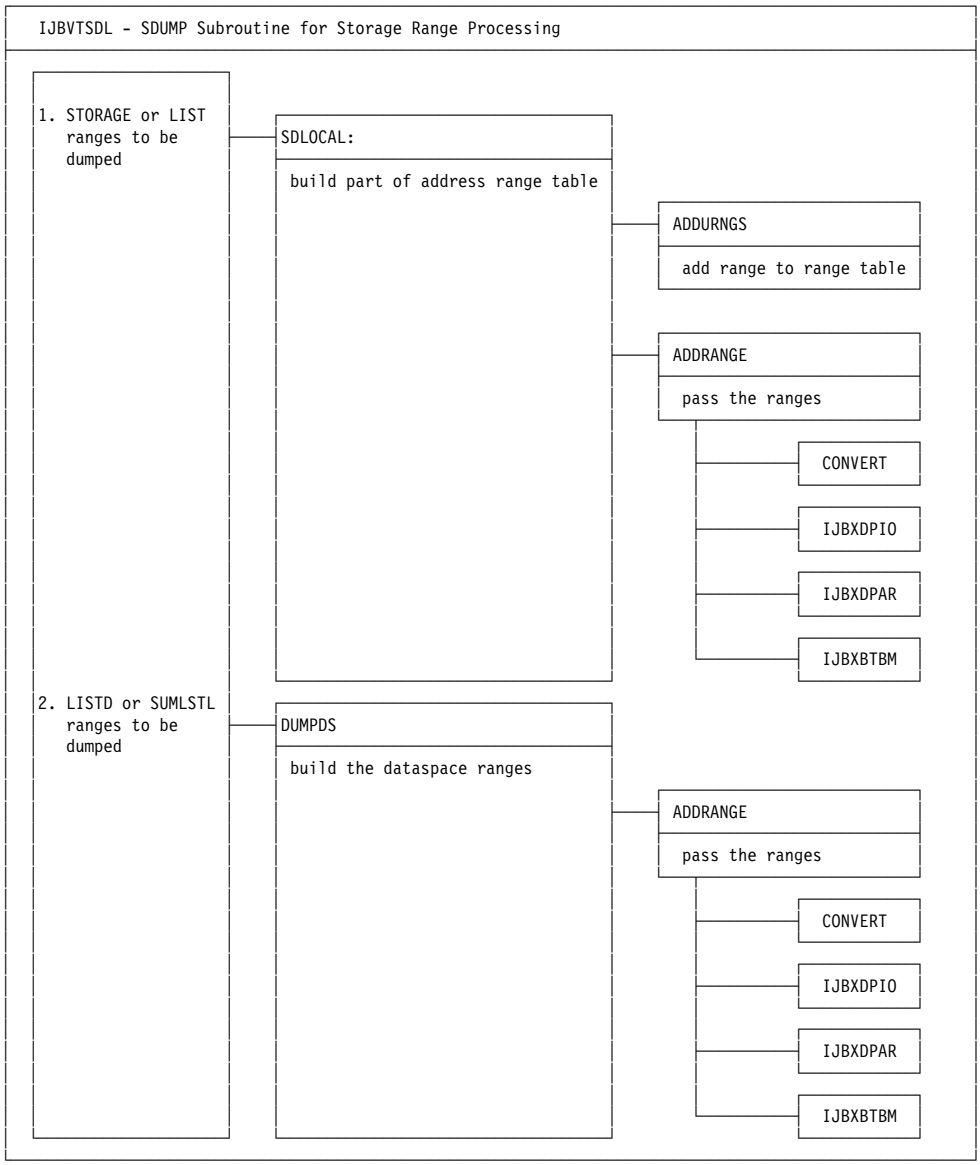


Figure 12. Control flow of IJBVTSDL

****

Chapter 7. Dump Utility: Introduction

Functions

The Dump Utility Program DOSVSDMP provides the following functions:

- Creating a Stand-Alone Dump Program
- Scanning a Dump Tape or Disk
- Printing a Dump Tape or Disk
- Printing an SDAID Tape
- Printing the IPL Diagnostics.

Creating a Stand-Alone Dump Tape or Disk

The program DOSVSDMP is used to create a Stand-Alone Dump Program on tape or disk to be used in emergency situations when the VSE system is in a hard or soft wait situation or in a system loop.

The Stand-Alone Dump Program writes the dump data to tape or disk when it is IPLed. The Dump Files which are written contain Dump Data and Symptom Records. The Symptom Records contain environmental data, error information, and control block locators (LBDs). The Symptom Records supply Info/Analysis with the data it needs to print and format the dump data.

Info/Analysis provides a facility to read a Stand-Alone Dump and transfer its contents into the VSE dump library (onload function). Info/Analysis then processes the dump.

The Stand-Alone Dump Program creation must write IPL records on the tape or disk. This allows the program to be IPLed at dump time. These IPL records are written in the first file on the Stand-Alone Dump Tape and at the beginning of the Disk for Stand-Alone Dump on CDK or FBA disks.

The Stand-Alone Dump Program can be created on a Work Disk or on a SYSRES disk. If the Stand-Alone Dump Program is created on a SYSRES disk, the Stand-Alone Dump IPL Records replace the "normal" system IPL records. When the Stand-Alone Dump Program is IPLed from a SYSRES disk, the dump program gets control, and when the dump program is complete, "normal" system IPL is called. This allows for the Stand-Alone Dump Program and SYSRES to be on the same disk.

Scanning a Dump Tape or Disk

The program DOSVSDMP is used to scan a Dump Tape or Disk and list what Dump Files are found. This function reads the Symptom Record at the beginning of each Dump File and lists the file number, the dump type, the name and the type of data contained in the file.

Printing a Dump Tape or Disk

Info/Analysis provides the facility to onload dumps into the VSE dump library. If this feature is used, the Info/Analysis functions for dump printing are available for the VSE Stand-Alone Dumps.

If the user does not have enough space in the dump library to onload a Stand-Alone Dump, he may print the Stand-Alone Dump Files with DOSVSDMP. DOSVSDMP prints the Symptom Record and the virtual dump without any formatting.

Printing an SDAID Tape

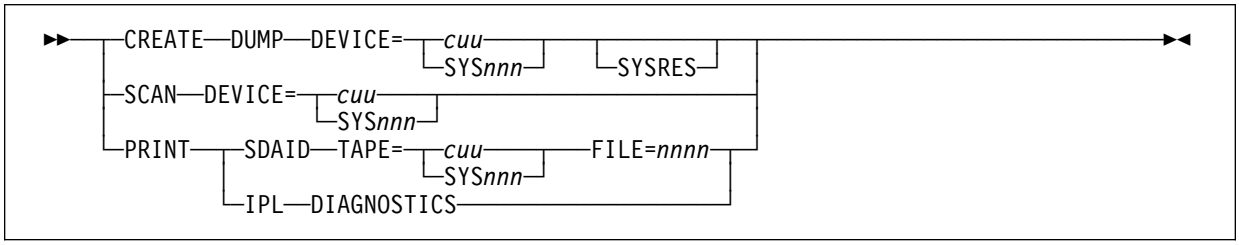
The program DOSVSDMP is used to print and format SDAID tapes. DOSVSDMP processes two types of SDAID tapes. The first type of tapes contains the dynamic output of an SDAID trace with OUTDEV assigned to a tape device. The other type of tapes contains SDAID buffer data. If the SDAID program writes its output into a wraparound buffer, the contents of this buffer can be written onto a tape with the attention DUMP command.

Printing IPL Diagnostics

The program DOSVSDMP is used to print the unattended node IPL diagnostics. The counter record from the unattended node control block is printed.

Invocation of DOSVSDMP

- Invocation from the console: DOSVSDMP prompts the console operator for the required functions. A parameter is not allowed from the console.
- Invocation from SYSRDR: DOSVSDMP may be invoked via `// EXEC DOSVSDMP,PARM='`*input parameter*`'`, an `EXEC` statement followed by an *input parameter* for a direct invocation of the DOSVSDMP functions (except Printing a Dump Tape or Disk). The *input parameter* format is as follows:



Chapter 8. Dump Utility and Stand-Alone Dump: Program Design

Modules

The **Dump Utility Program DOSVSDMP** consists of the following modules:

IJBXDMP Initialization Routine: Prompt console operator for additional input or check the operands of the parameter,

IJBXDM1 Dump Device input/output subroutines,

IJBXDM2 message writer for DOSVSDMP, console I/O routines, SYSLST output routines,

IJBXDM4 creates the Stand-Alone Dump Tape or Disk,

IJBXDM5 prints SDAID tapes and files from the Dump Tape or Disk.

The **Stand-Alone Dump Program** consists of the following modules:

IJBXDM7 the Stand-Alone Dump Base Program,

IJBXDM8 builds section 6 entries of the Symptom Record for the control blocks,

IJBXDM9 reads console messages from the hard-copy file and builds section 6 entries of the Symptom Record for them,

IJBXDM10 provides the Page Management capabilities for the Stand-Alone Dump Program to retrieve virtual storage pages from real storage and the page data set.

$$ACISS provides synchronous support for console integration. The message handling is dealed by this module when an integrated console is enable.

A brief description of the different modules is given in Chapter 10, “Dump Utility and Stand-Alone Dump: Module Descriptions” on page 191.

Macros

The macros **DMPSCTX**, **IJBXINT7**, **IJBXLIO**, **IJBXRC** and the **ADSxxxx** macros are required to compile and assemble the above modules:

DMPSCTX contains the common fields, constants and flags which are used for inter-module communication between the Dump Utlity Program DOSVSDMP and the Stand-Alone Dump Program at creation time,

IJBXINT7 is the interface between IJBXDM7, IJBXDM8, IJBXDM9 and IJBXDM10 at Stand-Alone Dump time,

IJBXLIO is the interface to the logical dump retrieval routine, the SYSLST, SYSLOG and the message handler routine of DOSVSDMP,

IJBXRC provides the Dump Record structure, and

ADSxxxx the macros **ADSSR**, **ADSLBD**, **ADSLBXT** and **ADSLBXX** provide the Symptom Record structure.

The **HCFCONTL**, **HCFIOMOD** and the **READ-**, **WRITE-**, **SKIP-** and **MODHCF** macros are required to assemble IJBXDM9, the Hardcopyfile Handler module.

Additionally, lots of system macros mapping system control blocks are required to compile and assemble the DOSVSDMP and Stand-Alone Dump Program modules.

Phases

The phase/module relationship is as follows:

Phase	Module	Function
Dump Utility Program		
DOSVSDMP	IJBXDMP	Main Module (Dialog)
	IJBXDM1	Dump Device I/O
	IJBXDM2	Message Writer
	IJBXDM4	SA Dump Program Creation
	IJBXDM5	Formatting and SYSLST I/O
Bootstraps		
$$A$DMP1		IPL Bootstrap for Tape
$$A$DMP2		IPL Bootstrap for CKD Disk
$$A$SMP3		IPL Bootstrap for FBA Disk
Stand-Alone Dump Program		
DOSVSDX7	IJBXDM7	Base Module
DOSVSDX8	IJBXDM8	Control Block Module
DOSVSDX9	IJBXDM9	Hardcopyfile Module
DOSVSDXA	IJBXDM10	Page Manager Module
$$ACISS	$$ACISS	Integrated Console Support Module

The phase $$A$DMP4 is just a dummy phase which is created during the assembly of IJBXDM7.

Flow of Control of DOSVSDMP

When DOSVSDMP is called from Job Control, the module IJBXDMP is entered. It loads the base registers and initializes the interface area to the other modules. When DOSVSDMP is invoked from SYSRDR, no prompting at all takes place. All necessary information has to be included in the *input parameter*. In prompting mode, the first selection menu is displayed. The operator has to specify the activity to be performed:

```
XX-000 4G01D SELECT ONE OF THE FOLLOWING FUNCTIONS:

  1   CREATE STAND-ALONE DUMP PROGRAM
  2   SCAN DUMP TAPE/DISK
  3   PRINT DUMP TAPE/DISK
  4   PRINT SDAID TAPE
  5   PRINT IPL DIAGNOSTICS
  R   END DOSVSDMP PROCESSING
```

Creating a Stand-Alone Dump Tape or Disk

If dump program creation is requested (via option 1 or the *input parameter*), the operator is prompted for the Dump Device:

```
XX-000 4G04D SPECIFY ADDRESS OF DUMP DEVICE (CUU OR SYSNNN)
```

If the Dump Device is a disk, the second selection menu is displayed:

```
XX-000 4G02D CREATE THE STAND-ALONE DUMP PROGRAM:

  1   ON A WORK DISK
  2   ON A SYSRES DISK
  R   END DOSVSDMP PROCESSING
```

The Stand-Alone Dump Program is created under control of IJBXDM4:

- the IPL Bootstraps (matching the Dump Device type),
- DOSVSDX7 (after initialization of the interface area DMPSCTX),
- DOSVSDX8,
- DOSVSDX9,
- DOSVSDXA and
- $$ACISS

are loaded and written to the Dump Device.

Scanning a Dump Tape or Disk

If scanning of a dump tape or disk is requested (via option 2 or the *input parameter*), an OC exit routine is activated, the operator is prompted for the Dump Device (see above) and scanning is performed under control of IJBXDM5: the Dump Device is scanned (via IJBXDM1) and the directory is built and printed (via IJBXDM2).

Printing a Dump Tape or Disk

If printing of a dump tape or disk is requested (via option 3; this function cannot be requested via the *input parameter*), an OC exit routine is activated, the operator is prompted for the Dump Device (see above) and the file number

```
XX-000 4G30D SPECIFY FILE NUMBER
```

and printing is performed under control of IJBXDM5: the Dump File is read (via IJBXDM1), formatted and printed (via IJBXDM2).

Printing an SDAID Tape

If SDAID tape printing is requested (via option 4 or the *input parameter*), an OC exit routine is activated, the operator is prompted for the tape unit and the file number and printing is performed under control of IJBXDM5. The module IJBXDM5 is able to process two types of SDAID tapes: the SDAID trace tapes and the SDAID buffer tapes built by the attention DUMP command. SDAID tapes which have been built by the attention DUMP command contain a Symptom Record. There is no difference in processing the SDAID data records for both types of SDAID tapes, because IJBXDM5 loads the SDAID formatting routine IJSDDEB and passes the SDAID data for formatting and printing.

Printing IPL Diagnostics

If printing of IPL diagnostics is requested (via option 5 or the *input parameter*), the IPL diagnostics are printed (via IJBXDM2).

When processing is complete, DOSVSDMP returns to Job Control via SVC 14.

The modules IJBXDM1 (containing the Dump Device I/O routines) and IJBXDM2 (containing the console I/O routines, the SYSLST output routines and a collection of all messages issued by the Dump Utility Program DOSVSDMP) provide functions which are called when they are required by other modules. When the requested service is completed, control is returned to the calling module.

The Generated Stand-Alone Dump Program

The generated Stand-Alone Dump Program is structured in such a way, that the loss of dumped data is kept at a minimum. The generated dump program is therefore divided into three parts:

- The first part are 2 small Bootstrap programs. The first one is loaded via IPL to `X'0' - X'13'` and loads the second one into a low core area reserved exclusively for the Stand-Alone Dump Program (`X'300' - X'3FF'`). The second Bootstrap saves six pages of low storage (`X'0' - X'5FFF'`) into a work file to get space for the main routine of the Stand-Alone Dump Program. These saved pages are later on inserted into the dump data file.
- The second part of the Stand-Alone Dump Program is the dump main routine DOSVSDX7. It is loaded into the already saved real storage location (`X'800' - X'5FFF'`). DOSVSDX7 dumps the real pages of the supervisor from real storage and is afterwards shifted to a higher storage location (the First Level Interrupt Handler (FLIH) area, which does not contain any relevant data areas).
- The third part of the Stand-Alone Dump Program, the phases DOSVSDX8, DOSVSDX9, DOSVSDXA and $$ACISS, are loaded adjacent to the shifted DOSVSDX7 (actually, DOSVSDX7 is shifted after the other 4 phases have been loaded). DOSVSDX8 and DOSVSDX9 create the control block and hardcopyfile message LBDs, and DOSVSDX7 dumps the virtual storage from real storage or from the page data set (via DOSVSDXA).

  When the integrated console is available and enable, all the messages will be handled by the $$ACISS module.

The Stand-Alone Dump Program destroys

- Real Storage locations
  - `X'0' - X'17'` (IPL of Bootstrap 1),
  - `X'B8' - X'BF'` (ESA) or `X'BA' - X'BB'` (370), respectively, (IPL I/O interrupt),
  - `X'300' - X'3FF'` (Bootstrap 2)
- and the current PSW.

A machine **Store Status** prior to IPLing the Stand-Alone Dump Device should be done to save the PSW. The destroyed real storage locations, if necessary, should be displayed and saved as hardcopy.

Flow of Control of the Stand-Alone Dump Program

IPL from Tape

The first record on the tape ($$A$DMP1) contains the CCWs to read the next record into storage location X'300' The record read into X'300' (TAPEIPL2) contains the CCWs to write the first six pages of storage to the tape and then read phase DOSVSDX7 into storage location X'800'. DOSVSDX7 is then passed control.

IPL from CKD Disk

The first record on the disk ($$A$DMP2) contains the CCWs to read the next record into storage location X'300'. The record read into X'300' (CKDIPL2) contains the CCWs to write the first 3 pages of storage to the disk at cylinder 0, head 4, record 1 and the second 3 pages of storage at cylinder 0, head 5, record 1. It also contains the CCWs to read phase DOSVSDX7 into storage location X'800'. DOSVSDX7 is then passed control.

IPL from FBA Disk

The first record on the disk ($$A$DMP3) contains the CCWs to read the next record into storage location X'300'. The record read into X'300' (FBAIPL2) contains the CCWs to write the first 6 pages of storage to the disk and the CCW's to read phase DOSVSDX7 into storage location X'800'. DOSVSDX7 is then passed control.

Stand-Alone Dump Program Main Routine

The Stand-Alone Dump main routine DOSVSDX7 is loaded at X'800'. The first part of DOSVSDX7 is the interface area DMPSCTX. This area contains information about the Dump Device. The information in DMPSCTX was built by IJBXDM4 at the time the dump tape or disk was created. The actual entry point for DOSVSDX7 is DMPENTRY.

The Stand-Alone Dump Program investigates a few items to determine if a dump is required and, if so, what type of dump.

- Check if machine is ESA
- If IPL with Action Clear, no dump is taken (Action Clear is assumed if the SYSCOM address, the SVC and I/O new PSW addresses and storage from X'420' - X'7FF' are all zero)
- Check Dump Device type
- If disk, check for correct format of data file
- Check for VSE/AF supervisor identification
- Check for running under VM
- Check for IPL complete

If the basic prerequisites are not met, the real storage will be dumped. If running under VM in VM mode, a real dump will be taken. Otherwise, a virtual dump (including pages from the page data set) will be taken.

The Supervisor and the Shared Virtual Area will be dumped.

DOSVSDX8, the Control Block Module, builds LBDs (locator block definitions) for specific control blocks and writes them as SCT6 entries to the Supervisor and SVA file.

DOSVSDX9, the Hardcopyfile Module, reads up to 200 console messages from the hardcopyfile and writes the LBDs to the Supervisor and SVA file.

The object manager HCFCONTL handles all I/O services requested for the HCF. These services are:

- POINTHCF --- Open HCF control block
- CLOSEHCF --- Close HCF control block
- WRITEHCF --- Write logical HCF record
- READHCF --- Read logical HCF record
- SKIPHCF --- Skip one or more logical records
- MODHCF --- Change direction of read

All services are supported by interface macros. The service gets the entry point to the general entry routine ('CONTROL') via the CRTTAB and branches to it (standard linkage conventions). The service routines branch back to the requestor after handling the request and passes an return code in R15.

The Page Manager Address Spaces (definitely), Partitions and Dataspaces (based on the JCL `OPTION SADUMP`) will be dumped in separate files after the Supervisor and SVA file.

Partitions and Dataspaces are dumped in priority order based on the job control `OPTION SADUMP`. The order is to dump all partitions with a priority of 9, then 8, then, 7 etc.. Partitions with a priority of zero will not be dumped. Dumping to disk will terminate when the disk file becomes full. When end of volume on a dump tape is reached, the Stand-Alone Dump Program prompts the operator to mount a new tape. Dumping continues when the tape becomes ready. Dumping to tape terminates when the whole storage is dumped or when the operator decides to reIPL VSE instead of mounting a new tape.

On tape, the Supervisor and SVA are contained in the same file. All other entities follow each in a separate file. The last file on the tape is followed by two tape marks.

On disk, all of the dumped data is in one physical file. However, the Supervisor and the SVA are contained in one logical file and all other entities follow each in a separate logical file. Each logical file ends with a dump record id of `X'00000004'` (End of File). The last logical file in the physical disk file is followed by a dump record with an id of `X'00000008'` (End of Dump Data File).

If the Stand-Alone Dump Program was IPLed from a non-SYSRES device, it issues one of the following hard wait codes at program termination.

If the Stand-Alone Dump Program was IPLed from SYSRES and no errors occurred during processing, VSE is reIPLed. In case of an error, those hard wait codes which indicate I/O problems with the IPL disk may be issued.

The format of the hard wait PSW is `X'000A000000CE`*nnnn*`'`, where *nnnn* indicates the Reason Code for the hard wait.

Code	Meaning
0000	OK
0001	I/O Error on SIO / SSCH
0002	Device not operational
0004	Channel Error
0008	Permanent I/O Error
0010	I/O Error during Error Recovery
0020	Unrecoverable Tape Error
0040	Console I/O Error
0080	End of Extent on Stand-Alone Dump Disk
0100	I/O Error on Tape IPL
0400	Program Check during IPL
0800	Program Check preparing Virtual Dump
1000	Program Check dumping Virtual Storage in IJBXDM10
2000	Program Check shifting IJBXDM7
4000	Program Check in IJBXDM8 or IJBXDM9

Chapter 9. Dump Utility and Stand-Alone Dump: Records and Formats

When the processor storage contains a VSE system and the page data set can be accessed by the Stand-Alone Dump Program, a virtual dump is taken. Otherwise, the real storage is dumped. Virtual Pages (pages from the Page Data Set) are only dumped if they are valid. To save space on the Dump Device, pages which contain all zeros are not dumped at all.

The Stand-Alone Dump Program writes different Files to the Dump Device. The logical content of a File is a complete entity: the Supervisor and the SVA, a Page Manager Address Space, a Partition or a Dataspace. Physically, a File contains

- Symptom Records (Header Record and Section 6 Extension Records) and
- Data Records.

Each of these records is 4112 bytes long (4112 is 4K (4096) for dump data plus 16 bytes space for control data).

On Tape, the Dump Files are physically separated by a Tape Mark. The last File on Tape is followed by 2 Tape Marks.

Since all Dump Files on Disk are contained in one big physical file (the Dump Data File), they are logically separated by

- Control Records.

The general Dump Record layout is given in macro IJBXRC.

The Symptom Record Layout is given in the ADSxxxx macros.

Symptom Record (X'00000001')

Symptom Records can be divided into 2 groups:

- Header Records and
- Section 6 Extension Records.

All Symptom Records together (that is, one Header Record and n Section 6 Extension Records) logically form the Symptom String.

The first record of a Dump File is always the Header Record. Section 6 Extension Records may occur interspersed between the Data Records, or they appear at the end of the Dump File.

The format of a Symptom Record is:

```
Field Name  Length  Description

DMPIDFLD       4    Type of Record
DMPSYMP     4096    Symptom Record
              12    reserved
```

The DMPIDFLD contains X'00000001'. This identification field does not belong to the Symptom Record, that means that offsets within the Symptom Record are calculated relative to the byte following the identification field.

Header Record

The first two bytes of the Header Record (following the identification field) contain C'SR'. The Stand-Alone Dump Program only fills in Section 1 (Environment Section), Section 2 (Offsets to Other Sections), and Section 5 (Status Section). Section 6 contains a 16-byte entry of binary zeros. This indicates that the Section 6 information follows in separate Section 6 Extension Records.

Section 1 contains the following fields:

```
Field Name  Length  Description

              (4)   (X'00000001') (Id Field)
ADSSRID        2    C'SR'
ADSSRCPM       4    CPU Model Number
ADSSRCPS       6    CPU Serial Number
ADSSRSTK       8    CPU Timer Value (stored by a STore ClocK instruction)
ADSSRSTU      22    binary zeros (not used)
ADSSRCMP       9    C'5686-032'
ADSSRREL       1    C'D'
ADSSRFEA       2    C'B6'
ADSSFL01       1    X'00' / X'40' if dump is from VM Machine
ADSSFL02       1    X'80' Date and Time are in STCK format
ADSSRDTP       8    C'SADUMP  '
               8    binary zeros (not used)
```

Section 2 contains offset pointers to the Sections 3, 4, 5, and 6.

The symptoms in Section 5 describe the status of the dumped system. Section 5 contains a set of self-explanatory symptoms, e.g.:

- MACHINE_STATUS_NOT_SAVED
- CLOCK_NOT_AVAILABLE

- MACHINE= XA / ESA
- MODE=PAGING / VM / OTHER_SYSTEM_(NO_VALID_VSE_SUPERVISOR)
- ACTIVE_SPACE_ID=xx
- DUMPED_DATA_FROM_SPACE_ID=xx
- PMR_ADDRESS_SPACE_ID=xx
- JOB_NAME=xxxxxxxx
- DATA_SPACE_NAME=xxxxxxxx
- DUMPED_DATA=SUPERVISOR+SVA / REAL_STORAGE / REAL_DUMP_REASON_CODE_xx / xx-PARTITION / xx-DATA_SPACE

Section 6 Extension Records

The first four bytes of the Section 6 Extension Record (following the identification field) contain the characters `C'SCT6'`. This constant is followed by 12 bytes of binary zeros. The following area of 4080 bytes is filled with control block locator (LBD) entries. The Dump File containing the Supervisor and the SVA also contains LBD entries for the following VSE control blocks:

PSW	Program Status Word (at time of failure)
AREGS	Access Registers *
FREGS	Floating Point Registers
GREGS	General Purpose Registers
CREGS	Control Registers
MESSAGE	Error messages and the last 200 messages from the Hard-Copy File
LOWCORE	Low address storage
SYSCOM	System Communication Region
UNATTCB	Re-IPL control block (previous Re-IPL invocation)
UNATTCBN	Hard Wait information (last Re-IPL invocation)
SMCOM	Storage Management Communication Area
CLIM	Class/System Limits Control Block
PCB	Partition Control Block
SCB	Space Control Block
PMRAS	Page Manager Address Space
ASTE	Address Space Number Second Table Entry *
PASNAL	Primary Address Space Number Access List *
COMREG	Partition Communication Region
PIBTAB	Partition Information Block
PIB2TAB	Partition Information Block Extension
LUBTAB	Logical Unit Block
PUBTAB	Physical Unit Block
PUB2TAB	Physical Unit Block Extension
ERBLOC	Error Recovery Block
CHQTAB	Channel Queue Table
CHNTAB	Channel Control Table
TIBATAB	Task Information Block Address Table
TIB	Task Information Block
TCB	Task Control Block
SAVAREA	Partition Save Areas
ACCREGS	Access Registers *
TDSE	Task's Data Space Extension *
DUCT	Dispatchable Unit Control Table *
DUAL	Dispatchable Unit Access List *
DSCB-SCB	Data Space Space Control Block *
LPT	Library Pointer Table
LDT	Library Definition Table

SDT	Sublibrary Definition Table
EDT	Extent Definition Table
DDT	Device Definition Table
SDBUFFER	SDAID Buffer

Note: * ESA mode on ESA hardware

The Section 6 Extension Record of a Dump File containing a Partition contains an LBDs for the partition's save area.

Data Record (X'00000002')

```
Field Name  Length  Description

DMPIDFLD       4    Type of Record
DMPVTADR       4    Virtual Address of Record
DMPRLADR       4    Real Address of Record
DMPRLKEY       1    Storage Key (if data in Real Storage)
DMPCNTFL       1    Control Flags
DMPKYFLG            X'10' Storage Key present
DMPRLFL3            X'08' Real Address present
DMPRLFL2            X'04' Page without Address Translation
DMPRLFLG            X'02' Page from Processor Storage
DMPVTFLG            X'01' Page from Page Data Set
               2    reserved
DMPDATA     4096    Dump Record Data
```

The DMPIDFLD contains X'00000002'.

Control Record (X'00000004') - (logical) End of File

```
Field Name  Length  Description

DMPIDFLD       4    Type of Record
            4108    reserved
```

The DMPIDFLD contains X'00000004'. This record only applies to Stand-Alone Dumps on disk. On tape, End of File is indicated by a Tape Mark.

Control Record (X'00000008') - (logical) End of Dump Data File

```
Field Name  Length  Description

DMPIDFLD       4    Type of Record
            4108    reserved
```

The DMPIDFLD contains X'00000008'. This record only applies to Stand-Alone Dumps on disk. On tape, End of Volume is indicated by two Tape Marks.

Format of the Stand-Alone Dump Tape

The first file of the Stand-Alone Dump Tape contains the Stand-Alone Dump Program. It is built by the program IJBXDM4 during Dump Program Creation. The second file is a Work File which is used by the Stand-Alone Dump Program to temporary save storage pages. The third file is the Main Dump File. The remaining files contain the other entities as described above.

The Dump Files are separated by single Tape Marks. The last file is followed by two Tape Marks. The format of the files is as follows:

File 1

The 24-Byte IPL Record ($$A$DMP1): Bootstrap 1

The 104-Byte IPL Record (TAPIPL2): Bootstrap 2

DOSVSDX7: The Stand-Alone Dump Program Base Module.

DOSVSDX8: The Stand-Alone Dump Program Control Block Module.

DOSVSDX9: The Stand-Alone Dump Program Hardcopyfile Module.

DOSVSDXA: The Stand-Alone Dump Program Page Manager Module.

$$ACISS: The Console Integration Synchronous Support Module.

File 2

The second file is a work file et for the dump program. It is used for temporary saving the first 6 pages of real storage. These pages are later on retrieved by DOSVSDX7 and copied into the Supervisor and SVA Dump File. The record length in file 2 is 4096 bytes.

File 3

File 3 is the Main Dump File. It contains

- a Symptom Record,
- the Supervisor storage,
- the output of DOSVSDX8 and DOVSDX9 and
- the SVA storage.

File 4 - n

These files contain the dumped storage

- of the Page Manager Address Spaces and
- of Partitions and Dataspaces in the order of the JCL `OPTION SADUMP` value from 9 to 1.

Each file contains

- Symptom Records and
- Data Records.

Format of the Stand-Alone Dump Disk (CKD)

The Stand-Alone Dump on disk consists of two parts. Part 1 is the Stand-Alone Dump Program File and part 2 is the Stand-Alone Dump Data File.

Stand-Alone Dump Program File

The 24-Byte IPL Record ($$A$DMP2): Bootstrap 1. This record is at cylinder 0, head 0, record 1.

The 144-Byte IPL Record (CKDIPL2): Bootstrap 2. This record is at cylinder 0, head 0, record 2.

DOSVSDX7: The Stand-Alone Dump Program Base Module. This record is at cylinder 0, head 2, record 1.

DOSVSDX8: The Stand-Alone Dump Program Control Block Module. This record is at cylinder 0, head 3, record 1.

DOSVSDX9: The Stand-Alone Dump Program Hardcopyfile Module. This record is at cylinder 0, head 3, record 2.

DOSVSDXA: The Stand-Alone Dump Program Page Manager Module. This record is at cylinder 0, head 3, record 3.

$$ACISS: The Console Integration Synchronous Support Module. This record is at cylinder 0, head 3, record 4.

The Stand-Alone Dump Work File: These records are a work file for the dump program. It is used for temporary saving the first 6 pages of real storage. These pages are later on retrieved by DOSVSDX7 and copied into the Supervisor and SVA Dump File. These records are at cylinder 0, head 4, record 1 and cylinder 0, head 5, record 1. The record length is 3*4096=12288 bytes.

Stand-Alone Dump Data File

This file must be on the same disk as the Stand-Alone Dump Program. This file is defined during dump creation using DASD labels DOSVSDX7 locates this file via information in the interface area DMPSCTX which was written on the disk at creation time by IJBXDM4.

This file contains "logical" files for each dumped entity. The logical files are separated by a dump record id of `X'00000004'` (End of File). The layout of each file is the same as described for tape. The last logical file in the physical disk file is followed by a dump record with an id of `X'00000008'` (End of Dump Data File).

Format of the Stand-Alone Dump Disk (FBA)

The Stand-Alone Dump on disk consists of two parts. Part 1 is the Stand-Alone Dump Program File and part 2 is the Stand-Alone Dump Data File.

Stand-Alone Dump Program File

The 24-Byte IPL Record ($$A$DMP3): Bootstrap 1.

The 48-Byte IPL Record (FBAIPL2): Bootstrap 2.

DOSVSDX7: The Stand-Alone Dump Program Base Module.

DOSVSDX8: The Stand-Alone Dump Program Control Block Module.

DOSVSDX9: The Stand-Alone Dump Program Hardcopyfile Module.

DOSVSDXA: The Stand-Alone Dump Program Page Manager Module.

$$ACISS: The Console Integration Synchronous Support Module.

The Stand-Alone Dump Work File: These records are a work file for the dump program. It is used for temporary saving the first 6 pages of real storage. These pages are later on retrieved by DOSVSDX7 and copied into the Supervisor and SVA Dump File.

Stand-Alone Dump Data File

This file must be on the same disk as the Stand-Alone Dump Program. This file is defined during dump creation using DASD labels DOSVSDX7 locates this file via information in the interface area DMPSCTX which was written on the disk at creation time by IJBXDM4.

This file contains "logical" files for each dumped entity. The logical files are separated by a dump record id of `X'00000004'` (End of File). The layout of each file is the same as described for tape. The last logical file in the physical disk file is followed by a dump record with an id of `X'00000008'` (End of Dump Data File).

Chapter 10. Dump Utility and Stand-Alone Dump: Module Descriptions

DOSVSDMP Module Descriptions

IJBXDMP - DOSVSDMP Main Module

```
* ---------------------------------------------------------- *
*                                                            *
* MODULE NAME  :   IJBXDMP                                   *
*                                                            *
* PHASE NAME   :   DOSVSDMP                                  *
*                                                            *
* LINKBOOK     :   IJBMDUXL                                  *
*                                                            *
* DESCRIPTIVE NAME:                                          *
*                                                            *
*   DUMP UTILITY PROGRAM DOSVSDMP                            *
*   (MODULE IJBXDMP - MAIN MODULE)                           *
*                                                            *
* FUNCTION     :   GENERATE STAND-ALONE DUMP PROGRAM         *
*                  SCAN DUMP TAPE/DISK                       *
*                  PRINT DUMP TAPE/DISK                      *
*                  PRINT SDAID TAPE                          *
*                  PRINT IPL DIAGNOSTICS                     *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
*                                                            *
* ENTRY POINT  :   IJBXDMP                                   *
*                                                            *
* CALLED BY    :   JOB CONTROL                               *
*                                                            *
* MODULES CALLED:                                            *
*                                                            *
*   IJBXDM1 (FOR DUMP DEVICE I/O)                            *
*   IJBXDM2 (FOR CONSOLE AND SYSLST I/O)                     *
*   IJBXDM4 (FOR DUMP PROGRAM CREATION)                      *
*   IJBXDM5 (PRINT DUMP OR SDAID TAPES OR IPL DIAGNOSTICS)   *
*                                                            *
* EXIT NORMAL  :   TO JOB CONTROL VIA EOJ MACRO              *
*                                                            *
* EXIT ERROR   :   NONE                                      *
*                                                            *
* MODULE / PHASE RELATIONSHIP:                               *
*                                                            *
*   LINKBOOK IJBMDUXL IS USED TO LINK THE DOSVSDMP PHASE:    *
*                                                            *
*   PHASE DOSVSDMP,*,NOAUTO                                  *
*   INCLUDE IJBXDMP  DOSVSDMP MONITOR ROUTINE                *
*   INCLUDE IJBXDM1  TAPE RETRIEVAL ROUTINE                  *
*   INCLUDE IJBXDM2  CONSOLE AND SYSLST I/O                  *
*   INCLUDE IJBXDM4  DUMP PROGRAM GENERATION                 *
*   INCLUDE IJBXDM5  PRINT SDAID OR DUMP TAPES               *
*                                                            *
* MACROS USED:                                               *
*                                                            *
*   IJBXSCT6   DMPSCTX   IJBXRC   IJBXLIO                    *
*                                                            *
* ---------------------------------------------------------- *
```

```
* ---------------------------------------------------------- *
*                                                            *
* INPUT:                                                     *
*                                                            *
*   FOR PROMPTING MODE:                                      *
*     OPERATOR RESPONSE TO CONSOLE MENUES                    *
*                                                            *
*   FOR PARAMETER INVOKATION:                                *
*     R1 = ADDRESS OF THE POINTER TO THE PARAMETER LIST      *
*     R15 = R1 IF NO PARAMETER WAS SPECIFIED                 *
*     FORMAT OF THE PARAMETER LIST:                          *
*     BYTE 0-1: LENGTH OF THE SPECIFIED PARAMETER STRING     *
*     BYTE 2..: PARAMETER CHARACTER STRING                   *
*                                                            *
*     FORMAT OF THE PARAMETER:                               *
*     PARM='CREATE DUMP DEVICE=CUU <SYSRES>'                 *
*     PARM='CREATE DUMP TAPE=CUU <SYSRES>' (COMPATIBILITY)   *
*     PARM='SCAN DEVICE=CUU'                                 *
*     PARM='PRINT SDAID TAPE=CUU FILE=NNNN'                  *
*     PARM='PRINT IPL DIAGNOSTICS'                           *
*                                                            *
*     SYSNNN MAY BE SPECIFIED INSTEAD OF CUU                 *
*                                                            *
* OUTPUT:                                                    *
*                                                            *
*   UPDATED INTERFACE BLOCK DMPSCTX                          *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
*                                                            *
* MESSAGES CAUSED:                                           *
*                                                            *
*   4G01D  'MENU 1'                                          *
*   4G02D  'MENU 2'                                          *
*   4G03I  DISK DOES NOT HAVE A VOL1 LABEL                   *
*   4G04D  SPECIFY ADDRESS OF DUMP DEVICE                    *
*   4G05D  SPECIFY ADDRESS OF SDAID TAPE                     *
*   4G09I  DUMP PROGRAM HAS BEEN CREATED                     *
*   4G11I  SELECTED OPTION IS INVALID                        *
*   4G13I  INVALID DEVICE SPECIFICATION                      *
*   4G15I  PROGRAMMER LOGICAL UNIT IS NOT ASSIGNED           *
*   4G16I  NO FREE PROGRAMMER LOGICAL UNIT AVAILABLE         *
*   4G19I  SYSLST IS NOT ASSIGNED                            *
*   4G23I  WRONG TAPE REEL                                   *
*   4G25I  DEVICE NOT AVAILABLE IN THE SYSTEM                *
*   4G26I  WRONG DEVICE SPECIFIED                            *
*   4G30D  SPECIFY FILE NUMBER ON TAPE                       *
*   4G31I  FILE NUMBER SPECIFIED INCORRECTLY                 *
*   4G32I  DEVICE ALREADY USED                               *
*   4G33I  PARAMETER ERROR                                   *
*                                                            *
* MESSAGES ISSUED:                                           *
*                                                            *
*   NONE                                                     *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
*                                                            *
* REGISTER USAGE:                                            *
*                                                            *
*   REGISTER 0-1   WORK REGISTER (MACRO EXPANSION)           *
*   REGISTER 2-4   WORK REGISTER                             *
*   REGISTER 5-6   BASE REGISTER                             *
*   REGISTER 7-11  WORK REGISTER                             *
*   REGISTER 12    SYSLOG INPUT AREA ADDRESS                 *
*   REGISTER 13    SAVE AREA ADDRESS FOR SUBROUTINE CALL     *
*   REGISTER 14    LINK REGISTER                             *
*   REGISTER 15    WORK REGISTER                             *
*                                                            *
* ---------------------------------------------------------- *
```

```
* ---------------------------------------------------------- *
*                                                            *
* SEQUENCE OF OPERATION:                                     *
*                                                            *
*   BEGIN:    LOAD BASE REGISTERS / INITIALIZATION           *
*             FOR PROMPTING MODE:                            *
*               ISSUE MENU 1 FOR REQUIRED FUNCTION           *
*             FOR PARAMETER MODE:                            *
*               CHECK OPERANDS FOR REQUIRED FUNCTION         *
*             BRANCH TO THE APPROPRIATE ROUTINE              *
*                                                            *
*   CREATION: DUMP PROGRAM CREATION (IJBXDM4)                *
*                                                            *
*   STOC:     OTHER OPTIONS (PRINT DUMP AND SDAID TAPE).     *
*             SET OC EXIT ROUTINE                            *
*                                                            *
*   DUMPEOJ:  CLOSE FILES AND RETURN TO EOJ                  *
*                                                            *
*   PACKRT:   PACK AND CHECK A DEVICE ADDRESS                *
*                                                            *
*   FINDFILE: FIND FILE NUMBER ON TAPE                       *
*                                                            *
*   GETDEVIC: CHECK SYSNNN OR CUU (USING PACKRT)             *
*                                                            *
* ---------------------------------------------------------- *
```

IJBXDM1 - DOSVSDMP Dump Device I/O Module

```
*/* ------------------------------------------------------ */
*/*                                                        */
*/* MODULE NAME:   IJBXDM1                                 */
*/*                                                        */
*/* PHASE NAME :   DOSVSDMP                                */
*/*                                                        */
*/* LINKBOOK   :   IJBMDUXL                                */
*/*                                                        */
*/* DESCRIPTIVE NAME:                                      */
*/*                                                        */
*/*   DUMP UTILITY PROGRAM DOSVSDMP                        */
*/*   (MODULE IJBXDM1 - DUMP DEVICE I/O MODULE)            */
*/*                                                        */
*/* FUNCTION  :   PERFORM ALL I/O FOR THE DUMP DEVICE      */
*/*                                                        */
*/* ------------------------------------------------------ */

*/* ------------------------------------------------------ */
*/*                                                        */
*/* ENTRY POINT:   IJBXDM1                                 */
*/*                                                        */
*/* CALLED BY  :   IJBXDMP, IJBXDM5 VIA THE MACRO IJBXLIO  */
*/*                                                        */
*/* MODULES CALLED:                                        */
*/*                                                        */
*/*   IJBXDM2   FOR PRINTING ON SYSLST AND SYSLOG          */
*/*                                                        */
*/* EXIT-NORMAL:   TO CALLER                               */
*/*                                                        */
*/*   RETURN CODE|REASON CODE:                             */
*/*                                                        */
*/*        0        -         COMPLETED SUCCESSFULLY       */
*/*        4        1         GETDATA: EOF RECORD (DISK)   */
*/*        4        2         GETDATA: TM OR END-OF EXTENT */
*/*        4        3         POSITION: EOF RECORD (DISK)  */
*/*        4        4         POSITION: END-OF EXTENT      */
*/*        8        -         END OF TAPE/DISK             */
*/*       12        -         WRONG LENGTH RECORD          */
*/*       16        -         COMMAND SEQUENCE             */
*/*       20        1         INVALID FUNCTION CALL        */
*/*       20        2         INVALID DUMP DATA TYPE       */
*/*                                                        */
*/* EXIT-ERROR :   NONE                                    */
*/*                                                        */
*/* ------------------------------------------------------ */

*/* ------------------------------------------------------ */
*/*                                                        */
*/* MESSAGES CAUSED:                                       */
*/*                                                        */
*/*   4G28I   DUMP FILE IS EMPTY                           */
*/*   4G29I   SDAID FILE IS EMPTY                          */
*/*                                                        */
*/* MESSAGES ISSUED: NONE                                  */
*/*                                                        */
*/* INPUT      :   DMPSCTX AS GLOBAL CONTROL BLOCK         */
*/*                                                        */
*/* OUTPUT     :   NONE                                    */
*/*                                                        */
*/* REGISTER CONVENTIONS:                                  */
*/*                                                        */
*/*   PLS LINKAGE CONVENTIONS                              */
*/*   REGISTER 12 IS PRIMARY BASE REGISTER                 */
*/*                                                        */
*/* MACROS USED  :    LIOCS MACROS FOR DUMP DEVICE I/O     */
*/*                                                        */
*/* ------------------------------------------------------ */
```

```
*/* ------------------------------------------------------- */
*/*                                                         */
*/* SEQUENCE OF OPERATION:                                  */
*/*                                                         */
*/*   IJBXDM1  : ENTRY POINT.                               */
*/*              INITIATE DTF FOR SDAID OR DUMP DEVICE.     */
*/*              ANALYZE THE REQUESTED FUNCTION.            */
*/*              DEPENDING ON THE FUNCTION CODE DO:         */
*/*                                                         */
*/*   TAPOPEN  : OPEN DUMP FILE.                            */
*/*              REWIND TAPE, OR INITIALIZE DISK EXTENTS    */
*/*                                                         */
*/*   TAPCLOSE : CLOSE DUMP FILE                            */
*/*                                                         */
*/*   TAPNEXT  : READ NEXT DUMP RECORD                      */
*/*                                                         */
*/*   TAPDATA  : READ DATA RECORD                           */
*/*                                                         */
*/*   TAPPOS   : POSITION TO FILE START                     */
*/*              CHECK VALIDITY OF SA, AR DUMP, E.G. THE    */
*/*              FIRST RECORD MUST BE A SYMPTOM RECORD      */
*/*                                                         */
*/* ------------------------------------------------------- */
```

IJBXDM2 - DOSVSDMP Message Writer Module

```
* ---------------------------------------------------------- *
*                                                            *
* MODULE NAME  :   IJBXDM2                                   *
*                                                            *
* PHASE NAME   :   DOSVSDMP                                  *
*                                                            *
* LINKBOOK     :   IJBMDUXL                                  *
*                                                            *
* DESCRIPTIVE NAME:                                          *
*                                                            *
*   DUMP UTILITY PROGRAM DOSVSDMP                            *
*   (MODULE IJBXDM2 - MESSAGE WRITER MODULE)                 *
*                                                            *
* FUNCTION:                                                  *
*                                                            *
*   THIS IS THE MESSAGE WRITER PHASE OF THE DUMP UTILITY     *
*   PROGRAM DOSVSDMP. IT IS CALLED TO DISPLAY MESSAGES       *
*   ON SYSLOG OR SYSLST                                      *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
*                                                            *
* ENTRY POINT  :   IJBXDM2                                   *
*                                                            *
* CALLED BY    :   IJBXDMP                                   *
*                  IJBXDM1                                   *
*                  IJBXDM5                                   *
*                                                            *
* NORMAL EXIT  :   RETURN TO CALLER                          *
*                                                            *
* ERROR EXIT   :   NONE                                      *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
*                                                            *
* INPUT        :   GLOBAL DATA AREA DMPSCTX                  *
*                                                            *
* OUTPUT       :   MESSAGE ON SYSLOG OR SYSLST               *
*                                                            *
* MACROS USED  :   LOGICAL IOCS MACROS                       *
*                                                            *
* REGISTERS:                                                 *
*                                                            *
*   R4   ADDRESS OF INTERFACE AREA DMPSCTX                   *
*   R5   BASE REGISTER                                       *
*   RF   RETURN CODE                                         *
*                                                            *
* MESSAGES CAUSED:                                           *
*                                                            *
*   NONE                                                     *
*                                                            *
* MESSAGES ISSUED:                                           *
*                                                            *
*   ALL MESSAGES CAUSED BY THE DUMP UTILITY PROGRAM DOSVSDMP *
*   ARE ISSUED BY THIS MODULE                                *
*                                                            *
* ---------------------------------------------------------- *
```

```
* ---------------------------------------------------------- *
*                                                            *
* SEQUENCE OF OPERATION:                                     *
*                                                            *
*   IJBXDM2 :  ESTABLISH BASE REGISTERS                      *
*                                                            *
*   BEGIN   :  DEPENDING ON THE FUNCTION CODE BRANCH TO ONE  *
*              OF THE FOLLOWING ROUTINES                     *
*                                                            *
*   OPENROUT:  OPEN SYSLST                                   *
*                                                            *
*   CLOSROUT:  CLOSE SYSLST                                  *
*                                                            *
*   PSYSLST :  SYSLST OUTPUT ROUTINE                         *
*                                                            *
*   MSYSLST :  DISPLAY THE REQUESTED MESSAGE ON SYSLST       *
*                                                            *
*   RSYSLOG :  READ THE OPERATOR RESPONSE TO SELECTION MENU  *
*              FROM SYSLOG                                   *
*                                                            *
*   MSYSLOG :  DISPLAY THE REQUESTED MESSAGE ON SYSLOG       *
*                                                            *
*   LOGLINE :  DISPLAY THE CONTENTS OF THE SYSLOG OUT AREA   *
*                                                            *
* ---------------------------------------------------------- *
```

IJBXDM4 - DOSVSDMP Installation Module

```
* ------------------------------------------------------------ *
*                                                              *
* MODULE NAME  :   IJBXDM4                                     *
*                                                              *
* PHASE NAME   :   DOSVSDMP                                    *
*                                                              *
* LINKBOOK     :   IJBMDUXL                                    *
*                                                              *
* DESCRIPTIVE NAME:                                            *
*                                                              *
*   DUMP UTILITY PROGRAM DOSVSDMP                              *
*   (MODULE IJBXDM4 - INSTALLATION PROGRAM)                    *
*                                                              *
* FUNCTION     : CREATES STAND-   ALONE DUMP PROGRAM ON        *
*                DISK (CKD OR FBA) OR ON TAPE                  *
*                                                              *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
*                                                              *
* ENTRY POINT  :   IJBXDM4                                     *
*                                                              *
* CALLED BY    :   IJBXDMP                                     *
*                                                              *
* EXIT NORMAL  :   RETURN TO CALLER                            *
*                                                              *
* EXIT ERROR   :   NONE                                        *
*                                                              *
* MACROS USED:                                                 *
*                                                              *
*   DMPSCTX, IJBXRC                                            *
*   LOAD, CCB, ASYSCOM, SYSCOM, EXTRACT ID=DVTY,               *
*   EXCP, WAIT, MAPCCB, SGSRVLVL, GENL                         *
*                                                              *
* REGISTER USAGE:                                              *
*                                                              *
*   REGISTER 0-1   WORK REGISTER (MACRO EXPANSION)             *
*   REGISTER 2     WORK REGISTER                               *
*   REGISTER 3-5   BASE REGISTER                               *
*   REGISTER 6-9   WORK REGISTER                               *
*   REGISTER 10    BASE REGISTER (INTERFACE AREA)              *
*   REGISTER 11-15 WORK REGISTER                               *
*                                                              *
* INPUT        :   DMPSCTX                                     *
*                                                              *
* OUTPUT       :   NONE                                        *
*                                                              *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
*                                                              *
* SEQUENCE OF OPERATION:                                       *
*                                                              *
*   PROGRAM ENTRY                                              *
*   READ ENTRY INFORMATION                                     *
*   IF DUMP DEVICE IS TAPE CALL TAPE ROUTINE                   *
*   CALL $$BSYSWR                                              *
*   GET CUU AND DEVICE TYPE OF SYSLOG DEVICE                   *
*                                                              *
* ------------------------------------------------------------ *
```

IJBXDM5 - DOSVSDMP Printout Module

```
* ------------------------------------------------------------ *
*                                                              *
* MODULE NAME  :   IJBXDM5                                     *
*                                                              *
* PHASE NAME   :   DOSVSDMP                                    *
*                                                              *
* LINKBOOK     :   IJBMDUXL                                    *
*                                                              *
* DESCRIPTIVE NAME:                                            *
*                                                              *
*   DUMP UTILITY PROGRAM DOSVSDMP                              *
*   (MODULE IJBXDMP - MAIN MODULE)                             *
*                                                              *
* FUNCTION     :   FORMAT AND PRINT SDAID TAPES                *
*                  PRINT DUMP FILES UNFORMATTED                *
*                  PRINT IPL DIAGNOSTICS                       *
*                                                              *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
*                                                              *
* ENTRY POINT  :   IJBXDM5                                     *
*                                                              *
* CALLED BY    :   IJBXDMP                                     *
*                                                              *
* NORMAL EXIT  :   RETURN TO CALLER                            *
*                                                              *
* ERROR  EXIT  :   NONE                                        *
*                                                              *
* PHASES FETCHED:                                              *
*                                                              *
*   IJSDDEB   SDAID ENTRY ROUTINE                              *
*   IJSDPWB   SDAID DEBLOCKING ROUTINE                         *
*   IJSDCVT   SDAID CONVERT ROUTINE                            *
*   IJSDNEM   SDAID ROUTINE FOR MNEMONIC OP                    *
*                                                              *
* MODULES CALLED:                                              *
*                                                              *
*   IJBXDM1   FOR TAPE I/O                                     *
*   IJBXDM2   FOR PRINTING ON SYSLST AND SYSLOG                *
*                                                              *
* MESSAGES CAUSED:                                             *
*                                                              *
*   4G17I   PRINTOUT CANCELED BY THE OPERATOR                  *
*   4G18I   FORMAT OF SDAID BUFFER INCORRECT                   *
*   4G20I   SDAID DEBLOCKING ROUTINES CANNOT BE LOADED         *
*                                                              *
* MESSAGES ISSUED:                                             *
*                                                              *
*   NONE                                                       *
*                                                              *
* ------------------------------------------------------------ *
```

```
* ------------------------------------------------------- *
*                                                         *
* REGISTER USAGE:                                         *
*                                                         *
*   R2   ADDRESS OF INPUT RECORD                          *
*   R3   SYSLST OUTPUT AREA                               *
*   R4   BASE REGISTER                                    *
*   R5   COMMON INTERFACE AREA ADDRESS                    *
*   RD   SAVE AREA ADDRESS                                *
*                                                         *
* INPUT        :   SDAID TAPE OR DUMP TAPE                *
*                  GLOBAL DATA AREA DMPSCTX               *
*                                                         *
* OUTPUT       :   PRINT OUTPUT                           *
*                                                         *
* SEQUENCE OF OPERATION:                                  *
*                                                         *
*   IJBXDM5 :   INITIALIZE BASE REGISTERS                 *
*                                                         *
*   BEGIN   :   DEPEND ON THE FUNCTION BYTE BRANCH        *
*               TO PRTDUMP OR SDAIDRT                     *
*                                                         *
*   PRTDUMP :   PRINT DUMP TAPE                           *
*                                                         *
*   SDAIDRT :   PRINT SDAID TAPE                          *
*                                                         *
*   SCANROUT:   SCAN DUMP TAPE/DISK AND PRINT REPORT OF   *
*               ITS CONTENTS                              *
*                                                         *
* ------------------------------------------------------- *
```

Stand-Alone Dump Program Module Descriptions

IJBXDM7 - SA Dump Program Base Module

```
* ---------------------------------------------------------- *
*                                                            *
* PHASE NAMES:                                               *
*                                                            *
*   $$A$DMP1:   IPL BOOTSTRAP RECORDS FOR TAPE               *
*   $$A$DMP2:   IPL BOOTSTRAP RECORDS FOR CKD DISK           *
*   $$A$SMP3:   IPL BOOTSTRAP RECORDS FOR FBA DISK           *
*   $$A$SMP4:   DUMMY PHASE                                  *
*   DOSVSDX7:   MAIN PROGRAM TO DUMP VSE TO TAPE OR DISK     *
*                                                            *
* MODULE NAME:  IJBXDM7                                      *
*                                                            *
* ENTRY POINT:  DMPENTRY                                     *
*                                                            *
* FUNCTION:     STAND-ALONE DUMP PROGRAM (MAIN MODULE)       *
*                                                            *
* SEQUENCE OF OPERATION:                                     *
*                                                            *
*   WRITE 6 4K PAGES TO WORK FILE VIA IPL BOOTSTRAP CCW'S    *
*   LOAD AND INITIALIZE DOSVSDX7                             *
*   WRITE SYMPTOM RECORD                                     *
*   COPY THE SAVED 6 PAGES TO THE DUMP DATA FILE             *
*   DUMP THE SUPERVISOR                                      *
*   SHIFT IJBXDM7 TO THE FLIH AREA                           *
*   LOAD DOSVSDX8, DOSVSDX9 AND DOSVSDXA                     *
*   CALL IJBXDM8 TO WRITE CONTROL BLOCK RECORDS              *
*   CALL IJBXDM9 TO WRITE HARDCOPY FILE RECORDS              *
*   DUMP THE SDAIDS AREA AND THE SVA                         *
*   DUMP THE PAGE MANAGER ADDRESS SPACES                     *
*   DUMP PARTITIONS AND DATASPACES ACCORDING TO THEIR        *
*     SA DUMP OPTION SETTING                                 *
*   IF DUMP PROGRAM IS ON SYSRES: CALL $$A$PLBX TO IPL VSE   *
*   OTHERWISE: HARDWAIT                                      *
*                                                            *
* SUBROUTINES:                                               *
*                                                            *
*   POSSYM ..... DUMP DATA FILE POSITIONING                  *
*   SHLDSUBR ... SHIFT DM7 AND LOAD DM8, DM9 AND DMA         *
*   PRIOSUBR ... DUMP PARTITIONS AND DATA SPACES             *
*   CONSSUBR ... HANDLE CONSOLE MESSAGE                      *
*   SIOSUBR .... HANDLE I/O (XA AND NON-XA)                  *
*   SYMRSUBR ... CREATE SYMPTOM RECORDS                      *
*   TAPESUBR ... TAPE DUMP FILE I/O AND ERROR RECOVERY       *
*   DISKSUBR ... DISK DUMP FILE I/O AND ERROR RECOVERY       *
*   PDSSUBR  ... PAGE DATA SET I/O AND ERROR RECOVERY        *
*                                                            *
* ---------------------------------------------------------- *
```

```
* ------------------------------------------------------- *
*                                                         *
* CALLED BY:    THE STAND-ALONE DUMP PROGRAM IS IPL'ED    *
*               (BY HARDWARE OR SOFTWARE RE-IPL)          *
*                                                         *
* CALLS TO:     DOSVSDX8 (MODULE: IJBXDM8)                *
*               THIS PHASE CREATES SYMPTOM RECORD SECTION  *
*               6 ENTRIES OF VSE CONTROL BLOCKS           *
*                                                         *
*               DOSVSDX9 (MODULE: IJBXDM9)                *
*               THIS PHASE CREATES SYMPTOM RECORD SECTION  *
*               6 ENTRIES OF VSE HARDCOPYFILE MESSAGES    *
*                                                         *
*               DOSVSDXA (MODULE: IJBXDM10)               *
*               THIS PHASE DUMPS THE VIRTUAL STUFF        *
*                                                         *
*               NOTE: THESE PHASES ARE READ FROM THE DUMP *
*               PROGRAM FILE ON THE DUMP TAPE OR DISK     *
*                                                         *
* MESSAGES:     4G06I CLEARED STORAGE FOUND, NO DUMP TAKEN *
*               4G07I END OF EXTENT ON STAND-ALONE DUMP DISK *
*               4G08I DUMP DATA FILE NOT FORMATTED        *
*               4G10I STAND-ALONE DUMP COMPLETE           *
*               4G34I STAND-ALONE DUMP IN PROGRESS ON TAPE | *
*                     DISK CUU                            *
*               4G35I PROBLEM ENC OUNTERED DURING SA DUMP *
*                     PROCESSING. REASON CODE NNNN        *
*               4G36I END OF VOLUME ON DUMP TAPE CUU. MOUNT *
*                     NEW TAPE OR RE-IPL VSE              *
*               4G37I ERROR ON DUMP TAPE CUU. MOUNT NEW TAPE *
*                     OR RE-IPL VSE                       *
*               4G40I VSE IPL IN PROGRESS                 *
*               4G44I PERMANENT ERROR ON DUMP DEVICE      *
*                                                         *
* INPUT:        NO DIRECT INPUT                           *
*                                                         *
*               ALL NEEDED INPUT DATA IS PLACED IN AN     *
*               INTERFACE AREA (DMPSCTX) BY DOSVSDMP WHEN *
*               THE SA DUMP PROGRAM IS CREATED ON THE SA  *
*               DUMP TAPE OR DISK. THE INTERFACE AREA IS  *
*               PART OF MODULE IJBXDM7 (CSECT)            *
*                                                         *
* OUTPUT:       DUMP ON TAPE OR DISK                      *
*                                                         *
* NORMAL EXIT:  IF DUMP IPL'ED FROM SYSRES                *
*               --> IPL OF SYSRES                         *
*               IF DUMP IPL'ED FROM NON-SYSRES DISK OR TAPE *
*               --> HARDWAIT (PSW X'000A000000CE0000')    *
*                                                         *
* ------------------------------------------------------- *
```

```
* ---------------------------------------------------------- *
*                                                            *
* ERROR EXITS:  HARDWAITS:                                   *
*                                                            *
*   PSW = X'000A000000CE0001'                                *
*         ... I/O ERROR ON SIO / SSCH                        *
*   PSW = X'000A000000CE0002'                                *
*         ... DEVICE NOT OPERATIONAL                         *
*   PSW = X'000A000000CE0004'                                *
*         ... CHANNEL ERROR                                  *
*   PSW = X'000A000000CE0008'                                *
*         ... PERMANENT I/O ERROR                            *
*   PSW = X'000A000000CE0010'                                *
*         ... I/O ERROR DURING ERROR RECOVERY                *
*   PSW = X'000A000000CE0020'                                *
*         ... UNRECOVERABLE TAPE ERROR                       *
*   PSW = X'000A000000CE0040'                                *
*         ... CONSOLE I/O ERROR                              *
*   PSW = X'000A000000CE0080'                                *
*         ... END OF EXTENT ON SA DUMP DISK                  *
*   PSW = X'000A000000CE0100'                                *
*         ... I/O ERROR ON TAPE IPL                          *
*   PSW = X'000A000000CE0400'                                *
*         ... PROGRAM CHECK DURING IPL                       *
*   PSW = X'000A000000CE0800'                                *
*         ... PROG CHECK PREPARING VIRTUAL DUMP              *
*   PSW = X'000A000000CE1000'                                *
*         ... PCK DUMPING VIRTUAL STORAGE IN IJBXDM10        *
*   PSW = X'000A000000CE2000'                                *
*         ... PCK SHIFTING IJBXDM7                           *
*   PSW = X'000A000000CE4000'                                *
*         ... PCK IN IJBXDM8 OR IJBXDM9                      *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
*                                                            *
* DATA AREAS:                                                *
*                                                            *
*   DMPSCTX .... DATA AREA DOSVSDMP VS SA DUMP               *
*   IJBXINT7 ... INTERNAL DATA AREA SA DUMP                  *
*   IJBXRC ..... DUMP FILE RECORD FORMAT                     *
*                                                            *
*   SGLOWC ..... LAY-OUT OF LOW CORE LOCATION                *
*   SGSRVLVL ... SUPERVISOR LEVEL MACRO                      *
*   ADSSR ...... SYMPTOM RECORD MAP                          *
*   ADSLBD ..... SCT 6 LBD MAP                               *
*   ADSLBXT .... SCT 6 TEXT EXTENSION                        *
*   ADSLBXX .... SCT 6 HEX DATA EXTENSION                    *
*   MAPCLIM .... DYN CLASS & SYS LIMITS TABLE                *
*   MAPCOMR .... COMREG                                      *
*   MAPDEVTY ... DEVICE TYPE CODES                           *
*   IRB ........ INTERRUPTION REQUEST BLOCK                  *
*   INLCLPT .... LIBRARY POINTER TABLE                       *
*   ORB ........ OPERATION REQUEST BLOCK                     *
*   MAPPCB ..... PARTITION CONTROL BLOCK                     *
*   MAPPIB ..... PARTITION INFORMATION BLOCK                 *
*   MAPSAVAR ... SAVE AREA                                   *
*   MAPSCB ..... SPACE CONTROL BLOCK                         *
*   MAPTCB ..... TASK CONTROL BLOCK                          *
*   MAPTDSE .... TCB DATA SPACE EXTENSION                    *
*   MAPTIB ..... TASK INFORMATION BLOCK                      *
*   SCBATAB .... SCB ADDRESS TABLE                           *
*   SCHIB ...... SUBCHANNEL INFORMATION BLOCK                *
*   SMCOM ...... STORAGE MANAGEMENT CB                       *
*   SYSCOM ..... SYSTEM COMMUNICATIONS REGION                *
*   UNATTCB .... UNATTENDED NODE CB                          *
*                                                            *
* ---------------------------------------------------------- *
```

```
* ---------------------------------------------------------- *
* IPL PHASES ($$A$DMP1, $$A$DMP2, AND $$A$DMP3) RECORD 1     *
* ---------------------------------------------------------- *
*                                                            *
*   CALLED BY IPL OF THE SYSTEM                              *
*                                                            *
*   LOADED INTO LOW STORAGE AT X'0'                          *
*                                                            *
*   FUNCTIONS:                                               *
*                                                            *
*     1. LOAD IPL2 USING INITIAL CCW'S                       *
*                                                            *
*        IPL RECORD 1 HAS THREE PARTS:                       *
*                                                            *
*         - PSW  START EXECUTION                             *
*              R=0  NO PROGRAM EVENT RECORDING               *
*              T=0  DAT IS OFF                               *
*             I/O=0  I/O INTERRUPT DISABLED                  *
*              E=0  EXTERNAL INTERRUPT DISABLED              *
*              KEY  KEY IS ZERO                              *
*              C=1  EXTENDED CONTROL MODE                    *
*              M=0  MACHINE CHECK DISABLED                   *
*              W=0  NO WAIT STATE                            *
*              P=0  SUPERVISOR STATE                         *
*                                                            *
*         - CCW  READ NEXT RECORD INTO LOCATION              *
*                SADUMPLA (X'300')                           *
*                                                            *
*         - CCW  TIC TO LOCATION SADUMPLA (X'300')           *
*                                                            *
* ---------------------------------------------------------- *

* ---------------------------------------------------------- *
* IPL PHASES ($$A$DMP1, $$A$DMP2, AND $$A$DMP3) RECORD 2     *
* ---------------------------------------------------------- *
*                                                            *
*   CALLED BY IPL RECORD 1 BOOTSTRAP                         *
*                                                            *
*   LOADED INTO LOW STORAGE AT X'300'                        *
*                                                            *
*   FUNCTIONS:                                               *
*                                                            *
*     1. WRITE FIRST 6 PAGES OF LOW STAORAGE TO WORK FILE    *
*                                                            *
*     2. READ IN IJBXDM7 FROM DUMP IPL DEVICE                *
*                                                            *
* ---------------------------------------------------------- *
```

```
* ------------------------------------------------------------ *
* FORMAT OF OUTPUT - FIXED PART: TAPE                          *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
* FILE   | REC  | RECORD |                                     *
* NUMBER | NUMB | LENGTH | CONTENTS                            *
* -------+------+--------+------------------------------------ *
*    1   |  1   |     24 | $$A$DMP1 - IPL BOOTSTRAP            *
* -------+------+--------+------------------------------------ *
*    1   |  2   |     48 | TAPEIPL2 - CCW TO WRT 1ST 6 PAGES   *
* -------+------+--------+------------------------------------ *
*    1   |  3   |  (15K) | DOSVSDX7 - SA DUMP PROGRAM          *
* -------+------+--------+------------------------------------ *
*    1   |  4   |   (2K) | DOSVSDX8 - SECTION 6 RECORD PROG    *
* -------+------+--------+------------------------------------ *
*    1   |  5   |   (6K) | DOSVSDX9 - HARDCOPY FILE PROGRAM    *
* -------+------+--------+------------------------------------ *
*    1   |  6   |   (2K) | DOSVSDXA - VIRTUAL STUFF PROGRAM    *
* -------+------+--------+------------------------------------ *
*        |      |        | TAPE MARK                           *
* -------+------+--------+------------------------------------ *
*    2   |  1   |   4096 | DUMPED STORAGE X'0000' - X'0FFF'    *
* -------+------+--------+------------------------------------ *
*    2   |  2   |   4096 | DUMPED STORAGE X'1000' - X'1FFF'    *
* -------+------+--------+------------------------------------ *
*    2   |  3   |   4096 | DUMPED STORAGE X'2000' - X'2FFF'    *
* -------+------+--------+------------------------------------ *
*    2   |  4   |   4096 | DUMPED STORAGE X'3000' - X'3FFF'    *
* -------+------+--------+------------------------------------ *
*    2   |  5   |   4096 | DUMPED STORAGE X'4000' - X'4FFF'    *
* -------+------+--------+------------------------------------ *
*    2   |  6   |   4096 | DUMPED STORAGE X'5000' - X'5FFF'    *
* -------+------+--------+------------------------------------ *
*        |      |        | TAPE MARK                           *
* -------+------+--------+------------------------------------ *

* ------------------------------------------------------------ *
* FORMAT OF OUTPUT - FIXED PART: CKD DISK                      *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
* DISK   |      |        |                                     *
* ADDR   | REC  | RECORD |                                     *
* CC HH  | NUMB | LENGTH | CONTENTS                            *
* -------+------+--------+------------------------------------ *
* 00 00  |  0   |     24 | $$A$DMP2 - IPL BOOTSTRAP            *
* -------+------+--------+------------------------------------ *
* 00 00  |  1   |     48 | CKDIPL2  - CCW TO WRT 1ST 6 PAGES   *
* -------+------+--------+------------------------------------ *
* 00 02  |  1   |  (15K) | DOSVSDX7 - SA DUMP PROGRAM          *
* -------+------+--------+------------------------------------ *
* 00 03  |  1   |   (2K) | DOSVSDX8 - SECTION 6 RECORD PROG    *
* -------+------+--------+------------------------------------ *
* 00 03  |  2   |   (6K) | DOSVSDX9 - HARDCOPY FILE PROGRAM    *
* -------+------+--------+------------------------------------ *
* 00 03  |  3   |   (2K) | DOSVSDXA - VIRTUAL STUFF PROGRAM    *
* -------+------+--------+------------------------------------ *
* 00 04  |  1   | 12,288 | DUMPED STORAGE X'0000' - X'2FFF'    *
* -------+------+--------+------------------------------------ *
* 00 05  |  1   | 12,288 | DUMPED STORAGE X'3000' - X'5FFF'    *
* ------------------------------------------------------------ *
```

```
* ------------------------------------------------------------ *
* FORMAT OF OUTPUT - FIXED PART: FBA DISK                      *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
* DISK   |      |        |                                      *
* ADDR   | REC  | RECORD |                                      *
* BLOCK  | NUMB | LENGTH | CONTENTS                             *
* -------+------+--------+------------------------------------ *
*    0   |  0   |     24 | $$A$IPL3 - IPL BOOTSTRAP             *
* -------+------+--------+------------------------------------ *
*    0   |  1   |     48 | FBAIPL2  - CCW TO WRT 1ST 6 PAGES    *
* -------+------+--------+------------------------------------ *
*   18   |  1   |  (15K) | DOSVSDX7 - SA DUMP PROGRAM           *
* -------+------+--------+------------------------------------ *
*        |  2   |   (2K) | DOSVSDX8 - SECTION 6 RECORD PROG     *
* -------+------+--------+------------------------------------ *
*        |  3   |   (6K) | DOSVSDX9 - HARDCOPY FILE PROGRAM     *
* -------+------+--------+------------------------------------ *
*        |  3   |   (2K) | DOSVSDXA - VIRTUAL STUFF PROGRAM     *
* -------+------+--------+------------------------------------ *
*        |  1   | 24,576 | DUMPED STORAGE X'0000' - X'5FFF'     *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
* FORMAT OF OUTPUT - VARIABLE PART (1/2)                       *
*                    SUPERVISOR, SDAID AREA AND SVA            *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
*        |      |        |                                      *
*        | REC  | RECORD |                                      *
*        | NUMB | LENGTH | CONTENTS                             *
* -------+------+--------+------------------------------------ *
*        |  1   |   4112 | SYMPTOM RECORD                       *
* -------+------+--------+------------------------------------ *
*        |  2.. |   4112 | SUPERVISOR                           *
* -------+------+--------+------------------------------------ *
*        |  ... |   4112 | SECTION 6 RECORDS FROM IJBXDM8       *
*        |      |        | (CONTROL BLOCK LBDS) AND IJBXDM9     *
*        |      |        | (HARDCOPYFILE RECORDS)               *
* -------+------+--------+------------------------------------ *
*        |  ... |   4112 | SDAID AREA, SVA                      *
* -------+------+--------+------------------------------------ *
*        |  ... |   4112 | SECTION 6 RECORD                     *
* -------+------+--------+------------------------------------ *
*        |  ... |   4112 | END OF FILE RECORD OR TAPE MARK      *
* -------+------+--------+------------------------------------ *
```

```
* ------------------------------------------------------------ *
* FORMAT OF OUTPUT - VARIABLE PART (2/2)                       *
*                    PAGE MANAGER ADDRESS SPACE FILES (DEF)    *
*                    AND PARTITION AND DATASPACE FILES         *
*                    (ACCORDING TO SA DUMP OPTION)             *
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
*        |      |         |                                    *
*        | REC  | RECORD  |                                    *
*        | NUMB | LENGTH  | CONTENTS                           *
* -------+------+---------+----------------------------------- *
*        |  1   |   4112  | SYMPTOM RECORD                     *
* -------+------+---------+----------------------------------- *
*        |  2.. |   4112  | DATA RECORDS                       *
* -------+------+---------+----------------------------------- *
*        |  ... |   4112  | SECTION 6 RECORD                   *
* -------+------+---------+----------------------------------- *
*        |  ... |   4112  | END OF FILE RECORD OR TAPE MARK    *
* -------+------+---------+----------------------------------- *

* -------+------+---------+----------------------------------- *
*        |      |   4112  | END OF VOLUME RECORD OR 2 TM'S     *
*        |      |         | AFTER THE LAST FILE                *
* ------------------------------------------------------------ *
```

IJBXDM8 - SA Dump Program Control Block Module

```
* ------------------------------------------------------- *
*                                                         *
*  PHASE NAME:          DOSVSDX8                          *
*                                                         *
*  MODULE NAME:         IJBXDM8                           *
*                                                         *
*  ENTRY POINT:         IJBXDM8                           *
*                                                         *
*  FUNCTION:            CREATE SYMPTOM RECORD SECTION 6   *
*                       ENTRIES FOR VSE CONTROL BLOCKS    *
*                                                         *
*  CALLED BY:           DOSVSDX7                          *
*                                                         *
*  MODULES CALLED:      I/O ROUTINE IN DOSVSDX7           *
*                                                         *
*  MESSAGES CAUSED:     NONE                              *
*                                                         *
*  MESSAGES ISSUED:     NONE                              *
*                                                         *
*  INPUT:               IN REGISTER R1 ADDRESS OF INTERFACE *
*                       AREA. SEE IJBXINT7 DSECT FOR LAYOUT *
*                                                         *
*                       REGISTER RD ADDRESS OF SAVEAREA   *
*                       THEN BASE FOR INTERFACE           *
*                                                         *
*  OUTPUT:              SECTION 6 ENTRIES IN BUFFER       *
*                                                         *
*  EXITS:               RETURN TO CALLER                  *
*                                                         *
*  REGISTERS USED       R1:  ADDRESS OF INTERFACE AREA    *
*                       R2:  ADDRESS OF LBD NAME FIELD    *
*                       R4:  LENGTH OF CONTROL BLOCK      *
*                       R5:  CONTROL BLOCK ADDRESS        *
*                       R6:  LENGTH OF LBD                *
*                       R8:  BRANCH AND LINK REGISTER     *
*                       RC:  BASE REGISTER                *
*                       RD:  ADDRESS OF SAVE AREA / INTERFACE *
*                            DURING EXECUTION             *
*                                                         *
*  SEQUENCE OF OPERATION:                                 *
*                                                         *
*  BEGIN:    ESTABLISH BASE REGISTERS                     *
*            SAVE CALLERS REGISTERS                       *
*                                                         *
*  WRITEREC: WRITE SECTION 6 RECORD IN BUFFER. IF THE BUFFER *
*            IS FULL, WRITE IT TO DUMP DEVICE             *
*                                                         *
*  MAKEENTR: MAKE A LBD ENTRY                             *
*                                                         *
*  ARRAYEN:  MAKE A LBXA ENTRY                            *
*                                                         *
* ------------------------------------------------------- *
```

IJBXDM9 - SA Dump Program Hardcopyfile Module

```
* ------------------------------------------------------- *
*                                                         *
*  PHASE:         DOSVSDX9                                *
*                                                         *
*  MODULE:        IJBXDM9                                 *
*                                                         *
*  FUNCTION:                                              *
*                                                         *
*       MOVE ABOUT 200 MESSEAGE LINES FROM                *
*       HARDCOPY FILE TO THE END OF SA DUMP               *
*       IN FORM OF SECTION 6 ENTRIES OF SYMPTOM RECORD    *
*                                                         *
*  EXTERNAL SERVICES:                                     *
*                                                         *
*       WRTFORC  --- WRITE CURRENT HCF BUFFER TO HCF      *
*       HCFREAD  --- READ ONE RECORD FROM HCF             *
*       SKIPHCF  --- SKIP ONE OR MORE LOG RECORD(S)       *
*       MODHCF   --- CHANGE DIRECTION OF READ             *
*       HCFUPD   --- UPDATE CURRENT DISK ADDR IN HCFCB    *
*                                                         *
*  MESSAGES CAUSED:                                       *
*                                                         *
*       4F02I HARD COPY FILE NOT OPEN                     *
*       4F03I INVALID EXTENTS IN HCFCB                    *
*       4F04I INVALID CURRENT DISK ADDRESS IN HCFCB       *
*       4F05I HARD COPY FILE DISK NOT READY|OPERATIONAL   *
*       4F06I READ ERROR DURING FIRST READ FROM HCF       *
*       4F07I INTERNAL ERROR DURING POSITIONING OF MSGPTR *
*       4F08I UNDEFINED ERROR                             *
*       4F09I ERROR DURING WRITE TO HARD COPY FILE        *
*       4F10I DIFFERENT MSG LEN FOUND - CONSOLE MSG LEN USED *
*       4F11I READ ERROR DURING READ FROM HARD COPY FILE  *
*                                                         *
*  REGISTER USAGE:                                        *
*                                                         *
*       RA  ADDR OF CRTSAV                                *
*       RB  ADDR OF HCFCB                                 *
*       RC  BASE REGISTER                                 *
*       RD  ADDR OF SAVEAREA                              *
*       RF  RETURN CODES FROM HCF-SERVICE                 *
*                                                         *
*  INPUT:                                                 *
*                                                         *
*       R01  POINTS TO INTERFACE BLOCK                    *
*       R14  RETURN ADDRESS                               *
*       R15  CALL REGISTER                                *
*                                                         *
*                                                         *
*  RETURN CODES:  R0                                      *
*                                                         *
*  RETURN:        TO CALLER                               *
*                                                         *
*  ERROR EXITS:   NONE                                    *
*                                                         *
* ------------------------------------------------------- *
```

IJBXDM10 - SA Dump Program Page Manager Module

```
* ---------------------------------------------------------- *
*                                                            *
*  PHASE NAME:         DOSVSDXA                              *
*                                                            *
*  MODULE NAME:        IJBXDM10                              *
*                                                            *
*  ENTRY POINT:        IJBXDM10                              *
*                                                            *
*  FUNCTION:           RETRIEVE PAGES FROM PAGE DATA SET     *
*                                                            *
*  CALLED BY:          DOSVSDX7                              *
*                                                            *
*  MODULES CALLED:     I/O ROUTINE IN DOSVSDX7               *
*                                                            *
*  MESSAGES CAUSED:    NONE                                  *
*                                                            *
*  MESSAGES ISSUED:    NONE                                  *
*                                                            *
*  INPUT:                                                    *
*                                                            *
*       R1        ADDRESS OF INTERFACE AREA                 *
*                 SEE IJBXINT7 DSECT FOR LAYOUT             *
*       RD        ADDRESS OF SAVEAREA / INTERFACE           *
*       LDUMPADD  START ADDRESS OF DUMP INTERVAL            *
*       HDUMPADD  END ADDRESS OF DUMP INTERVAL              *
*       SEGTABAD  ADDRESS OF SEGMENT TABLE                  *
*                                                            *
*  OUTPUT:             PAGE FROM PDS IN DUMP RECORD BUFFER   *
*                                                            *
*  EXITS:              RETURN TO CALLER                      *
*                                                            *
*  REGISTERS USED:                                           *
*                                                            *
*       R1   ADDRESS OF INTERFACE AREA                       *
*       R2   CURRENT SEGMENT TABLE ENTRY POINTER             *
*       R3   CURRENT PAGE TABLE ENTRY POINTER                *
*       R8   ADDR OF PAGE TO BE DUMPED                       *
*       R9   REL PAGE NO WITHIN SEGMENT                      *
*       RC   BASE REGISTER                                   *
*       RD   ADDRESS OF SAVE AREA /INTERFACE                 *
*                                                            *
* ---------------------------------------------------------- *
```

$$ACISS - Console Integration Synchronous Support Module

```
* ---------------------------------------------------------- *
*                                                            *
*  PHASE NAME:         $$ACISS                               *
*                                                            *
*                                                            *
*  MODULE NAME:        $$ACISS                               *
*                                                            *
*  DESCRIPTIVE NAME: CONSOLE INTEGRATION SYNCHRONOUS SUPPORT *
*                                                            *
*  FUNCTION: PROVIDES SYNCHRONOUS SUPPORT FOR CONSOLE        *
*            INTEGRATION DURING INITIALIZATION OF THE SYSTEM. *
*                                                            *
* ENTRY POINT= LOAD POINT                                    *
*                                                            *
*      LINKAGE= R15  IS ENTRY POINT                          *
*               R14  IS RETURN ADDRESS                       *
*                                                            *
* INPUT= R1 CONTAINS ADDRESS OF CIRPL                        *
*                                                            *
* OUTPUT= R15 CONTAINS RETURN CODE                           *
*                                                            *
* MESSAGES ISSUED= NONE                                      *
*                                                            *
* EXIT NORMAL= RETURN TO IJBXDM7 (RC=0                       *
*                                                            *
* EXIT ERROR=  RETURN TO IJBXDM7 (RC=8)                      *
*                                                            *
* EXTERNAL REFERENCES=                                       *
*                                                            *
*      ROUTINES= NONE                                        *
*                                                            *
*      CONTROL BLOCKS= SGLOWC                                *
*                                                            *
*                                                            *
* ---------------------------------------------------------- *
```

Chapter 11. Dump Utility and Stand-Alone Dump: Interfaces, Macros and Messages

Interface Area DMPSCTX

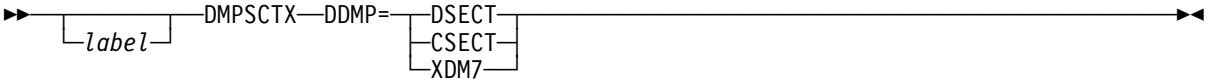
The interface area for the communication between the Dump Utility Program DOSVSDMP and the Stand-Alone Dump Program is generated by the macro DMPSCTX. DMPSCTX contains the following data:

- I/O area addresses
- return information from logical I/O module (IJBXDM1)
- device type flags and Dump Device constants
- function codes for dump retrieval
- function codes for I/O on SYSLOG and SYSLST
- equates for symbolic addressing of messages
- operation codes for printing

DMPSCTX is generated as CSECT in IJBXDMP and as DSECT in the other modules of DOSVSDMP. It is initialized and copied to the beginning of IJBXDM7 by IJBXDM4.

DMPSCTX is generated as CSECT (DDMP=XDM7) in IJBXDM7 and as DSECT in the other modules of the Stand-Alone Dump Program.

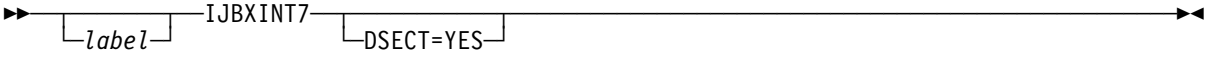
Format of DMPSCTX



Interface Area IJBXINT7

The macro IJBXINT7 contains the interface area within the Stand-Alone Dump Program (CSECT (default) in IJBXDM7, DSECT in all other modules). It contains pointers passed by IJBXDM7 to IJBXDM8, IJBXDM9 and IJBXDM10. It is also used by IJBXDM8, IJBXDM9 and IJBXDM10 to call I/O routines in IJBXDM7.

Format of IJBXINT7



DOSVSDMP Logical I/O Macro IJBXLIO

The macro IJBXLIO provides an interface to the logical dump retrieval routine, the SYSLST routine, the SYSLOG routine and the message handler routine.

All modules which use the logical I/O macro must contain the interface area DMPSCTX as DSECT or CSECT.

Format of IJBXLIO

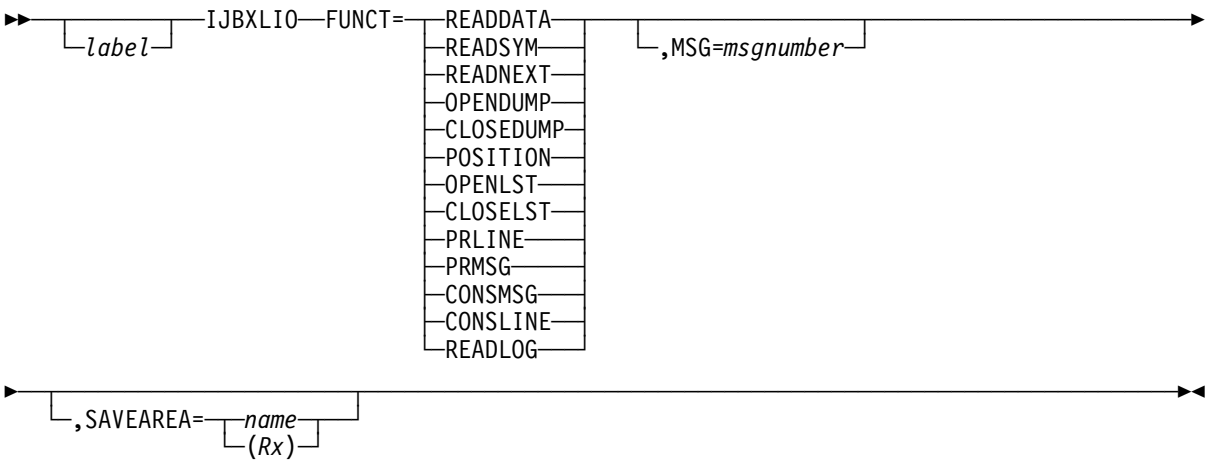


Figure 13. Description of IJBXLIO

Operand	Keyword and Description	
FUNCT	READNEXT	Reads next record
	OPENDUMP	Opens the dump medium
	CLOSEDUMP	Closes the dump medium
	POSITION	Positions tape to file start
	OPENLST	Opens SYSLST
	CLOSELST	Closes SYSLST
	PRLINE	Prints contents of print I/O area
	PRMSG	Prints a specified message
	CONSMSG	Writes a specified message or menu to the console
	CONSLINE	Writes a line on SYSLOG
	READLOG	Reads operator response from SYSLOG
MSG	*msgnumber*	Specifies a message to print on SYSLST or to display on SYSLOG (valid only if PRMSG or CONSMSG specified)
SAVAREA	*name*	Specifies the name of a register save area
	(*Rx*)	Specifies a register which contains the address of a register save area
		If the SAVAREA operand is omitted register 13 must point to a save area

Message Cross Reference

Message	Module
4G01D	IJBXDMP
4G02D	IJBXDMP
4G03I	IJBXDMP
4G04D	IJBXDMP
4G05D	IJBXDMP
4G06I	IJBXDM7
4G07I	IJBXDM7
4G08I	IJBXDM7
4G09I	IJBXDMP
4G10I	IJBXDM7
4G11I	IJBXDMP
4G12I	IJBXDMP
4G13I	IJBXDMP
4G15I	IJBXDMP
4G16I	IJBXDMP
4G17I	IJBXDM5
4G18I	IJBXDM5 (IJBXSDA)
4G19I	IJBXDMP
4G20I	IJBXDM5
4G23I	IJBXDMP
4G24I	IJBXDMP
4G25I	IJBXDMP
4G26I	IJBXDMP
4G27I	IJBXDMP
4G28I	IJBXDM1
4G29I	IJBXDM1
4G30D	IJBXDMP
4G31I	IJBXDMP
4G32I	IJBXDMP
4G33I	IJBXDMP
4G34I	IJBXDM7
4G35I	IJBXDM7
4G36I	IJBXDM7
4G37I	IJBXDM7
4G40I	IJBXDM7
4G44I	IJBXDM7
4F02I-4F11I	IJBXDM9 (HCF Macros)

Chapter 12. Dump Analysis Programs

Introduction

The VSE Dump Analysis Programs IJBXDBUG, IJBXCSMG and IJBXSDA are written to do post-dump analysis for VSE Stand-Alone Dumps.

The data produced by these programs will assist the user in recognizing errors and duplicate problems without printing a dump and without analysis work by experts.

IJBXDBUG, IJBXCSMG and IJBXSDA are implemented in the form of Exit Routines to Info/Analysis.

Stand-Alone Dump Analysis Program IJBXDBUG

INFO/ANALYSIS passes dump data to the Dump Analysis Program IJBXDBUG. Results of the dump analysis are returned to INFO/ANALYSIS for further processing.

The dump analysis routine returns the analysis data in the form of text extension entries (LBXT) entries. These LBXT entries are added to the dump member in the library. They will remain available for other INFO/ANALYSIS sessions.

The INFO/ANALYSIS user may request an overview list showing the names of all control blocks (LBD), text data (LBXT) and hex data (LBXX). The user may print the formatted dump which will include the analysis program output.

LBXT entries built by analysis routines will get a special flag to distinguish them from pre-dump analysis data. Moreover the entries built by the dump analysis routines will get the name IJBXDBUG in the component identification field to distinguish them from other post-dump analysis data. IJBXDBUG extracts environmental information, status information and error information. It looks for incorrect pointer data, inconsistent system status, bound states, etc.

IJBXDBUG supplies general information, which can be extracted from all Stand-Alone Dumps, and specific information which is only available in distinct error situations.

The following data will be supplied on all dumps:

- Service level identifier
- Supervisor ID
- Supervisor name
- Date the dump was taken
- Dump type
- System status
- Current task
- Owner of LTA and transient name (if active)
- DOC Screen image buffer (if applicable)
- ASYNOC reply status (if applicable).

IJBXDBUG Messages

The Dump Analysis Program does not produce error messages for errors in the dump. Any unusual condition detected in the Stand-Alone Dump Data is returned to INFO/ANALYSIS in the form of an LBXT entry. These 'messages' are available to the system analyst as dump analysis data. The Dump Analysis Program IJBXDBUG may produce error messages which indicate errors with the interface to INFO/ANALYSIS. These messages indicate internal errors in the Dump Analysis Program or INFO/ANALYSIS.

4G80I IJBXDBUG Analysis Output already exists for this Dump. IJBXDBUG terminated

4G82I Dump Analysis Routine "IJBXDBUG" completed successfully

4G83I IJBXDBUG Call Error. Reason Code: X

4G87I Nonzero Return Code from INFO/ANALYSIS Dump Access. Return Code: XX. Reason Code: XX

4G88I Nonzero Return Code from INFO/ANALYSIS Symptom Record Update. Section: X. Return Code: XXX. Reason Code: XXX

4G89I IJBXDBUG internal Symptom Record Update Error. Invalid Section Number: X

4G90I Internal Error in IJBXDBUG or Dump File. More than 15 LBDs built

These messages will be written by the analysis routine using the INFO/ANALYSIS print routine.

Additional information type messages:

- IJBXDBUG cannot analyze non-VSE dumps.
- IJBXDBUG cannot analyze XXXX type dumps. (XXXX equals dump type.)
- Save system data not available.
- Low core not available in this dump. IJBXDBUG cannot continue.
- SYSCOM not available in this dump. IJBXDBUG cannot continue.
- STORE STATUS not done.
- Problem program begin address is zero.
- SYSCOM indicates IPL in progress.
- Program old PSW indicates BC mode.
- Dump indicates no free channel queue entries.
- Instruction length code zero.
- SYSCOM address is zero. IJBXDBUG cannot continue.
- Comparing failing instruction against current running system indicates possible overlay, instruction currently: XXXXXX
- Unable to locate BG COMREG.
- Invalid address XXXXXX encountered during analysis.

Address of: (Area / Block name)

- Address referenced 500 times.
- Possible loop within control blocks.

IJBXDBUG Logic

IJBXDBUG is called via the INFO/ANALYSIS "call exit" function of dump viewing. INFO/ANALYSIS passes a parameter list to the exit. This list supplies the exit with the addresses of the INFO/ANALYSIS Dump Access Routine, Symptom Record Access Routine, Symptom Record Update Routine, Print Routine, and other various pointers. IJBXDBUG uses these addresses for the INFO/ANALYSIS interface. When requesting a service from INFO/ANALYSIS a parameter list is passed back to INFO/ANALYSIS for that particular service. For more information reference INFO/ANALYSIS documentation.

IJBXDBUG analyzes the dump supplied by INFO/ANALYSIS by requesting portion of the dump data using the INFO/ANALYSIS Dump Access Routine. As the data is analyzed, symptoms are placed in section 5 of the dump Symptom Record using the INFO/ANALYSIS Symptom Record Update Routine. As the section 5 symptoms are being built text data is also being built. These text data entries are passed to INFO/ANALYSIS via the Symptom Record Update Routine to be placed in section 6 of the dump. If an error occurs during section 5 update, no additional section 5 updates will be attempted. IJBXDUG will continue to build text entries for section 6. On section 5 and/or section 6 update errors, a message will be issued listing the section number, the return code, and the reason code which INFO/ANALYSIS generated.

IJBXDBUG Macros

The following macros are required to assemble IJBXDBUG:

GSXVALID Internal macro for address validation

GSXADDR Internal macro to get dump data from INFO/ANALYSIS

GSXSYMAC Internal macro to pass LBD to INFO/ANALYSIS

Additionally, lots of system macros mapping system control blocks are required for the assembly.

GSXVALID

```
* ------------------------------------------------------------------ *
* GSXVALID MACRO - USED TO VALIDATE ADDRESS'S                        *
* ------------------------------------------------------------------ *
*                                                                    *
* FORMAT - GSXVALID REG,FLAG,WORD1,WORD2,WORD3,WORDN                 *
*          REG   = REGISTER WHICH CONTAINS ADDRESS TO VALIDATE       *
*          FLAG  = MSG OR NOMSG                                      *
*                  MSG   = ISSUE INVALID ADDRESS MESSAGE IF FIELD    *
*                          VALID IS NON ZERO (VALID DESCRIBED BELOW). *
*                  NOMSG = DO NOT ISSUE MESSAGE ON NON ZERO VALID    *
*          WORD1 = NAME OF AREA POINTED TO BY REGISTER. UP TO 10     *
*                  WORDS CAN BE SPECIFIED. WORDS MUST BE SEPARATED   *
*                  BY COMMA'S. MAXIMUM LENGTH OF WORDS IS DECIMAL 32. *
*                                                                    *
* OUTPUT - REGISTER F AND FIELD 'VALID' CONTAIN RETURN CODE          *
*          RETURN CODE = 00 ADDRESS IN SUPERVISOR                    *
*                        01 ADDRESS IN OVERLAYED PART OF SUPERVISOR  *
*                           BELOW X'10', FLOG, OR DUMPAREA (OVERAREA) *
*                        02 ADDRESS IN PARTITION AREA (PARTAREA)     *
*                        04 ADDRESS IN SVA AREA (SVAAREA)            *
*                        08 ADDRESS UNKNOWN (UNKWAREA)               *
*                        10 ADDR REF'D 500 TIMES IN A ROW (LOOPFLGA) *
*                        20 AREA REF'D 1000 TIMES IN A ROW (LOOPFLGB) *
*                        CODE 10 OR 20 MAY BE COMBINED WITH OTHER    *
*                        CODES. ALL CODES ARE IN HEX.                *
*                                                                    *
* MESSAGE = INVALID ADDRESS XXXXXXXX ENCOUNTERED DURING ANALYSIS.    *
*           ADDRESS OF:  (BLOCK/AREA NAME)                           *
*                                                                    *
* ------------------------------------------------------------------ *
```

GSXADDR

```
* ------------------------------------------------------------------ *
* GSXADDR MACRO - USED TO REQUEST DATA FROM INFO/ANALYSIS (DUMP)     *
* ------------------------------------------------------------------ *
*                                                                    *
* FORMAT - GSXADDR REG,LENGTH,BUFFER                                 *
*          REG    = REGISTER WHICH CONTAINS ADDRESS OF DATA          *
*          LENGTH = LENGTH OF DATA REQUESTED.  MAXIMUM IS X'800'.    *
*          BUFFER = NAME OF BUFFER TO RETURN DATA IN OR 0            *
*                   IF 0 IS SPECIFIED THE DATA IS ONLY AVAILABLE     *
*                   UNTIL THE NEXT ENTRY TO ROUTINE GSXADDR VIA THIS *
*                   MACRO OR VIA DIRECT BRANCH.                      *
*                                                                    *
* OUTPUT - REGISTER F CONTAINS RETURN CODE                           *
*          RETURN CODE = 00 DATA AVAILABLE                           *
*                        0F DATA NOT AVAILABLE                       *
*                                                                    *
*          ADDRESS OF THE BUFFER CONTAINING THE DATA IS RETURNED IN  *
*          THE REGISTER SPECIFIED IN 'REG' PARAMETER.                *
*                                                                    *
* MESSAGE = ALL OR PART OF REQUESTED DATA NOT AVAILABLE FROM         *
*           DUMP FILE.                                               *
*           ADDRESS OF REQUESTED DATA: XXXXXXXX                      *
*           LENGTH OF REQUESTED DATA: XXXXXX                         *
*           ADDRESS OF:  (BLOCK/AREA NAME)                           *
*                                                                    *
*           (IF GSXADDR ISSUES THIS MESSAGE THE BLOCK/AREA NAME      *
*           IS GOTTEN FROM THE FIELD 'LASTBLK'. THIS FIELD IS SET    *
*           BY THE GSXVALID MACRO. THEREFORE, IF THE ADDRESS         *
*           REQUESTED BY GSXADDR IS NOT THE ADDRESS OF THE LAST      *
*           AREA VALIDATED BY GSXVALID, THE FIELD 'LASTBLK' SHOULD   *
*           BE CLEARED AND SET TO THE PROPER NAME JUST PRIOR TO      *
*           ISSUING GSXADDR MACRO OR BRANCHING TO GSXADDR ROUTINE.)  *
*                                                                    *
* ------------------------------------------------------------------ *
```

GSXSYMAC

```
* ------------------------------------------------------------------- *
* GSXSYMAC MACRO - USED TO REQUEST SECT 6 SYMPTOM FROM INFO/ANALYSIS  *
* ------------------------------------------------------------------- *
*                                                                     *
* FORMAT - GSXSYMAC TYPE,LBD,BLOCK                                    *
*          TYPE  = TYPE OF REQUEST                                    *
*                  B = SECTION 6 RECORDS                              *
*                  E = ENTIRE SYMPTOM RECORD                          *
*          LBD   = TYPE OF SECTION 6 LBD REQUESTED                    *
*          BLOCK = NAME OF SECTION 6 BLOCK REQUESTED                  *
*                                                                     *
* OUTPUT - REGISTER F CONTAINS RETURN CODE                            *
*          RETURN CODE = 00 RECORD AVAILABLE                          *
*                        0F RECORD NOT AVAILABLE                      *
*                                                                     *
*          REGISTER 1 POINTS TO THE RECORD REQUESTED.                 *
*                                                                     *
* ------------------------------------------------------------------- *
```

IJBXDBUG Module Description

```
* ---------------------------------------------------------------- *
*                                                                  *
*  MODULE NAME:        IJBXDBUG                                    *
*                                                                  *
*  ENTRY POINT:        IJBXDBUG                                    *
*                                                                  *
*  FUNCTION:           DO POST DUMP ANALYSIS ON STAND ALONE DUMPS AND *
*                      SYSTEM DUMPS (DUMP COMMAND) OF THE SUPERVISOR  *
*                      PRODUCE A MINIMAL AMOUNT OF MEANINGFUL      *
*                      OUTPUT FOR PROBLEM DETERMINATION, PROBLEM   *
*                      SOURCE IDENTIFICATION, AND/OR PROBLEM       *
*                      RESOLUTION.                                 *
*                                                                  *
*  CALLED BY:          INFO/ANALYSIS CALL EXIT COMMAND             *
*                                                                  *
*  REGISTER USAGE:     R0 - N/A                                    *
*                      R1 - PARAMETER LIST POINTER                 *
*                      R2 - POINTER TO LOW CORE BUFFER             *
*                      R3 - BASE FOR SECTIONS MAIN, WAITFFF, ETC.  *
*                      R4 - BASE 1 FOR SUBROUTINE & CONSTANT SECTIONS *
*                      R5 - BASE 2 FOR SUBROUTINE & CONSTANT SECTIONS *
*                      R6 - WORK REGISTER                          *
*                      R7 - WORK REGISTER / SYSCOM                 *
*                      R8 - WORK REGISTER / TIB                    *
*                      R9 - WORK REGISTER                          *
*                      RA - WORK REGISTER                          *
*                      RB - WORK REGISTER                          *
*                      RC - WORK REGISTER                          *
*                      RD - SAVE AREA POINTER / WORK REGISTER      *
*                      RE - LINK REGISTER                          *
*                      RF - RETURN CODES & INFO/ANALYSIS ROUTINES  *
*                                                                  *
*  PATCH AREA LABEL:   PATCH AND PATCH2                            *
*                                                                  *
*  MESSAGES CAUSED:    BLN9003I                                    *
*                                                                  *
*  MESSAGES ISSUED:    4G80I THROUGH 4G90I                         *
*                                                                  *
*  INPUT:              R1 POINTS TO INFO/ANALYSIS PARAMETER LIST PTR  *
*                                                                  *
*  OUTPUT:             OUTPUT WILL BE IN ONE OF TWO FORMS,         *
*                      LBD ENTRIES(SECTION 5 AND/OR 6) OR PRINT.   *
*                      THE TYPE OF OUTPUT IS PRINT ONLY IF         *
*                      INFO/ANALYSIS DOES NOT SUPPLY A SYMPTOM RECORD *
*                      UPDATE ROUTINE FOR ADDING LBD'S TO SECTION 5   *
*                      AND 6 OF THE DUMP.                          *
*                                                                  *
*  LBD NAMES:          DBUGHDR - BASIC INFO (STATUS, ETC.)         *
*                      DBUGR07 - REGISTERS 0 THRU 7 FOR HARD WAITS    *
*                      DBUGR8F - REGISTERS 8 THRU F FOR HARD WAITS    *
*                      DBUGSCR - CRT SCREEN BUFFER IF APPLICABLE   *
*                                                                  *
*  MACRO'S USED:       GSXVALID - INTERNAL ADDRESS VALIDATION      *
*                      GSXADDR  - GET DUMP DATA FROM INFO/ANALYSIS    *
*                      GSXSYMAC - REQUEST SYMPTOM RECORD DATA      *
*                      MAPPING  - SEE SECTION SYSTEM OWNED DSECTS  *
*                                                                  *
*  NORMAL EXIT:        PLACE TERMINATION MESSAGE IN INFO/ANALYSIS  *
*                      MESSAGE BUFFER.                             *
*                      RETURN TO INFO/ANALYSIS VIA REGISTER 14.    *
*                                                                  *
*  ERROR EXIT:         PRINT ERROR MESSAGE AND/OR PLACE MESSAGE IN    *
*                      INFO/ANALYSIS MESSAGE BUFFER.               *
*                      RETURN TO INFO/ANALYSIS VIA REGISTER 14.    *
*                                                                  *
* ---------------------------------------------------------------- *
```

```
* ---------------------------------------------------------------- *
*                                                                  *
*  FLOW OF CONTROL:                                                *
*                                                                  *
*       GSXENTRY:      PROVIDE ADDRESSABILITY                      *
*                      CHECK PARAMETERS PASSED BY INFO/ANALYSIS    *
*                      MAKE SURE THAT A SUPPORTED VSE DUMP         *
*                      HAS BEEN PASSED FOR ANALYSIS.               *
*                      ON ERRORS RETURN VIA INFOERR.               *
*                                                                  *
*       READCORE:      READ IMPORTANT DATA AREAS AND CONTROL BLOCKS *
*                      DO PLAUSIBILITY CHECKING                    *
*                      SET VALUES FOR ADDRESS VALIDATION           *
*                      ON PROBABLE ERRORS PROVIDE ERROR            *
*                      MESSAGE IN LBXT FORMAT.                     *
*                                                                  *
*       CKSTSTAT:      CHECK STATUS OF DUMPED SYSTEM.              *
*                      BRANCH TO APPROPRIATE ROUTINE:              *
*                          NOWAIT:   RUNNING SYSTEM, LOOP, ETC.    *
*                          SOFTWAIT: SOFTWAIT STATE                *
*                          HARDWAIT: HARDWAIT FFF                  *
*                          NONFFF:   NON FFF HARDWAIT.             *
*                                                                  *
*       NOWAIT:        GIVE INFORMATION MESSAGE AND CURRENT PSW PTR *
*                      THEN CONTINUE VIA SOFTWAIT                  *
*                                                                  *
*       SOFTWAIT:      DISPLAY STATUS OF ACTIVE NON TP DEVICES     *
*                      DISPLAY STATUS OF ACTIVE TASKS              *
*                      ANALYZE BOUND CONDITIONS                    *
*                                                                  *
*       HARDWAIT:      DISPLAY WAIT CODE, INTERRUPT CODE,          *
*                      FAILING ADDRESS, FAILING INSTRUCTION,       *
*                      GIVE ADDITIONAL INFO IF FAILURE             *
*                      IN PTA, LTA, CRT, RTA, SVA.                 *
*                      DISPLAY CURRENT TASK,                       *
*                      DISPLAY REGISTER VALUES AND DATA.           *
*                                                                  *
*       NONFFF:        DISPLAY WAIT CODE, CURRENT TASK,            *
*                      ADDITIONAL INFORMATION FOR SPECIFIC         *
*                      TYPE WAITS (SENSE, PHASE NOT FOUND, ETC.)   *
*                      DISPLAY REGISTER VALUES AND DATA.           *
*                                                                  *
* SUBROUTINES:                                                     *
*                                                                  *
*       GSXADDR:       GET A STORAGE INTERVAL FROM INFO/ANALYSIS   *
*       GSXSYMAC:      RETRIEVE LBD ENTRIES FROM INFO/ANALYSIS     *
*       GSXSYMUP:      UPDATE LBD ENTRIES                          *
*       GSXSYMS3:      ADD A SYMPTOM TO SECTION 3                  *
*       GSXSYMS4:      ADD A SYMPTOM TO SECTION 4                  *
*       GSXSYMS5:      ADD A SYMPTOM TO SECTION 5                  *
*       REGDATA:       DISPLAY REGISTER VALUES                     *
*       FINDTASK:      FIND TASK NAME                              *
*       SVAFAIL:       CHECK FOR FAILURE IN SVA                    *
*       APARSCAN:      SCAN PHASE FOR APAR ID (DYNNNNN)            *
*       LTAACTIV:      CHECK FOR LTA ACTIVE                        *
*       ACTTID:        FIND CURRENT TASK                           *
*       CHECKORE:      CHECK FOR REPLIES OUTSTANDING               *
*       FDBGCMRG:      FIND BG COMREG                              *
*       VALADDR:       VALIDATE DUMP ADDRESS                       *
*       GSXPRT1,2,3:   PRINT DATA OR ADD DATA TO LBD               *
*       GSXPRINT:      PRINT DATA ONLY                             *
*       INFOCALL:      EXIT TO INFO/ANALYSIS FOR SERVICE/FUNCTION  *
*       INFOERR:       EXIT TO INFO/ANALYSIS ERROR MSG PASSED      *
*       INFOEXIT:      EXIT TO INFO/ANALYSIS AT IJBXDBUG TERMINATION *
*                                                                  *
* ---------------------------------------------------------------- *
```

```
* ------------------------------------------------------------------ *
* IJBXDEBUG REQUIREMENTS - SUPERVISOR AND INFO/ANALYSIS              *
* ------------------------------------------------------------------ *
*                                                                    *
* IJBXDBUG IS DEPENDENT ON THE SUPERVISOR AND INFO/ANALYSIS IN MANY  *
* KEY AREAS. THE VALUES LISTED IN THE SECTION IJBXDBUG EQU'S MUST    *
* MATCH THE SUPERVISOR AND/OR INFO/ANALYSIS. IF A VALUE IS CHANGED,  *
* REFERENCES TO THAT VALUE SHOULD BE CHECKED TO INSURE PROPER OUTPUT.*
* THE VALUES ARE LISTED TO AID IN MAKING CHANGES TO IJBXDBUG WHEN    *
* THE SUPERVISOR AND/OR INFO/ANALYSIS CHANGES.                       *
*                                                                    *
*        IJBXDBUG VALUES WHICH MUST MATCH THE SUPERVISOR             *
*                                                                    *
* DSECTS   - SECTIONS: SYSTEM OWNED DSECTS AND IJBXDBUG OWNED DSECTS *
* EQU'S    - SECTION: IJBXDBUG OWNED EQUS                            *
* TABLE'S  - SECTION: TABLES / BLOCKS                                *
* INTABLE  - TABLE OF INTERRUPTS (PGMINTC VALUES)                    *
* RUNTABLE - TABLE OF RUN CODES AND MEANING (TIBRQID VALUES)         *
* TIDBASE  - TABLE OF TASK ID'S AND NAMES (TIBRBYTE VALUES)          *
* BNDTABLE - TABLE OF RUN CODE ROUTINES (TIBRBYTE VALUES)            *
* DEVTABLE - TABLE OF DEVICE TYPES AND NAMES (PUBDEVTY VALUES)       *
* SUBSYID  - TABLE OF SUBSYSTEM NAMES (BASED ON PCBSSFLG)            *
* PIKBG    - TABLE OF PARTITION ID'S (BASED ON BG THRU FB)           *
*                                                                    *
*        IJBSDBUG VALUES WHICH MUST MATCH INFO/ANALYSIS              *
*                                                                    *
* DSECTS   - SECTIONS: IJBXDBUG OWNED DSECTS                         *
* EQU'S    - SECTION: IJBXDBUG OWNED EQUS                            *
* TABLE'S  - SECTION: TABLES / BLOCKS                                *
* ADSPXRPL - POST DUMP ANALYSIS EXIT PARAMETER LIST                  *
* ADSPXDAC - DUMP ACCESS PARAMETER LIST                              *
* ADSPXPDS - PRINT/DISPLAY PARAMETER LIST                            *
* ADSPXUSS - SYMPTOM RECORD UPDATE PARAMETER LIST                    *
* ADSPXSRS - SYMPTOM RECORD ACCESS PARAMETER LIST                    *
* ADSPXMSG - MESSAGE BUFFER LAYOUT                                   *
*                                                                    *
* ------------------------------------------------------------------ *
```

Stand-Alone Dump Analysis Routine IJBXSDA

The Info/Analysis exit routine IJBXSDA formats the SDAID buffer in a Stand-Alone Dump, if SDAID was active at the time the Stand-Alone Dump was taken. IJBXSDA decompresses the encoded SDAID records and prints them on SYSLST.

IJBXSDA Module Description

```
* ------------------------------------------------------------------ *
*                                                                    *
*  MODULE NAME:        IJBXSDA                                       *
*                                                                    *
*  ENTRY POINT:        IJBXSDA                                       *
*                                                                    *
*                                                                    *
*  FUNCTION:           IJBXSDA FORMATS THE SDAID BUFFER              *
*                      IN A STAND-ALONE DUMP                         *
*                      (IF SDAID WAS ACTIVE AT THE TIME WHEN THE     *
*                      STAND-ALONE DUMP WAS TAKEN).                  *
*                      THIS ALLOWS TO SEE THE VERY LAST ACTIONS      *
*                      OF THE SYSTEM BEFORE THE SA-DUMP WAS TAKEN.   *
*                                                                    *
*  CALLED BY:          INFO/ANALYSIS AS AN EXIT ROUTINE              *
*                                                                    *
*  INPUT PARAMETERS:   R1 POINTS TO THE ADDRESS OF THE               *
*                      INFO/ANALYSIS PARAMETER LIST ADSPXRPL.        *
*                                                                    *
*  INPUT:              IJBXSDA RETRIEVES ITS INPUT DATA FROM THE     *
*                      VSE DUMP LIBRARY VIA THE INFO/ANALYSIS        *
*                      INTERFACE ROUTINES.                           *
*                      THE ADDRESS AND THE LENGTH OF THE SDAID BUFFER *
*                      ARE TAKEN FROM A CONTROL BLOCK LOCATOR ENTRY  *
*                      (LBD ENTRY) FROM WITHIN THE STAND-ALONE DUMP. *
*                                                                    *
*  OUTPUT:             IJBXSDA SUBMITS THE PRINT RECORDS TO THE      *
*                      INFOANA PRINT/DISPLAY ROUTINE                 *
*                                                                    *
*  EXIT:               THE PROGRAM MOVES A COMPLETION MESSAGE TO THE *
*                      INFO/ANALYSIS MESSAGE BUFFER AND RETURNS TO   *
*                      INFO/ANALYSIS WITH A RETURN CODE OF ZERO.     *
*                                                                    *
*  ERROR EXIT:         THE PROGRAM MOVES AN ERROR MESSAGE TO THE     *
*                      INFO/ANALYSIS MESSAGE BUFFER AND RETURNS TO   *
*                      INFO/ANALYSIS WITH A NON-ZERO RETURN CODE.    *
*                                                                    *
*  RETURN CODES:       0  IF SDAID BUFFER WAS FORMATTED              *
*                         SUCCESSFULLY, OR THE STAND-ALONE DUMP      *
*                         DID NOT CONTAIN AN SDAID BUFFER,           *
*                      8  ERROR DURING FREEVIS (VIA INFO/ANALYSIS),  *
*                         THE SDAID BUFFER HAS BEEN FORMATTED        *
*                         SUCCESSFULLY BUT FREEVIS FAILED.           *
*                      12 THIS CODE IS RETURNED IN SEVERE ERROR      *
*                         CASES WHICH PREVENT FROM FORMATTING        *
*                         THE SDAID BUFFER.                          *
*                                                                    *
*  MESSAGES:    MESSAGES ARE PASSED TO INFO/ANALYSIS                 *
*               IN A MESSAGE BUFFER.                                 *
*                                                                    *
*               4G18I FORMAT OF SDAID BUFFER IS INCORRECT            *
*               4G21I PHASE IJSDDEB NOT FOUND                        *
*               4G21I PHASE IJSDPWB NOT FOUND                        *
*               4G21I PHASE IJSDCVT NOT FOUND                        *
*               4G21I PHASE IJSDNEM NOT FOUND                        *
*               4G22I SDAID BUFFER WAS FORMATTED SUCCESSFULLY        *
*               4G70I DUMP TO BE PROCESSED WAS NOT PRODUCED BY VSE   *
*               4G71I DUMP DOES NOT CONTAIN AN SDAID BUFFER          *
*               4G72I NO DUMP DATA FOUND FOR SDAID BUFFER            *
*               4G73I WRONG DUMP TYPE. IJBXSDA PROCESSES STAND       *
*                     ALONE DUMPS ONLY                               *
```

```
*              4G74I GETVIS FOR SDAID BUFFER FAILED                  *
*              4G75I FREEVIS FOR SDAID BUFFER FAILED                 *
*              4G76I IJBXSDA CALL ERROR. REASON CODE: X              *
*                    X=1 DUMP DOES NOT CONTAIN A SYMPTOM RECORD      *
*                    X=2 NO INFO/ANALYSIS DUMP ACCESS ROUTINE        *
*                    X=3 NO INFO/ANALYSIS PRINT ROUTINE              *
*                    X=4 NO INFO/ANALYSIS S.-R. ACCESS ROUTINE       *
*                    X=5 NO INFO/ANALYSIS GETVIS/FREEVIS ROUTINE     *
*              4G77I INFOANA PRINT ROUTINEFAILED.                    *
*                    RETURN CODE: XXX, REASON CODE: XXX              *
*                                                                    *
*  PROGRAM LOGIC:                                                    *
*            PART 1:  CHECK INFO/ANALYSIS INTERFACE:                 *
*                     IF AN ERROR IS DETECTED THEN PREPARE           *
*                     AN ERROR MESSAGE AND RETURN TO INFO/ANALYSIS:  *
*                     MESSAGES:                                      *
*                      DUMP NOT PRODUCED BY VSE            (4G70I)   *
*                      DUMP DOES NOT CONTAIN A SYMPT.REC   (4G76I-1) *
*                      DUMP ACCESS ROUTINE NOT AVAILABLE   (4G76I-2) *
*                      INFOANA PRINT ROUTINE NOT AVAILABLE (4G76I-3) *
*                      SYMPTOM ACCESS ROUTINE NOT AVAILABLE(4G76I-4) *
*                                                                    *
*            PART 2:  RETRIEVE LBD ENTRY 'SDBUFFER'.                 *
*                     CALCULATE LENGTH OF SDAID PHASES.              *
*                     ASK INFO/ANALYSIS FOR THE REQUIRED GETVIS SPACE.*
*                     LENGTH OF GETVIS SPACE =                       *
*                       LENGTH OF SDAID BUFFER  +                    *
*                       LENGTH OF PHASE IJSDDEB +                    *
*                       LENGTH OF PHASE IJSDPWB +                    *
*                       LENGTH OF PHASE IJSDCVT +                    *
*                       LENGTH OF PHASE IJSDNEM                      *
*                                                                    *
*                     IF LBD ENTRY FOR 'SDBUFFER' NOT FOUND  (4G71I) *
*                     OR THE SDAID PHASES CANNOT BE LOADED   (4G21I) *
*                     OR NO GETVIS/FREEVIS ROUTINE AVAILABLE (4G76I-5)*
*                     OR NOT ENOUGH GETVIS SPACE AVAILABLE   (4G74I) *
*                     THEN PREPARE AN ERROR MESSAGE AND              *
*                     RETURN TO INFO/ANALYSIS.                       *
*                                                                    *
*            PART 3:  PREPARE THE SDAID BUFFER AND LOAD              *
*                     THE REQUIRED SDAID PHASES.                     *
*                     IF AN ERROR IS DETECTED THEN                   *
*                     FREE THE GETVIS SPACE, PREPARE AN ERROR MESSAGE *
*                     AND RETURN TO INFO/ANALYSIS:                   *
*                                                                    *
*                     REQUEST DUMP DATA (CONTENTS OF SDAID BUFFER)   *
*                     FROM INFO/ANALYSIS AND MOVE IT TO GETVIS BUFFER *
*                     IF 'NO DUMP DATA FOR SDBUFFER'  THEN MSG 4G72I *
*                                                                    *
*                     LOAD SDAID PHASE IJSDDEB.                      *
*                     IF 'LOAD FAILURE FOR IJSDDEB'   THEN MSG 4G21I *
*                     LOAD SDAID PHASE IJSDPWB.                      *
*                     IF 'LOAD FAILURE FOR IJSDPWB'   THEN MSG 4G21I *
*                     LOAD SDAID PHASE IJSDCVT.                      *
*                     IF 'LOAD FAILURE FOR IJSDCVT'   THEN MSG 4G21I *
*                     LOAD SDAID PHASE IJSDNEM.                      *
*                     IF 'LOAD FAILURE FOR IJSDNEM'   THEN MSG 4G21I *
*                                                                    *
*                     STORE LOAD ADDRESSES OF IJSDPWB, IJSDCVT,      *
*                     AND IJSDNEM INTO THE INTERFACE AREA OF IJSDDEB *
*                                                                    *
*            PART 4:  PROCESS THE SDAID TRACE BUFFER.                *
*                                                                    *
*                     CALL IJSDDEB (FUNCTION=INIT)                   *
*                     INITIALIZE THE PRINT BUFFER                    *
*                                                                    *
*                     CALL IJSDDEB (FUNCTION=PRINT)                  *
*                     THE TRACE RECORDS ARE PASSED TO THE            *
*                     PRINT ROUTINE OF INFO/ANALYSIS.                *
*                     IF BUFFER INVALID (R.C.>0)     THEN MSG 4G18I  *
*                                                                    *
*                     FREE GETVIS SPACE.                             *
```

```
*                                                                     *
*                    IF BUFFER HAS BEEN PROCESSED SUCCESSFULLY,       *
*                    THEN SET RETURN CODE TO ZERO, AND RETURN         *
*                    MESSAGE 4G22I TO INFO/ANALYSIS.                  *
*                                                                     *
*  REGISTER USAGE:     R0 - WORK REGISTER (ESPECIALLY FOR SVC'S)       *
*                      R1 - WORK REGISTER (ESPECIALLY FOR SVC'S)       *
*                      R2 - WORK REGISTER                             *
*                      R3 - WORK REGISTER                             *
*                      R4 - WORK REGISTER                             *
*                      R5 - WORK REGISTER                             *
*                      R6 - RPL POINTER                               *
*                      R7 - WORK REGISTER                             *
*                      R8 - WORK REGISTER                             *
*                      R9 - WORK REGISTER                             *
*                      RA - WORK REGISTER                             *
*                      RB - RESERVED FOR SECOND BASE REGISTER         *
*                      RC - BASE REGISTER                             *
*                      RD - SAVE AREA POINTER / POINTER TO DUMP CAUSE *
*                      RE - LINK REGISTER                             *
*                      RF - RETURN CODES & INFO/ANALYSIS ROUTINES     *
*                                                                     *
*  SUBROUTINES:    (1) IJBXSDA USES THE FOLLOWING INFO/ANALYSIS       *
*                      ROUTINES:                                      *
*                                                                     *
*                         DUMP DATA ACCESS ROUTINE                    *
*                         SYMPTOM RECORD ACCESS ROUTINE               *
*                         SYMPTOM RECORD UPDATE ROUTINE               *
*                         GETVIS/FREEVIS SERVICE ROUTINE              *
*                                                                     *
*                   (2) WRITELBD.                                     *
*                      THIS ROUTINE IS USED BY THE SDAID MODULE       *
*                      IJSDDEB TO PASS LBXT ENTRIES TO INFO/ANALYSIS. *
*                                                                     *
*  MODULES CALLED:     IJSDDEB (TO FORMAT THE SDAID BUFFER).          *
*                                                                     *
*                      THE ADDRESS OF AN INTERFACE AREA (DUMPIF)      *
*                      IS PASSED TO IJSDDEB IN REGISTER 1.            *
*                                                                     *
*  MACROS USED:          LOAD        LOAD A PHASE INTO GETVIS STORAGE *
*                        MAPDNTRY    STRUCTURE OF DIRECTORY ENTRY     *
*                        GENL        GENERATE A LOAD LIST             *
*                        IJBXSTC6    SYMPTOM RECORD INTERFACE         *
*                        ADSPXRPL    ANALYSIS ROUTINE INTERFACE       *
*                                                                     *
*  PATCH AREA LABEL:   PATCH                                          *
*                                                                     *
***********************************************************************
```

Stand-Alone Dump Analysis Routine IJBXCSMG

IJBXCSMG Logic

IJBXCSMG is an exit routine of Info/Analysis which processes the console events by redisplaying about the last most recent 20 messages and inputs including the timestamp and the console name where the source is coming from. This is performed by calling the Console Router.

The Console Router provides a set of services to redisplay the console messages and inputs in a timing order.

IJBXCSMG calls the Console Router code by passing the parameter list in the register R1.

After completion, the Console Router code IJBCSSA

1. updates the queue output area with an array of entries,

   where each entry has the following content:

   - the line length
   - the timestamp
   - the console id in case of a input and
   - the content of message text or console input,

2. and passes a return code in R15.

Additionally to access the dump data a common routine IJBXCSDA is shared between IJBXCSMG and IJBCSSA.

IJBXCSMG Module Description

IJBCSMSA

```
* ---------------------------------------------------------------- *
*                                                                  *
* MACRO NAME:  IJBCSMSA                                            *
*                                                                  *
* DESCRIPTIVE NAME:  Console router interface to the standalone    *
*                    (Info/Analysis) routine                       *
*                                                                  *
*  FUNCTION:                                                       *
*              Pass the last console router queue entries to       *
*              Info/Analysis for printing                          *
*                                                                  *
*  METHOD OF ACCESS =                                              *
*                  PLX   - Specify %INCLUDE SYSLIB(IJBCSMSA)       *
*                                                                  *
*                                                                  *
* ---------------------------------------------------------------- *
```

IJBXCSMG

```
* ------------------------------------------------------------------ *
*                                                                    *
* MODULE-NAME:      IJBXCSMG                                         *
*                                                                    *
* DESCRIPTIVE NAME: Support the Standalone dump utility              *
*                                                                    *
* FUNCTION:   This module retrieves the last messages from           *
*             the console router queue and writes it into            *
*             an area passed by the caller.                          *
*             The number of message written is defined by            *
*             the size of that area.                                 *
*             Prior to writing these message, the                    *
*             outstanding replies are written into this area.        *
*                                                                    *
* NOTES:                                                             *
*                                                                    *
*   DEPENDENCIES =  VSE/ESA 2.1.0 or later                           *
*                                                                    *
*   REGISTER-CONVENTIONS:                                            *
*                                                                    *
*     PLX     GENERAL REGISTER CONVENTIONS                           *
*     R09 = Base Register                                            *
*     R12 = Data Register                                            *
*                                                                    *
*                                                                    *
* MODULE-TYPE = Procedure                                            *
*                                                                    *
*   PROCESSOR = PLX 1.3.0 or later                                   *
*                                                                    *
*   MODULE-SIZE = see assembler listing                              *
*                                                                    *
*   ATTRIBUTES  = non-reentrant                                      *
*                                                                    *
* ENTRY-POINT = IJBXCSMG                                             *
*                                                                    *
*   LINKAGE   = R1 LOADED WITH THE ADDRESS OF THE PARAMETER LIST     *
*               R14 LOADED WITH RETURN ADDRESS                       *
*               R15 LOADED WITH ENTRY POINT                          *
*                                                                    *
* CALLED BY   - INFO/ANALYSIS  (Exit Routine)                        *
*                                                                    *
* OUTPUT      = none                                                 *
*                                                                    *
* MESSAGES ISSUED = none                                             *
*                                                                    *
* EXIT-NORMAL =  Return to caller (RetCode passed in Reg15)          *
*                RC = 00  Area filled with outstanding replies,      *
*                         if any, and last messages from queue.      *
*                RC = 04  Area filled with outstanding replies,      *
*                         if any, and last messages from queue.      *
*                         End of console router queue reached        *
*                                                                    *
* EXIT-ERROR  =  Return to caller (RetCode passed in Reg15)          *
*                RC = 08  End of passed output area reached while    *
*                         filling with outstanding replies.          *
*                         Probably not all outstanding replies       *
*                         could be written. No messages from the     *
*                         console router queue were written yet.     *
*                RC = 16  Console router queue for messages          *
*                         is empty.                                  *
*                         Outstanding replies were written into      *
*                         the output area if there were any.         *
*                RC = 32  the IJBXCSDA (Info/Analysis) service routine*
*                         has returned a bad return code.            *
*                         Cannot continue.                           *
```

```
*                                                                     *
*                                                                     *
* CONTROL BLOCKS:  (DSECTS)                                          *
*      IJBCSMSA  contains mappings of parameters passed from/to      *
*                caller                                              *
*                                                                     *
***********************************************************************
```

IJBCSDA

```
* ------------------------------------------------------------- *
*                                                               *
* MODULE-NAME:      IJBXCSDA                                    *
*                                                               *
* DESCRIPTIVE NAME: Support the Standalone dump utility         *
*                                                               *
* FUNCTION:   This module retrieves the dump data               *
*             using the dump access routine of Info/Analysis.   *
*                                                               *
* NOTES:                                                        *
*                                                               *
*   REGISTER-CONVENTIONS:                                       *
*                                                               *
*     PLX     GENERAL REGISTER CONVENTIONS                      *
*     R09 = Base Register                                       *
*     R12 = Data Register                                       *
*                                                               *
* MODULE-TYPE = Procedure                                       *
*                                                               *
*   PROCESSOR = PLX 1.3.0 or later                              *
*                                                               *
*   ATTRIBUTES  = non-reentrant                                 *
*                                                               *
* ENTRY-POINT = IJBXCSDA                                        *
*                                                               *
*   LINKAGE   = R1 LOADED WITH THE ADDRESS OF THE PARAMETER LIST*
*               R14 LOADED WITH RETURN ADDRESS                  *
*               R15 LOADED WITH ENTRY POINT                     *
*                                                               *
* CALLED BY   - IJBXCSMG, IJBCSSA                               *
*                                                               *
* CONTROL BLOCKS:  (DSECTS)                                     *
*      IJBCSMSA  contains mappings of parameters passed from/to *
*                caller                                         *
*                                                               *
* ------------------------------------------------------------- *
```

IJBCSSA

```
* ------------------------------------------------------------- *
*                                                               *
* MODULE-NAME:      IJBCSSA                                     *
*                                                               *
* DESCRIPTIVE NAME:  Support the Standalone dump utility        *
*                                                               *
* FUNCTION:   This module retrieves the last messages from      *
*             the console router queue and writes it into       *
*             an area passed by the caller.                     *
*             The number of message written is defined by       *
*             the size of that area.                            *
*             Prior to writing these message, the               *
*             outstanding replies are written into this         *
*             area                                              *
*                                                               *
*   DEPENDENCIES =  VSE/ESA 2.1.0 or later                      *
*                                                               *
*   RESTRICTIONS = NONE                                         *
*                                                               *
*   REGISTER-CONVENTIONS:                                       *
*                                                               *
*     PLX     GENERAL REGISTER CONVENTIONS                      *
*     R09 = Base Register                                       *
*     R12 = Data Register                                       *
*                                                               *
*   PATCH-LABEL  = NONE                                         *
*                                                               *
* MODULE-TYPE = Procedure                                       *
*                                                               *
*   PROCESSOR = PLX 1.3.0 or later                              *
*                                                               *
*   ATTRIBUTES  = non-reentrant                                 *
*                                                               *
* ENTRY-POINT = IJBCSSA                                         *
*                                                               *
*   PURPOSE   = See function                                    *
*                                                               *
*   LINKAGE   = R1 LOADED WITH THE ADDRESS OF THE PARAMETER LIST *
*               R14 LOADED WITH RETURN ADDRESS                  *
*               R15 LOADED WITH ENTRY POINT                     *
*                                                               *
* CALLED BY   - IJBXCSMG (standalone dump utility)              *
*                                                               *
* OUTPUT      = none                                            *
*                                                               *
* MESSAGES ISSUED = none                                        *
*                                                               *
* EXIT-NORMAL =  Return to caller (RetCode passed in Reg15)     *
*                RC = 00  Area filled with outstanding replies, *
*                         if any, and last messages from queue. *
*                RC = 04  Area filled with outstanding replies, *
*                         if any, and last messages from queue. *
*                         End of console router queue reached   *
*                                                               *
* EXIT-ERROR  =  Return to caller (RetCode passed in Reg15)     *
*                RC = 08  End of passed output area reached while *
*                         filling with outstanding replies.     *
*                         Probably not all outstanding replies  *
*                         could be written. No messages from the *
*                         console router queue were written yet. *
*                RC = 16  Console router queue for messages     *
*                         is empty.                             *
*                         Outstanding replies were written into *
*                         the output area if there were any.    *
*                RC = 32  the GetDData (Info/Analysis) service  *
*                         routine has returned a bad return code. *
*                         Cannot continue.                      *
*                                                               *
```

```
*                                                                     *
* CONTROL BLOCKS:  (DSECTS)                                          *
*      IJBCSMSA  contains mappings of parameters passed from/to      *
*                caller                                               *
*                                                                     *
* ------------------------------------------------------------------- *
```

Chapter 13. Info/Analysis: Introduction

This chapter contains general information describing the Info/Analysis for VSE Program Product.

Program Overview

Info/Analysis is a component of VSE/ESA Version 2 Release 1. It provides batch facilities for dump analysis, dump management, and dump symptom extraction.

Info/Analysis consists of a group of interrelated functional routines for:

- Dump data set management
- Dump symptom display
- Dump data viewing
- Offload and onload of dump data

Info/Analysis provides facilities to support the Symptom Record Architecture. These facilities include:

- Display of the problem failure symptom record
- Display of dump data using pre-defined control block and data locators and formatting descriptors
- Invocation of system and component dump analysis routines

Info/Analysis Functional Components

Figure 14 is an overview of the functional components of Info/Analysis.

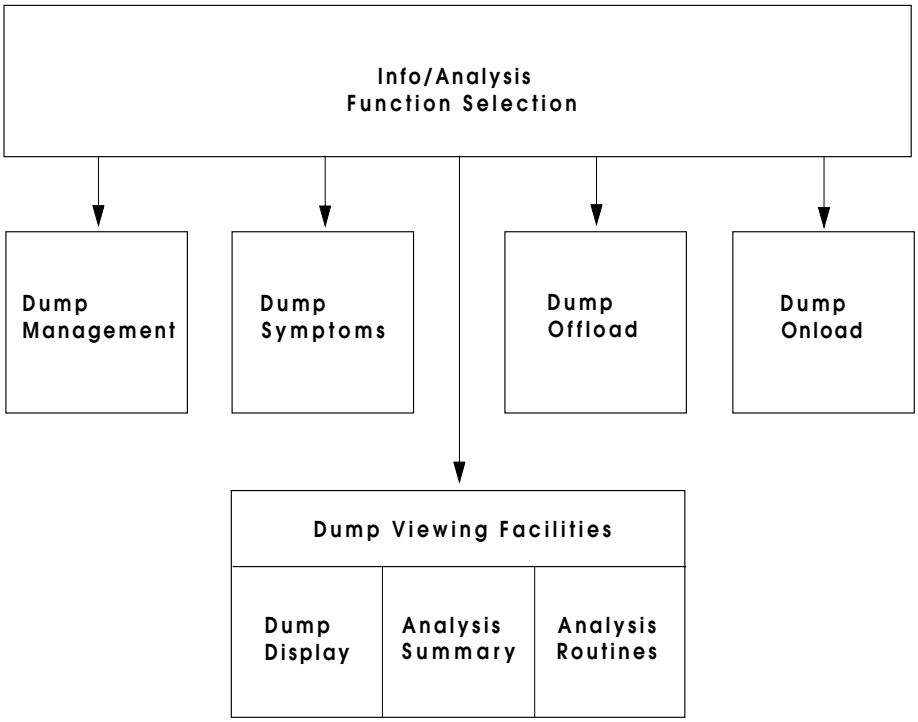


Figure 14. Info/Analysis Functional Components

With Info/Analysis, you can simplify the task of using dump data to solve software problems. Info/Analysis assists you in this task through the following functions:

- Dump data set management - to list the dumps being managed by Info/Analysis, to add or delete dumps from that list, and to delete dumps from the system.
- Dump symptom display - to display problem failure information collected by the dumping component and by subsequent analysis routines. This information may be used both locally and in an IBM maintenance database for problem determination.
- Dump data viewing - to display dump data in hexadecimal format, to locate, format, and display control blocks and other dump data that may be pertinent to the problem, to mask out (overlay) sensitive data in specific areas of the dump, to invoke dump analysis routines, and to display the results of those routines.
- Offload of dump data - to copy a dump to tape for later retrieval or for forwarding to an IBM software support center.
- Onload of dump data - to copy an offloaded dump, a dump from another system, or a stand-alone dump to the system.

Info/Analysis processes storage dumps that result from errors within the system supervisor or within subsystem or user programs running on the system. The dumps are created by system dump and stand-alone dump utilities. Info/Analysis does not directly access the dump data. Rather, it uses system facilities to retrieve and update dump data and the symptom record. The symptom record is a collection of problem-related information stored in the dump and its extensions.

Info/Analysis uses a dump management file to maintain information about dumps. A dump must be identified in this file before it can be processed by Info/Analysis. This file is maintained using the dump management function.

Info/Analysis also uses an external routines file. This file contains a list of analysis routines that you may invoke to process dump data. The file identifies user exit routines and dump access routines called by Info/Analysis.

Invocation of Info/Analysis

Info/Analysis runs in a static or dynamic VSE/ESA partition. You may enter control statements in two modes:

- Line mode - from the operator console
- Reader mode - from the system input device

From a VSE/Advanced Functions partition, all output from batch operations is routed to the SYSLST device assigned to the partition. In line mode, messages are sent to the console as well as to SYSLST.

In the batch environment certain services, such as display, are not available.

Physical Characteristics

Info/Analysis is comprised of reentrant phases only.

Chapter 14. Info/Analysis: Technical Overview

This chapter provides an overview of the structure of Info/Analysis as well as information on resources and security.

Info/Analysis Structure

Info/Analysis interacts with several external programs and components that perform specialized services. These programs and components are:

- VSE Dump Access
- BLX

Figure 15 illustrates the structure of Info/Analysis.

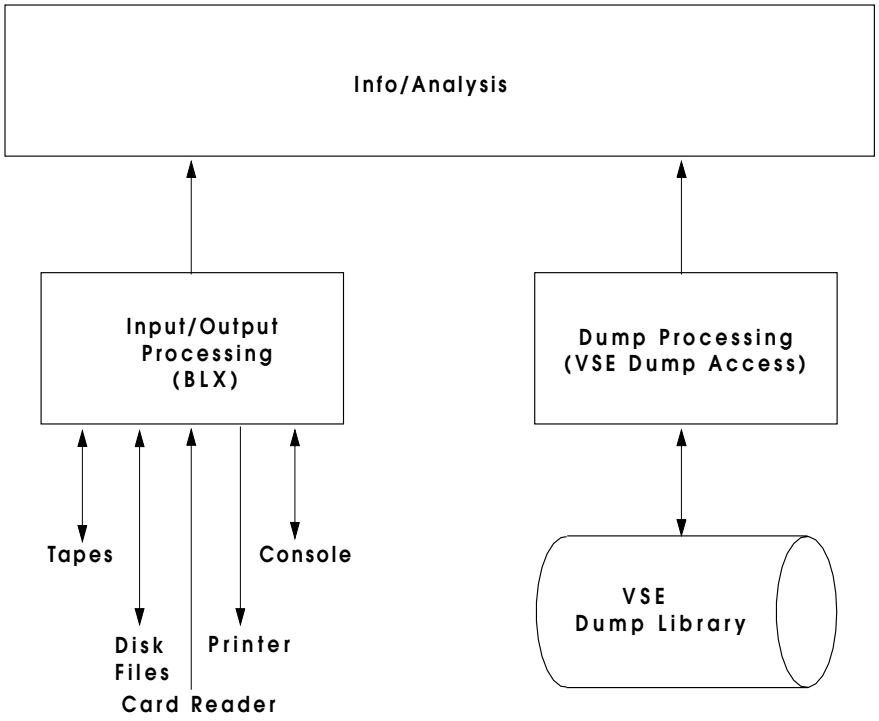


Figure 15. Info/Analysis Structure

The different external components of Info/Analysis have been isolated to facilitate future requirements and problem determination. Each of the components is implemented as a separate load module.

Each component has its own method of problem determination; therefore, when a problem does occur, it is important to identify which of these components caused the problem.

All accesses to the VSE dump library are performed by a dump access service written specifically for VSE. Control is transferred to dump access when a request is made to read, write, or update dump data.

All I/O operations are isolated in BLX services. These services provide system independent macros for all I/O operations. Control is transferred to BLX when I/O operations are requested.

Info/Analysis Libraries and Files

All VSE libraries and files must be defined to the VSE operating system by the user before Info/Analysis is initialized. Info/Analysis requires auxiliary storage space for its own programs. The current estimate is 190 tracks of 3330 disk storage for Info/Analysis, dump access and BLX services.

The customer determines the allocation and size of the library for dump data storage. This determination depends on the number of dumps that the user site is prepared to maintain at any one time. Space for three dumps of 2M bytes each may be a valid initial specification, subject to change based on experience.

Info/Analysis requires a data set allocation to contain the list of dumps currently being managed. This data set is estimated to be one track of 3380 storage, depending on the number of dumps being managed. This data set contains control information for all dumps known to Info/Analysis whether they are currently stored in the VSE dump library or not. This data set is maintained by Info/Analysis and may not be modified by the user.

Info/Analysis requires a data set allocation to contain the list of post dump analysis exit routines available. This data set is estimated to be one track of 3380 storage. The precise size depends on the number of dump analysis routines available. This data set contains the name and an optional description of each routine available for use with Info/Analysis. The data set may be modified by the user with system utilities. The format of each record in the data set is as follows:

- The first nonblank characters in each record of the data set are the character string 'ANEXIT'. This string indicates that this record contains the name and description of an analysis exit routine. Any other starting string of characters is ignored.
- The name of the routine follows the 'ANEXIT' string. There must be at least one blank between 'ANEXIT' and the routine name. The routine name may be one to eight characters in length, followed by at least one blank.
- An optional description may be included following the routine name and the blank character delimiter. This description is free-form text and is placed alongside the name of the routine in the displayed list. Info/Analysis separates the description from the routine name with six blank characters in the displayed list. Continuation cards are not displayed or printed.

Examples of this record are:

```
ANEXIT   IJBXSDA    FORMAT SDAID BUFFER
ANEXIT IJBXDBUG ANALYSE VSE SUPERVISOR STATUS
```

Note that there may be more than one space between 'ANEXIT' and the routine name and between the routine name and its description. Info/Analysis uses the information in this record for display and selection of an analysis routine.

Info/Analysis requires tape volumes to onload and offload dump information.

Library and File Protection

Info/Analysis makes use of the VSE enqueue facilities as provided by BLX services. This use covers access to dump data sets and other auxiliary data sets maintained on external storage. Conflicts in VSE library access by Info/Analysis and the VSE librarian or by multiple Info/Analysis users are controlled through this facility.

Info/Analysis serializes read and write activities to the same file with a BLXENQ request for exclusive use using the logical device name (LNAME).

This BLXENQ is unconditional (UNCOND) with an option of WAIT. UNCOND specifies that if a previous BLXENQ has been issued for the same resource, without an intervening BLXDEQ, the task is to abend. WAIT specifies that if a resource is not immediately available, the user is to be put into a wait state until it can be enqueued. This may result in a temporary wait if the library is being written into by another Info/Analysis user. This wait is usually of short duration. If successful, this enqueue remains in effect for the entire time Info/Analysis has control of the library.

System Integrity

Info/Analysis executes entirely as a nonprivileged problem program in the host environment. Therefore, no system integrity problems are anticipated.

Info/Analysis Host System Control Blocks

Info/Analysis does not refer to or utilize any host system control blocks. Communication with system facilities is done through standard system interfaces such as system macros or system-provided functions (dump access).

Info/Analysis Storage Requirements

Info/Analysis is expected to require approximately 300K bytes, including BLX system services. Under most conditions, a partition size of 896K bytes is considered adequate; a GETVIS area of 596K is recommended.

Info/Analysis Equipment Supported

Info/Analysis operates on any configuration and CPU that meets the minimum specifications for VSE/ESA Version 2 Release 1.

Chapter 15. Info/Analysis: Diagnostic Aids

This chapter contains the following information, which can help in diagnosing problems within Info/Analysis and Dump Access:

- Register usage
- Abend information
- I/O diagnostic aids
- Messages and codes
- Module/message cross-reference for Info/Analysis
- Message/module cross-reference for Info/Analysis
- Module/reason cross-reference for Info/Analysis
- Reason/module cross-reference for Info/Analysis
- Calling/called module cross-reference for Info/Analysis
- Called/calling module cross-reference for Info/Analysis
- Module/reason cross-reference for dump access
- Reason/module cross-reference for dump access
- Calling/called module cross-reference for dump access
- Called/calling module cross-reference for dump access

Register Usage

When an Info/Analysis or Dump Access program passes control to a separately compiled Info/Analysis, Dump Access, or BLX program, the following linkage conventions are generally observed:

Register	Contents
1	Pointer to parameter list; each address in the list, in turn, points to a parameter.
2 - 12	Work registers.
13	Address of an 80-byte save area that includes the normal 72-byte save area and eight bytes of BLX control information.
14	Return address.
15	When a program is entered, contains its entry address; upon return to the invoking program, contains its return code.

Within Info/Analysis programs, the following register conventions are used:

Register	Contents
1	Involved in program linkage.
2 - 11	Used differently by each program.
12	Address of the storage containing the program and its literals (base register).
13	Address of an 80-byte save area which includes the normal 72-byte save area and eight bytes of BLX control information followed by the local dynamic storage associated with the module.
14 - 15	Involved in program linkages.

Abend Information

When an abend occurs in Info/Analysis, information about the abend is normally provided in an internal dump (IDUMP). When a dump is taken a symptom string is built and placed at the beginning of the dump. The symptom string can be printed using Info/Analysis.

The symptom string is a line of text that contains specific items in the condensed format described below. It may help an installation or an IBM representative in tracking problems because it can be easily compared to the symptom strings of known problems. The string will contain at least the following required symptoms:

- AB/abend code prefix, abend code

  The prefix is "S" for a system abend code or "U" for a user code.

- PIDS/component identifier

  If the abend occurred in Info/Analysis, or if the location of the abend cannot be identified, the Info/Analysis component ID is shown.

- RIDS/routine identifier

  The Info/Analysis module where the abend occurred.

- REGS/register number, PSW-register difference This item shows the result of comparing the value of each general register with the address in the PSW at the time of the abend. The results are shown for the two highest numbered registers whose values are less than the PSW, but only by an amount less than 4096 bytes. The hexadecimal register number occupies the first two positions, and the hexadecimal difference occupies the last three positions.

  If no registers satisfy this condition, this item appears as "REGS/FFFFF". If the PSW address is less than 512, this item appears as "REGS/FExxx" where xxx is the hexadecimal representation of the PSW address.

The IDUMP

An IDUMP consists of the entire partition in which Info/Analysis is running. If the address of the abend is within the SVA, a phase of the SVA is also dumped.

The following information is provided by an IDUMP:

- IDUMP comment

  The comment at the top of each page identifies Info/Analysis and the name of the Info/Analysis module in which the abend occurred. The comment also contains the address of the IDUMP information area within the dump, which is described below.

- Registers

  The contents of the general registers at the time of the abend are listed at the start of the IDUMP.

- IDUMP information area

  The IDUMP information area contains a copy of the symptom string described above, followed by a 24-byte "ABEND SAVEAREA" label and an 80-byte field in partition save area format. This 80-byte field consists of the name of the abended module (8 bytes), the abend-time PSW (8 bytes), and registers 9 through 8 at abend time (64 bytes).

Decimal ("User") Abend Codes

Info/Analysis does not issue any abend codes. BLX abend codes are contained in the *VSE/ESA Messages and Codes Manual*.

Abend codes issued by other programs or components can be found in the appropriate program or component documentation.

I/O Diagnostic Aids

All Info/Analysis library and sequential file access has been isolated within the external components. BLX codes are contained in the *VSE/ESA Messages and Codes Manual*. Any codes issued by other programs or components can be found in the appropriate program or component documentation.

Messages and Codes

Messages serve to communicate status information to the user. These messages pertain to error conditions and requests for verification and confirmation that may occur while using the operations and functions of Info/Analysis. BLX and Info/Analysis message services are both utilized to accomplish this task.

BLX message services are used to construct the skeleton forms of the message. BLX message macros are used to route the output. The BLX message macro is used to route the messages to the printer or output device.

Each Info/Analysis message is identified by a string of eight characters called the message ID. This message ID has the format BLNnnnna, where:

- BLN is the Info/Analysis component identifier
- nnnn is the assigned decimal message number
- a is the action code

The first two digits of the message number identify the functional area to which the message applies, as follows:

09 Symptom record access

10 Function selection

20 Dump management

30 Dump onload

40 Dump offload

50 Dump viewing

70 Analysis summary and routines

90 General usage

The following action codes are used in Info/Analysis:

D = Decision You must choose between courses of action.

I = Information This message provides error or status information; you should correct the error if one is indicated.

When an action or decision is necessary, Info/Analysis usually waits until you enter an acceptable reply from the keyboard, or perform an action such as mounting a disk pack, readying a device, or placing cards in the card reader.

Variable data in the message text is indicated in lowercase. The variable that appears in the message documentation is replaced in an issued message as follows:

addr - address

block - control block name

disp - DISPLAY value

dumpid - dump name

entry - a command, control statement, or data entry field value

fil - file name

func - function

jobname - job name

maxrec - maximum number of records

mod - module

msg - message

ofst - offset

oper - operand

rcd - record number

req - request

rsc - reason code

rtc - return code

rtn - external routine

serv - service

tbl - table

userec - number of records currently in use

Reason codes may be used within a message to further describe the error condition that caused the message. These reason codes assist the user in identifying and correcting error situations.

More severe error conditions that the user cannot correct by altering input may cause an abend to occur. In these cases, a reason code is entered in internal control blocks before the dump is taken. These reason codes are then used during dump analysis to identify the problem.

The reason codes that may be included in Info/Analysis messages are associated with particular functions, as follows:

0 - 899	Dump access interface
900 - 999	Symptom record access
1000 - 1999	Function selection and batch control statements
2000 - 2999	Dump management
3000 - 3999	Dump onload
4000 - 4999	Dump offload
5000 - 5999	Dump viewing
7000 - 7499	Analysis summary
7500 - 7999	Analysis routines
9000 - 9999	General

Info/Analysis Module/Message Cross-Reference

In the following list, object modules that use messages are listed alphabetically by module name. Each module name is followed by the message ID of each message that the module issues, the possible environments for each message, the type of message (error or informational), and the variables that are inserted in the message. The possible environment modes are:

B - Batch

C - Console

Module Issuing	Message ID	Modes	Type	Inserts
BLNALADF	BLN0929I	B/C	INFO	rcd, ofst
BLNARDLO	BLN9001I	B/C	ERROR	
BLNARDSP	BLN7540I	B/C	ERROR	
BLNAREXA	BLN9001I	B/C	ERROR	
BLNAREXA	BLN9007I	B/C	ERROR	rsc
BLNAREXC	BLN7540I	B/C	ERROR	
BLNAREXD	BLN7515I	B/C	ERROR	rtn
BLNAREXD	BLN7520I	B/C	ERROR	
BLNAREXD	BLN7540I	B/C	ERROR	
BLNAREXG	BLN9002I	B/C	ERROR	rtc, rsc
BLNAREXL	BLN7516I	B/C	ERROR	rtn, rsc
BLNARGRN	BLN9001I	B/C	ERROR	
BLNARINT	BLN9007I	B/C	ERROR	rsc
BLNARPD6	BLN9001I	B/C	ERROR	
BLNARPRN	BLN9001I	B/C	ERROR	
BLNARPRT	BLN5011I	B/C	ERROR	
BLNARPRT	BLN7540I	B/C	ERROR	
BLNARPRT	BLN9012I	B/C	INFO	
BLNARPRT	BLN9022I	B/C	ERROR	
BLNARS6U	BLN9001I	B/C	ERROR	
BLNARXPL	BLN9001I	B/C	ERROR	
BLNASADF	BLN0923I	B/C	INFO	
BLNASADF	BLN0930I	B/C	INFO	
BLNASADL	BLN0923I	B/C	INFO	
BLNASADL	BLN0929I	B/C	INFO	rcd, ofst
BLNASADL	BLN0930I	B/C	INFO	
BLNASAXA	BLN0923I	B/C	INFO	
BLNASAXA	BLN0929I	B/C	INFO	rcd, ofst
BLNASAXA	BLN0930I	B/C	INFO	
BLNASAXA	BLN0931I	B/C	INFO	
BLNASAXC	BLN0923I	B/C	INFO	
BLNASAXC	BLN0929I	B/C	INFO	rcd, ofst
BLNASAXC	BLN0930I	B/C	INFO	
BLNASAXC	BLN0931I	B/C	INFO	
BLNASAXK	BLN0923I	B/C	INFO	
BLNASAXK	BLN0929I	B/C	INFO	rcd, ofst
BLNASAXK	BLN0930I	B/C	INFO	
BLNASAXK	BLN0931I	B/C	INFO	
BLNASAXT	BLN0923I	B/C	INFO	
BLNASAXT	BLN0929I	B/C	INFO	rcd, ofst
BLNASAXT	BLN0930I	B/C	INFO	
BLNASAXT	BLN0931I	B/C	INFO	
BLNASAXX	BLN0923I	B/C	INFO	
BLNASAXX	BLN0929I	B/C	INFO	rcd, ofst
BLNASAXX	BLN0930I	B/C	INFO	
BLNASAXX	BLN0933I	B/C	INFO	

Module Issuing	Message ID	Modes	Type	Inserts
BLNASFMT	BLN0921I	B/C	ERROR	
BLNASGET	BLN0921I	B/C	ERROR	
BLNASLCB	BLN0920I	B/C	ERROR	
BLNASLCB	BLN0921I	B/C	ERROR	
BLNASLD1	BLN0921I	B/C	ERROR	
BLNASLD1	BLN0926I	B/C	ERROR	
BLNASLOC	BLN0926I	B/C	ERROR	
BLNASLST	BLN0921I	B/C	ERROR	
BLNASMAP	BLN0921I	B/C	ERROR	
BLNASMAP	BLN0927I	B/C	ERROR	
BLNASMD	BLN0923I	B/C	INFO	
BLNASMD	BLN0929I	B/C	INFO	rcd, ofst
BLNASMD	BLN0932I	B/C	INFO	
BLNASMD	BLN0934I	B/C	INFO	
BLNASMP1	BLN0921I	B/C	ERROR	
BLNASRSR	BLN0921I	B/C	ERROR	
BLNASRSR	BLN0927I	B/C	ERROR	
BLNASRSR	BLN0928I	B/C	ERROR	rsc
BLNASTRM	BLN0920I	B/C	ERROR	
BLNAXPD2	BLN9012I	B/C	INFO	
BLNAXPD2	BLN9022I	B/C	ERROR	
BLNAXPD3	BLN7521I	B/C	INFO	
BLNAXPD3	BLN7522I	B/C	INFO	
BLNAXPD4	BLN7522I	B/C	INFO	
BLNAXPD6	BLN9022I	B/C	ERROR	
BLNAXSU1	BLN7518I	B/C	ERROR	
BLNBABH1	BLN1007I	B/C	ERROR	
BLNBABH1	BLN9004I	B/C	ERROR	entry
BLNBABH1	BLN9019I	B/C	ERROR	dumpid
BLNBABH1	BLN9032I	B/C	ERROR	dumpid
BLNBABH2	BLN1004I	B	INFO	
BLNBADM1	BLN1007I	B/C	ERROR	
BLNBADM1	BLN9004I	B/C	ERROR	entry
BLNBADM1	BLN9016I	B/C	ERROR	
BLNBADM2	BLN1003I	B/C	ERROR	
BLNBADM2	BLN9004I	B/C	ERROR	entry
BLNBADS1	BLN9004I	B/C	ERROR	entry
BLNBADS1	BLN9016I	B/C	ERROR	
BLNBADV1	BLN9004I	B/C	ERROR	entry
BLNBAFLS	BLN1004I	B	INFO	
BLNBAINT	BLN1008I	B/C	INFO	
BLNBAINT	BLN9001I	B/C	ERROR	
BLNBAPF1	BLN4006I	B/C	ERROR	
BLNBAPF1	BLN9004I	B/C	ERROR	entry
BLNBAPF1	BLN9016I	B/C	ERROR	
BLNBAPN1	BLN9004I	B/C	ERROR	entry
BLNBAPN1	BLN9016I	B/C	ERROR	
BLNBARDR	BLN1004I	B	INFO	
BLNBARDR	BLN1005D	C	INFO	func
BLNBARDR	BLN1006I	B	ERROR	
BLNBARDR	BLN1009I	B/C	INFO	
BLNBARDR	BLN9008I	B/C	ERROR	mod, rsc
BLNCMCHD	BLN9006I	B/C	INFO	
BLNCMCHD	BLN9019I	B/C	ERROR	dumpid
BLNCMERM	BLN9002I	B/C	ERROR	rtc, rsc
BLNCMERM	BLN9003I	B/C	ERROR	msg
BLNCMERM	BLN9031I	B/C	ERROR	rtc, rsc

Module Issuing	Message ID	Modes	Type	Inserts
BLNCMFIL	BLN9004I	B/C	ERROR	entry
BLNCMHEX	BLN9004I	B/C	ERROR	entry
BLNCMVOL	BLN4001I	B/C	ERROR	
BLNCMVOL	BLN9001I	B/C	ERROR	
BLNCMVOL	BLN9004I	B/C	ERROR	entry
BLNCMXCF	BLN9004I	B/C	ERROR	entry
BLNCMXDE	BLN9030I	B/C	ERROR	req, fil, rsc
BLNDDCDB	BLN5030I	B/C	INFO	
BLNDDCDB	BLN9001I	B/C	ERROR	
BLNDDDPT	BLN9012I	B/C	INFO	
BLNDDFDO	BLN5013I	B/C	INFO	
BLNDDFDZ	BLN5030I	B/C	INFO	
BLNDDFMT	BLN9001I	B/C	ERROR	
BLNDDFPP	BLN5030I	B/C	INFO	
BLNDDFPP	BLN9001I	B/C	ERROR	
BLNDDFPP	BLN9007I	B/C	ERROR	rsc
BLNDDFPP	BLN9022I	B/C	ERROR	
BLNDDFPT	BLN9012I	B/C	INFO	
BLNDDHDO	BLN5014I	B/C	INFO	block
BLNDDINT	BLN9001I	B/C	ERROR	
BLNDDMDT	BLN5014I	B/C	INFO	block
BLNDDMDT	BLN9010I	B/C	INFO	
BLNDDNBU	BLN9001I	B/C	ERROR	
BLNDDPOP	BLN5011I	B/C	ERROR	
BLNDDPOP	BLN5022I	B/C	ERROR	
BLNDDPOP	BLN9004I	B/C	ERROR	entry
BLNDDPRT	BLN9012I	B/C	INFO	
BLNDDPRT	BLN9022I	B/C	ERROR	
BLNDDRDF	BLN9021I	B/C	INFO	
BLNDDRPT	BLN5014I	B/C	INFO	block
BLNDDRPT	BLN9010I	B/C	INFO	
BLNDDRPT	BLN9012I	B/C	INFO	
BLNDDRPT	BLN9022I	B/C	ERROR	
BLNDDSYI	BLN9001I	B/C	ERROR	
BLNDDVFA	BLN5022I	B/C	ERROR	
BLNDDVFA	BLN9004I	B/C	ERROR	entry
BLNDDVFN	BLN9004I	B/C	ERROR	entry
BLNDMADU	BLN2006I	B/C	ERROR	
BLNDMADU	BLN9018I	B/C	INFO	dumpid, func
BLNDMAD1	BLN2019I	B/C	INFO	
BLNDMAD1	BLN9006I	B/C	INFO	
BLNDMCNT	BLN2007I	B/C	INFO	maxrec, userec
BLNDMDDL	BLN2017I	B/C	ERROR	
BLNDMDDL	BLN9001I	B/C	ERROR	
BLNDMDEL	BLN2009I	B/C	ERROR	
BLNDMDEL	BLN2017I	B/C	ERROR	
BLNDMDEL	BLN9018I	B/C	INFO	dumpid, func
BLNDMDMU	BLN2013I	B/C	ERROR	rsc
BLNDMDMU	BLN2014I	B/C	ERROR	
BLNDMDMU	BLN2020I	B/C	INFO	
BLNDMGNT	BLN9008I	B/C	ERROR	mod, rsc
BLNDMPDL	BLN2017I	B/C	ERROR	
BLNDMPDL	BLN9001I	B/C	ERROR	
BLNDMPDL	BLN9012I	B/C	INFO	
BLNDMRDF	BLN2013I	B/C	ERROR	
BLNDMUAD	BLN9004I	B/C	ERROR	entry
BLNDMWDF	BLN2013I	B/C	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNDSCEN	BLN9006I	B/C	INFO	
BLNDSDSP	BLN9007I	B/C	ERROR	rsc
BLNDSGBF	BLN9001I	B/C	ERROR	
BLNDSPRT	BLN9007I	B/C	ERROR	rsc
BLNDSPRT	BLN9012I	B/C	INFO	
BLNDSTRM	BLN9007I	B/C	ERROR	rsc
BLNDSYMP	BLN9007I	B/C	ERROR	rsc
BLNDVKCB	BLN9001I	B/C	ERROR	
BLNDVSIA	BLN9001I	B/C	ERROR	
BLNDVSIN	BLN9007I	B/C	ERROR	rsc
BLNFNARC	BLN9007I	B/C	ERROR	rsc
BLNFNDDC	BLN9007I	B/C	ERROR	rsc
BLNFNDMC	BLN9007I	B/C	ERROR	rsc
BLNFNDSC	BLN9007I	B/C	ERROR	rsc
BLNFNDVS	BLN9007I	B/C	ERROR	rsc
BLNFNMSK	BLN9007I	B/C	ERROR	rsc
BLNFNPFC	BLN9007I	B/C	ERROR	rsc
BLNFNPNC	BLN9007I	B/C	ERROR	rsc
BLNFNSMC	BLN9007I	B/C	ERROR	rsc
BLNLBARC	BLN9001I	B/C	ERROR	
BLNLBARR	BLN9001I	B/C	ERROR	
BLNLBCAC	BLN9001I	B/C	ERROR	
BLNLBCAE	BLN9001I	B/C	ERROR	
BLNLBCAI	BLN9001I	B/C	ERROR	
BLNLBCAL	BLN9001I	B/C	ERROR	
BLNLBCHC	BLN9001I	B/C	ERROR	
BLNLBCHE	BLN9001I	B/C	ERROR	
BLNLBCHI	BLN9001I	B/C	ERROR	
BLNLBCHL	BLN9001I	B/C	ERROR	
BLNLBCHR	BLN9001I	B/C	ERROR	
BLNLBCHW	BLN9001I	B/C	ERROR	
BLNLBLAL	BLN9001I	B/C	ERROR	
BLNLBSIM	BLN9001I	B/C	ERROR	
BLNLBVER	BLN7013I	B/C	ERROR	block, rsc
BLNMACBC	BLN1013I	B/C	ERROR	
BLNMACBG	BLN9002I	B/C	ERROR	rtc, rsc
BLNMKBMD	BLN9001I	B/C	ERROR	
BLNMKDSP	BLN9007I	B/C	ERROR	rsc
BLNMKINT	BLN9007I	B/C	ERROR	rsc
BLNMKMSK	BLN9007I	B/C	ERROR	rsc
BLNMKQRY	BLN5030I	B/C	INFO	
BLNMKVMR	BLN5022I	B/C	ERROR	
BLNMKVMR	BLN9004I	B/C	ERROR	entry
BLNOFFLD	BLN4003I	B/C	ERROR	
BLNOFFLD	BLN4004I	B/C	INFO	dumpid, ERASED
BLNOFFLD	BLN4005I	B/C	ERROR	rsc
BLNOFFLD	BLN4007I	B/C	ERROR	
BLNOFFLD	BLN4010I	B/C	INFO	
BLNOFFLD	BLN9019I	B/C	ERROR	dumpid
BLNONEND	BLN3002I	B/C	ERROR	rsc
BLNONEND	BLN9018I	B/C	INFO	dumpid, func
BLNONINT	BLN3002I	B/C	INFO	rsc
BLNONINT	BLN4007I	B/C	ERROR	
BLNSMBLE	BLN7010I	B/C	INFO	
BLNSMBLE	BLN9007I	B/C	ERROR	rsc
BLNSMBLF	BLN9001I	B/C	ERROR	
BLNSMBLO	BLN9001I	B/C	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNSMBMA	BLN9001I	B/C	ERROR	
BLNSMBME	BLN9007I	B/C	ERROR	rsc
BLNSMCMO	BLN9001I	B/C	ERROR	
BLNSMDSP	BLN7010I	B/C	INFO	
BLNSMFND	BLN7010I	B/C	INFO	
BLNSMFRE	BLN9001I	B/C	ERROR	
BLNSMGRE	BLN9001I	B/C	ERROR	
BLNSMHXA	BLN9001I	B/C	ERROR	
BLNSMHXA	BLN9007I	B/C	ERROR	rsc
BLNSMHXO	BLN9001I	B/C	ERROR	
BLNSMINT	BLN9007I	B/C	ERROR	rsc
BLNSMLKB	BLN9001I	B/C	ERROR	
BLNSMLKC	BLN9001I	B/C	ERROR	
BLNSMLKD	BLN9001I	B/C	ERROR	
BLNSMLKL	BLN5030I	B/C	INFO	
BLNSMLKL	BLN9001I	B/C	ERROR	
BLNSMLKO	BLN9001I	B/C	ERROR	
BLNSMOCL	BLN9001I	B/C	ERROR	
BLNSMPCD	BLN9007I	B/C	ERROR	rsc
BLNSMPCD	BLN9022I	B/C	ERROR	
BLNSMPHP	BLN9001I	B/C	ERROR	
BLNSMPRT	BLN5011I	B/C	ERROR	
BLNSMPRT	BLN9007I	B/C	ERROR	rsc
BLNSMPRT	BLN9012I	B/C	INFO	
BLNSMPSO	BLN9001I	B/C	ERROR	
BLNSMPSW	BLN9001I	B/C	ERROR	
BLNSMSEL	BLN7010I	B/C	INFO	
BLNSMTRM	BLN9007I	B/C	ERROR	rsc
BLNSMTXA	BLN9001I	B/C	ERROR	
BLNSMTXA	BLN9007I	B/C	ERROR	rsc
BLNSMTXO	BLN9001I	B/C	ERROR	

Info/Analysis Message/Module Cross-Reference

In the following list, Info/Analysis message IDs are listed numerically. Each message ID is followed by the name of the module that issues the message, the possible environments for each message, the type of message (error or informational), and the variables that are inserted in the message. (Refer to "Info/Analysis Module/Message Cross-Reference" on page 248 for additional information.)

Message ID	Module Issuing	Modes	Type	Inserts
BLN0920I	BLNASLCB	B/C	ERROR	
BLN0920I	BLNASTRM	B/C	ERROR	
BLN0921I	BLNASFMT	B/C	ERROR	
BLN0921I	BLNASGET	B/C	ERROR	
BLN0921I	BLNASLCB	B/C	ERROR	
BLN0921I	BLNASLD1	B/C	ERROR	
BLN0921I	BLNASLST	B/C	ERROR	
BLN0921I	BLNASMAP	B/C	ERROR	
BLN0921I	BLNASMP1	B/C	ERROR	
BLN0921I	BLNASRSR	B/C	ERROR	
BLN0923I	BLNASADF	B/C	INFO	
BLN0923I	BLNASADL	B/C	INFO	
BLN0923I	BLNASAXA	B/C	INFO	
BLN0923I	BLNASAXC	B/C	INFO	
BLN0923I	BLNASAXK	B/C	INFO	
BLN0923I	BLNASAXT	B/C	INFO	
BLN0923I	BLNASAXX	B/C	INFO	
BLN0923I	BLNASMD	B/C	INFO	
BLN0926I	BLNASLD1	B/C	ERROR	
BLN0926I	BLNASLOC	B/C	ERROR	
BLN0927I	BLNASMAP	B/C	ERROR	
BLN0927I	BLNASRSR	B/C	ERROR	
BLN0928I	BLNASRSR	B/C	ERROR	rsc
BLN0929I	BLNALADF	B/C	INFO	rcd, ofst
BLN0929I	BLNASADL	B/C	INFO	rcd, ofst
BLN0929I	BLNASAXA	B/C	INFO	rcd, ofst
BLN0929I	BLNASAXC	B/C	INFO	rcd, ofst
BLN0929I	BLNASAXK	B/C	INFO	rcd, ofst
BLN0929I	BLNASAXT	B/C	INFO	rcd, ofst
BLN0929I	BLNASAXX	B/C	INFO	rcd, ofst
BLN0929I	BLNASMD	B/C	INFO	rcd, ofst
BLN0930I	BLNASADF	B/C	INFO	
BLN0930I	BLNASADL	B/C	INFO	
BLN0930I	BLNASAXA	B/C	INFO	
BLN0930I	BLNASAXC	B/C	INFO	
BLN0930I	BLNASAXK	B/C	INFO	
BLN0930I	BLNASAXT	B/C	INFO	
BLN0930I	BLNASAXX	B/C	INFO	
BLN0931I	BLNASAXA	B/C	INFO	
BLN0931I	BLNASAXC	B/C	INFO	
BLN0931I	BLNASAXK	B/C	INFO	
BLN0931I	BLNASAXT	B/C	INFO	
BLN0932I	BLNASMD	B/C	INFO	
BLN0933I	BLNASAXX	B/C	INFO	
BLN0934I	BLNASMD	B/C	INFO	
BLN1003I	BLNBADM2	B/C	ERROR	
BLN1004I	BLNBABH2	B	INFO	
BLN1004I	BLNBAFLS	B	INFO	
BLN1004I	BLNBARDR	B	INFO	

Message ID	Module Issuing	Modes	Type	Inserts
BLN1005D	BLNBARDR	C	INFO	func
BLN1006I	BLNBARDR	B	ERROR	
BLN1007I	BLNBABH1	B/C	ERROR	
BLN1007I	BLNBADM1	B/C	ERROR	
BLN1007I	BLNSFDID	B/C	ERROR	
BLN1008I	BLNBAINT	B/C	INFO	
BLN1013I	BLNMACBC	B/C	ERROR	
BLN2006I	BLNDMADU	B/C	ERROR	
BLN2007I	BLNDMCNT	B/C	INFO	maxrec, userec
BLN2009I	BLNDMDEL	B/C	ERROR	
BLN2013I	BLNDMDMU	B/C	ERROR	rsc
BLN2013I	BLNDMRDF	B/C	ERROR	
BLN2013I	BLNDMWDF	B/C	ERROR	
BLN2014I	BLNDMDMU	B/C	ERROR	
BLN2017I	BLNDMDDL	B/C	ERROR	
BLN2017I	BLNDMDEL	B/C	ERROR	
BLN2017I	BLNDMPDL	B/C	ERROR	
BLN2019I	BLNDMAD1	B/C	INFO	
BLN3002I	BLNONEND	B/C	ERROR	rsc
BLN3002I	BLNONINT	B/C	ERROR	rsc
BLN3002I	BLNSFDID	B/C	ERROR	rsc
BLN4001I	BLNCMVOL	B/C	ERROR	
BLN4003I	BLNOFFLD	B/C	ERROR	
BLN4004I	BLNOFFLD	B/C	INFO	dumpid, ERASED
BLN4005I	BLNOFFLD	B/C	ERROR	rsc
BLN4006I	BLNBAPF1	B/C	ERROR	
BLN4006I	BLNSFOFL	B/C	ERROR	
BLN4007I	BLNOFFLD	B/C	ERROR	
BLN4007I	BLNONINT	B/C	ERROR	
BLN4007I	BLNSFOFL	B/C	ERROR	
BLN4007I	BLNSFONL	B/C	ERROR	
BLN4010I	BLNOFFLD	B/C	INFO	
BLN5022I	BLNDDPOP	B/C	ERROR	
BLN5022I	BLNDDVFA	B/C	ERROR	
BLN5022I	BLNMKVMR	B/C	ERROR	
BLN7010I	BLNSMBLE	B/C	INFO	
BLN7010I	BLNSMDSP	B/C	INFO	
BLN7010I	BLNSMFND	B/C	INFO	
BLN7010I	BLNSMSEL	B/C	INFO	
BLN7013I	BLNLBVER	B/C	ERROR	block, rsc
BLN7515I	BLNAREXD	B/C	ERROR	rtn
BLN7516I	BLNAREXL	B/C	ERROR	rtn, rsc
BLN7518I	BLNAXSU1	B/C	ERROR	
BLN7520I	BLNAREXD	B/C	ERROR	
BLN7521I	BLNAXPD3	B/C	INFO	
BLN9001I	BLNARDLO	B/C	ERROR	
BLN9001I	BLNAREXA	B/C	ERROR	
BLN9001I	BLNARGRN	B/C	ERROR	
BLN9001I	BLNARPRN	B/C	ERROR	
BLN9001I	BLNARXPL	B/C	ERROR	
BLN9001I	BLNARPD6	B/C	ERROR	
BLN9001I	BLNARS6U	B/C	ERROR	
BLN9001I	BLNBAINT	B/C	ERROR	
BLN9001I	BLNCMVOL	B/C	ERROR	
BLN9001I	BLNDDCDB	B/C	ERROR	
BLN9001I	BLNDDFMT	B/C	ERROR	
BLN9001I	BLNDDFPP	B/C	ERROR	

Message ID	Module Issuing	Modes	Type	Inserts
BLN9001I	BLNDDINT	B/C	ERROR	
BLN9001I	BLNDDNBU	B/C	ERROR	
BLN9001I	BLNDDSYI	B/C	ERROR	
BLN9001I	BLNDMDDL	B/C	ERROR	
BLN9001I	BLNDMPDL	B/C	ERROR	
BLN9001I	BLNDSGBF	B/C	ERROR	
BLN9001I	BLNDVKCB	B/C	ERROR	
BLN9001I	BLNDVSIA	B/C	ERROR	
BLN9001I	BLNLBARC	B/C	ERROR	
BLN9001I	BLNLBARR	B/C	ERROR	
BLN9001I	BLNLBCAC	B/C	ERROR	
BLN9001I	BLNLBCAE	B/C	ERROR	
BLN9001I	BLNLBCAI	B/C	ERROR	
BLN9001I	BLNLBCAL	B/C	ERROR	
BLN9001I	BLNLBCHC	B/C	ERROR	
BLN9001I	BLNLBCHE	B/C	ERROR	
BLN9001I	BLNLBCHI	B/C	ERROR	
BLN9001I	BLNLBCHL	B/C	ERROR	
BLN9001I	BLNLBCHR	B/C	ERROR	
BLN9001I	BLNLBCHW	B/C	ERROR	
BLN9001I	BLNLBLAL	B/C	ERROR	
BLN9001I	BLNLBSIM	B/C	ERROR	
BLN9001I	BLNMKBMD	B/C	ERROR	
BLN9001I	BLNSFVDF	B/C	ERROR	
BLN9001I	BLNSMBLO	B/C	ERROR	
BLN9001I	BLNSMBMA	B/C	ERROR	
BLN9001I	BLNSMCMO	B/C	ERROR	
BLN9001I	BLNSMFRE	B/C	ERROR	
BLN9001I	BLNSMGRE	B/C	ERROR	
BLN9001I	BLNSMHXA	B/C	ERROR	
BLN9001I	BLNSMHXO	B/C	ERROR	
BLN9001I	BLNSMLKB	B/C	ERROR	
BLN9001I	BLNSMLKC	B/C	ERROR	
BLN9001I	BLNSMLKD	B/C	ERROR	
BLN9001I	BLNSMLKL	B/C	ERROR	
BLN9001I	BLNSMLKO	B/C	ERROR	
BLN9001I	BLNSMOCL	B/C	ERROR	
BLN9001I	BLNSMPHP	B/C	ERROR	
BLN9001I	BLNSMPSO	B/C	ERROR	
BLN9001I	BLNSMPSW	B/C	ERROR	
BLN9001I	BLNSMTXA	B/C	ERROR	
BLN9001I	BLNSMTXO	B/C	ERROR	
BLN9002I	BLNAREXG	B/C	ERROR	rtc, rsc
BLN9002I	BLNCMERM	B/C	ERROR	rtc, rsc
BLN9002I	BLNMACBG	B/C	ERROR	rtc, rsc
BLN9003I	BLNCMERM	B/C	ERROR	msg
BLN9004I	BLNBABH1	B/C	ERROR	entry
BLN9004I	BLNBADM1	B/C	ERROR	entry
BLN9004I	BLNBADM2	B/C	ERROR	entry
BLN9004I	BLNBADS1	B/C	ERROR	entry
BLN9004I	BLNBADV1	B/C	ERROR	entry
BLN9004I	BLNBAPF1	B/C	ERROR	entry
BLN9004I	BLNBAPN1	B/C	ERROR	entry
BLN9004I	BLNCMFIL	B/C	ERROR	entry
BLN9004I	BLNCMHEX	B/C	ERROR	entry
BLN9004I	BLNCMVOL	B/C	ERROR	entry
BLN9004I	BLNCMXCF	B/C	ERROR	entry

Message ID	Module Issuing	Modes	Type	Inserts
BLN9004I	BLNDDPOP	B/C	ERROR	entry
BLN9004I	BLNDDVFA	B/C	ERROR	entry
BLN9004I	BLNDDVFN	B/C	ERROR	entry
BLN9004I	BLNDMUAD	B/C	ERROR	entry
BLN9006I	BLNCMCHD	B/C	INFO	
BLN9006I	BLNDMAD1	B/C	INFO	
BLN9006I	BLNDSCEN	B/C	INFO	
BLN9007I	BLNAREXA	B/C	ERROR	rsc
BLN9007I	BLNARINT	B/C	ERROR	rsc
BLN9007I	BLNDDFPP	B/C	ERROR	rsc
BLN9007I	BLNDSDSP	B/C	ERROR	rsc
BLN9007I	BLNDSPRT	B/C	ERROR	rsc
BLN9007I	BLNDSTRM	B/C	ERROR	rsc
BLN9007I	BLNDSYMP	B/C	ERROR	rsc
BLN9007I	BLNDVSIN	B/C	ERROR	rsc
BLN9007I	BLNFNARC	B/C	ERROR	rsc
BLN9007I	BLNFNDDC	B/C	ERROR	rsc
BLN9007I	BLNFNDMC	B/C	ERROR	rsc
BLN9007I	BLNFNDSC	B/C	ERROR	rsc
BLN9007I	BLNFNDVS	B/C	ERROR	rsc
BLN9007I	BLNFNMSK	B/C	ERROR	rsc
BLN9007I	BLNFNPFC	B/C	ERROR	rsc
BLN9007I	BLNFNPNC	B/C	ERROR	rsc
BLN9007I	BLNFNSMC	B/C	ERROR	rsc
BLN9007I	BLNMKDSP	B/C	ERROR	rsc
BLN9007I	BLNMKINT	B/C	ERROR	rsc
BLN9007I	BLNMKMSK	B/C	ERROR	rsc
BLN9007I	BLNSMBLE	B/C	ERROR	rsc
BLN9007I	BLNSMBME	B/C	ERROR	rsc
BLN9007I	BLNSMHXA	B/C	ERROR	rsc
BLN9007I	BLNSMINT	B/C	ERROR	rsc
BLN9007I	BLNSMPCD	B/C	ERROR	rsc
BLN9007I	BLNSMPRT	B/C	ERROR	rsc
BLN9007I	BLNSMTRM	B/C	ERROR	rsc
BLN9007I	BLNSMTXA	B/C	ERROR	rsc
BLN9008I	BLNBARDR	B/C	ERROR	mod, rsc
BLN9008I	BLNDMGNT	B/C	ERROR	mod, rsc
BLN9010I	BLNDDMDT	B/C	INFO	
BLN9010I	BLNDDRPT	B/C	INFO	
BLN9012I	BLNARPRT	B/C	INFO	
BLN9012I	BLNAXPD2	B/C	INFO	
BLN9012I	BLNDDDPT	B/C	INFO	
BLN9012I	BLNDDFPT	B/C	INFO	
BLN9012I	BLNDDPRT	B/C	INFO	
BLN9012I	BLNDDRPT	B/C	INFO	
BLN9012I	BLNDMPDL	B/C	INFO	
BLN9012I	BLNDSPRT	B/C	INFO	
BLN9012I	BLNSMPRT	B/C	INFO	
BLN9016I	BLNBADM1	B/C	ERROR	
BLN9016I	BLNBADS1	B/C	ERROR	
BLN9016I	BLNBAPF1	B/C	ERROR	
BLN9016I	BLNBAPN1	B/C	ERROR	
BLN9018I	BLNDMADU	B/C	INFO	dumpid, func
BLN9018I	BLNDMDEL	B/C	INFO	dumpid, func
BLN9018I	BLNONEND	B/C	INFO	dumpid, func
BLN9019I	BLNBABH1	B/C	ERROR	dumpid
BLN9019I	BLNCMCHD	B/C	ERROR	dumpid

Message ID	Module Issuing	Modes	Type	Inserts
BLN9019I	BLNOFFLD	B/C	ERROR	dumpid
BLN9019I	BLNSFDID	B/C	ERROR	dumpid
BLN9021I	BLNDDRDF	B/C	INFO	
BLN9022I	BLNARPRT	B/C	ERROR	
BLN9022I	BLNAXPD2	B/C	ERROR	
BLN9022I	BLNAXPD6	B/C	ERROR	
BLN9022I	BLNDDFPP	B/C	ERROR	
BLN9022I	BLNDDPRT	B/C	ERROR	
BLN9022I	BLNDDRPT	B/C	ERROR	
BLN9022I	BLNSMPCD	B/C	ERROR	
BLN9030I	BLNCMXDE	B/C	ERROR	req, fil, rsc
BLN9031I	BLNCMERM	B/C	ERROR	rtc, rsc
BLN9032I	BLNBABH1	B/C	ERROR	dumpid
BLN9032I	BLNSFDID	B/C	ERROR	dumpid
BLN9042I	BLNRASEC	B/C	ERROR	
BLN9043I	BLNCMERM	B/C	ERROR	
BLN9050I	BLNCMERM	B/C	ERROR	
BLN9051I	BLNCMERM	B/C	ERROR	

Info/Analysis Module/Reason Cross-Reference

In the following list, object modules that use reason codes are listed alphabetically by module name. Each module name is followed by the reason code that the module uses, the message ID in which it may be inserted, and a description of the reason code.

Module Issuing	Reason Code	Message ID	Description
BLNADIFI	0004		Invalid dump access entry point
BLNADIFI	0008		Storage allocation failure
BLNADIFI	0012		Unable to get ADCB
BLNADIFR	0900		Unable to get ADCB
BLNADIFX	0900		Unable to get ADCB
BLNADIFX	0904		Unable to delete ADCB from BLX
BLNARDLO	7550		Storage allocation failure
BLNARDSP	7540		No analysis routines list available
BLNARDSP	7541		Initialization error
BLNAREXA	7544	BLN9007I	Unable to add ARCB to BLX
BLNAREXA	7592		Storage allocation failure
BLNAREXC	7570		Initialization error
BLNAREXG	7581	BLN9002I	Unable to delete ARCB from BLX
BLNAREXL	7582	BLN7516I	Load of analysis routine failed
BLNARFND	7567		Initialization error
BLNARFND	7568		No analysis routines list available
BLNARGRN	7503		Storage allocation failure
BLNARGRN	7504		Storage allocation failure
BLNARINT	7502	BLN9007I	Duplicate initialization
BLNARPRN	7505		Storage allocation failure
BLNARPRT	7560		Initialization not done
BLNARPRT	7567		Initialization error
BLNARXPL	7583		Storage allocation failure
BLNASADF	0923		Format error in section 6
BLNASADL	0923		Format error in section 6
BLNASAXA	0923		Format error in section 6
BLNASAXC	0923		Format error in section 6
BLNASAXK	0923		Format error in section 6
BLNASAXT	0923		Format error in section 6
BLNASAXX	0923		Format error in section 6
BLNASFMT	0921		Storage allocation failure
BLNASFSY	0925		Data not found
BLNASGET	0921		Storage allocation failure
BLNASLCB	0920		Unable to get ASCB
BLNASLCB	0921		Storage allocation failure
BLNASLD1	0921		Storage allocation failure
BLNASLD1	0925		Data not found
BLNASLD1	0926		Error in parameter list
BLNASLD6	0925		Data not found
BLNASLOC	0926		Error in parameter list
BLNASLST	0921		Storage allocation failure
BLNASLST	0925		Data not found
BLNASMAP	0921		Storage allocation failure
BLNASMAP	0927		Invalid dump name
BLNASMD	0923		Format error in section 6
BLNASMP1	0921		Storage allocation failure
BLNASRSR	0921		Storage allocation failure
BLNASRSR	0925		Data not found
BLNASRSR	0928		Dump access error
BLNASTRM	0920		Unable to get ASCB

Module Issuing	Reason Code	Message ID	Description
BLNAXDAC	0100		Invalid environment pointer
BLNAXDA1	0016		Invalid request type
BLNAXDA1	0040		Dump name changed
BLNAXDA1	0100		Invalid environment pointer
BLNAXFMT	0100		Invalid environment pointer
BLNAXFM1	0008		Storage length too small
BLNAXFM1	0016		Storage allocation failure
BLNAXFM1	0096		Unable to access ARCB
BLNAXGST	0100		Invalid environment pointer
BLNAXGS1	0096		Unable to access ARCB
BLNAXGS2	0004		Storage allocation failure
BLNAXGS2	0008		Invalid length request
BLNAXGS3	0012		Invalid free address passed
BLNAXPDS	0100		Invalid environment pointer
BLNAXPD1	0016		No output device specified
BLNAXPD1	0020		Invalid buffer length
BLNAXPD1	0096		Unable to access ARCB
BLNAXPD3	0004		Output routed to alternate device
BLNAXPD3	0096		Unable to access ARCB
BLNAXPD5	0096		Unable to access ARCB
BLNAXPD6	0096		Unable to access ARCB
BLNAXPD7	0096		Unable to access ARCB
BLNAXSAC	0100		Invalid environment pointer
BLNAXSA1	0100		Invalid environment pointer
BLNAXSA1	0926		Error in parameter list
BLNAXSUA	0024		Symptom record update error
BLNAXSUA	0092		Storage allocation failure
BLNAXSUP	0100		Invalid environment pointer
BLNAXSU1	0004		Symptom record has been modified
BLNAXSU1	0016		Symptom record format invalid
BLNAXSU1	0020		Dump name changed
BLNAXSU1	0092		Storage allocation failure
BLNAXSU1	0096		Unable to access ARCB
BLNAXS3U	0092		Storage allocation failure
BLNAXS4U	0092		Storage allocation failure
BLNAXS5U	0092		Storage allocation failure
BLNAXS6U	0024		Symptom record update error
BLNAXS6U	0092		Storage allocation failure
BLNAXS6V	0012		Section 6 updates invalid
BLNAXS6V	0024		Symptom record update error
BLNCMCHD	0925		Data not found
BLNCMXDE	9501		BLX allocation - invalid unit
BLNCMXDE	9502		BLX allocation - extract macro error
BLNCMXDE	9503		BLX allocation - device not supported
BLNCMXDE	9504		BLX allocation - RECFM not supported
BLNCMXDE	9505		BLX allocation - blocked records not supported
BLNCMXDE	9506		BLX allocation - access mode not supported
BLNCMXDE	9507		BLX allocation - DLBL statement not supported
BLNCMXDE	9508		BLX allocation - open VTOC not supported
BLNCMXDE	9509		BLX allocation - close VTOC not supported
BLNCMXDE	9510		BLX allocation - extent VOLSER invalid
BLNCMXDE	9511		BLX allocation - extent logical unit invalid
BLNCMXDE	9512		BLX allocation - DDNAME not found
BLNCMXDE	9519		BLX free error
BLNCMXDE	9521		BLX open - permanent open error
BLNCMXDE	9522		BLX open - invalid FSEQ keyword
BLNCMXDE	9523		BLX open - data set already open

Module Issuing	Reason Code	Message ID	Description
BLNCMXDE	9531		BLX close - permanent close error
BLNCMXDE	9532		BLX close - end of extent
BLNCMXDE	9541		BLX read - update not allowed
BLNCMXDE	9542		BLX read - wrong access type
BLNCMXDE	9543		BLX read - buffer length too small
BLNCMXDE	9544		BLX read - invalid relative record number
BLNCMXDE	9545		BLX read - VSAM position error
BLNCMXDE	9546		BLX read - invalid record length
BLNCMXDE	9547		BLX read - invalid key length
BLNCMXDE	9548		BLX read - VSAM MODCB verb failed
BLNCMXDE	9549		BLX read - undefined request
BLNCMXDE	9591		Unknown error condition
BLNDDBAS	5006		Invalid characters in BASE
BLNDDBAS	5007		Imbedded blanks in BASE
BLNDDCDB	5023		Storage allocation failure
BLNDDFDZ	5015		Dump data not available
BLNDDFMT	5014		Storage allocation failure
BLNDDFND	5018		Invalid find command
BLNDDFPP	5019		Storage allocation failure
BLNDDFPP	5020		Storage allocation failure
BLNDDFPP	5021	BLN9007I	Error freeing buffer chain
BLNDDFPP	5022		Error printing buffers
BLNDDINT	5002		Storage allocation failure
BLNDDINT	5003		Storage allocation failure
BLNDDNBU	5016		Storage allocation failure
BLNDDPRT	5017		Invalid print command
BLNDDQAL	5008		Invalid characters in QUAL
BLNDDQAL	5009		Imbedded blanks in QUAL
BLNDDRDF	5010		Locator not found
BLNDDRDF	5011		Invalid base in locator
BLNDDRDF	5012		Invalid length in locator
BLNDDSYI	5004		Storage allocation failure
BLNDDVRF	5013		DISPLAY and BASE fields blank
BLNDMAD1	0925		Data not found
BLNDMDEL	2204		Dump access failure
BLNDMDEL	2208		No dump delete verification
BLNDMRDF	2004	BLN2013I	Dump management file open error
BLNDMRDF	2008	BLN2013I	Storage allocation failure
BLNDMRDF	2012	BLN2013I	Dump management file close error
BLNDMRDF	2020	BLN2013I	Dump management file read error
BLNDMRDF	2028	BLN2013I	Invalid control record in dump management file
BLNDMRD1	2024	BLN2013I	Invalid data record in dump management file
BLNDMUAI	2008		Storage allocation failure
BLNDMUDI	0925		Data not found
BLNDMUDI	2008		Storage allocation failure
BLNDMWDF	2004	BLN2013I	Dump management file open error
BLNDMWDF	2012	BLN2013I	Dump management file close error
BLNDMWDF	2016	BLN2013I	Dump management file write error
BLNDSCEN	6020		No symptom record in dump
BLNDSDSP	6004	BLN9007I	Damage to DSCB suspected
BLNDSGBF	6008		Storage allocation failed
BLNDSPRT	6004	BLN9007I	Damage to DSCB suspected
BLNDSPRT	6012		Permanent I/O error on print
BLNDSPRT	6016		BLX error
BLNDSTRM	6004	BLN9007I	Damage to DSCB suspected
BLNDSYMP	6000	BLN9007I	Unknown subfunction request
BLNDVKCP	5901		Storage allocation failure

Module Issuing	Reason Code	Message ID	Description
BLNDVSIA	5921		Storage allocation failure
BLNDVSIA	5922		Storage allocation failure
BLNDVSIA	5923		Storage allocation failure
BLNDVSIA	5924		Storage allocation failure
BLNDVSIA	5925		Storage allocation failure
BLNDVSIN	5902	BLN9007I	Unable to access DVVT
BLNFNARC	7501	BLN9007I	Unknown subfunction request
BLNFNDDC	5001	BLN9007I	Unknown subfunction request
BLNFNDMC	1101	BLN9007I	Unable to access DMCB
BLNFNDMC	1103	BLN9007I	Unable to access DSCB
BLNFNDVS	5701	BLN9007I	Unknown subfunction request
BLNFNMSK	5901	BLN9007I	Unknown subfunction request
BLNFNPFC	1101	BLN9007I	Unable to access DMCB
BLNFNPNC	1101	BLN9007I	Unable to access DMCB
BLNFNSMC	7099	BLN9007I	Unknown subfunction request
BLNLBARC	7406		Storage allocation failure
BLNLBARC	7407		Storage allocation failure
BLNLBARR	7408		Storage allocation failure
BLNLBCAC	7464		Storage allocation failure
BLNLBCAC	7477		Storage allocation failure
BLNLBCAE	7465		Storage allocation failure
BLNLBCAE	7475		Storage allocation failure
BLNLBCAI	7466		Storage allocation failure
BLNLBCAI	7476		Storage allocation failure
BLNLBCAL	7467		Storage allocation failure
BLNLBCAL	7468		Storage allocation failure
BLNLBCHC	7473		Storage allocation failure
BLNLBCHC	7483		Storage allocation failure
BLNLBCHE	7472		Storage allocation failure
BLNLBCHE	7478		Storage allocation failure
BLNLBCHI	7471		Storage allocation failure
BLNLBCHI	7481		Storage allocation failure
BLNLBCHL	7469		Storage allocation failure
BLNLBCHL	7479		Storage allocation failure
BLNLBCHR	7474		Storage allocation failure
BLNLBCHW	7470		Storage allocation failure
BLNLBLAL	7460		Storage allocation failure
BLNLBSIM	7405		Storage allocation failure
BLNLBVER	7401	BLN7013I	Unknown number of elements in array
BLNLBVER	7402	BLN7013I	Array extension type = B, no chain extension
BLNLBVER	7403	BLN7013I	Invalid array extension type field
BLNLBVER	7411	BLN7013I	Invalid end condition field
BLNLBVER	7412	BLN7013I	Invalid end condition length field
BLNLBVER	7413	BLN7013I	Invalid chain extension type field
BLNLBVER	7414	BLN7013I	Invalid chain extension address list pointer
BLNLBVER	7421	BLN7013I	Invalid hex extension length
BLNLBVER	7431	BLN7013I	Invalid text extension length
BLNLBVER	7441	BLN7013I	Keyfield length is zero
BLNLBVER	7442	BLN7013I	Invalid keyfield format field
BLNLBVER	7443	BLN7013I	Invalid keyfield type field
BLNLBVER	7451	BLN7013I	Number of entries in format descriptor is zero
BLNLBVER	7451	BLN7013I	Number of entries in linkage descriptor is zero
BLNLBVER	7452	BLN7013I	Length error in format descriptor
BLNLBVER	7452	BLN7013I	Length error in linkage descriptor
BLNLBVER	7453	BLN7013I	Invalid offset attribute type
BLNLBVER	7454	BLN7013I	Invalid sequential offset format
BLNLBVER	7455	BLN7013I	Invalid specific offset format

Module Issuing	Reason Code	Message ID	Description
BLNMACBG	1104	BLN9002I	Unable to access control blocks
BLNOFFLD	4004	BLN4005I	Error deleting dump from library
BLNOFFLD	4008	BLN4005I	Error adding VOLID to dump management file
BLNOFFLD	4016	BLN4005I	Storage allocation failure
BLNOFFLD	4020	BLN4005I	No VOLID specified
BLNOFFLD	4024	BLN4005I	Dump not in dump management list
BLNOFFLD	4028	BLN4005I	Read of dump management list failed
BLNOFFLD	4030	BLN4005I	Invalid bypass-no previous offload
BLNOFFLD	4032	BLN4005I	Invalid bypass-dump altered
BLNOFFLD	4036	BLN4005I	Dump not in library
BLNOFFLD	4040	BLN4005I	Job submit failed
BLNOFOUT	4012	BLN4005I	Error writing dump to tape
BLNOFOUT	4040	BLN4005I	Tape allocate error
BLNOFOUT	4044	BLN4005I	Tape open error
BLNOFOUT	4048	BLN4005I	Tape close error
BLNOFOUT	4052	BLN4005I	Tape deallocate error
BLNOFUDM	4028	BLN4005I	Read of dump management list failed
BLNONDDT	3016	BLN3002I	Read dump tape failed
BLNONEND	3016	BLN3002I	Read dump tape failed
BLNONEND	3020	BLN3002I	Tape open error
BLNONEND	3024	BLN3002I	Tape allocate error
BLNONEND	3036	BLN3002I	Write dump management file error
BLNONINT	3020	BLN3002I	Tape open error
BLNONINT	3024	BLN3002I	Tape allocate error
BLNONINT	3032	BLN3002I	Error reading dump management file
BLNONINT	3040	BLN3002I	Error processing dump management file
BLNONINT	3044	BLN4007I	No VOLID specified
BLNONINT	3048	BLN3002I	Dump already in library
BLNONINT	3052	BLN3002I	Multi-dump not allowed
BLNONINT	3056	BLN3002I	Job submit failed
BLNONLOD	3012	BLN3002I	Error writing dump to system
BLNONMUD	3008	BLN3002I	Undefined dump type
BLNONMUD	3016	BLN3002I	Read dump tape failed
BLNONMUD	3020	BLN3002I	Tape open error
BLNONMUD	3022	BLN3002I	Tape close error
BLNONMUD	3060	BLN3002I	No operator requested dump
BLNONTRW	3012	BLN3002I	Error writing dump to system
BLNONTRW	3016	BLN3002I	Read dump tape failed
BLNONVSE	3008	BLN3002I	Undefined dump type
BLNONVSE	3016	BLN3002I	Read dump tape failed
BLNONVSE	3020	BLN3002I	Tape open error
BLNONVSE	3022	BLN3002I	Tape close error
BLNSMBLE	0925		Data not found
BLNSMBLE	7025	BLN9007I	Internal error
BLNSMBLF	7026		Storage allocation failure
BLNSMBLO	7062		Storage allocation failure
BLNSMBMA	7021		Storage allocation failure
BLNSMBME	7004	BLN9007I	Internal error
BLNSMCMO	7024		Storage allocation failure
BLNSMDSP	7023		Initialization error
BLNSMFND	7100	BLN9007I	Initialization error
BLNSMFND	7101		Data not available on find
BLNSMFND	7102		Invalid form of find command
BLNSMFRE	7094		Storage allocation failure
BLNSMGRE	7093		Storage allocation failure
BLNSMHXA	7076		Storage allocation failure
BLNSMHXA	7078		Storage allocation failure

Module Issuing	Reason Code	Message ID	Description
BLNSMHXO	7080		Storage allocation failure
BLNSMINT	7002	BLN9007I	Double initialization
BLNSMLKC	7086		Storage allocation failure
BLNSMLKD	7090		Storage allocation failure
BLNSMLKL	7082		Storage allocation failure
BLNSMLKL	7083		Control block locator not found
BLNSMLKO	7088		Starting point not found
BLNSMLKO	7089		Storage allocation failure
BLNSMOCL	7095		Storage allocation failure
BLNSMPCD	7122	BLN9007I	Internal error freeing buffers
BLNSMPHP	7018		Storage allocation failure
BLNSMPRT	7084	BLN9007I	Initialization error
BLNSMPSO	7064		Storage allocation failure
BLNSMPSW	7092		Storage allocation failure
BLNSMSEL	7110	BLN9007I	Initialization error
BLNSMTRM	7003	BLN9007I	Double termination
BLNSMTXA	7066	BLN9007I	LBD not found in symptom record
BLNSMTXA	7068		Storage allocation failure
BLNSMTXA	7070		Storage allocation failure
BLNSMTXO	7072		Storage allocation failure

Info/Analysis Reason/Module Cross-Reference

In the following list, Info/Analysis reason codes are listed numerically. Each reason code is followed by name of the module that issues the reason code, the message ID in which it may be inserted, and a description of the reason code. (Refer to “Info/Analysis Module/Reason Cross-Reference” on page 258 for additional information).

Reason Code	Module Issuing	Message ID	Description
0004	BLNADIFI		Invalid dump access entry point
0004	BLNAXGS2		Storage allocation failure
0004	BLNAXPD3		Output routed to alternate device
0004	BLNAXSU1		Symptom record has been modified
0008	BLNADIFI		Storage allocation failure
0008	BLNAXFM1		Storage length too small
0008	BLNAXGS2		Invalid length request
0012	BLNADIFI		Unable to get ADCB
0012	BLNAXGS3		Invalid free address passed
0012	BLNAXS6V		Section 6 updates invalid
0016	BLNAXDA1		Invalid request type
0016	BLNAXFM1		Storage allocation failure
0016	BLNAXPD1		No output device specified
0016	BLNAXSU1		Symptom record format invalid
0020	BLNAXPD1		Invalid buffer length
0020	BLNAXSU1		Dump name changed
0024	BLNAXSUA		Symptom record update error
0024	BLNAXS6U		Symptom record update error
0024	BLNAXS6V		Symptom record update error
0040	BLNAXDA1		Dump name changed
0092	BLNAXSUA		Storage allocation failure
0092	BLNAXSU1		Storage allocation failure
0092	BLNAXS3U		Storage allocation failure
0092	BLNAXS4U		Storage allocation failure
0092	BLNAXS5U		Storage allocation failure
0092	BLNAXS6U		Storage allocation failure
0096	BLNAXFM1		Unable to access ARCB
0096	BLNAXGS1		Unable to access ARCB
0096	BLNAXPD1		Unable to access ARCB
0096	BLNAXPD3		Unable to access ARCB
0096	BLNAXPD5		Unable to access ARCB
0096	BLNAXPD6		Unable to access ARCB
0096	BLNAXPD7		Unable to access ARCB
0096	BLNAXSU1		Unable to access ARCB
0100	BLNAXDAC		Invalid environment pointer
0100	BLNAXFMT		Invalid environment pointer
0100	BLNAXGST		Invalid environment pointer
0100	BLNAXPDS		Invalid environment pointer
0100	BLNAXSAC		Invalid environment pointer
0100	BLNAXSA1		Invalid environment pointer
0100	BLNAXSUP		Invalid environment pointer
0900	BLNADIFR		Unable to get ADCB
0900	BLNADIFX		Unable to get ADCB
0904	BLNADIFX		Unable to delete ADCB from BLX
0920	BLNASLCB		Unable to get ASCB
0920	BLNASTRM		Unable to get ASCB
0921	BLNASFMT		Storage allocation failure
0921	BLNASGET		Storage allocation failure
0921	BLNASLCB		Storage allocation failure

Reason Code	Module Issuing	Message ID	Description
0921	BLNASLD1		Storage allocation failure
0921	BLNASLST		Storage allocation failure
0921	BLNASMAP		Storage allocation failure
0921	BLNASMP1		Storage allocation failure
0923	BLNASADF		Format error in section 6
0923	BLNASADL		Format error in section 6
0923	BLNASAXA		Format error in section 6
0923	BLNASAXC		Format error in section 6
0923	BLNASAXK		Format error in section 6
0923	BLNASAXT		Format error in section 6
0923	BLNASAXX		Format error in section 6
0923	BLNASMD		Format error in section 6
0925	BLNASFSY		Data not found
0925	BLNASLD1		Data not found
0925	BLNASLD6		Data not found
0925	BLNASLST		Data not found
0925	BLNASMAP		Data not found
0925	BLNASRSR		Data not found
0925	BLNCMCHD		Data not found
0925	BLNDMAD1		Data not found
0925	BLNDMUDI		Data not found
0925	BLNSMBLE		Data not found
0926	BLNASLD1		Error in parameter list
0926	BLNASLOC		Error in parameter list
0926	BLNAXSA1		Error in parameter list
0927	BLNASMAP		Invalid dump name
0928	BLNASRSR		Dump access error
1101	BLNFNDMC	BLN9007I	Unable to access DMCB
1101	BLNFNPFC	BLN9007I	Unable to access DMCB
1101	BLNFNPNC	BLN9007I	Unable to access DMCB
1103	BLNFNDMC	BLN9007I	Unable to access DSCB
1104	BLNMACBG	BLN9002I	Unable to access control blocks
2004	BLNDMRDF	BLN2013I	Dump management file open error
2004	BLNDMWDF	BLN2013I	Dump management file open error
2008	BLNDMRDF	BLN2013I	Storage allocation failure
2008	BLNDMUAI		Storage allocation failure
2008	BLNDMUDI		Storage allocation failure
2012	BLNDMRDF	BLN2013I	Dump management file close error
2012	BLNDMWDF	BLN2013I	Dump management file close error
2016	BLNDMWDF	BLN2013I	Dump management file write error
2020	BLNDMRDF	BLN2013I	Dump management file read error
2024	BLNDMRD1	BLN2013I	Invalid data record in dump management file
2028	BLNDMRDF	BLN2013I	Invalid control record in dump management file
2204	BLNDMDEL		Dump access failure
2208	BLNDMDEL		No dump delete verification
3008	BLNONMUD	BLN3002I	Undefined dump type
3008	BLNONVSE	BLN3002I	Undefined dump type
3012	BLNONLOD	BLN3002I	Error writing dump to system
3012	BLNONSC1	BLN3002I	Error writing dump to system
3012	BLNONTRW	BLN3002I	Error writing dump to system
3016	BLNONDDT	BLN3002I	Read dump tape failed
3016	BLNONEND	BLN3002I	Read dump tape failed
3016	BLNONMUD	BLN3002I	Read dump tape failed
3016	BLNONSCM	BLN3002I	Read dump tape failed
3016	BLNONSC1	BLN3002I	Read dump tape failed
3016	BLNONTRW	BLN3002I	Read dump tape failed
3016	BLNONVSE	BLN3002I	Read dump tape failed

Reason Code	Module Issuing	Message ID	Description
3020	BLNONEND	BLN3002I	Tape open error
3020	BLNONINT	BLN3002I	Tape open error
3020	BLNONMUD	BLN3002I	Tape open error
3020	BLNONSCM	BLN3002I	Tape open error
3020	BLNONVSE	BLN3002I	Tape open error
3022	BLNONMUD	BLN3002I	Tape close error
3022	BLNONSCM	BLN3002I	Tape close error
3022	BLNONVSE	BLN3002I	Tape close error
3024	BLNONEND	BLN3002I	Tape allocate error
3024	BLNONINT	BLN3002I	Tape allocate error
3028	BLNONSCM	BLN3002I	Storage allocation failure
3032	BLNONINT	BLN3002I	Error reading dump management file
3036	BLNONEND	BLN3002I	Write dump management file error
3040	BLNONINT	BLN3002I	Error processing dump management file
3044	BLNONINT	BLN4007I	No VOLID specified
3048	BLNONINT	BLN3002I	Dump already in library
3052	BLNONINT	BLN3002I	Multi-dump not allowed
3056	BLNONINT	BLN3002I	Job submit failed
3060	BLNONMUD	BLN3002I	No operator requested dump
4004	BLNOFFLD	BLN4005I	Error deleting dump from library
4008	BLNOFFLD	BLN4005I	Error adding VOLID to dump management file
4012	BLNOFOUT	BLN4005I	Error writing dump to tape
4016	BLNOFFLD	BLN4005I	Storage allocation failure
4020	BLNOFFLD	BLN4005I	No VOLID specified
4024	BLNOFFLD	BLN4005I	Dump not in dump management list
4028	BLNOFFLD	BLN4005I	Read of dump management list failed
4028	BLNOFUDM	BLN4005I	Read of dump management list failed
4030	BLNOFFLD	BLN4005I	Invalid bypass-no previous offload
4032	BLNOFFLD	BLN4005I	Invalid bypass-dump altered
4036	BLNOFFLD	BLN4005I	Dump not in library
4040	BLNOFFLD	BLN4005I	Job submit failed
4040	BLNOFOUT	BLN4005I	Tape allocate error
4044	BLNOFOUT	BLN4005I	Tape open error
4048	BLNOFOUT	BLN4005I	Tape close error
4052	BLNOFOUT	BLN4005I	Tape deallocate error
5001	BLNFNDDC	BLN9007I	Unknown subfunction request
5002	BLNDDINT		Storage allocation failure
5003	BLNDDINT		Storage allocation failure
5004	BLNDDSYI		Storage allocation failure
5006	BLNDDBAS		Invalid characters in BASE
5007	BLNDDBAS		Imbedded blanks in BASE
5008	BLNDDQAL		Invalid characters in QUAL
5009	BLNDDQAL		Imbedded blanks in QUAL
5010	BLNDDRDF		Locator not found
5011	BLNDDRDF		Invalid base in locator
5012	BLNDDRDF		Invalid length in locator
5013	BLNDDVRF		DISPLAY and BASE fields blank
5014	BLNDDFMT		Storage allocation failure
5015	BLNDDFDZ		Dump data not available
5016	BLNDDNBU		Storage allocation failure
5017	BLNDDPRT		Invalid print command
5018	BLNDDFND		Invalid find command
5019	BLNDDFPP		Storage allocation failure
5020	BLNDDFPP		Storage allocation failure
5021	BLNDDFPP	BLN9007I	Error freeing buffer chain
5022	BLNDDFPP		Error printing buffers
5023	BLNDDCDB		Storage allocation failure

Reason Code	Module Issuing	Message ID	Description
5701	BLNFNDVS	BLN9007I	Unknown subfunction request
5901	BLNFNMSK	BLN9007I	Unknown subfunction request
5901	BLNDVKCP		Storage allocation failure
5902	BLNDVSIN	BLN9007I	Unable to access DVVT
5902	BLNMKINT	BLN9007I	Initialization error
5903	BLNMKDSP	BLN9007I	Initialization error
5904	BLNMKBMD		Storage allocation failure
5905	BLNMKMSK	BLN9007I	Initialization error
5921	BLNDVSIA		Storage allocation failure
5922	BLNDVSIA		Storage allocation failure
5923	BLNDVSIA		Storage allocation failure
5924	BLNDVSIA		Storage allocation failure
5925	BLNDVSIA		Storage allocation failure
6000	BLNDSYMP	BLN9007I	Unknown subfunction request
6004	BLNDSDSP	BLN9007I	Damage to DSCB suspected
6004	BLNDSPRT	BLN9007I	Damage to DSCB suspected
6004	BLNDSTRM	BLN9007I	Damage to DSCB suspected
6008	BLNDSGBF		Storage allocation failure
6012	BLNDSPRT		Permanent I/O error on print
6016	BLNDSPRT		BLX error
6020	BLNDSCEN		No symptom record in dump
7002	BLNSMINT	BLN9007I	Double initialization
7003	BLNSMTRM	BLN9007I	Double termination
7004	BLNSMBME	BLN9007I	Internal error
7018	BLNSMPHP		Storage allocation failure
7021	BLNSMBMA		Storage allocation failure
7023	BLNSMDSP		Initialization error
7024	BLNSMCMO		Storage allocation failure
7025	BLNSMBLE	BLN9007I	Internal error
7026	BLNSMBLF		Storage allocation failure
7062	BLNSMBLO		Storage allocation failure
7064	BLNSMPSO		Storage allocation failure
7066	BLNSMTXA	BLN9007I	LBD not found in symptom record
7068	BLNSMTXA		Storage allocation failure
7070	BLNSMTXA		Storage allocation failure
7072	BLNSMTXO		Storage allocation failure
7076	BLNSMHXA		Storage allocation failure
7078	BLNSMHXA		Storage allocation failure
7080	BLNSMHXO		Storage allocation failure
7082	BLNSMLKL		Storage allocation failure
7083	BLNSMLKL		Control block locator not found
7084	BLNSMPRT	BLN9007I	Initialization error
7086	BLNSMLKC		Storage allocation failure
7088	BLNSMLKO		Starting point not found
7089	BLNSMLKO		Storage allocation failure
7090	BLNSMLKD		Storage allocation failure
7092	BLNSMPSW		Storage allocation failure
7093	BLNSMGRE		Storage allocation failure
7094	BLNSMFRE		Storage allocation failure
7095	BLNSMOCL		Storage allocation failure
7099	BLNFNSMC	BLN9007I	Unknown subfunction request
7100	BLNSMFND	BLN9007I	Initialization error
7101	BLNSMFND		Data not available on find
7102	BLNSMFND		Invalid form of find command
7110	BLNSMSEL	BLN9007I	Initialization error
7122	BLNSMPCD	BLN9007I	Internal error freeing buffers
7401	BLNLBVER	BLN7013I	Unknown number of elements in array

Reason Code	Module Issuing	Message ID	Description
7402	BLNLBVER	BLN7013I	Array extension type = B, no chain extension
7403	BLNLBVER	BLN7013I	Invalid array extension type field
7405	BLNLBSIM		Storage allocation failure
7406	BLNLBARC		Storage allocation failure
7407	BLNLBARC		Storage allocation failure
7408	BLNLBARR		Storage allocation failure
7411	BLNLBVER	BLN7013I	Invalid end condition field
7412	BLNLBVER	BLN7013I	Invalid end condition length field
7413	BLNLBVER	BLN7013I	Invalid chain extension type field
7414	BLNLBVER	BLN7013I	Invalid chain extension address list pointer
7421	BLNLBVER	BLN7013I	Invalid hex extension length
7431	BLNLBVER	BLN7013I	Invalid text extension length
7441	BLNLBVER	BLN7013I	Keyfield length is zero
7442	BLNLBVER	BLN7013I	Invalid keyfield format field
7443	BLNLBVER	BLN7013I	Invalid keyfield type field
7451	BLNLBVER	BLN7013I	Number of entries in format descriptor is zero
7451	BLNLBVER	BLN7013I	Number of entries in linkage descriptor is zero
7452	BLNLBVER	BLN7013I	Length error in format descriptor
7452	BLNLBVER	BLN7013I	Length error in linkage descriptor
7453	BLNLBVER	BLN7013I	Invalid offset attribute type
7454	BLNLBVER	BLN7013I	Invalid sequential offset format
7455	BLNLBVER	BLN7013I	Invalid specific offset format
7460	BLNLBLAL		Storage allocation failure
7464	BLNLBCAC		Storage allocation failure
7465	BLNLBCAE		Storage allocation failure
7466	BLNLBCAI		Storage allocation failure
7467	BLNLBCAL		Storage allocation failure
7468	BLNLBCAL		Storage allocation failure
7469	BLNLBCHL		Storage allocation failure
7470	BLNLBCHW		Storage allocation failure
7471	BLNLBCHI		Storage allocation failure
7472	BLNLBCHE		Storage allocation failure
7473	BLNLBCHC		Storage allocation failure
7474	BLNLBCHR		Storage allocation failure
7475	BLNLBCAE		Storage allocation failure
7476	BLNLBCAI		Storage allocation failure
7477	BLNLBCAC		Storage allocation failure
7478	BLNLBCHE		Storage allocation failure
7479	BLNLBCHL		Storage allocation failure
7481	BLNLBCHI		Storage allocation failure
7483	BLNLBCHC		Storage allocation failure
7501	BLNFNARC	BLN9007I	Unknown subfunction request
7502	BLNARINT	BLN9007I	Duplicate initialization
7503	BLNARGRN		Storage allocation failure
7504	BLNARGRN		Storage allocation failure
7505	BLNARPRN		Storage allocation failure
7540	BLNARDSP		Initialization error
7541	BLNARDSP		No analysis routines list available
7544	BLNAREXA	BLN9007I	Unable to add ARCB to BLX
7550	BLNARDLO		Storage allocation failure
7560	BLNARPRT		Initialization not done
7567	BLNARFND		Initialization error
7568	BLNARFND		No analysis routines list available
7570	BLNAREXC		Initialization error
7581	BLNAREXG	BLN9002I	Unable to delete ARCB from BLX
7582	BLNAREXG	BLN7516I	Load of analysis routine failed
7583	BLNARXPL		Storage allocation failure

Reason Code	Module Issuing	Message ID	Description
7592	BLNAREXA		Storage allocation failure
9501	BLNCMXDE		BLX allocation - invalid unit
9502	BLNCMXDE		BLX allocation - extract macro error
9503	BLNCMXDE		BLX allocation - device not supported
9504	BLNCMXDE		BLX allocation - RECFM not supported
9505	BLNCMXDE		BLX allocation - blocked records not supported
9506	BLNCMXDE		BLX allocation - access mode not supported
9507	BLNCMXDE		BLX allocation - DLBL statement not supported
9508	BLNCMXDE		BLX allocation - open VTOC not supported
9509	BLNCMXDE		BLX allocation - close VTOC not supported
9510	BLNCMXDE		BLX allocation - extent VOLSER invalid
9511	BLNCMXDE		BLX allocation - extent logical unit invalid
9512	BLNCMXDE		BLX allocation - DDNAME not found
9519	BLNCMXDE		BLX free error
9521	BLNCMXDE		BLX open - permanent open error
9522	BLNCMXDE		BLX open - invalid FSEQ keyword
9523	BLNCMXDE		BLX open - data set already open
9531	BLNCMXDE		BLX close - permanent close error
9532	BLNCMXDE		BLX close - end of extent
9541	BLNCMXDE		BLX read - update not allowed
9542	BLNCMXDE		BLX read - wrong access type
9543	BLNCMXDE		BLX read - buffer length too small
9544	BLNCMXDE		BLX read - invalid relative record number
9545	BLNCMXDE		BLX read - VSAM position error
9546	BLNCMXDE		BLX read - invalid record length
9547	BLNCMXDE		BLX read - invalid key length
9548	BLNCMXDE		BLX read - VSAM MODCB verb failed
9549	BLNCMXDE		BLX read - undefined request
9591	BLNCMXDE		Unknown error condition

Info/Analysis Calling/Called Module Cross-Reference

This section lists, alphabetically, the names of Info/Analysis calling modules (modules that call other modules). Following each calling module name are the names of the modules that it calls. In addition, the following information on calling modules may be useful:

- Modules used to interface with other components:
  - BLNADIFR
  - BLNASLOC
- Modules used to control dumps in Dump Management file:
  - BLNUSADM
  - BLNUSDDM
- Modules used to execute an analysis routine:
  - BLNAXGST
  - BLNAXPDS
  - BLNAXDAC
  - BLNAXFMT
  - BLNAXSUP
  - BLNAXSAC
- Modules referenced by the BLNRAS service:
  - BLN6ADCB
  - BLN6ARCB
  - BLN6ASRC
  - BLN6BACB
  - BLN6CCB
  - BLN6DDCB
  - BLN6DMCB
  - BLN6DSCB
  - BLN6DVVT
  - BLN6MKCB
  - BLN6SCLC
  - BLN6SFCB
  - BLN6SMCB
- Modules referenced by the BLX service:
  - BLNCONS
  - BLNPRTS
  - BLNCOPT
  - BLNTRLX
  - BLNSTOP
- Data set allocation models referenced:
  - BLNMODEL
  - BLNLMODL
- Link book referenced
  - BLNLINK
- Module referenced by BLNBARDR:
  - BLNBAHLP

- Mnemonic used to denote when dump access is invoked:
  - DUMPACC
- Mnemonic used to denote when an analysis routine is invoked:
  - EXITRTN

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
BLNADIFI		BLNASADL	BLNERCMB	BLNASTRM	BLNASTMP
BLNADIFR	BLNADIFT	BLNASAXA	BLNERCMB	BLNASTRM	BLNERCMB
BLNADIFR	DUMPACC	BLNASAXC	BLNERCMB	BLNASUPD	BLNASLCB
BLNADIFT	DUMPACC	BLNASAXK	BLNERCMB	BLNASUPD	BLNASUP6
BLNADIFX	BLNADIFT	BLNASAXT	BLNERCMB	BLNASUP6	BLNASMP6
BLNARDLO		BLNASAXX	BLNASMP1	BLNASUP6	BLNASRSR
BLNARDSP	BLNARDLO	BLNASAXX	BLNERCMB	BLNAXCMB	BLNAXMSG
BLNARDSP	BLNCMALN	BLNASCME	BLNASGET	BLNAXDAC	BLNAXDA1
BLNAREXA	BLNDVFPE	BLNASCME	BLNASIST	BLNAXDA1	BLNADIFR
BLNAREXC	BLNAREXD	BLNASFMT	BLNASRFL	BLNAXFMT	BLNAXFM1
BLNAREXC	BLNAREXE	BLNASFMT	BLNASRSR	BLNAXFM1	BLNAXGS2
BLNAREXD	BLNCMPRS	BLNASFMT	BLNERCMB	BLNAXFM1	BLNAXMSG
BLNAREXE	BLNAREXA	BLNASFSY		BLNAXFM1	BLNCMFBC
BLNAREXE	BLNAREXF	BLNASGET	BLNERCMB	BLNAXFM1	BLNDDFMT
BLNAREXE	BLNAREXL	BLNASIST	BLNASGET	BLNAXFM1	BLNDDLSZ
BLNAREXE	BLNARXPL	BLNASLCB	BLNERCMB	BLNAXGST	BLNAXGS1
BLNAREXE	EXITRTN	BLNASLD1	BLNASFSY	BLNAXGS1	BLNAXGS2
BLNAREXF	BLNAREXG	BLNASLD1	BLNERCMB	BLNAXGS1	BLNAXGS3
BLNAREXF	BLNCMERM	BLNASLD6	BLNASFMT	BLNAXGS2	
BLNAREXG		BLNASLOC	BLNASLCB	BLNAXGS3	
BLNAREXL		BLNASLOC	BLNASLD1	BLNAXMSG	BLNAXGS2
BLNARFCS	BLNARDSP	BLNASLOC	BLNASLD6	BLNAXPDS	BLNAXPD1
BLNARFCS	BLNCMFBC	BLNASLOC	BLNASMAP	BLNAXPD1	BLNAXGS3
BLNARFML	BLNARFML	BLNASLOC	BLNERCMB	BLNAXPD1	BLNAXPD2
BLNARFND	BLNARDSP	BLNASLST	BLNASLCB	BLNAXPD1	BLNAXPD3
BLNARFND	BLNARFCS	BLNASLST	BLNASMAP	BLNAXPD2	BLNAXCMB
BLNARFND	BLNCMPRS	BLNASLST	BLNERCMB	BLNAXPD2	BLNCMPRT
BLNARGDN	BLNARFML	BLNASMAP	BLNASMP1	BLNAXPD3	BLNAXCMB
BLNARGDN	BLNARPRN	BLNASMAP	BLNASMP6	BLNAXPD3	BLNAXPD2
BLNARGDN	BLNCMXDE	BLNASMAP	BLNASRSR	BLNAXPD3	BLNAXPD4
BLNARGDN	BLNMSGS	BLNASMAP	BLNASTMP	BLNAXPD3	BLNAXPD7
BLNARGRN	BLNARFML	BLNASMAP	BLNERCMB	BLNAXPD4	BLNAXPD4
BLNARGRN	BLNARGDN	BLNASMD	BLNASADF	BLNAXPD4	BLNAXPD5
BLNARGRN	BLNMSGS	BLNASMD	BLNASADL	BLNAXPD4	BLNAXPD6
BLNARINT	BLNARGRN	BLNASMD	BLNASAXA	BLNAXPD5	BLNAXPD6
BLNARINT	BLNMSGS	BLNASMD	BLNASAXC	BLNAXPD6	BLNAXCMB
BLNARPDI		BLNASMD	BLNASAXK	BLNAXPD6	BLNCMPRT
BLNARPRN	BLNARPDI	BLNASMD	BLNASAXT	BLNAXPD7	
BLNARPRN	BLNMSGS	BLNASMD	BLNASAXX	BLNAXSAC	BLNAXSA1
BLNARPRT	BLNARDLO	BLNASMD	BLNASCME	BLNAXSA1	BLNASLOC
BLNARPRT	BLNCMALN	BLNASMD	BLNERCMB	BLNAXSA1	BLNAXGS2
BLNARPRT	BLNCMFBC	BLNASMDA	BLNASCME	BLNAXSRV	
BLNARPRT	BLNCMPRS	BLNASMP1	BLNERCMB	BLNAXSUA	BLNADIFR
BLNARPRT	BLNCMPRT	BLNASMP6	BLNASMD	BLNAXSUA	BLNASUPD
BLNARPRT	BLNSMPLA	BLNASMP6	BLNASMDA	BLNAXSUA	BLNAXSUA
BLNARTRM	BLNARFML	BLNASRFL		BLNAXSUP	BLNAXSU1
BLNARXPL	BLNASLOC	BLNASRSR	BLNADIFR	BLNAXSUR	BLNAXS3U
BLNARXPL	BLNAXGS2	BLNASRSR	BLNERCMB	BLNAXSUR	BLNAXS4U
BLNARXPL	BLNCMERM	BLNASTMP		BLNAXSUR	BLNAXS5U
BLNASADF	BLNERCMB	BLNASTRM	BLNASLCB	BLNAXSU1	BLNAXCMB

Calling Module	Called Module
BLNAXSU1	BLNAXSRV
BLNAXSU1	BLNAXSUA
BLNAXSU1	BLNAXSUR
BLNAXSU1	BLNAXS6U
BLNAXS3U	BLNAXGS2
BLNAXS3U	BLNAXGS3
BLNAXS3U	BLNAXS4U
BLNAXS4U	BLNAXGS2
BLNAXS4U	BLNAXGS3
BLNAXS5U	BLNAXGS2
BLNAXS5U	BLNAXGS3
BLNAXS6U	BLNASUPD
BLNAXS6U	BLNAXCMB
BLNAXS6U	BLNAXS6V
BLNAXS6V	BLNADIFR
BLNBABH1	BLNBABH2
BLNBABH1	BLNBADM1
BLNBABH1	BLNBADM2
BLNBABH1	BLNBADS1
BLNBABH1	BLNBADV1
BLNBABH1	BLNBAFLS
BLNBABH1	BLNBAPF1
BLNBABH1	BLNBAPN1
BLNBABH1	BLNBARDR
BLNBABH2	BLNBAMSG
BLNBABH2	BLNBARDR
BLNBADM1	BLNBABH2
BLNBADM1	BLNBADM2
BLNBADM1	BLNBARDR
BLNBADM1	BLNFNDMC
BLNBADM2	
BLNBADS1	BLNBABH2
BLNBADS1	BLNBARDR
BLNBADS1	BLNFNDSC
BLNBADV1	BLNBABH2
BLNBADV1	BLNBADV2
BLNBADV1	BLNBADV3
BLNBADV1	BLNBARDR
BLNBADV1	BLNFNDDC
BLNBADV2	BLNFNDVS
BLNBADV3	BLNFNARC
BLNBAFLS	BLNBAMSG
BLNBAFLS	BLNBARDR
BLNBAINT	BLNBABH1
BLNBAINT	BLNBAIPF
BLNBAINT	BLNBAMSG
BLNBAIPF	BLNBAVSE
BLNBAMSG	
BLNBAPF1	BLNBARDR
BLNBAPF1	BLNCMVOL
BLNBAPF1	BLNFNPFC
BLNBAPN1	BLNBAMSG
BLNBAPN1	BLNBARDR
BLNBAPN1	BLNCMFIL
BLNBAPN1	BLNCMVOL
BLNBAPN1	BLNFNPNC
BLNBAPRS	
BLNBARDR	BLNBAMSG
BLNBARDR	BLNBAPRS
BLNBARDR	BLNCMLST
BLNBAVSE	
BLNCMALN	
BLNCMBUT	
BLNCMCHD	BLNASLOC
BLNCMCHD	BLNCMPRI
BLNCMCHD	BLNCMTDS
BLNCMDEC	
BLNCMERM	BLNMSGS
BLNCMFBC	
BLNCMFIL	BLNCMDEC
BLNCMHEX	
BLNCMLST	
BLNCMMOB	
BLNCMMSP	
BLNCMPRI	
BLNCMPRO	BLNINPRT
BLNCMPRS	
BLNCMPRT	BLNCMPRO
BLNCMPRT	BLNINPRT
BLNCMTDS	
BLNCMTSZ	
BLNCMVOL	
BLNCMXBI	
BLNCMXCF	
BLNCMXDC	
BLNCMXDE	
BLNCMXEB	
BLNCMXPF	
BLNCONS	
BLNCOPT	BLNCONS
BLNCOPT	BLNPRTS
BLNDDBAS	BLNCMHEX
BLNDDBAS	BLNCMPRS
BLNDDBAS	BLNCMXPF
BLNDDBAS	BLNDDSYM
BLNDDBOD	BLNCMXPF
BLNDDBTE	BLNCMBUT
BLNDDBTE	BLNDDNLI
BLNDDCDB	BLNDDFMT
BLNDDCDB	BLNDDLSZ
BLNDDCDB	BLNDDNLI
BLNDDCDB	BLNDDREQ
BLNDDCDB	BLNDDVRF
BLNDDDPT	BLNCMPRT
BLNDDDPT	BLNDDCDB
BLNDDDPT	BLNDDFPP
BLNDDDPT	BLNDDRPT
BLNDDDSP	BLNDDCDB
BLNDDEBC	BLNCMXEB
BLNDDFDO	BLNCMXPF
BLNDDFDO	BLNDDFDZ
BLNDDFDO	BLNDDNLI
BLNDDFDO	BLNDVFFR
BLNDDFDZ	BLNCMMOB
BLNDDFDZ	BLNDDNLI
BLNDDFDZ	BLNDDREQ
BLNDDFFN	BLNCMFBC
BLNDDFMT	BLNASLOC
BLNDDFMT	BLNCMERM
BLNDDFMT	BLNCMFBC
BLNDDFMT	BLNDDBTE
BLNDDFMT	BLNDDFDO
BLNDDFMT	BLNDDHDO
BLNDDFMT	BLNLBVER
BLNDDFND	BLNCMFBC
BLNDDFND	BLNDDCDB
BLNDDFND	BLNDDFFN
BLNDDFND	BLNDDHFN
BLNDDFND	BLNDDLSZ
BLNDDFND	BLNDDVFN
BLNDDFPD	BLNDDCDB
BLNDDFPD	BLNDDFPP
BLNDDFPD	BLNFNSMC
BLNDDFPL	BLNDDFPD
BLNDDFPP	BLNCMFBC
BLNDDFPP	BLNCMMOB
BLNDDFPP	BLNCMPRT
BLNDDFPS	BLNDDFPL
BLNDDFPS	BLNFNSMC
BLNDDFPS	BLNSMPCD
BLNDDFPT	BLNDDFPS
BLNDDFPT	BLNDVKCB
BLNDDFPT	BLNDVRCB
BLNDDFPT	BLNSMPCD
BLNDDHBL	BLNCMXPF
BLNDDHBL	BLNDDBOD
BLNDDHBL	BLNDDEBC
BLNDDHBL	BLNDDMDT
BLNDDHDO	BLNCMMOB
BLNDDHDO	BLNCMXPF
BLNDDHDO	BLNDDHBL
BLNDDHDO	BLNDDNBU
BLNDDHDO	BLNDDNLI
BLNDDHFN	BLNCMHEX
BLNDDHFN	BLNCMXPF
BLNDDHFN	BLNDDCDB
BLNDDHFN	BLNDDQRY
BLNDDHFN	BLNDDREQ
BLNDDINT	BLNDDSYI
BLNDDLSZ	
BLNDDMDT	BLNCMMOB
BLNDDMDT	BLNCMXPF
BLNDDMDT	BLNDDQRY
BLNDDMDT	BLNDDREQ
BLNDDNBU	
BLNDDNLI	BLNDDNBU
BLNDDPOP	BLNADIFR
BLNDDPOP	BLNCMHEX
BLNDDPOP	BLNCMPRS
BLNDDPRT	BLNDDDPT
BLNDDPRT	BLNDDFPT

Calling Module	Called Module
BLNDDPRT	BLNDDPOP
BLNDDPRT	BLNDDRPT
BLNDDQAL	BLNCMHEX
BLNDDQAL	BLNCMPRS
BLNDDQAL	BLNCMXPF
BLNDDQRY	BLNADIFR
BLNDDQRY	BLNCMERM
BLNDDRDF	BLNASLOC
BLNDDRDF	BLNCMERM
BLNDDRDF	BLNCMXPF
BLNDDRDF	BLNLBVZA
BLNDDREQ	BLNADIFR
BLNDDREQ	BLNCMERM
BLNDDRPT	BLNCMFBC
BLNDDRPT	BLNCMMOB
BLNDDRPT	BLNCMPRT
BLNDDRPT	BLNCMXPF
BLNDDRPT	BLNDDFMT
BLNDDRPT	BLNDDLSZ
BLNDDRPT	BLNDDQRY
BLNDDRPT	BLNDDREQ
BLNDDSYI	BLNASLOC
BLNDDSYI	BLNCMERM
BLNDDSYI	BLNLBVER
BLNDDSYM	
BLNDDTRM	
BLNDDVFA	BLNCMHEX
BLNDDVFN	BLNCMHEX
BLNDDVFN	BLNCMPRS
BLNDDVFN	BLNDDVFA
BLNDDVRF	BLNCMXPF
BLNDDVRF	BLNDDBAS
BLNDDVRF	BLNDDQAL
BLNDDVRF	BLNDDRDF
BLNDMADU	BLNDMAD1
BLNDMADU	BLNDMCHK
BLNDMADU	BLNDMCNT
BLNDMADU	BLNDMRDF
BLNDMADU	BLNDMSRT
BLNDMADU	BLNDMWDF
BLNDMAD1	BLNASLOC
BLNDMAD1	BLNCMERM
BLNDMAD1	BLNCMTDS
BLNDMCHK	BLNDMCK1
BLNDMCHK	BLNDMCK2
BLNDMCK1	BLNDMCK2
BLNDMCK1	BLNDMCK3
BLNDMCK2	BLNADIFR
BLNDMCK2	BLNCMERM
BLNDMCK2	BLNDMAD1
BLNDMCK2	BLNDMWDF
BLNDMCK3	
BLNDMCNT	
BLNDMDDL	BLNDMCHK
BLNDMDDL	BLNDMRDF
BLNDMDDL	BLNDMSRT
BLNDMDEL	BLNADIFR
BLNDMDEL	BLNASTRM
BLNDMDEL	BLNCMERM
BLNDMDEL	BLNDMCHK
BLNDMDEL	BLNDMCNT
BLNDMDEL	BLNDMRDF
BLNDMDEL	BLNDMUDI
BLNDMDEL	BLNDMWDF
BLNDMDMU	BLNDMRDF
BLNDMDMU	BLNDMTRM
BLNDMDMU	BLNDMWDF
BLNDMGNT	BLNDMADU
BLNDMGNT	BLNDMDDL
BLNDMGNT	BLNDMDEL
BLNDMGNT	BLNDMDMU
BLNDMGNT	BLNDMPDL
BLNDMGNT	BLNDMTRM
BLNDMGNT	BLNDMUAI
BLNDMGNT	BLNOFFLD
BLNDMGNT	BLNONLOD
BLNDMPDL	BLNCMFBC
BLNDMPDL	BLNCMPRT
BLNDMPDL	BLNDMCHK
BLNDMPDL	BLNDMRDF
BLNDMPDL	BLNDMSRT
BLNDMRDF	BLNDMRD1
BLNDMRD1	
BLNDMSRT	
BLNDMTRM	
BLNDMUAB	BLNDMUAD
BLNDMUAD	BLNDMADU
BLNDMUAI	BLNDMRDF
BLNDMUAI	BLNUSADM
BLNDMUDI	BLNASLOC
BLNDMUDI	BLNCMERM
BLNDMUDI	BLNUSDDM
BLNDMWDF	
BLNDSBCL	BLNDSGBF
BLNDSBSO	BLNDSBCL
BLNDSBSO	BLNDSGBF
BLNDSCEN	BLNASLOC
BLNDSCEN	BLNCMERM
BLNDSCEN	BLNDSFED
BLNDSCEN	BLNDSGBF
BLNDSCEN	BLNMSGS
BLNDSCNS	BLNASLOC
BLNDSCNS	BLNCMERM
BLNDSCNS	BLNDSBSO
BLNDSCNS	BLNDSGBF
BLNDSCOL	BLNDSCEN
BLNDSCOL	BLNDSCNS
BLNDSCOL	BLNDSCRQ
BLNDSCOL	BLNDSCST
BLNDSCRQ	BLNASLOC
BLNDSCRQ	BLNCMERM
BLNDSCRQ	BLNDSBSO
BLNDSCRQ	BLNDSGBF
BLNDSCST	BLNASLOC
BLNDSCST	BLNCMERM
BLNDSCST	BLNDSBSO
BLNDSCST	BLNDSGBF
BLNDSDSP	BLNCMFBC
BLNDSDSP	BLNDSCOL
BLNDSDSP	BLNMSGS
BLNDSFED	BLNCMTDS
BLNDSFED	BLNDSGBF
BLNDSGBF	BLNMSGS
BLNDSPRT	BLNCMFBC
BLNDSPRT	BLNCMPRT
BLNDSPRT	BLNDSCOL
BLNDSPRT	BLNDSGBF
BLNDSPRT	BLNMSGS
BLNDSTRM	BLNCMFBC
BLNDSTRM	BLNMSGS
BLNDSYMP	BLNDSDSP
BLNDSYMP	BLNDSPRT
BLNDSYMP	BLNDSTRM
BLNDSYMP	BLNMSGS
BLNDVFFR	BLNCMXBI
BLNDVFFR	BLNCMXDC
BLNDVFFR	BLNCMXEB
BLNDVFFR	BLNCMXPF
BLNDVFPE	
BLNDVKCB	BLNFNSMC
BLNDVRCB	BLNFNSMC
BLNDVRCB	BLNSMFES
BLNDVSIA	
BLNDVSIN	BLNDVFPE
BLNDVSIN	BLNDVSIA
BLNDVSIN	BLNDVSTM
BLNDVSIN	BLNFNDDO
BLNDVSTM	
BLNERCMB	
BLNFNARC	BLNARDSP
BLNFNARC	BLNAREXC
BLNFNARC	BLNARFND
BLNFNARC	BLNARINT
BLNFNARC	BLNARPRT
BLNFNARC	BLNARTRM
BLNFNARC	BLNMSGS
BLNFNDC2	
BLNFNDC3	
BLNFNDDC	BLNDDDSP
BLNFNDDC	BLNDDFND
BLNFNDDC	BLNDDINT
BLNFNDDC	BLNDDPRT
BLNFNDDC	BLNDDTRM
BLNFNDDO	
BLNFNDMC	BLNCMCHD
BLNFNDMC	BLNDMGNT
BLNFNDMC	BLNFNDC2
BLNFNDSC	BLNDSYMP
BLNFNDSC	BLNFNDC3
BLNFNDVS	BLNDVSIN
BLNFNDVS	BLNDVSTM

Calling Module	Called Module
BLNFNMSK	BLNMKDSP
BLNFNMSK	BLNMKINT
BLNFNMSK	BLNMKMSK
BLNFNMSK	BLNMKTRM
BLNFNPC1	
BLNFNPC2	
BLNFNPFC	BLNDMGNT
BLNFNPFC	BLNFNPC2
BLNFNPNC	BLNDMGNT
BLNFNPNC	BLNFNPC1
BLNFNSMC	BLNSMDSP
BLNFNSMC	BLNSMFND
BLNFNSMC	BLNSMINT
BLNFNSMC	BLNSMPRT
BLNFNSMC	BLNSMSEL
BLNFNSMC	BLNSMTRM
BLNINIT	BLNMAIN
BLNINPRT	CDP
BLNINPRT	CTA
BLNITSPF	BLNMAIN
BLNLBARB	BLNLBARC
BLNLBARB	BLNLBCHN
BLNLBARB	BLNLBSIM
BLNLBARC	BLNLBBCI
BLNLBARR	BLNLBCHR
BLNLBARR	BLNLBCNT
BLNLBARY	BLNLBARB
BLNLBARY	BLNLBARC
BLNLBBCI	BLNADIFR
BLNLBBCI	BLNLBCEF
BLNLBBCI	BLNLBCMA
BLNLBBCI	BLNLBRCB
BLNLBBCI	BLNSMLKC
BLNLBBLR	BLNLBARR
BLNLBBLR	BLNLBARY
BLNLBBLR	BLNLBCHN
BLNLBBLR	BLNLBCHR
BLNLBBLR	BLNLBSIM
BLNLBCAC	BLNLBBCI
BLNLBCAC	BLNLBCNT
BLNLBCAC	BLNLBLAL
BLNLBCAE	BLNLBBCI
BLNLBCAE	BLNLBCEF
BLNLBCAE	BLNLBLAL
BLNLBCAI	BLNLBBCI
BLNLBCAI	BLNLBLAL
BLNLBCAL	BLNLBBCI
BLNLBCAL	BLNLBCEF
BLNLBCAL	BLNLBLAL
BLNLBCEF	
BLNLBCHC	BLNLBBCI
BLNLBCHC	BLNLBCLN
BLNLBCHC	BLNLBCNT
BLNLBCHE	BLNLBBCI
BLNLBCHE	BLNLBCEF
BLNLBCHE	BLNLBCLN
BLNLBCHI	BLNLBBCI
BLNLBCHI	BLNLBCLN
BLNLBCHL	BLNLBBCI
BLNLBCHL	BLNLBCEF
BLNLBCHL	BLNLBCLN
BLNLBCHN	BLNLBCAC
BLNLBCHN	BLNLBCAE
BLNLBCHN	BLNLBCAI
BLNLBCHN	BLNLBCAL
BLNLBCHN	BLNLBCHC
BLNLBCHN	BLNLBCHE
BLNLBCHN	BLNLBCHI
BLNLBCHN	BLNLBCHL
BLNLBCHN	BLNLBCHW
BLNLBCHR	BLNADIFR
BLNLBCHR	BLNCMERM
BLNLBCHW	BLNLBBCI
BLNLBCHW	BLNLBCLN
BLNLBCLN	BLNADIFR
BLNLBCLN	BLNCMERM
BLNLBCMA	
BLNLBCNT	BLNADIFR
BLNLBCNT	BLNCMERM
BLNLBLAL	BLNADIFR
BLNLBLAL	BLNCMERM
BLNLBRCB	BLNADIFR
BLNLBRCB	BLNCMERM
BLNLBSIM	BLNLBBCI
BLNLBVER	
BLNLBVZA	
BLNMACBC	BLNMACBG
BLNMACBC	BLNMAPDF
BLNMACBG	
BLNMAIN	BLNADIFI
BLNMAIN	BLNBAINT
BLNMAIN	BLNMACBC
BLNMAIN	BLNMAOCM
BLNMAIN	BLNRAS
BLNMAIN	BLNSFINT
BLNMAIN	BLNTERM
BLNMAOCM	
BLNMAPDF	
BLNMSGS	
BLNOFFLD	BLNADIFR
BLNOFFLD	BLNCMERM
BLNOFFLD	BLNDMCHK
BLNOFFLD	BLNDMRDF
BLNOFFLD	BLNOFOUT
BLNOFFLD	BLNOFUDM
BLNOFFLD	BLNSUBMT
BLNOFOUT	BLNADIFR
BLNOFOUT	BLNCMERM
BLNOFUDM	BLNDMCHK
BLNOFUDM	BLNDMRDF
BLNOFUDM	BLNDMWDF
BLNONDDT	BLNONVSE
BLNONEND	BLNADIFR
BLNONEND	BLNCMERM
BLNONEND	BLNDMAD1
BLNONEND	BLNDMCHK
BLNONEND	BLNDMRDF
BLNONEND	BLNDMSRT
BLNONEND	BLNDMWDF
BLNONINT	BLNDMCHK
BLNONINT	BLNDMRDF
BLNONINT	BLNSUBMT
BLNONLOD	BLNONDDT
BLNONLOD	BLNONEND
BLNONLOD	BLNONINT
BLNONLOD	BLNONMUD
BLNONLOD	BLNONSCM
BLNONLOD	BLNONTRW
BLNONMUD	
BLNONSCM	BLNONSC1
BLNONSC1	BLNADIFR
BLNONSC1	BLNCMERM
BLNONTRW	BLNADIFR
BLNONTRW	BLNCMERM
BLNONVSE	
BLNPRTS	BLNCMLST
BLNPRTS	BLNCMMSP
BLNRAADB	BLNRALBD
BLNRAADB	BLNRAS6S
BLNRAARB	BLNRALBD
BLNRAARB	BLNRAS6S
BLNRAASB	BLNRALBD
BLNRAASB	BLNRAS6S
BLNRABAB	BLNRALBD
BLNRABAB	BLNRAS6S
BLNRACCB	BLNRALBD
BLNRACCB	BLNRAS6S
BLNRADDB	BLNRALBD
BLNRADDB	BLNRAS6S
BLNRADMB	BLNRALBD
BLNRADMB	BLNRAS6S
BLNRADSB	BLNRALBD
BLNRADSB	BLNRAS6S
BLNRADVT	BLNRALBD
BLNRADVT	BLNRAS6S
BLNRAEIN	
BLNRALBD	
BLNRAMKB	BLNRALBD
BLNRAMKB	BLNRAS6S
BLNRAS	BLNRASEC
BLNRASCB	BLNRALBD
BLNRASCB	BLNRAS6S
BLNRASEC	BLNRAEIN
BLNRASEC	BLNRAS4B
BLNRASEC	BLNRAS5B
BLNRASEC	BLNRAS6B
BLNRASFB	BLNRALBD
BLNRASFB	BLNRAS6S
BLNRASMB	BLNRALBD
BLNRASMB	BLNRAS6S
BLNRAS4B	

Calling Module	Called Module
BLNRAS5B	
BLNRAS6B	BLNRAADB
BLNRAS6B	BLNRAARB
BLNRAS6B	BLNRAASB
BLNRAS6B	BLNRABAB
BLNRAS6B	BLNRACCB
BLNRAS6B	BLNRADDB
BLNRAS6B	BLNRADMB
BLNRAS6B	BLNRADSB
BLNRAS6B	BLNRADVT
BLNRAS6B	BLNRAMKB
BLNRAS6B	BLNRASCB
BLNRAS6B	BLNRASFB
BLNRAS6B	BLNRASMB
BLNRAS6S	
BLNSMBHX	BLNCMALN
BLNSMBHX	BLNSMHXA
BLNSMBHX	BLNSMHXO
BLNSMBLB	BLNLBBLR
BLNSMBLB	BLNLBVER
BLNSMBLE	BLNASLOC
BLNSMBLE	BLNCMERM
BLNSMBLE	BLNSMBLF
BLNSMBLF	BLNLBVZA
BLNSMBLF	BLNSMBLH
BLNSMBLH	BLNSMBLB
BLNSMBLK	BLNCMALN
BLNSMBLK	BLNSMLKL
BLNSMBLK	BLNSMLKO
BLNSMBLL	BLNCMALN
BLNSMBLL	BLNSMBLE
BLNSMBLL	BLNSMBLO
BLNSMBLN	
BLNSMBLO	BLNSMBOL
BLNSMBLO	BLNSMLSL
BLNSMBMA	BLNSMBME
BLNSMBME	BLNSMBLN
BLNSMBME	BLNSMBMN
BLNSMBML	BLNASLST
BLNSMBML	BLNCMERM
BLNSMBML	BLNSMBMA
BLNSMBMN	
BLNSMBOK	BLNADIFR
BLNSMBOK	BLNDVFFR
BLNSMBOL	BLNCMXPF
BLNSMBOL	BLNSMBOK
BLNSMBPE	BLNCMALN
BLNSMBPE	BLNSMPSO
BLNSMBSI	BLNSMBHX
BLNSMBSI	BLNSMBLK
BLNSMBSI	BLNSMBPE
BLNSMBSI	BLNSMBTX
BLNSMBTX	BLNCMALN
BLNSMBTX	BLNSMTXA
BLNSMBTX	BLNSMTXO
BLNSMCML	BLNCMALN
BLNSMCML	BLNSMCMO
BLNSMCML	BLNSMFES
BLNSMCMO	
BLNSMDSP	BLNSMCML
BLNSMDSP	BLNSMSDC
BLNSMFCD	BLNSMFC1
BLNSMFC1	BLNSMFC1
BLNSMFES	BLNSMFCD
BLNSMFES	BLNSMFLD
BLNSMFES	BLNSMFTD
BLNSMFES	BLNSMFXD
BLNSMFLD	
BLNSMFND	BLNCMPRS
BLNSMFND	BLNSMDSP
BLNSMFND	BLNSMFN1
BLNSMFN1	BLNCMFBC
BLNSMFN1	BLNSMDSP
BLNSMFRE	BLNCMXPF
BLNSMFTD	
BLNSMFXD	
BLNSMGRE	BLNCMXPF
BLNSMHXA	BLNASLOC
BLNSMHXA	BLNCMERM
BLNSMHXA	BLNLBVER
BLNSMHXA	BLNMSGS
BLNSMHXA	BLNSMFRE
BLNSMHXA	BLNSMGRE
BLNSMHXA	BLNSMOCL
BLNSMHXA	BLNSMPSW
BLNSMHXO	
BLNSMINT	BLNSMBML
BLNSMLKB	BLNSMLKC
BLNSMLKB	BLNSMLKD
BLNSMLKB	BLNSMLKN
BLNSMLKC	BLNSMBLB
BLNSMLKD	
BLNSMLKG	BLNASLOC
BLNSMLKG	BLNCMERM
BLNSMLKG	BLNLBVZA
BLNSMLKI	BLNSMLKG
BLNSMLKL	BLNASLOC
BLNSMLKL	BLNCMERM
BLNSMLKL	BLNLBVER
BLNSMLKL	BLNSMLKB
BLNSMLKL	BLNSMLKI
BLNSMLKN	BLNASLOC
BLNSMLKN	BLNCMERM
BLNSMLKN	BLNSMLKN
BLNSMLKO	BLNSMLKP
BLNSMLKP	BLNSMBOL
BLNSMLKP	BLNSMLKP
BLNSMLSL	
BLNSMLSP	BLNSMLSP
BLNSMOCL	BLNCMXPF
BLNSMOCL	BLNDDBOD
BLNSMOCL	BLNDDEBC
BLNSMOCL	BLNDDLSZ
BLNSMPCD	BLNCMFBC
BLNSMPCD	BLNCMPRT
BLNSMPCD	BLNSMDSP
BLNSMPCD	BLNSMPHP
BLNSMPCD	BLNSMPLA
BLNSMPHP	
BLNSMPLA	
BLNSMPRT	BLNCMPRS
BLNSMPRT	BLNSMPCD
BLNSMPSO	
BLNSMPSW	BLNCMXPF
BLNSMSDC	BLNSMBLL
BLNSMSDC	BLNSMBSI
BLNSMSDC	BLNSMFES
BLNSMSEL	BLNSMSEM
BLNSMSEL	BLNSMSNI
BLNSMSEM	
BLNSMSNC	BLNCMXPF
BLNSMSNC	BLNSMLSP
BLNSMSNI	BLNSMSNC
BLNSMSNI	BLNSMSNL
BLNSMSNL	BLNCMXPF
BLNSMSNL	BLNSMLSL
BLNSMTRM	BLNSMFES
BLNSMTXA	BLNASLOC
BLNSMTXA	BLNCMBUT
BLNSMTXA	BLNCMERM
BLNSMTXA	BLNLBVER
BLNSMTXA	BLNMSGS
BLNSMTXO	
BLNSUBMT	
BLNTERM	BLNADIFX
BLNTERM	BLNASTRM
BLNTERM	BLNCMFBC
BLNUSADM	
BLNUSDDM	
CDP	
CTA	
DUMPACC	
EXITRTN	
ISPLINK	

Info/Analysis Called/Calling Module Cross-Reference

This section lists, alphabetically, the names of Info/Analysis modules that are called by other modules. Following each called module name are the names of the modules that call it.
(Refer to “Info/Analysis Calling/Called Module Cross-Reference” on page 270 for additional information.)

Called Module	Calling Module	Called Module	Calling Module	Called Module	Calling Module
BLNADIFI	BLNMAIN	BLNARTRM	BLNFNARC	BLNASRSR	BLNASFMT
BLNADIFR	BLNASRSR	BLNARXPL	BLNAREXE	BLNASRSR	BLNASMAP
BLNADIFR	BLNAXDA1	BLNASADF	BLNASMD	BLNASRSR	BLNASUP6
BLNADIFR	BLNAXSUA	BLNASADL	BLNASMD	BLNASTMP	BLNASMAP
BLNADIFR	BLNAXS6V	BLNASAXA	BLNASMD	BLNASTMP	BLNASTRM
BLNADIFR	BLNDDPOP	BLNASAXC	BLNASMD	BLNASTRM	BLNDMDEL
BLNADIFR	BLNDDQRY	BLNASAXK	BLNASMD	BLNASTRM	BLNTERM
BLNADIFR	BLNDDREQ	BLNASAXT	BLNASMD	BLNASUPD	BLNAXSUA
BLNADIFR	BLNDMCK2	BLNASAXX	BLNASMD	BLNASUPD	BLNAXS6U
BLNADIFR	BLNDMDEL	BLNASCME	BLNASMD	BLNASUP6	BLNASUPD
BLNADIFR	BLNLBBCI	BLNASCME	BLNASMDA	BLNAXCMB	BLNAXPD2
BLNADIFR	BLNLBCHR	BLNASFMT	BLNASLD6	BLNAXCMB	BLNAXPD3
BLNADIFR	BLNLBCLN	BLNASFSY	BLNASLD1	BLNAXCMB	BLNAXPD6
BLNADIFR	BLNLBCNT	BLNASGET	BLNASCME	BLNAXCMB	BLNAXSU1
BLNADIFR	BLNLBLAL	BLNASGET	BLNASIST	BLNAXCMB	BLNAXS6U
BLNADIFR	BLNLBRCB	BLNASIST	BLNASCME	BLNAXDAC	
BLNADIFR	BLNMKMSK	BLNASLCB	BLNASLOC	BLNAXDA1	BLNAXDAC
BLNADIFR	BLNOFFLD	BLNASLCB	BLNASLST	BLNAXFMT	
BLNADIFR	BLNOFOUT	BLNASLCB	BLNASTRM	BLNAXFM1	BLNAXFMT
BLNADIFR	BLNONEND	BLNASLCB	BLNASUPD	BLNAXGST	
BLNADIFR	BLNONSC1	BLNASLD1	BLNASLOC	BLNAXGS1	BLNAXGST
BLNADIFR	BLNONTRW	BLNASLD6	BLNASLOC	BLNAXGS2	BLNARXPL
BLNADIFR	BLNSMBOK	BLNASLOC	BLNARXPL	BLNAXGS2	BLNAXFM1
BLNADIFT	BLNADIFR	BLNASLOC	BLNAXSA1	BLNAXGS2	BLNAXGS1
BLNADIFT	BLNADIFX	BLNASLOC	BLNCMCHD	BLNAXGS2	BLNAXMSG
BLNADIFX	BLNTERM	BLNASLOC	BLNDDFMT	BLNAXGS2	BLNAXSA1
BLNARDLO	BLNARDSP	BLNASLOC	BLNDDRDF	BLNAXGS2	BLNAXS3U
BLNARDLO	BLNARPRT	BLNASLOC	BLNDDSYI	BLNAXGS2	BLNAXS4U
BLNARDSP	BLNARFCS	BLNASLOC	BLNDMAD1	BLNAXGS2	BLNAXS5U
BLNARDSP	BLNARFND	BLNASLOC	BLNDMUDI	BLNAXGS3	BLNAXGS1
BLNARDSP	BLNFNARC	BLNASLOC	BLNDSCEN	BLNAXGS3	BLNAXPD1
BLNAREXA	BLNAREXE	BLNASLOC	BLNDSCNS	BLNAXGS3	BLNAXS3U
BLNAREXC	BLNFNARC	BLNASLOC	BLNDSCRQ	BLNAXGS3	BLNAXS4U
BLNAREXD	BLNAREXC	BLNASLOC	BLNDSCST	BLNAXGS3	BLNAXS5U
BLNAREXE	BLNAREXC	BLNASLOC	BLNSMBLE	BLNAXMSG	BLNAXCMB
BLNAREXF	BLNAREXE	BLNASLOC	BLNSMHXA	BLNAXMSG	BLNAXFM1
BLNAREXG	BLNAREXF	BLNASLOC	BLNSMLKG	BLNAXPDS	
BLNAREXL	BLNAREXE	BLNASLOC	BLNSMLKL	BLNAXPD1	BLNAXPDS
BLNARFCS	BLNARFND	BLNASLOC	BLNSMLKN	BLNAXPD2	BLNAXPD1
BLNARFML	BLNARFML	BLNASLOC	BLNSMTXA	BLNAXPD2	BLNAXPD3
BLNARFML	BLNARGDN	BLNASLST	BLNSMBML	BLNAXPD3	BLNAXPD1
BLNARFML	BLNARGRN	BLNASMAP	BLNASLOC	BLNAXPD4	BLNAXPD3
BLNARFML	BLNARTRM	BLNASMAP	BLNASLST	BLNAXPD4	BLNAXPD4
BLNARFND	BLNFNARC	BLNASMD	BLNASMP6	BLNAXPD5	BLNAXPD4
BLNARGDN	BLNARGRN	BLNASMDA	BLNASMP6	BLNAXPD6	BLNAXPD4
BLNARGRN	BLNARINT	BLNASMP1	BLNASAXX	BLNAXPD6	BLNAXPD5
BLNARINT	BLNFNARC	BLNASMP1	BLNASMAP	BLNAXPD7	BLNAXPD3
BLNARPDI	BLNARPRN	BLNASMP6	BLNASMAP	BLNAXSAC	
BLNARPRN	BLNARGDN	BLNASMP6	BLNASUP6	BLNAXSA1	BLNAXSAC
BLNARPRT	BLNFNARC	BLNASRFL	BLNASFMT	BLNAXSRV	BLNAXSU1

Called Module	Calling Module
BLNAXSUA	BLNAXSUA
BLNAXSUA	BLNAXSU1
BLNAXSUP	
BLNAXSUR	BLNAXSU1
BLNAXSU1	BLNAXSUP
BLNAXS3U	BLNAXSUR
BLNAXS4U	BLNAXSUR
BLNAXS4U	BLNAXS3U
BLNAXS5U	BLNAXSUR
BLNAXS6U	BLNAXSU1
BLNAXS6V	BLNAXS6U
BLNBABH1	BLNBAINT
BLNBABH2	BLNBABH1
BLNBABH2	BLNBADM1
BLNBABH2	BLNBADS1
BLNBABH2	BLNBADV1
BLNBADM1	BLNBABH1
BLNBADM2	BLNBABH1
BLNBADM2	BLNBADM1
BLNBADS1	BLNBABH1
BLNBADV1	BLNBABH1
BLNBADV2	BLNBADV1
BLNBADV3	BLNBADV1
BLNBAFLS	BLNBABH1
BLNBAINT	BLNMAIN
BLNBAIPF	BLNBAINT
BLNBAMSG	BLNBABH2
BLNBAMSG	BLNBAFLS
BLNBAMSG	BLNBAINT
BLNBAMSG	BLNBAPN1
BLNBAMSG	BLNBARDR
BLNBAPF1	BLNBABH1
BLNBAPN1	BLNBABH1
BLNBAPRS	BLNBARDR
BLNBARDR	BLNBABH1
BLNBARDR	BLNBABH2
BLNBARDR	BLNBADM1
BLNBARDR	BLNBADS1
BLNBARDR	BLNBADV1
BLNBARDR	BLNBAFLS
BLNBARDR	BLNBAPF1
BLNBARDR	BLNBAPN1
BLNBAVSE	BLNBAIPF
BLNCMALN	BLNARDSP
BLNCMALN	BLNARPRT
BLNCMALN	BLNSMBHX
BLNCMALN	BLNSMBLK
BLNCMALN	BLNSMBLL
BLNCMALN	BLNSMBPE
BLNCMALN	BLNSMBTX
BLNCMALN	BLNSMCML
BLNCMBUT	BLNDDBTE
BLNCMBUT	BLNSFBUM
BLNCMBUT	BLNSFCMQ
BLNCMBUT	BLNSMTXA
BLNCMCHD	BLNFNDMC
BLNCMDEC	BLNCMFIL
BLNCMERM	BLNAREXF
BLNCMERM	BLNARXPL
BLNCMERM	BLNDDFMT
BLNCMERM	BLNDDQRY
BLNCMERM	BLNDDRDF
BLNCMERM	BLNDDREQ
BLNCMERM	BLNDDSYI
BLNCMERM	BLNDMAD1
BLNCMERM	BLNDMCK2
BLNCMERM	BLNDMDEL
BLNCMERM	BLNDMUDI
BLNCMERM	BLNDSCEN
BLNCMERM	BLNDSCNS
BLNCMERM	BLNDSCRQ
BLNCMERM	BLNDSCST
BLNCMERM	BLNLBCHR
BLNCMERM	BLNLBCLN
BLNCMERM	BLNLBCNT
BLNCMERM	BLNLBLAL
BLNCMERM	BLNLBRCB
BLNCMERM	BLNMKMSK
BLNCMERM	BLNOFFLD
BLNCMERM	BLNOFOUT
BLNCMERM	BLNONEND
BLNCMERM	BLNONSC1
BLNCMERM	BLNONTRW
BLNCMERM	BLNSMBLE
BLNCMERM	BLNSMBML
BLNCMERM	BLNSMHXA
BLNCMERM	BLNSMLKG
BLNCMERM	BLNSMLKL
BLNCMERM	BLNSMLKN
BLNCMERM	BLNSMTXA
BLNCMFBC	BLNARFCS
BLNCMFBC	BLNARPRT
BLNCMFBC	BLNAXFM1
BLNCMFBC	BLNDDFFN
BLNCMFBC	BLNDDFMT
BLNCMFBC	BLNDDFND
BLNCMFBC	BLNDDFPP
BLNCMFBC	BLNDDRPT
BLNCMFBC	BLNDMPDL
BLNCMFBC	BLNDSDSP
BLNCMFBC	BLNDSPRT
BLNCMFBC	BLNDSTRM
BLNCMFBC	BLNSFBST
BLNCMFBC	BLNSFMSK
BLNCMFBC	BLNSMFN1
BLNCMFBC	BLNSMPCD
BLNCMFBC	BLNTERM
BLNCMFIL	BLNBAPN1
BLNCMHEX	BLNDDBAS
BLNCMHEX	BLNDDHFN
BLNCMHEX	BLNDDPOP
BLNCMHEX	BLNDDQAL
BLNCMHEX	BLNDDVFA
BLNCMHEX	BLNDDVFN
BLNCMHEX	BLNMKVMR
BLNCMHEX	BLNSFDD3
BLNCMLST	BLNBARDR
BLNCMLST	BLNPRTS
BLNCMMOB	BLNDDFDZ
BLNCMMOB	BLNDDFPP
BLNCMMOB	BLNDDHDO
BLNCMMOB	BLNDDMDT
BLNCMMOB	BLNDDRPT
BLNCMMOB	BLNMKBMD
BLNCMMSP	BLNPRTS
BLNCMPRI	BLNCMCHD
BLNCMPRO	BLNCMPRT
BLNCMPRS	BLNAREXD
BLNCMPRS	BLNARFND
BLNCMPRS	BLNARPRT
BLNCMPRS	BLNDDBAS
BLNCMPRS	BLNDDPOP
BLNCMPRS	BLNDDQAL
BLNCMPRS	BLNDDVFN
BLNCMPRS	BLNMKVMR
BLNCMPRS	BLNSFMDA
BLNCMPRS	BLNSFMDB
BLNCMPRS	BLNSMFND
BLNCMPRS	BLNSMPRT
BLNCMPRT	BLNARPRT
BLNCMPRT	BLNAXPD2
BLNCMPRT	BLNAXPD6
BLNCMPRT	BLNDDDPT
BLNCMPRT	BLNDDFPP
BLNCMPRT	BLNDDRPT
BLNCMPRT	BLNDMPDL
BLNCMPRT	BLNDSPRT
BLNCMPRT	BLNSMPCD
BLNCMTDS	BLNCMCHD
BLNCMTDS	BLNDMAD1
BLNCMTDS	BLNDSFED
BLNCMVOL	BLNBAPF1
BLNCMVOL	BLNBAPN1
BLNCMXBI	BLNDVFFR
BLNCMXDC	BLNDVFFR
BLNCMXDE	BLNARGDN
BLNCMXEB	BLNDDEBC
BLNCMXEB	BLNDVFFR
BLNCMXPF	BLNDDBAS
BLNCMXPF	BLNDDBOD
BLNCMXPF	BLNDDFDO
BLNCMXPF	BLNDDHBL
BLNCMXPF	BLNDDHDO
BLNCMXPF	BLNDDHFN
BLNCMXPF	BLNDDMDT
BLNCMXPF	BLNDDQAL
BLNCMXPF	BLNDDRDF
BLNCMXPF	BLNDDRPT
BLNCMXPF	BLNDDVRF
BLNCMXPF	BLNDVFFR
BLNCMXPF	BLNMKVMR

Called Module	Calling Module
BLNCMXPF	BLNSFDD3
BLNCMXPF	BLNSMBOL
BLNCMXPF	BLNSMFRE
BLNCMXPF	BLNSMGRE
BLNCMXPF	BLNSMOCL
BLNCMXPF	BLNSMPSW
BLNCMXPF	BLNSMSNC
BLNCMXPF	BLNSMSNL
BLNCONS	BLNCOPT
BLNCOPT	
BLNDDBAS	BLNDDVRF
BLNDDBOD	BLNDDHBL
BLNDDBOD	BLNSMOCL
BLNDDBTE	BLNDDFMT
BLNDDCDB	BLNDDDPT
BLNDDCDB	BLNDDDSP
BLNDDCDB	BLNDDFND
BLNDDCDB	BLNDDFPD
BLNDDCDB	BLNDDHFN
BLNDDDPT	BLNDDPRT
BLNDDDSP	BLNFNDDC
BLNDDEBC	BLNDDHBL
BLNDDEBC	BLNSMOCL
BLNDDFDO	BLNDDFMT
BLNDDFDZ	BLNDDFDO
BLNDDFFN	BLNDDFND
BLNDDFMT	BLNAXFM1
BLNDDFMT	BLNDDCDB
BLNDDFMT	BLNDDRPT
BLNDDFMT	BLNMKBMD
BLNDDFND	BLNFNDDC
BLNDDFPD	BLNDDFPL
BLNDDFPL	BLNDDFPS
BLNDDFPP	BLNDDDPT
BLNDDFPP	BLNDDFPD
BLNDDFPS	BLNDDFPT
BLNDDFPT	BLNDDPRT
BLNDDHBL	BLNDDHDO
BLNDDHDO	BLNDDFMT
BLNDDHFN	BLNDDFND
BLNDDINT	BLNFNDDC
BLNDDLSZ	BLNAXFM1
BLNDDLSZ	BLNDDCDB
BLNDDLSZ	BLNDDFND
BLNDDLSZ	BLNDDRPT
BLNDDLSZ	BLNMKSMB
BLNDDLSZ	BLNSMOCL
BLNDDMDT	BLNDDHBL
BLNDDNBU	BLNDDHDO
BLNDDNBU	BLNDDNLI
BLNDDNLI	BLNDDBTE
BLNDDNLI	BLNDDCDB
BLNDDNLI	BLNDDFDO
BLNDDNLI	BLNDDFDZ
BLNDDNLI	BLNDDHDO
BLNDDPOP	BLNDDPRT
BLNDDPRT	BLNFNDDC
BLNDDQAL	BLNDDVRF
BLNDDQRY	BLNDDHFN
BLNDDQRY	BLNDDMDT
BLNDDQRY	BLNDDRPT
BLNDDRDF	BLNDDVRF
BLNDDREQ	BLNDDCDB
BLNDDREQ	BLNDDFDZ
BLNDDREQ	BLNDDHFN
BLNDDREQ	BLNDDMDT
BLNDDREQ	BLNDDRPT
BLNDDREQ	BLNMKBMD
BLNDDREQ	BLNMKQRY
BLNDDRPT	BLNDDDPT
BLNDDRPT	BLNDDPRT
BLNDDSYI	BLNDDINT
BLNDDSYM	BLNDDBAS
BLNDDTRM	BLNFNDDC
BLNDDVFA	BLNDDVFN
BLNDDVFN	BLNDDFND
BLNDDVRF	BLNDDCDB
BLNDMADU	BLNDMGNT
BLNDMADU	BLNDMUAD
BLNDMAD1	BLNDMADU
BLNDMAD1	BLNDMCK2
BLNDMAD1	BLNONEND
BLNDMCHK	BLNDMADU
BLNDMCHK	BLNDMDDL
BLNDMCHK	BLNDMDEL
BLNDMCHK	BLNDMPDL
BLNDMCHK	BLNOFFLD
BLNDMCHK	BLNOFUDM
BLNDMCHK	BLNONEND
BLNDMCHK	BLNONINT
BLNDMCK1	BLNDMCHK
BLNDMCK2	BLNDMCHK
BLNDMCK2	BLNDMCK1
BLNDMCK3	BLNDMCK1
BLNDMCNT	BLNDMADU
BLNDMCNT	BLNDMDEL
BLNDMDDL	BLNDMGNT
BLNDMDEL	BLNDMGNT
BLNDMDMU	BLNDMGNT
BLNDMGNT	BLNFNDMC
BLNDMGNT	BLNFNPFC
BLNDMGNT	BLNFNPNC
BLNDMPDL	BLNDMGNT
BLNDMRDF	BLNDMADU
BLNDMRDF	BLNDMDDL
BLNDMRDF	BLNDMDEL
BLNDMRDF	BLNDMDMU
BLNDMRDF	BLNDMPDL
BLNDMRDF	BLNDMUAI
BLNDMRDF	BLNOFFLD
BLNDMRDF	BLNOFUDM
BLNDMRDF	BLNONEND
BLNDMRDF	BLNONINT
BLNDMRD1	BLNDMRDF
BLNDMSRT	BLNDMADU
BLNDMSRT	BLNDMDDL
BLNDMSRT	BLNDMPDL
BLNDMSRT	BLNONEND
BLNDMTRM	BLNDMDMU
BLNDMTRM	BLNDMGNT
BLNDMUAB	
BLNDMUAD	BLNDMUAB
BLNDMUAI	BLNDMGNT
BLNDMUDI	BLNDMDEL
BLNDMWDF	BLNDMADU
BLNDMWDF	BLNDMCK2
BLNDMWDF	BLNDMDEL
BLNDMWDF	BLNDMDMU
BLNDMWDF	BLNOFUDM
BLNDMWDF	BLNONEND
BLNDSBCL	BLNDSBSO
BLNDSBSO	BLNDSCNS
BLNDSBSO	BLNDSCRQ
BLNDSBSO	BLNDSCST
BLNDSCEN	BLNDSCOL
BLNDSCNS	BLNDSCOL
BLNDSCOL	BLNDSDSP
BLNDSCOL	BLNDSPRT
BLNDSCRQ	BLNDSCOL
BLNDSCST	BLNDSCOL
BLNDSDSP	BLNDSYMP
BLNDSFED	BLNDSCEN
BLNDSGBF	BLNDSBCL
BLNDSGBF	BLNDSBSO
BLNDSGBF	BLNDSCEN
BLNDSGBF	BLNDSCNS
BLNDSGBF	BLNDSCRQ
BLNDSGBF	BLNDSCST
BLNDSGBF	BLNDSFED
BLNDSGBF	BLNDSPRT
BLNDSPRT	BLNDSYMP
BLNDSTRM	BLNDSYMP
BLNDSYMP	BLNFNDSC
BLNDVFFR	BLNDDFDO
BLNDVFFR	BLNSMBOK
BLNDVFPE	BLNAREXA
BLNDVFPE	BLNDVSIN
BLNDVKCB	BLNDDFPT
BLNDVRCB	BLNDDFPT
BLNDVSIA	BLNDVSIN
BLNDVSIN	BLNFNDVS
BLNDVSTM	BLNDVSIN
BLNDVSTM	BLNFNDVS
BLNERCMB	BLNASADF
BLNERCMB	BLNASADL
BLNERCMB	BLNASAXA
BLNERCMB	BLNASAXC
BLNERCMB	BLNASAXK
BLNERCMB	BLNASAXT
BLNERCMB	BLNASAXX
BLNERCMB	BLNASFMT

Called Module	Calling Module
BLNERCMB	BLNASGET
BLNERCMB	BLNASLCB
BLNERCMB	BLNASLD1
BLNERCMB	BLNASLOC
BLNERCMB	BLNASLST
BLNERCMB	BLNASMAP
BLNERCMB	BLNASMD
BLNERCMB	BLNASMP1
BLNERCMB	BLNASRSR
BLNERCMB	BLNASTRM
BLNFNARC	BLNBADV3
BLNFNARC	BLNSFCAR
BLNFNARC	BLNSFCA1
BLNFNARC	BLNSFMDB
BLNFNDC2	BLNFNDMC
BLNFNDC3	BLNFNDSC
BLNFNDDC	BLNBADV1
BLNFNDDC	BLNSFDD2
BLNFNDDC	BLNSFDVD
BLNFNDDC	BLNSFSCL
BLNFNDDO	BLNDVSIN
BLNFNDDO	BLNSFDVO
BLNFNDMC	BLNBADM1
BLNFNDMC	BLNSFDMD
BLNFNDMC	BLNSFDMG
BLNFNDMC	BLNSFNDN
BLNFNDSC	BLNBADS1
BLNFNDSC	BLNSFDSP
BLNFNDVS	BLNBADV2
BLNFNDVS	BLNSFDVS
BLNFNMSK	BLNSFMSK
BLNFNPC1	BLNFNPNC
BLNFNPC2	BLNFNPFC
BLNFNPFC	BLNBAPF1
BLNFNPFC	BLNSFOFL
BLNFNPNC	BLNBAPN1
BLNFNPNC	BLNSFONL
BLNFNSMC	BLNDDFPD
BLNFNSMC	BLNDDFPS
BLNFNSMC	BLNDVKCB
BLNFNSMC	BLNDVRCB
BLNFNSMC	BLNSFASM
BLNFNSMC	BLNSFAS1
BLNFNSMC	BLNSFDAT
BLNFNSMC	BLNSFLNK
BLNFNSMC	BLNSFLN1
BLNFNSMC	BLNSFLOC
BLNFNSMC	BLNSFMDA
BLNINIT	
BLNINPRT	BLNCMPRO
BLNINPRT	BLNCMPRT
BLNITSPF	
BLNLBARB	BLNLBARY
BLNLBARC	BLNLBARB
BLNLBARC	BLNLBARY
BLNLBARR	BLNLBBLR
BLNLBARY	BLNLBBLR
BLNLBBCI	BLNLBARC
BLNLBBCI	BLNLBCAC
BLNLBBCI	BLNLBCAE
BLNLBBCI	BLNLBCAI
BLNLBBCI	BLNLBCAL
BLNLBBCI	BLNLBCHC
BLNLBBCI	BLNLBCHE
BLNLBBCI	BLNLBCHI
BLNLBBCI	BLNLBCHL
BLNLBBCI	BLNLBCHW
BLNLBBCI	BLNLBSIM
BLNLBBLR	BLNSMBLB
BLNLBCAC	BLNLBCHN
BLNLBCAE	BLNLBCHN
BLNLBCAI	BLNLBCHN
BLNLBCAL	BLNLBCHN
BLNLBCEF	BLNLBBCI
BLNLBCEF	BLNLBCAE
BLNLBCEF	BLNLBCAL
BLNLBCEF	BLNLBCHE
BLNLBCEF	BLNLBCHL
BLNLBCHC	BLNLBCHN
BLNLBCHE	BLNLBCHN
BLNLBCHI	BLNLBCHN
BLNLBCHL	BLNLBCHN
BLNLBCHN	BLNLBARB
BLNLBCHN	BLNLBBLR
BLNLBCHR	BLNLBARR
BLNLBCHR	BLNLBBLR
BLNLBCHW	BLNLBCHN
BLNLBCLN	BLNLBCHC
BLNLBCLN	BLNLBCHE
BLNLBCLN	BLNLBCHI
BLNLBCLN	BLNLBCHL
BLNLBCLN	BLNLBCHW
BLNLBCMA	BLNLBBCI
BLNLBCNT	BLNLBARR
BLNLBCNT	BLNLBCAC
BLNLBCNT	BLNLBCHC
BLNLBLAL	BLNLBCAC
BLNLBLAL	BLNLBCAE
BLNLBLAL	BLNLBCAI
BLNLBLAL	BLNLBCAL
BLNLBRCB	BLNLBBCI
BLNLBSIM	BLNLBARB
BLNLBSIM	BLNLBBLR
BLNLBVER	BLNDDFMT
BLNLBVER	BLNDDSYI
BLNLBVER	BLNSMBLB
BLNLBVER	BLNSMHXA
BLNLBVER	BLNSMLKL
BLNLBVER	BLNSMTXA
BLNLBVZA	BLNDDRDF
BLNLBVZA	BLNSMBLF
BLNLBVZA	BLNSMLKG
BLNMACBC	BLNMAIN
BLNMACBG	BLNMACBC
BLNMAIN	BLNINIT
BLNMAIN	BLNITSPF
BLNMAOCM	BLNMAIN
BLNMAPDF	BLNMACBC
BLNMKBMD	BLNMKDSP
BLNMKDSP	BLNFNMSK
BLNMKINT	BLNFNMSK
BLNMKMSK	BLNFNMSK
BLNMKQRY	BLNMKDSP
BLNMKSMB	BLNMKDSP
BLNMKTRM	BLNFNMSK
BLNMKVMR	BLNMKDSP
BLNMSGS	BLNARGDN
BLNMSGS	BLNARGRN
BLNMSGS	BLNARINT
BLNMSGS	BLNARPRN
BLNMSGS	BLNCMERM
BLNMSGS	BLNDSCEN
BLNMSGS	BLNDSDSP
BLNMSGS	BLNDSGBF
BLNMSGS	BLNDSPRT
BLNMSGS	BLNDSTRM
BLNMSGS	BLNDSYMP
BLNMSGS	BLNFNARC
BLNMSGS	BLNSMHXA
BLNMSGS	BLNSMTXA
BLNOFFLD	BLNDMGNT
BLNOFOUT	BLNOFFLD
BLNOFUDM	BLNOFFLD
BLNONDDT	BLNONLOD
BLNONEND	BLNONLOD
BLNONINT	BLNONLOD
BLNONLOD	BLNDMGNT
BLNONLOD	
BLNONMUD	BLNONLOD
BLNONSCM	BLNONLOD
BLNONSC1	BLNONSCM
BLNONTRW	BLNONLOD
BLNONVSE	BLNONDDT
BLNPRTS	BLNCOPT
BLNRAADB	BLNRAS6B
BLNRAARB	BLNRAS6B
BLNRAASB	BLNRAS6B
BLNRABAB	BLNRAS6B
BLNRACCB	BLNRAS6B
BLNRADDB	BLNRAS6B
BLNRADMB	BLNRAS6B
BLNRADSB	BLNRAS6B
BLNRADVT	BLNRAS6B
BLNRAEIN	BLNRASEC
BLNRALBD	BLNRAADB
BLNRALBD	BLNRAARB
BLNRALBD	BLNRAASB
BLNRALBD	BLNRABAB
BLNRALBD	BLNRACCB
BLNRALBD	BLNRADDB
BLNRALBD	BLNRADMB

Called Module	Calling Module
BLNRALBD	BLNRADSB
BLNRALBD	BLNRADVT
BLNRALBD	BLNRAMKB
BLNRALBD	BLNRASCB
BLNRALBD	BLNRASFB
BLNRALBD	BLNRASMB
BLNRAMKB	BLNRAS6B
BLNRAS	BLNMAIN
BLNRASCB	BLNRAS6B
BLNRASEC	BLNRAS
BLNRASFB	BLNRAS6B
BLNRASMB	BLNRAS6B
BLNRAS4B	BLNRASEC
BLNRAS5B	BLNRASEC
BLNRAS6B	BLNRASEC
BLNRAS6S	BLNRAADB
BLNRAS6S	BLNRAARB
BLNRAS6S	BLNRAASB
BLNRAS6S	BLNRABAB
BLNRAS6S	BLNRACCB
BLNRAS6S	BLNRADDB
BLNRAS6S	BLNRADMB
BLNRAS6S	BLNRADSB
BLNRAS6S	BLNRADVT
BLNRAS6S	BLNRAMKB
BLNRAS6S	BLNRASCB
BLNRAS6S	BLNRASFB
BLNRAS6S	BLNRASMB
BLNSFAMS	BLNSFIMP
BLNSMBHX	BLNSMBSI
BLNSMBLB	BLNSMBLH
BLNSMBLB	BLNSMLKC
BLNSMBLE	BLNSMBLL
BLNSMBLF	BLNSMBLE
BLNSMBLH	BLNSMBLF
BLNSMBLK	BLNSMBSI
BLNSMBLL	BLNSMSDC
BLNSMBLN	BLNSMBME
BLNSMBLO	BLNSMBLL
BLNSMBMA	BLNSMBML
BLNSMBME	BLNSMBMA
BLNSMBML	BLNSMINT
BLNSMBMN	BLNSMBME
BLNSMBOK	BLNSMBOL
BLNSMBOL	BLNSMBLO
BLNSMBOL	BLNSMLKP
BLNSMBPE	BLNSMBSI
BLNSMBSI	BLNSMSDC
BLNSMBTX	BLNSMBSI
BLNSMCML	BLNSMDSP
BLNSMCMO	BLNSMCML
BLNSMDSP	BLNFNSMC
BLNSMDSP	BLNSMFND
BLNSMDSP	BLNSMFN1
BLNSMDSP	BLNSMPCD
BLNSMFCD	BLNSMFES
BLNSMFC1	BLNSMFCD
BLNSMFC1	BLNSMFC1
BLNSMFES	BLNDVRCB
BLNSMFES	BLNSMCML
BLNSMFES	BLNSMSDC
BLNSMFES	BLNSMTRM
BLNSMFLD	BLNSMFES
BLNSMFND	BLNFNSMC
BLNSMFN1	BLNSMFND
BLNSMFRE	BLNSMHXA
BLNSMFTD	BLNSMFES
BLNSMFXD	BLNSMFES
BLNSMGRE	BLNSMHXA
BLNSMHXA	BLNSMBHX
BLNSMHXO	BLNSMBHX
BLNSMINT	BLNFNSMC
BLNSMLKB	BLNSMLKL
BLNSMLKC	BLNLBBCI
BLNSMLKC	BLNSMLKB
BLNSMLKD	BLNSMLKB
BLNSMLKG	BLNSMLKI
BLNSMLKI	BLNSMLKL
BLNSMLKL	BLNSMBLK
BLNSMLKN	BLNSMLKB
BLNSMLKN	BLNSMLKN
BLNSMLKO	BLNSMBLK
BLNSMLKP	BLNSMLKO
BLNSMLKP	BLNSMLKP
BLNSMLSL	BLNSMBLO
BLNSMLSL	BLNSMSNL
BLNSMLSP	BLNSMLSP
BLNSMLSP	BLNSMSNC
BLNSMOCL	BLNSMHXA
BLNSMPCD	BLNDDFPS
BLNSMPCD	BLNDDFPT
BLNSMPCD	BLNSMPRT
BLNSMPHP	BLNSMPCD
BLNSMPLA	BLNARPRT
BLNSMPLA	BLNSMPCD
BLNSMPRT	BLNFNSMC
BLNSMPSO	BLNSMBPE
BLNSMPSW	BLNSMHXA
BLNSMSDC	BLNSMDSP
BLNSMSEL	BLNFNSMC
BLNSMSEM	BLNSMSEL
BLNSMSNC	BLNSMSNI
BLNSMSNI	BLNSMSEL
BLNSMSNL	BLNSMSNI
BLNSMTRM	BLNFNSMC
BLNSMTXA	BLNSMBTX
BLNSMTXO	BLNSMBTX
BLNSUBMT	BLNOFFLD
BLNSUBMT	BLNONINT
BLNTERM	BLNMAIN
BLNUSADM	BLNDMUAI
BLNUSDDM	BLNDMUDI
CDP	BLNINPRT
CTA	BLNINPRT
DUMPACC	BLNADIFR
DUMPACC	BLNADIFT
EXITRTN	BLNAREXE

Dump Access Module/Reason Cross-Reference

In the following list, Dump Access modules are listed alphabetically by module name. Each module name is followed by the reason code that the module issues, the name of the message in which the reason code is inserted, and a description of the reason code.

The symbols under reason code have the following meaning:

- LBR - Librarian reason code
- rtc - GETVIS/FREEVIS return code

Module Issuing	Reason Code	Message ID	Description
IJBXAALC	LBR	BLN9031I	
IJBXABLD	LBR	BLN9031I	
IJBXACMP	100	BLN9002I	Invalid dump record
IJBXACTL	012	BLN9002I	Invalid base address
IJBXACTL	016	BLN9002I	Invalid request type
IJBXACTL	104	BLN9002I	No storage available
IJBXAFND	112	BLN9002I	Member not found
IJBXAFND	LBR	BLN9031I	
IJBXAFVS	800+rtc	BLN9031I	Free storage error
IJBXAGAC	116	BLN9002I	Truncation occurred
IJBXAGAC	120	BLN9002I	Exceeds member
IJBXAGAC	LBR	BLN9031I	
IJBXAGET	109	BLN9002I	Truncation occurred
IJBXAGET	110	BLN9002I	Exceeds member
IJBXAGET	LBR	BLN9031I	
IJBXAGVS	700+rtc	BLN9002I	Get storage error
IJBXAINS	124	BLN9002I	Get storage error
IJBXAINT	150	BLN9002I	Initialization failure
IJBXALCN	130	BLN9002I	No sublibrary
IJBXALCN	134	BLN9002I	Sublibrary full
IJBXALCN	138	BLN9002I	Library connect error
IJBXALCN	LBR	BLN9031I	
IJBXALDC	LBR	BLN9031I	
IJBXALNB	128	BLN9002I	Invalid dump name
IJBXAMDC	LBR	BLN9031I	
IJBXANOT	LBR	BLN9031I	
IJBXAPAC	134	BLN9002I	Sublibrary full
IJBXAPAC	LBR	BLN9031I	
IJBXAPNT	112	BLN9002I	Member not found
IJBXAPNT	LBR	BLN9031I	
IJBXAPSQ	LBR	BLN9031I	
IJBXAQDA	028	BLN9002I	Data not found
IJBXAQDA	004	BLN9002I	Invalid mode request
IJBXARDD	028	BLN9002I	Data not found
IJBXARDD	024	BLN9002I	Partial data returned
IJBXARDD	144	BLN9002I	Invalid length
IJBXARDD	148	BLN9002I	Invalid length
IJBXARDR	004	BLN9002I	Invalid mode request
IJBXARSP	008	BLN9002I	Invalid QUAL request
IJBXARSP	028	BLN9002I	Data not found
IJBXARSR	008	BLN9002I	Invalid QUAL request
IJBXARSR	028	BLN9002I	Data not found
IJBXARWR	004	BLN9002I	Invalid mode request
IJBXASTW	LBR	BLN9031I	
IJBXATSR	144	BLN9002I	Stow error

Module Issuing	Reason Code	Message ID	Description
IJBXAWUP	140	BLN9002I	Invalid length
IJBXAWUP	028	BLN9002I	Data not found

Dump Access Reason/Module Cross-Reference

In the following list, Dump Access reason codes are listed alphabetically. Each reason code is followed by the name of the module that issues the reason code, the message in which it is inserted, and a description of the reason code. (Refer to “Dump Access Module/Reason Cross-Reference” on page 281 for additional information.)

Reason Code	Module Issuing	Message ID	Description
LBR	IJBXAALC	BLN9031I	
LBR	IJBXABLD	BLN9031I	
LBR	IJBXAFND	BLN9031I	
LBR	IJBXAGAC	BLN9031I	
LBR	IJBXAGET	BLN9031I	
LBR	IJBXALCN	BLN9031I	
LBR	IJBXALDC	BLN9031I	
LBR	IJBXAMDC	BLN9031I	
LBR	IJBXANOT	BLN9031I	
LBR	IJBXAPAC	BLN9031I	
LBR	IJBXAPNT	BLN9031I	
LBR	IJBXAPSQ	BLN9031I	
LBR	IJBXASTW	BLN9031I	
004	IJBXAQDA	BLN9002I	Invalid mode request
004	IJBXARDR	BLN9002I	Invalid mode request
004	IJBXARWR	BLN9002I	Invalid mode request
008	IJBXARSP	BLN9002I	Invalid QUAL request
008	IJBXARSR	BLN9002I	Invalid QUAL request
012	IJBXACTL	BLN9002I	Invalid base address
016	IJBXACTL	BLN9002I	Invalid type request
024	IJBXARDD	BLN9002I	Partial data returned
028	IJBXAQDA	BLN9002I	Data not found
028	IJBXARDD	BLN9002I	Data not found
028	IJBXARSP	BLN9002I	Data not found
028	IJBXARSR	BLN9002I	Data not found
028	IJBXAWUP	BLN9002I	Data not found
100	IJBXACMP	BLN9002I	Invalid dump record
104	IJBXACTL	BLN9002I	No storage available
109	IJBXAGET	BLN9002I	Truncation occurred
110	IJBXAGET	BLN9002I	Exceeds member
112	IJBXAFND	BLN9002I	Member not found
112	IJBXAPNT	BLN9002I	Member not found
116	IJBXAGAC	BLN9002I	Truncation occurred
120	IJBXAGAC	BLN9002I	Exceeds member
124	IJBXAINS	BLN9002I	Get storage error
128	IJBXALNB	BLN9002I	Invalid dump name
130	IJBXALCN	BLN9002I	No sublibrary
134	IJBXALCN	BLN9002I	Sublibrary full
134	IJBXAPAC	BLN9002I	Sublibrary full
138	IJBXALCN	BLN9002I	Library connect error
140	IJBXAWUP	BLN9002I	Invalid length
144	IJBXARDD	BLN9002I	Invalid length
144	IJBXATSR	BLN9002I	Stow error
148	IJBXARDD	BLN9002I	Invalid length
150	IJBXAINT	BLN9002I	Initialization failure
700+rtc	IJBXAGVS	BLN9002I	Get storage error
800+rtc	IJBXAFVS	BLN9002I	Free storage error

Dump Access Calling/Called Module Cross-Reference

This section lists, alphabetically, the names of Dump Access calling modules (modules that call other modules). Following each calling module name are the names of the modules that it calls. In addition, the following information on calling modules may be useful:

- Link book referenced: - IJBXALNK

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
IJBXAALC	IJBXAFVS	IJBXAFND	IJBXAGVS	IJBXAPSQ	
IJBXAALC	IJBXAGVS	IJBXAFND	IJBXALDC	IJBXAQDA	IJBXAINT
IJBXAALC	IJBXALDC	IJBXAFND	IJBXAMDC	IJBXARBM	
IJBXAALC	IJBXAMDC	IJBXAFVS		IJBXARBS	
IJBXAAMP	IJBXAALC	IJBXAGAC		IJBXARBU	IJBXAGAC
IJBXAAMP	IJBXACMP	IJBXAGET	IJBXANOT	IJBXARBU	IJBXARBM
IJBXAAMP	IJBXAFND	IJBXAGET		IJBXARBU	IJBXARBS
IJBXAAMP	IJBXAFVS	IJBXAGMA		IJBXARDA	IJBXAMAP
IJBXAAMP	IJBXAGET	IJBXAGMP	IJBXAGVS	IJBXARDA	IJBXARBU
IJBXAAMP	IJBXALDC	IJBXAGVS		IJBXARDD	IJBXARDA
IJBXAAMP	IJBXAMDC	IJBXAIND	IJBXALCN	IJBXARDR	IJBXARDD
IJBXAAMP	IJBXASTW	IJBXAIND	IJBXALNB	IJBXARDR	IJBXARSP
IJBXAAMP	IJBXAWMR	IJBXAIND	IJBXAMCN	IJBXARDR	IJBXARSR
IJBXAAVL	IJBXABLD	IJBXAINE	IJBXACMP	IJBXARMP	IJBXAGET
IJBXAAVL	IJBXAFVS	IJBXAINE	IJBXAFND	IJBXARMP	IJBXAGVS
IJBXAAVL	IJBXAGVS	IJBXAINE	IJBXAGET	IJBXARSP	IJBXARBU
IJBXAAVL	IJBXALCN	IJBXAINE	IJBXAGVS	IJBXARSR	IJBXARBU
IJBXAAVL	IJBXALDC	IJBXAINF	IJBXACMP	IJBXARWR	IJBXAINT
IJBXAAVL	IJBXALNB	IJBXAINF	IJBXAGET	IJBXARWR	IJBXARDR
IJBXAAVL	IJBXAMDC	IJBXAINF	IJBXAGVS	IJBXARWR	IJBXAWAP
IJBXABLD		IJBXAINF	IJBXAPNT	IJBXARWR	IJBXAWUP
IJBXACMP	IJBXAMES	IJBXAINM	IJBXAAMP	IJBXASME	
IJBXACMP		IJBXAINM	IJBXAFND	IJBXASTW	
IJBXADEL	IJBXAFND	IJBXAINM	IJBXARMP	IJBXATDD	IJBXAFVS
IJBXADEL	IJBXAFVS	IJBXAINS	IJBXAGVS	IJBXATDD	IJBXALDC
IJBXADEL	IJBXAINT	IJBXAINT	IJBXAIND	IJBXATDD	IJBXAMDC
IJBXADEL	IJBXALCN	IJBXAINT	IJBXAINE	IJBXATDD	IJBXASTW
IJBXADEL	IJBXALDC	IJBXAINT	IJBXAINF	IJBXATDD	IJBXATMD
IJBXADEL	IJBXALNB	IJBXAINT	IJBXAINM	IJBXATMD	IJBXAFVS
IJBXADEL	IJBXAMCN	IJBXAINT	IJBXAINS	IJBXATMD	IJBXALDC
IJBXADEL	IJBXAMDC	IJBXALCN	IJBXAFVS	IJBXATMD	IJBXAMDC
IJBXADEL	IJBXASTW	IJBXALCN	IJBXAGVS	IJBXATMD	IJBXASTW
IJBXACTL	IJBXAAVL	IJBXALDC		IJBXATMD	IJBXAWMR
IJBXACTL	IJBXADEL	IJBXALNB		IJBXATRM	IJBXATDD
IJBXACTL	IJBXAGMA	IJBXAMAP		IJBXATRM	IJBXATSR
IJBXACTL	IJBXAGVS	IJBXAMCN	IJBXAFND	IJBXATSR	IJBXAFVS
IJBXACTL	IJBXAINT	IJBXAMCN	IJBXAFVS	IJBXAWAP	IJBXACMP
IJVXACTL	IJBXAQDA	IJBXAMCN	IJBXAGET	IJBXAWAP	IJBXAEMC
IJVXACTL	IJBXARWR	IJBXAMCN	IJBXAGVS	IJBXAWAP	IJBXAPAC
IJVXACTL	IJBXATRM	IJBXAMDC		IJBXAWAP	IJBXAWNM
IJBXAEMC	IJBXAALC	IJBXAMES	IJBXAGMP	IJBXAWDR	IJBXAFVS
IJBXAEMC	IJBXAFVS	IJBXAMES	IJBXASME	IJBXAWDR	IJBXAPAC
IJBXAEMC	IJBXAINE	IJBXANOT		IJBXAWMR	IJBXAFVS
IJBXAEMC	IJBXALDC	IJBXAPAC		IJBXAWMR	IJBXAPSQ
IJBXAEMC	IJBXAMDC	IJBXAPNT	IJBXAFVS	IJBXAWMR	IJBXASME
IJBXAEMC	IJBXAPAC	IJBXAPNT	IJBXAGVS	IJBXAWNM	IJBXAALC
IJBXAEMC	IJBXASTW	IJBXAPNT	IJBXALDC	IJBXAWNM	IJBXACMP
IJBXAFND	IJBXAFVS	IJBXAPNT	IJBXAMDC	IJBXAWNM	IJBXAGVS

Calling Module	Called Module
IJBXAWNM	IJBXAPAC
IJBXAWUP	IJBXAGVS
IJBXAWUP	IJBXARDA
IJBXAWUP	IJBXAWDR

Dump Access Called/Calling Module Cross-Reference

This section lists, alphabetically, the names of Dump Access modules that are called by other modules. Following each called module name are the names of the modules that call it.
(Refer to “Dump Access Calling/Called Module Cross-Reference” on page 284 for additional information.)

Called Module	Calling Module
IJBXAALC	IJBXAAMP
IJBXAALC	IJBXAEMC
IJBXAALC	IJBXAWNM
IJBXAAMP	IJBXAINM
IJBXAAVL	IJBXACTL
IJBXABLD	IJBXAAVL
IJBXACMP	IJBXAAMP
IJBXACMP	IJBXAINE
IJBXACMP	IJBXAINF
IJBXACMP	IJBXAWAP
IJBXACMP	IJBXAWNM
IJBXADEL	IJBXACTL
IJBXACTL	
IJBXAEMC	IJBXAWAP
IJBXAFND	IJBXAAMP
IJBXAFND	IJBXADEL
IJBXAFND	IJBXAINE
IJBXAFND	IJBXAINM
IJBXAFND	IJBXAMCN
IJBXAFVS	IJBXAALC
IJBXAFVS	IJBXAAMP
IJBXAFVS	IJBXAAVL
IJBXAFVS	IJBXADEL
IJBXAFVS	IJBXAEMC
IJBXAFVS	IJBXAFND
IJBXAFVS	IJBXALCN
IJBXAFVS	IJBXAMCN
IJBXAFVS	IJBXAPNT
IJBXAFVS	IJBXATDD
IJBXAFVS	IJBXATMD
IJBXAFVS	IJBXATSR
IJBXAFVS	IJBXAWDR
IJBXAFVS	IJBXAWMR
IJBXAGAC	IJBXARBU
IJBXAGET	IJBXAAMP
IJBXAGET	IJBXAINE
IJBXAGET	IJBXAINF
IJBXAGET	IJBXAMCN
IJBXAGET	IJBXARMP
IJBXAGMA	IJBXACTL
IJBXAGMP	IJBXAMES
IJBXAGVS	IJBXAALC
IJBXAGVS	IJBXAAVL
IJBXAGVS	IJBXACTL
IJBXAGVS	IJBXAFND
IJBXAGVS	IJBXAGMP
IJBXAGVS	IJBXAINE
IJBXAGVS	IJBXAINF
IJBXAGVS	IJBXAINS
IJBXAGVS	IJBXALCN
IJBXAGVS	IJBXAMCN
IJBXAGVS	IJBXAPNT
IJBXAGVS	IJBXARMP
IJBXAGVS	IJBXAWNM
IJBXAGVS	IJBXAWUP
IJBXAIND	IJBXAINT
IJBXAINE	IJBXAEMC
IJBXAINE	IJBXAINT
IJBXAINF	IJBXAINT
IJBXAINM	IJBXAINT
IJBXAINS	IJBXAINT
IJBXAINT	IJBXADEL
IJBXAINT	IJBXACTL
IJBXAINT	IJBXAQDA
IJBXAINT	IJBXADEL
IJBXAINT	IJBXACTL
IJBXAINT	IJBXAQDA
IJBXAINT	IJBXARWR
IJBXALCN	IJBXAAVL
IJBXALCN	IJBXADEL
IJBXALCN	IJBXAIND
IJBXALDC	IJBXAALC
IJBXALDC	IJBXAAMP
IJBXALDC	IJBXAAVL
IJBXALDC	IJBXADEL
IJBXALDC	IJBXAEMC
IJBXALDC	IJBXAFND
IJBXALDC	IJBXAPNT
IJBXALDC	IJBXATDD
IJBXALDC	IJBXATMD
IJBXALNB	IJBXAAVL
IJBXALNB	IJBXADEL
IJBXALNB	IJBXAIND
IJBXAMAP	IJBXARDA
IJBXAMCN	IJBXADEL
IJBXAMCN	IJBXAIND
IJBXAMDC	IJBXAALC
IJBXAMDC	IJBXAAMP
IJBXAMDC	IJBXAAVL
IJBXAMDC	IJBXADEL
IJBXAMDC	IJBXAEMC
IJBXAMDC	IJBXAFND
IJBXAMDC	IJBXAPNT
IJBXAMDC	IJBXATDD
IJBXAMDC	IJBXATMD
IJBXAMES	IJBXACMP
IJBXANOT	IJBXAGET
IJBXAPAC	IJBXAEMC
IJBXAPAC	IJBXAWAP
IJBXAPAC	IJBXAWDR
IJBXAPAC	IJBXAWNM
IJBXAPNT	IJBXAINF
IJBXAPSQ	IJBXAWMR
IJBXAQDA	IJBXACTL
IJBXARBM	IJBXARBU
IJBXARBS	IJBXARBU
IJBXARBU	IJBXARDA
IJBXARBU	IJBXARSP
IJBXARBU	IJBXARSR
IJBXARDA	IJBXARDD
IJBXARDA	IJBXAWUP
IJBXARDD	IJBXARDR
IJBXARDR	IJBXARWR
IJBXARMP	IJBXAINM
IJBXARSP	IJBXARDR
IJBXARSR	IJBXARDR
IJBXARWR	IJBXACTL
IJBXASME	IJBXAMES
IJBXASME	IJBXAWMR
IJBXASTW	IJBXAAMP
IJBXASTW	IJBXADEL
IJBXASTW	IJBXAEMC
IJBXASTW	IJBXATDD
IJBXASTW	IJBXATMD
IJBXATDD	IJBXATRM
IJBXATMD	IJBXATDD
IJBXATRM	IJBXACTL
IJBXATSR	IJBXATRM
IJBXAWAP	IJBXARWR
IJBXAWDR	IJBXAWUP
IJBXAWMR	IJBXAAMP
IJBXAWMR	IJBXATMD
IJBXAWNM	IJBXAWAP
IJBXAWUP	IJBXARWR

Information Analysis Module Flow

This chapter contains the following module flows that can help in diagnosing problems within Info/Analysis:

- Module flow for batch processing
- Module flow for analysis routines
- Module flow for symptom record access
- Module flow for analysis routines services
- Module flow for dump management
- Module flow for dump symptoms
- Module flow for dump viewing
- Module flow for locator blocks
- Module flow for dump offload
- Module flow for dump onload
- Module flow for RAS
- Module flow for analysis summary

MODULE FLOW FOR BATCH PROCESSING

In the following list, the module flow for analysis routine functions are shown. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Batch Processing Routines:

```
BLNINIT ......... ...... Batch driver invoked from JCL
 BLNMAIN ........ ...... Main entry routine
  BLNADIFI ...... ...... Dump Access initialization
   BLNMACBC ..... ...... Initialize CCB
   BLNMAOCM ..... ...... Allocate and open printer
   BLNTERM  ..... ...... Terminate Information Analysis name
   BLNBAINT ..... ...... Allocate and open Batch reader
                         Call Batch handler
    BLNBAIPF .... ...... Find out from where to read commands
    BLNBAMSG .... ...... Batch message writer
    BLNBABH1 .... ...... Batch function handler
     BLNBABH2 ... ...... Batch return code handler
     BLNBAFLS ... ...... Batch flush routine
     BLNBARDR ... ...... Batch reader routine
     BLNBADM1 ... ...... Dump management handler part 1
      BLNFNDMC .. ...... Dump management function handler
     BLNBADM2 ... ...... Dump management handler-2 (validate dump name)
     BLNBADS1 .... ..... Dump symptoms handler
      BLNFNDSC ... ..... Dump symptoms funtion handler
     BLNBADV1 .... ..... Dump viewing handler part 1
      BLNFNDDC ... ..... Dump display function control routine
       BLNDDINT ... .....
       BLNDDDSP ... .....
       BLNDDPRT ... .....
       BLNDDFND ... .....
       BLNDDTRM ... .....
      BLNBADV2 ... ..... Dump viewing handler part 2
       BLNFNDVS .. ..... Dump viewing selection function routine
      BLNBADV3 ... ..... Dump viewing handler part 3(CALL statement)
       BLNFNARC .. ..... Analysis routines function control
     BLNBAPF1 .... ..... Dump offload handler
       BLNCMVOL .. ..... VOLID processor
       BLNFNPFC .. ..... Dump offload selection routine
     BLNBAPN1 .... ..... Dump onload handler
       BLNCMFIL .. ..... File operand processor
       BLNCMVOL .. ..... VOLID processor
       BLNFNPNC .. ..... Dump onload selection handler
```

MODULE FLOW FOR ANALYSIS ROUTINES

In the following list, the module flow for analysis routine functions are shown. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Initialize Analysis Routines:

```
BLNFNARC ........ ........ Analysis routines function control
 BLNARINT ....... ........ Analysis routines initialization
  BLNARGRN ...... ........ Get list of analysis routines
   BLNARGDN ..... ........ Get records from external data set
    BLNCMXDE .... ........ Handle I/O error reason codes
    BLNARPRN .... ........ Process input line for routine name
     BLNARPDI ... ........ Parse input line for name and description
      BLNARFML .. ........ Free ARML chain
   BLNARFML ..... ........ Free ARML chain if errors occurred
```

Display Analysis Routines List:

```
BLNFNARC ........ ........ Analysis routines function control
 BLNARDSP ....... ........ Analysis routines list display
  BLNCMALN ...... ........ Adjust line numbers
  BLNARDLO ...... ........ Build output lines for routine names
```

Execute Analysis Routine:

```
BLNFNARC ........ ........ Analysis routines function control
 BLNAREXC ....... ........ Analysis routines execution
  BLNAREXD ...... ........ Get analysis routine name
   BLNCMPRS ..... ........ Get analysis routine name
  BLNAREXE ...... ........ Load and execute analysis routine
   BLNAREXL ..... ........ Load analysis routine
   BLNAREXA ..... ........ Set up ARCB for analysis routine exit interface
    BLNDVFPE .... ........ Free previous analysis routine display
   BLNARXPL ..... ........ Build parameter list for analysis routine exit
    BLNASLOC .... ........ Get symptom record for analysis routine
    BLNCMERM .... ........ Put error message onto message queue
    BLNAXGS2 .... ........ Get storage for symptom -record copy
   EXITRTN ...... ........ Analysis exit routine
   BLNAREXF ..... ........ Cleanup after analysis exit routine
    BLNCMERM .... ........ Put error message onto message queue
    BLNAREXG .... ........ Free control block storage
```

Print Data for Analysis Routines:

```
BLNFNARC ........ ........ Analysis routines function control
 BLNARPRT ....... ........ Analysis routines list print
  BLNCMALN ...... ........ Adjust line numbers
  BLNARDLO ...... ........ Build output lines for routine names
  BLNCMPRT ...... ........ Print output lines
  BLNCMFBC ...... ........ Free print buffer chain
  BLNSMPLA ...... ........ Adjust print line data as needed
```

Find String for Analysis Routines:

```
BLNFNARC ........ ........ Analysis routines function control
 BLNARFND ....... ........ Analysis routines find
  BLNCMPRS ...... ........ Adjust for starting line
  BLNARFCS ...... ........ Find character string
   BLNARDSP ..... ........ Build output buffer
   BLNCMFBC ..... ........ Free buffer chain
  BLNARDSP ...... ........ Build output buffer
```

Terminate Analysis Routines:

```
BLNFNARC ........ ........ Analysis routines function control
 BLNARTRM ....... ........ Analysis routines termination
  BLNARFML ...... ........ Free list of names
   BLNARFML ..... ........ Recursive call to free additional names
```

MODULE FLOW FOR SYMPTOM RECORD ACCESS

This section shows the module flow for symptom record access functions are shown. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Locate Data in Symptom Record:

```
BLNASLOC* ....... ........ Locate data in symptom record
 BLNASLCB* ...... ........ Load control block
 BLNASMAP* ...... ........ Map dump data (see full list of modules below)
 BLNASLD1* ...... ........ Locate data in sections 1-5
  BLNASFSY ...... ........ Find symptom within section
 BLNASLD6 ....... ........ Locate data in section 6
  BLNASFMT* ..... ........ Format data for return to caller
   BLNASRSR* .... ........ Read symptom record from dump
    BLNADIFR .... ........ Dump access interface routine
   BLNASRFL ..... ........ Reformat LBDA to look like simple LBD
```

Update Symptom Record Mapping Tables:

```
BLNASUPD ........ ........ Locate data in symptom record
 BLNASLCB* ...... ........ Load control block
 BLNASUP6 ....... ........ Update section 6 map
  BLNASRSR* ..... ........ Read symptom record from dump
   BLNADIFR ..... ........ Dump access interface routine
  BLNASMP6* ..... ........ Map section 6 data (see full list of modules below)
```

List Locators in Section 6:

```
BLNASLST* ....... ........ List section 6 entries
 BLNASLCB* ...... ........ Load control block
BLNASMAP* ....... ........ Map dump data (see full list of modules below)
```

Terminate Symptom Record Access:

```
BLNASTRM* ....... ........ Terminate symptom record access
 BLNASLCB* ...... ........ Locate control block
 BLNASTMP ....... ........ Terminate mapping tables
```

Map Dump Data:

```
BLNASMAP* ....... ........ Map dump data
 BLNASTMP ....... ........ Terminate old mapping tables
 BLNASRSR* ...... ........ Read symptom record from dump
  BLNADIFR ...... ........ Dump access interface routine
 BLNASMP1* ...... ........ Map sections 1-5 of symptom record
 BLNASMP6* ...... ........ Map section 6 data
  BLNASMD* ...... ........ Map LBD type
   BLNASAXA* .... ........ Audit LBXA extension
   BLNASAXC* .... ........ Audit LBXC extension
   BLNASAXT* .... ........ Audit LBXT extension
   BLNASAXX* .... ........ Audit LBXX extension
    BLNASMP1* ... ........ Map sections 1-5
   BLNASAXK* .... ........ Audit LBXK extension
   BLNASADL* .... ........ Audit LBDL extension
   BLNASADF* .... ........ Audit LBDF extension
   BLNASCME* .... ........ Create map entry
    BLNASGET* ... ........ Get another block of mapping tables
    BLNASIST .... ........ Insert a text LBD in mapping table in sequence
     BLNASGET* .. ........ Get another block of mapping tables
  BLNASMDA ...... ........ Map LBDA type
   BLNASCME ..... ........ Create map entry
    BLNASGET* ... ........ Get another block of mapping tables
    BLNASIST .... ........ Insert a text LBD in mapping table in sequence
     BLNASGET* .. ........ Get another block of mapping tables
```

* = Module may call BLNERCMB to create a message in the symptom record access message buffer

MODULE FLOW FOR ANALYSIS ROUTINE SERVICES:

The following lists show the module flow for analysis routine services functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Analysis Routines Get Storage Service:

```
BLNAXGST ........ ........ Get storage service
 BLNAXGS1 ....... ........ Get storage service processor
  BLNAXGS2 ...... ........ Allocate storage service
  BLNAXGS3 ...... ........ Free storage service
```

Analysis Routines Print/Display Service:

```
BLNAXPDS ........ ........ Print/display service
 BLNAXPD1 ....... ........ Print/display service processor
  BLNAXPD2 ...... ........ Print service processor
   BLNCMPRT ..... ........ Info/Analysis print routine
   BLNAXCMB ..... ........ Create message buffer
  BLNAXPD3 ...... ........ Display service processor
   BLNAXPD4 ..... ........ Route excess display output
    BLNAXPD5 .... ........ Print previous display output
     BLNAXPD6 ... ........ Print data block for SMPD
      BLNCMPRT .. ........ Print data routine
      BLNAXCMB .. ........ Create message buffer
    BLNAXPD6 .... ........ Print data block for SMPD
     BLNCMPRT ... ........ Print data routine
     BLNAXMSG ... ........ Get messages onto queue
    BLNAXPD4 .... ........ Recursive call to route additional output
   BLNAXCMB ..... ........ Create message buffer
   BLNAXPD7 ..... ........ Put buffer on display output
   BLNAXPD2 ..... ........ Print service processor
    BLNCMPRT .... ........ Info/Analysis print routine
    BLNAXCMB .... ........ Create message buffer
  BLNAXGS3 ...... ........ Free storage service
```

Analysis Routines Dump Access Service:

```
BLNAXDAC ........ ........ Dump access service
 BLNAXDA1 ....... ........ Dump access service processor
  BLNADIFR ...... ........ Dump access interface
```

Analysis Routines Formatting Service:

```
BLNAXFMT ........ ........ Formatting service
 BLNAXFM1 ....... ........ Formatting service processor
  BLNDDLSZ ...... ........ Determine line size
  BLNDDFMT ...... ........ Dump data formatter
  BLNAXMSG ...... ........ Enqueue message on chain
   BLNAXGST ..... ........ Get storage service
  BLNAXGS2 ...... ........ Get storage service
  BLNCMFBC ...... ........ Free buffer chain
```

Analysis Routines Symptom Record Access Service:

```
BLNAXSAC ........ ........ Symptom record access service
 BLNAXSA1 ....... ........ Symptom record access service processor
  BLNASLOC ...... ........ Symptom record access
  BLNAXGS2 ...... ........ Get storage service
```

Analysis Routines Symptom Record Update Service:

```
BLNAXSUP ........ ........ Symptom record update service
 BLNAXSU1 ....... ........ Symptom record update service processor
  BLNAXSRV ...... ........ Symptom record verify
  BLNAXSUR ...... ........ Symptom record update
   BLNAXS3U ..... ........ Symptom record update section 3
    BLNAXS4U .... ........ Symptom record update section 4
    BLNAXGS2 .... ........ Get storage for new symptom record
    BLNAXGS3 .... ........ Free old symptom record storage
   BLNAXS4U ..... ........ Symptom record update section 4
    BLNAXGS2 .... ........ Get storage for new symptom record
    BLNAXGS3 .... ........ Free old symptom record storage
   BLNAXS5U ..... ........ Symptom record update section 5
    BLNAXGS2 .... ........ Get storage for new symptom record
    BLNAXGS3 .... ........ Free old symptom record storage
  BLNAXCMB ...... ........ Create message buffer
  BLNAXSUA ...... ........ Symptom record update for dump
   BLNAXSUA ..... ........ Recursive calls to add second
   BLNADIFR ..... ........ Append symptom record to dump
   BLNASUPD ..... ........ Tell symptom record access of update
  BLNAXS6U ...... ........ Symptom record update section 6
   BLNAXCMB ..... ........ Storage allocate fail message
   BLNAXS6V ..... ........ Symptom record update section 6
    BLNADIFR .... ........ Append section 6 record to dump
   BLNASUPD ..... ........ Tell symptom record access of update
```

MODULE FLOW FOR DUMP MANAGEMENT

The following lists show the module flow for dump management functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Dump Management User Add Interface:

```
BLNFNDMC ........ ........ Dump management function control
 BLNDMGNT ....... ........ Dump management driver
  BLNDMUAI ...... ........ User add interface
   BLNDMRDF ..... ........ Read dump management file control record
    BLNDMRD1 .... ........ Read dump management file data records
   BLNUSADM ..... ........ Search library for new dumps
    BLNDMUAB .... ........ Reestablish BLX environment
```

Dump Management User Add Interface:

```
    BLNDMUAD .... ........ Process new dump found in library
    BLNDMADU .... ........ Add/select dump (see below for details)
```

Display/Print Dump Management List:

```
BLNFNDMC ........ ........ Dump management function control
 BLNDMGNT ....... ........ Dump management driver
  BLNDMPDL ...... ........ Print dump management list
   BLNCMFBC ..... ........ Free buffer chain
   BLNCMPRT ..... ........ Print output buffer
   BLNDMCHK ..... ........ Locate dump management record
   BLNDMRDF ..... ........ Read dump management file control record
    BLNDMRD1 .... ........ Read dump management file data records
   BLNDMSRT ..... ........ Sort dump management file
  BLNSMDDL ...... ........ Display dump management list
   BLNDMCHK ..... ........ Locate dump management record
   BLNDMRDF ..... ........ Read dump management file control record
    BLNDMRD1 .... ........ Read dump management file data record
   BLNDMSRT ..... ........ Sort dump management file
```

Add/Select Dump:

```
BLNFNDMC ........ ........ Dump management function control
 BLNDMGNT ....... ........ Dump management driver
  BLNDMADU ...... ........ Add/select dump
   BLNDMRDF ..... ........ Read dump management file control record
    BLNDMRD1 .... ........ Read dump management file data record
   BLNDMCHK ..... ........ Check dump management file
    BLNDMCK1 .... ........ Check if entry exists in file
     BLNDMCK2 ... ........ Check if dump available in library
     BLNDMCK3 ... ........ Find next available slot in list
    BLNDMCK2 .... ........ Check if dump available in library
     BLNADIFR ... ........ Check availability of dump in library
     BLNCMERM ... ........ Handle external messages if any
     BLNDMAD1 ... ........ User add interface
     BLNDMWDF ... ........ Rewrite updated dump management file
   BLNDMAD1 ..... ........ Build line in list
    BLNASLOC .... ........ Read symptom record from dump
    BLNCMERM .... ........ Handle external messages if any
    BLNCMTDS .... ........ Convert data/time stamp to printable form
   BLNDMCNT ..... ........ Adjust count of entries in list
   BLNDMSRT ..... ........ Sort list into data/time sequence
   BLNDMWDF ..... ........ Write dump list to file
```

Delete Dump from Library and/or File:

```
BLNFNDMC ........ ........ Dump management function control
 BLNDMGNT ....... ........ Dump management driver
  BLNDMDEL ...... ........ Dump management delete
   BLNDMRDF ..... ........ Read dump management file control record
    BLNDMRD1 .... ........ Read dump management file data records
   BLNDMCHK ..... ........ Check dump management file
    BLNDMCK1 .... ........ Check if entry exists in file
     BLNDMCK2 ... ........ Check if dump available in library
     BLNDMCK3 ... ........ Find next available slot in list
    BLNDMCK2 .... ........ Check if dump available in library
     BLNADIFR ... ........ Check availability of dump in library
     BLNCMERM ... ........ Handle external messages if any
     BLNDMAD1 ... ........ User add interface
     BLNDMWDF ... ........ Rewrite updated dump management file
   BLNDMUDI ..... ........ User delete interface
    BLNASLOC .... ........ Read symptom record of dump
    BLNCMERM .... ........ Handle external messages if any
    BLNUSDDM .... ........ User delete interface module
   BLNADIFR ..... ........ Dump access interface
   BLNASTRM ..... ........ Terminate symptom record access
   BLNCMERM ..... ........ Handle external messages if any
   BLNDMCNT ..... ........ Adjust count of entries in list
   BLNDMWDF ..... ........ Rewrite updated dump management file
```

Execute Dump Management Utility:

```
BLNFNDMC ........ ........ Dump management function ccntrol
 BLNDMGNT ....... ........ Dump management driver
  BINDMDMU ...... ........ Dump management utility
   BLNDMRDF ..... ........ Read dump management file control record
    BLNDMRD1 .... ........ Read dump management file data records
   BLNDMTRM ..... ........ Terminate dump management
   BLNDMWDF ..... ........ Rewrite updated dump management file
```

Terminate Dump Management:

```
BLNFNDMC ........ ........ Dump management function control
 BLNDMGNT ....... ........ Dump management driver
  BLNDMTRM ...... ........ Terminate dump management
```

MODULE FLOW FOR DUMP SYMPTOMS

The following lists show the module flow for dump symptoms functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Display Dump Symptoms:

```
BLNFNDSC ........ ........ Dump symptoms function control
 BLNDSYMP ....... ........ Dump symptoms driver
  BLNDSDSP ...... ........ Build display output buffer
   BLNCMFBC ..... ........ Free buffer chain
   BLNDSCOL ..... ........ Dump symptoms collection control
    BLNDSCEN .... ........ Collect environment section
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSFED ... ........ Format environment data
     BLNDSGBF ... ........ Get next buffer in chain
    BLNDSCRQ .... ........ Collect required symptoms
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSBSO ... ........ Build first line of symptom output
      BLNDSBCL .. ........ Build continuation line of symptom output
       BLNDSGBF . ........ Get next buffer in chain
      BLNDSGBF .. ........ Get next buffer in chain
     BLNDSGBF ... ........ Get next buffer in chain
    BLNDSCST .... ........ Collect SBD formatted symptoms
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSBSO ... ........ Build first line of symptom output
      BLNDSBCL .. ........ Build continuation line of symptom output
       BLNDSGBF . ........ Get next buffer in chain
      BLNDSGBF .. ........ Get next buffer in chain
     BLNDSGBF ... ........ Get next buffer in chain
    BLNDSCNS .... ........ Collect non-SBD formatted symptoms
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSBSO ... ........ Build first line of symptom output
      BLNDSBCL .. ........ Build continuation line of symptom output
       BLNDSGBF . ........ Get next buffer in chain
      BLNDSGBF .. ........ Get next buffer in chain
     BLNDSGBF ... ........ Get next buffer in chain
```

Print Dump Symptoms:

```
BLNFNDSC ........ ........ Dump symptoms function control
 BLNDSYMP ....... ........ Dump symptoms driver
  BLNDSPRT ...... ........ Build print output buffer
   BLNCMFBC ..... ........ Free buffer chain
   BLNDSGBF ..... ........ Get next buffer in chain
   BLNDSCOL ..... ........ Dump symptoms collection control
    BLNDSCEN .... ........ Collect environment section
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSFED ... ........ Format environment data
     BLNDSGBF ... ........ Get next buffer in chain
    BLNDSCRQ .... ........ Collect required svmptoms
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSBSO ... ........ Build first line of symptom output
      BLNDSBCL .. ........ Build continuation line of symptom output
       BLNDSGBF . ........ Get next buffer in chain
      BLNDSGBF .. ........ Get next buffer in chain
     BLNDSGBF ... ........ Get next buffer in chain
    BLNDSCST .... ........ Collect SBD formatted symptoms
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSBSO ... ........ Build first line of symptom output
      BLNDSBCL .. ........ Build continuation line of symptom output
       BLNDSGBF . ........ Get next buffer in chain
      BLNDSGBF .. ........ Get next buffer in chain
     BLNDSGBF ... ........ Get next buffer in chain
    BLNDSCNS .... ........ Collect non-SBD formatted symptoms
     BLNASLOC ... ........ Read symptom record
     BLNCMERM ... ........ Handle external error messages if any
     BLNDSBSO ... ........ Build first line of symptom output
      BLNDSBCL .. ........ Build continuation line of symptom output
       BLNDSGBF . ........ Get next buffer in chain
      BLNDSGBF .. ........ Get next buffer in chain
     BLNDSGBF ... ........ Get next buffer in chain
   BLNCMPRT ..... ........ Print buffer chain
```

Terminate Dump Symptoms:

```
BLNFNDSC ........ ........ Dump symptoms function control
 BLNDSYMP ....... ........ Dump symptoms driver
  BLNDSTRM ...... ........ Terminate dump symptoms
   BLNCMFBC ..... ........ Free buffer chain
```

MODULE FLOW FOR DUMP VIEWING

The following lists show the module flow for dump viewing functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Select Dump Viewing Selection:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNDVSIN ....... ........ Dump viewing selection initialization
  BLNDVSIA ...... ........ Dump viewing selection allocate space
  BLNFNDDO ...... ........ Set dump display options
  BLNDVFPE ...... ........ Free pseudo section 6 data
  BLNDVSTM ...... ........ Terminate Dump viewing selection
 BLNDVSTM ....... ........ Terminate Dump viewing selection
```

Initialize Dump Display:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNFNDDC ....... ........ Dump display control
  BLNDDINT ...... ........ Dump display initialization
   BLNDDSYI ..... ........ Initialize symbol table
    BLNASLOC .... ........ Get registers from symptom record access
    BLNLBVER .... ........ Validate LBD
    BLNCMERM .... ........ Error message output
```

Request Dump Display Buffer:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNFNDDC ....... ........ Dump display control
  BLNDDDSP ...... ........ Dump display build output
   BLNDDCDB ..... ........ Common data build routine
    BLNDDLSZ .... ........ Get line size for hex dump output
    BLNDDVRF .... ........ Validate input request
     BLNCMXPF ... ........ Convert hex to EBCDIC
     BLNDDBAS ... ........ Get base address for request
      BLNCMPRS .. ........ Parse base field
      BLNDDSYM .. ........ Scan symbol table
      BLNCMHEX .. ........ Validate and convert base field to hex
      BLNCMXPF .. ........ Convert hex base to EBCDIC
     BLNDDQAL ... ........ Get QUAL field for request '
      BLNCMPRS .. ........ Parse QUAL field
      BLNCMHEX .. ........ Validate and convert QUAL field to hex
      BLNCMXPF .. ........ Convert hex QUAL to EBCDIC
     BLNDDRDF ... ........ Check for locator
      BLNASLOC .. ........ Get section 6 locator
      BLNCMERM .. ........ Put out messages if any
      BLNCMXPF .. ........ Convert hex base to EBCDIC
      BLNLBVZA .. ........ Verify zero address in LBD
    BLNDDREQ .... ........ Get raw dump data
     BLNADIFR ... ........ Make dump data request
     BLNCMERM ... ........ Put out messages if any
    BLNDDFMT .... ........ Format buffer for output
     BLNASLOC ... ........ Get section 6 locator
     BLNCMERM ... ........ Put out messages if any
     BLNLBVER ... ........ Validate LBD
    BLNDDBTE .... ........ Format text extension
     BLNCMBUT ... ........ Break up text by lines
     BLNDDNLI ... ........ Get pointer to new line
      BLNDDNBU .. ........ Get new buffer and chain it
    BLNDDHDO .... ........ Format for hexadecimal output
     BLNDDHBL ... ........ Format dump data in hex
      BLNCMXPF .. ........ Convert offset or address to EBCDIC
      BLNDDBOD .. ........ Convert dump data and storage key -
       BLNCMXPF . ........ Convert hexadecimal to EBCDIC
      BLNDDEBC .. ........ Convert dump data to EBCDIC
      BLNDDMDT .. ........ Get more raw dump data
       BLNDDREQ . ........ Get raw dump data
        BLNADIFR  ........ Make dump data request
        BLNCMERM  ........ Put out messages if any
       BLNDDQRY . ........ Get next available address in dump
        BLNADIFR  ........ Make dump data request
        BLNCMERM  ........ Put out messages if any
       BLNCMXPF . ........ Convert from range to EBCDIC
       BLNCMMOB . ........ Put message into output buffer
     BLNDDNBU ... ........ Get new buffer and chain it
     BLNDDNLI ... ........ Get pointer to new line
      BLNDDNBU .. ........ Get new buffer and chain it
     BLNCMXPF ... ........ Convert base to EBCDIC
     BLNCMMOB ... ........ Put message into output buffer
```

```
   BLNDDFDO .... ........ Format dump data per field
    BLNDDFDZ ... ........ Get raw dump data to format
      BLNDDREQ . ........ Get raw dump data
       BLNADIFR  ........ Make dump data request
       BLNCMERM  ........ Put out messages if any
    BLNDDNLI ... ........ Get pointer to new line
     BLNDDNBU .. ........ Get new buffer and chain it
    BLNCMXPF ... ........ Convert from range to EBCDIC
    BLNDVFFR ... ........ Format dump data per data type
     BLNCMXPF .. ........ Convert hexadecimal to EBCDIC
     BLNCMXBI .. ........ Convert to binary
     BLNCMXEB .. ........ Convert binary to EBCDIC
     BLNCMXDC .. ........ Convert to decimal characters
   BLNCMFBC .... ........ Free output buffer chain
```

Process Dump Display Print Format Request:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNFNDDC ....... ........ Dump display control
  BLNDDPRT ...... ........ Print requested dump data
   BLNDDPOP ..... ........ Validate PRINT command operands
    BLNCMPRS .... ........ Parse command
    BLNCMHEX .... ........ Convert fixed binary to hex
   BLNDDFPT ..... ........ Process PRINT FORMAT command
    BLNDVKCB .... ........ Save control blocks
    BLNSMPCD .... ........ Print data lines for overview
    BLNDDFPS .... ........ Print each entry from overview list
     BLNFN MC ... ........ Select analysis summary item
     BLNSMPCD ... ........ Print data lines for item selected
     BLNDDFPL ... ........ Print locator dump data
      BLNDDFPD .. ........ Print dump data for each locator
       BLNFNSMC . ........ Select analysis summary locator item
       BLNDDCDB . ........ Common data build routine
       BLNDDFPP . ........ Print data buffer
        BLNCMMOB  ........ Put message into output buffer
        BLNCMPRT  ........ Print data buffers
        BLNCMFBC  ........ Free data buffers
    BLNDVRCB .... ........ Restore control blocks
```

Process Dump Display Print Data Request:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNFNDDC ....... ........ Dump display control
  BLNDDPRT ...... ........ Print requested dump data
   BLNDDPOP ..... ........ Validate PRINT command operands
    BLNCMPRS .... ........ Parse command
    BLNCMHEX .... ........ Convert fixed binary to hex
   BLNDDDPT ..... ........ Process PRINT DATA command
    BLNDDCDB .... ........ Common data build routine
    BLNDDFPP .... ........ Print data buffer
     BLNCMMOB ... ........ Put message into output buffer
     BLNCMPRT ... ........ Print data buffers
     BLNCMFBC ... ........ Free data buffers
    BLNDDRPT .... ........ Print remainder of block over 4K
    BLNCMPRT .... ........ Print requested data
```

Process Dump Display Print Range Request:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNFNDDC ....... ........ Dump display control
 BLNDDPRT ....... ........ Print requested dump data
   BLNDDPOP ..... ........ Validate PRINT command operands
    BLNCMPRS .... ........ Parse command
    BLNCMHEX .... ........ Convert fixed binary to hex
   BLNDDRPT ..... ........ Process PRINT RANGE command
    BLNCMXPF .... ........ Convert from/to range to EBCDIC
    BLNDDLSZ .... ........ Get line size for hex dump output
    BLNDDREQ .... ........ Get raw dump data
     BLNADIFR ... ........ Dump access interface
     BLNCMERM ... ........ Put out message if necessary
    BLNDDFMT .... ........ Format buffer FOR output
    BLNCMPRT .... ........ Print dump data
    BLNCMFBC .... ........ Free buffer
   BLNDDQRY ..... ........ Get next available address in dump
    BLNADIFR .... ........ Dump access interface
    BLNCMERM .... ........ Put out message if necessary
   BLNCMMOB ..... ........ Put message to buffer
  BLNCMMOB ...... ........ Put message to buffer
  BLNCMPRT ...... ........ Print message line
```

Terminate Dump Display:

```
BLNFNDVS ........ ........ Dump viewing selection
 BLNFNDDC ....... ........ Dump display control
  BLNDDTRM ...... ........ Terminate dump display
```

MODULE FLOW FOR LOCATOR BLOCKS

The following lists show the module flow for locator block functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Build Blocks for Simple Locator:

```
BLNLBBLR ........ ........ Build remaining blocks for locator
 BLNLBSIM ....... ........ Build simple locator entry
  BLNLBBCI ...... ........ Build LOCB entries for block (see below)
```

Build Blocks for Array of Locators:

```
BLNLBBLR ........ ........ Build remaining blocks for locator
 BLNLBARR ....... ........ Build array rules for locator
  BLNLBCNT ...... ........ Read count field from dump
   BLNADIFR ..... ........ Dump access interface
   BLNCMERM ..... ........ Handle external messages if any
  BLNLBCHR ...... ........ Build chain rules for locator
   BLNADIFR ..... ........ Dump access interface
   BLNCMERM ..... ........ Handle external messages if any
 BLNLBARY ....... ........ Build array entry
  BLNLBARC ...... ........ Build locator blocks for contiguous array
   BLNLBBCI ..... ........ Build LOCB entries for one array block
  BLNLBARB ...... ........ Build locator blocks for array blocks
   BLNLBSIM ..... ........ Build simple locator entry
   BLNLBARC ..... ........ Build locator blocks for contiguous array
    BLNLBBCI .... ........ Build LOCB entries for block (see below)
   BLNLBCHN ..... ........ Build locator blocks for chained array
                           (calling sequence of BLNLBCHN depends on
                           end condition and type of link - see below)
```

Build Blocks for Chained Locators:

```
BLNLBBLR ........ ........ Build remaining blocks for locator
 BLNLBCHR ....... ........ Build chain rules for locator
 BLNLBCHN ....... ........ Build locator blocks for chain
  BLNLBCAL ...... ........ Build chain for address list end 'L'
  BLNLBCAI ...... ........ Build chain for address list end 'I'
  BLNLBCAE ...... ........ Build chain for address list end 'E'
  BLNLBCAC ...... ........ Build chain for address list end 'C'
  BLNLBCHL ...... ........ Build chain for linked list end 'L'
  BLNLBCHI ...... ........ Build chain for linked list end 'I'
  BLNLBCHE ...... ........ Build chain for linked list end 'E'
  BLNLBCHW ...... ........ Build chain for linked list end 'W'
  BLNLBCHC ...... ........ Build chain for linked list end 'C'
```

Build Blocks for Chained Locators Linked by Address List

With End Condition of 'L':

```
BLNLBCAL ........ ........ Build chain for address list end 'L'
 BLNLBLAL ....... ........ Load address list into storage
  BLNADIFR ...... ........ Get address list data area
    BLNCMERM .... ........ Handle external messages if any
   BLNLBBCI ..... ........ Build LOCB entries for block (see below)
   BLNLBCEF ..... ........ Test for end condition in address list
```

Build Blocks for Chained Locators Linked by Address List

With End Condition of 'I':

```
BLNLBCAI ........ ........ Build chain for address list end 'I'
 BLNLBLAL ....... ........ Load address list into storage
  BLNADIFR ...... ........ Get address list data area
  BLNCMERM ...... ........ Handle external messages if any
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
```

Build Blocks for Chained Locators Linked by Address List

With End Condition of 'E':

```
BLNLBCAE ........ ........ Build chain for address list end 'E'
 BLNLBLAL ....... ........ Load address list into storage
  BLNADIFR ...... ........ Get address list data area
  BLNCMERM ...... ........ Handle external messages if any
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
 BLNLBCEF ....... ........ Test for end condition in data field
```

Build Blocks for Chained Locators Linked by Address List

With End Condition of 'C':

```
BLNLBCAC ........ ........ Build chain for address list end 'C'
 BLNLBCNT ....... ........ Get count of entries in address list
  BLNADIFR ...... ........ Get count data from LBD in dump
  BLNCMERM ...... ........ Handle external messages if any
 BLNLBLAL ....... ........ Load address list into storage
  BLNADIFR ...... ........ Get address list data area
  BLNCMERM ...... ........ Handle external messages if any
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
```

Build Blocks for Chained Locators Linked by Linked List

With End Condition of 'L':

```
BLNLBCHL ........ ........ Build chain for linked list end 'L'
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
   BLNLBCEF ..... ........ Test for end condition in data field
    BLNLBCLN .... ........ Read link address from control block
     BLNADIFR ... ........ Read control block from dump
     BLNCMERM ... ........ Handle external messages if any
```

Build Blocks for Chained Locators Linked by Linked List

With End Condition of 'I':

```
BLNLBCHI ........ ........ Build chain for linked list end 'I'
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
 BLNLBCLN ....... ........ Read link address from control block
  BLNADIFR ...... ........ Read control block from dump
  BLNCMERM ...... ........ Handle external messages if any
```

Build Blocks for Chained Locators Linked by Linked List

With End Condition of 'E':

```
BLNLBCHE ........ ........ Build chain for linked list end 'E'
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
 BLNLBCEF ....... ........ Test for end condition in data field
 BLNLBCLN ....... ........ Read link address from control block
  BLNADIFR ...... ........ Read control block from dump
  BLNCMERM ...... ........ Handle external messages if any
```

Build Blocks for Chained Locators Linked by Linked List

With End Condition of 'W':

```
BLNLBCHW ........ ........ Build chain for linked list end 'W'
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
 BLNLBCLN ....... ........ Read link address from control block
  BLNADIFR ...... ........ Read control block from dump
  BLNCMERM ...... ........ Handle external messages if any
```

Build Blocks for Chained Locators Linked by Linked List

With End Condition of 'C':

```
BLNLBCHC ........ ........ Build chain for linked list end 'C'
 BLNLBCNT ....... ........ Get count field from LBD
  BLNADIFR ...... ........ Get data from dump
  BLNCMERM ...... ........ Put external messages on queue
 BLNLBBCI ....... ........ Build LOCB entries for block (see below)
 BLNLBCLN ....... ........ Read link address from control block
  BLNADIFR ...... ........ Read control block from dump
  BLNCMERM ...... ........ Handle external messages if any
```

Build LOCB Entry for Block:

```
BLNLBBCI ........ ........ Build LOCB entry for block
 BLNLBRCB ....... ........ Read end condition from control block
  BLNADIFR ...... ........ Dump access interface
  BLNCMERM ...... ........ Handle external messages if any
 BLNLBCEF ....... ........ Check for end condition in field
 BLNLBCMA ....... ........ Check for duplicate locator
 BLNADIFR ....... ........ Get address of next entry in linkage
 BLNSMLKC ....... ........ Process lower level of linkago
```

MODULE FLOW FOR DUMP OFFLOAD

The following lists show the module flow for dump offload functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Invoke Dump Offload:

```
BLNOFFLD ........ ........ Invoke offload function
 BLNDMRDF ....... ........ Read dump management file
 BLNDMCHK ....... ........ Check dump management table
 BLNSUBMT ....... ........ Submit job to batch
 BLNOFOUT ....... ........ Write dump record on tape
  BLNADIFR ...... ........ Call dump access to read record from library
  BLNCMERM ...... ........ Issue dump access messages
  BLNADIFR ...... ........ Call dump access to terminate dump
  BLNCMERM ...... ........ Issue dump access messages
 BLNOFUDM ....... ........ Update dump management file with volid
  BLNDMRDF ...... ........ Read dump management file
  BLNDMCHK ...... ........ Check dump management table
  BLNDMWDF ...... ........ Write dump managment file
 BLNADIFR ....... ........ Call dump access to delete dump
 BLNCMERM ....... ........ Issue dump access messages
```

MODULE FLOW FOR DUMP ONLOAD

The following lists show the module flow for dump onload functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Invoke Dump Onload:

```
BLNONLOD ........ ........ Invoke onload function
 BLNONINT ....... ........ Initialize onload function
  BLNDMRDF ...... ........ Read dump management file
  BLNDMCHK ...... ........ Check dump management table
  BLNSUBMT ...... ........ Submit job to batch
 BLNONMUD ....... ........ Process multiple dumps
 BLNONDDT ....... ........ Determine dump type
  BLNONVSE ...... ........ Handle DOS/VSE dump tape
 BLNONTRW ....... ........ Read dump tape record
  BLNADIFR ...... ........ Call dump access to write record to library
  BLNCMERM ...... ........ Issue dump access messages
 BLNONEND ....... ........ Terminate onload function
  BLNADIFR ...... ........ Call dump access to terminate dump
  BLNCMERM ...... ........ Issue dump access messages
  BLNDMRDF ...... ........ Read dump management file
  BLNDMCHK ...... ........ Check dump management table
 BLNDMAD1 ....... ........ Update dump management table
 BLNDMSRT ....... ........ Sort dump management table
 BLNDMWDF ....... ........ Write dump managment file
```

MODULE FLOW FOR RAS

The following lists show the module flow for RAS functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Process User Exit STAE:

```
BLNRAUSE ........ ........ Entry point
 BLNRASRC ....... ........ Symptom record build routine
 BLNRAS6C ....... ........ Section 6 build routine
  BLNRACCB ...... ........ Section 6 record build for CCB control block
   BLNRALBD ..... ........ Build LBD in section 6 buffer
  BLNRADVB ...... ........ Section 6 record build for DVCB control block
   BLNRALBD ..... ........ Build LBD in section 6 buffer
  BLNRADMB ...... ........ Section 6 record build for DMCB control block
   BLNRALBD ..... ........ Build LBD in section 6 buffer
  BLNRADSB ...... ........ Section 6 record build for DSCB control block
   BLNRALBD ..... ........ Build LBD in section 6 buffer
  BLNRABAB ...... ........ Section 6 record build for BACB control block
   BLNRALBD ..... ........ Build LBD in section 6 buffer
 BLNIDUMP ....... ........ Create dump in system data set
```

Invoke STAE Exit:

```
BLNRASEC ........ ........ Entry point
 BLNRAEIN ....... ........ Build BLNRACB and save termination data
 BLNRAS4B ....... ........ Section 4 build routine
 BLNRAS5B ....... ........ Section 5 build routine
 BLXSYMPP ....... ........ Symptom record build routine
 BLNRAS6B ....... ........ Section 6 build routine
  BLNRACCB ...... ........ Section 6 record build for CCB control block
   BLNRALBD ..... ........ Build LBD element in section 6 buffer
    BLNRAS6S .... ........ Get storage for section 6 buffer
  BLNRADVB ...... ........ Section 6 record build for DVCB control block
   BLNRALBD ..... ........ Build LBD element in section 6 buffer
    BLNRAS6S .... ........ Get storage for section 6 buffer
  BLNRADVT ...... ........ Section 6 record build for DVVT control block
   BLNRALBD ..... ........ Build LBD element in section 6 buffer
    BLNRAS6S .... ........ Get storage for section 6 buffer
  BLNRASFB ...... ........ Section 6 record build for SFCB control block
   BLNRALBD ..... ........ Build LBD element in section 6 buffer
     BLNRAS6S ... ........ Get storage for section 6 buffer
   BLNRADMB ..... ........ Section 6 record build for DMCB control block
    BLNRALBD .... ........ Build LBD element in section 6 buffer
     BLNRAS6S ... ........ Get storage for section 6 buffer
   BLNRADSB ..... ........ Section 6 record build for DSCB control block
    BLNRALBD .... ........ Build LBD element in section 6 buffer
     BLNRAS6S ... ........ Get storage for section 6 buffer
   BLNRABAB ..... ........ Section 6 record build for BACB control block
    BLNRALBD .... ........ Build LBD element in section 6 buffer
     BLNRAS6S ... ........ Get storage for section 6 buffer
   BLNRAASB ..... ........ Section 6 record build for ASRCB control bloc
    BLNRALBD .... ........ Build LBD element in section 6 buffer
     BLNRAS6S ... ........ Get storage for section 6 buffer
   BLXSIDMP ..... ........ Create dump in system data set
```

MODULE FLOW FOR ANALYSIS SUMMARY

The following lists show the module flow for analysis summary functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Initialize Analysis Summary:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMINT ....... ........ Analysis summary initialization
  BLNSMBML ...... ........ Build master list for analysis summary
   BLNASLST ..... ........ Get list of section 6 entries from symptom    record
                           access
   BLNCMERM ..... ........ Put external messages on queue
   BLNSMBMA ..... ........ Set up area for the section 6 entries list
    BLNSMBME .... ........ Set up the elements of the summary lists
     BLNSMBMN ... ........ Set up the entry for the master list
     BLNSMBLN ... ........ Set up the entry for the locator list
```

Request Analysis Summary Overview List:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMDSP ....... ........ Analysis summary display function
  BLNSMCML ...... ........ Construct overview output list
   BLNSMFES ..... ........ Free storage for previous display
    BLNSMFXD .... ........ Free storage for hex entry
    BLNSMFTD .... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
    BLNSMFCD .... ........ Free storage for linkage entry
     BLNSMFC1 ... ........ Free storage for LOCH and LOCB
   BLNCMALN ..... ........ Adjust line numbers for display
   BLNSMCMO ..... ........ Construct lines for overview output list
```

Request Analysis Summary Locator List:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMDSP ....... ........ Analysis summary display function
  BLNSMSDC ...... ........ Control build for specific data entry
   BLNSMFES ..... ........ Free storage for previous entry
    BLNSMFXD .... ........ Free storage for hex entry
    BLNSMFTD .... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
    BLNSMFCD .... ........ Free storage for linkage entry
     BLNSMFC1 ... ........ Free storage for LOCH and LOCB
   BLNSMBLL ..... ........ Build locators list data for output
    BLNCMALN .... ........ Adjust line numbers
    BLNSMBLE .... ........ Build locators list entries
     BLNASLOC ... ........ Get LBD for locator entry
     BLNCMERM ... ........ Put external messages on queue
     BLNSMBLF ... ........ Prepare to build locator data for entry
      BLNLBVZA .. ........ Validate address for LBD
      BLNSMBLH .. ........ Build locator header
       BLNSMBLB . ........ Build locator blocks for header
        BLNLBVER  ........ Verify LBD and extensions
        BLNLBBLR  ........ Build remaining blocks for locators
    BLNSMBLO .... ........ Build locators output list data
     BLNSMLSL ... ........ Get starting point for locator list
     BLNSMBOL ... ........ Build locator output line
      BLNCMXPF .. ........ Convert hex to character
      BLNSMBOK .. ........ Build keyfield data for locator
       BLNADIFR . ........ Get keyfield data from dump
       BLNDVFFR . ........ Format data field for locator output
```

Request Analysis Routines Display Output:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMDSP ....... ........ Analysis summary display function
  BLNSMSDC ...... ........ Control build for specific data entry
   BLNSMFES ..... ........ Free storage for previous entry
    BLNSMFXD .... ........ Free storage for hex entry
    BLNSMFTD .... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
    BLNSMFCD .... ........ Free storage for linkage entry
     BLNSMFC1 ... ........ Free storage for LOCH and LOCB
   BLNSMBSI ..... ........ Build specific entry item for output
    BLNSMBPE .... ........ Build pseudo section 6 entry for output
     BLNCMALN ... ........ Adjust line numbers
     BLNSMPSO ... ........ Build pseudo section 6 output
```

Request Analysis Summary Text Output:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMDSP ....... ........ Analysis summary display function
  BLNSMSDC ...... ........ Control build for specific data entry
   BLNSMFES ..... ........ Free storage for previous entry
    BLNSMFXD .... ........ Free storage for hex entry
    BLNSMFTD .... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
    BLNSMFCD .... ........ Free storage for linkage entry
     BLNSMFC1 ... ........ Free storage for I CH and LOCB
   BLNSMBSI ..... ........ Build specific entry item for output
    BLNSMBTX .... ........ Build text entry for output
     BLNSMTXA ... ........ Build text data from LBD entry
      BLNASLOC .. ........ Get LBD entry for text DATA
      BLNCMERM .. ........ Put external messages on queue
      BLNLBVER .. ........ Verify LBD read from symptom record
     BLNCMALN ... ........ Adjust line numbers
     BLNSMTXO ... ........ Build text data for output
```

Request Analysis Summary Hex Output:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMDSP ....... ........ Analysis summary display function
  BLNSMSDC ...... ........ Control build for specific data entry
   BLNSMFES ............. Free storage for previous entry
    BLNSMFXD .... ........ Free storage for hex entry
    BLNSMFTD .... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
    BLNSMFCD .... ........ Free storage for linkage entry
     BLNSMFC1 ... ........ Free storage for LOCH and LOCB
   BLNSMBSI ..... ........ Build specific entry item for output
    BLNSMBHX .... ........ Build hex entry output
     BLNSMHXA ... ........ Build data area for hex data
      BLNASLOC .. ........ Get LBD for hex data
      BLNCMERM .. ........ Put external messages on queue
      BLNSMPSW .. ........ Build PSW data output
      BLNSMFRE .. ........ Build floating point data output
      BLNSMGRE .. ........ Build general register data output
      BLNSMOCL .. ........ Build hex data output
     BLNCMALN ... ........ Adjust line numbers
     BLNSMHXO ... ........ Build hex output buffer
```

Request Analysis Summary Linkage Listing:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMDSP ....... ........ Analysis summary display function
  BLNSMSDC ...... ........ Control build for specific data entry
   BLNSMFES ..... ........ Free storage for previous entry
    BLNSMFXD .... ........ Free storage for hex entry
    BLNSMFTD .... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
    BLNSMFCD .... ........ Free storage for linkage entry
     BLNSMFC1 ... ........ Free storage for LOCH and LOCB
   BLNSMBSI ..... ........ Build specific entry item for output
    BLNSMBLK .... ........ Build linkage descriptor output
     BLNSMLKL ... ........ Build linkage descriptor data list
      BLNASLOC .. ........ Get LBD from symptom record access
      BLNCMERM .. ........ Put external messages on queue
      BLNLBVER .. ........ Validate linkage descriptor
      BLNSMLKI .. ........ Build linkage data Control blocks
       BLNSMLKG . ........ Set up linkage data Control blocks
        BLNASLOC  ........ Get LBD from symptom record access
        BLNCMERM  ........ Put external messages on queue
        BLNLBVZA  ........ Validate base address of LBD
       BLNSMLKB .. ........ Build linkage locator blocks
        BLNSMLKD . ........ Build locator for data not available
        BLNSMLKN . ........ Get locators for links
         BLNASLOC  ........ Get LBD from symptom record access
         BLNCMERM  ........ Put external messages on queue
         BLNSMLKN  ........ Get locators for links
        BLNSMLKC . ........ Get locators for links
         BLNSMBLB  ........ Build locator blocks for locator list
          BLNLBVER ........ Validate linkage descriptor
          BLNLBBLR ........ Build remaining blocks for locator
         BLNADIFR  ........ Get address of next link entry
         BLNSMLKC  ........ Get locators for links
     BLNCMALN ... ........ Adjust line numbers
     BLNSMLKO ... ........ Build linkage descriptor output data
      BLNSMLSP .. ........ Locate starting point in linkage
      BLNSMLKP .. ........ Process linkage chain
       BLNSMBOL . ........ Build output line for entry
        BLNCMXPF  ........ Convert hex to character
        BLNSMBOK  ........ Build keyfield data for locator
```

Select Analysis Summary Item:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMSEL ....... ........ Analysis summary select
  BLNSMSEM ...... ........ Select entry from master list
  BLNSMSNI ...... ........ Select item from named entry
   BLNSMSNL ..... ........ Select item from locators list
    BLNSMLSL .... ........ Get locator list item per line number
   BLNSMSNC ..... ........ Select item from linkage descriptor
    BLNSMLSP .... ........ Get linkage list item per line number
```

Find String in Analysis Summary Item:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMFND ....... ........ Analysis summary find
  BLNCMPRS ...... ........ Parse command argument
  BLNSMFN1 ...... ........ Search for data
   BLNSMDSP ..... ........ Get data display for search
   BLNCMFBC ..... ........ Free buffer chain
  BLNSMDSP ...... ........ Display data for argument found
```

Print Data for Analysis Summary Item:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMPRT ....... ........ Analysis summary print
  BLNCMPRS ...... ........ Parse command argument
  BLNDVKCB ...... ........ Save analysis summary environment
  BLNSMPCD ...... ........ Print current data item
   BLNSMDSP ..... ........ Display current data item
   BLNSMPLA ..... ........ Adjust print line data as needed
   BLNSMPHP ..... ........ Put in header line for data as needed
   BLNCMPRT ..... ........ Print buffer data
   BLNCMFBC ..... ........ Free buffer chain
  BLNDVRCB ...... ........ Restore analysis summary environment
```

Terminate Analysis Summary Function:

```
BLNFNSMC ........ ........ Analysis summary function control
 BLNSMTRM ....... ........ Analysis summary termination
  BLNSMFES ...... ........ Free storage for previous entry
   BLNSMFXD ..... ........ Free storage for hex entry
   BLNSMFTD ..... ........ Free storage for text entry
    BLNSMFLD .... ........ Free storage for locator entry
   BLNSMFCD ..... ........ Free storage for linkage entry
    BLNSMFC1 .... ........ Free storage for LOCH and LOCB
```

INFO/SERVICES MODULE FLOWS

This chapter contains information that can help in diagnosing problems within Information/services (BLX environment). Information/services provides a comprehensive set of SCP-independent operating system supervisor and data access services. The functions are modeled upon those provided by the MVS operating system. They are invoked via an SCP-independent macro interface. The control flow structures which follow are devided into three major groups:

- BLX environment initialization and termination.
  This function set creates and terminates the operating environment upon which the BLX services macros are dependent.
- System Supervisor services.
  This function set provides SCP-independent services normally provided by an operating system supervisor.
  - BLXDEQ - Release serially reusable resource
  - BLXENQ - Request serially reusable resource
  - BLXESTAE - Extended specify task abnormal exit
  - BLXEXTRT - Extract control block information
  - BLXFREEM - Free virtual storage
  - BLXGETM - Allocate virtual storage
  - BLXLOAD - Bring a load module into virtual storage
  - BLXLOCK - Manipulate a lock cell
  - BLXMACB - Manage anchor control blocks
  - BLXMVT - Manage vector tables
  - BLXSNAP - Dump virtual storage and continue
  - BLXSTIME - Set interval timer
  - BLXTIME - Provide time and date
  - BLXWTO - Write to operator
  - BLXWTOR - Write to operator with reply
- Data set access services.
  This function set provides SCP-independent access to auxiliary storage data sets.
  - BLXALLOC - Allocate a physical resource
  - BLXFREE - Free a physical resource
  - BLXOPEN - Open a logical resource
  - BLXCLOSE - Close a logical resource
  - BLXFRDS - Close and free resources
  - BLXGET - Retrieve record from a logical resource
  - BLXPUT - Write record to a logical resource

- Message text build and routing services.
  This function provides a means to implement the requirement that message text be separate from program code, a flexible message construction interface that permits complex text and data inserts, and an SCP-independent message routing interface.
  - BLXDMSGG - Message CSECT generation macro
  - BLXDMSG - Message text build and/or route

MODULE FLOW FOR BLX INITIALIZATION/TERMINATION FUNCTION

The initialization function is contained in the phase BLXSINIT. The BLXINIT macro generates code to load the BLXSINIT phase into virtual storage. It remains in storage only until initialization is complete. The BLX tertmination function is contained in the phase BLXSTERM.

BLX Initialization:

```
BLXSINIT ........ ........ Initialization load 0
 BLXSINI1 ....... ........ Initialization load 1
```

BLX Termination:

```
BLXSTERM ........ ........ BLX Termination
```

MODULE FLOWS FOR BLX SUPERVISOR FUNCTIONS.

All BLX supervisor functions are contained in the phase named BLXSSERV. It is loaded during BLX initialization and remains in storage until termination of BLX services is requested.

Release Serially Reusable Resource Control:

```
BLXSRTRO ........ ........ BLX service router
 BLXSDEQO ....... ........ Release reusable resource
```

Request Serially Reusable Resource Control:

```
BLXSRTRO ........ ........ BLX service router
 BLXSENQO ....... ........ Get control reusable resource
```

Extended Specify Task Abnormal Exit:

```
BLXSRTRO ........ ........ BLX service router
 BLXSESTO ....... ........ BLX ESTAE processor
  BLXSCBCR ...... ........ Create SCB
  BLXSCBDL ...... ........ Delete SCB
```

Extract Control Block Information:

```
BLXSRTRO ........ ........ BLX service router
 BLXSEXTO ....... ........ Manage extraction requests
```

Explicit Storage Manager - Release:

```
BLXSRTRO ........ ........ BLX service router
 BLXSESMR ....... ........ Manage explicit storage
  BLXSCSMR ...... ........ Storage cell manager
   BLXSFROO ..... ........ Release virtual storage
```

Explicit Storage Manager - Allocate:

```
BLXSRTRO ........ ........ BLX service router
 BLXSESMR ....... ........ Manage explicit storage
  BLXSCSMR ...... ........ Storage cell manager
   BLXSGTOO ..... ........ Acquire virtual storage
```

Bring a Load Module into Virtual Storage:

```
BLXSRTRO ........ ........ BLX service router
 BLXSLDOO ....... ........ Module load
```

Manipulate a Lock Cell:

```
BLXSRTRO ........ ........ BLX service router
 BLXSLKOO ....... ........ Lock service
```

Manage Anchor Control Blocks:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSANC0 ....... ........ Manage anchor blocks
```

Manage Vector Tables:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSVT00 ....... ........ Manage vector tables
```

Dump Virtual Storage and Continue:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSNAPO ....... ........ Take snapshot of storage
  BLXSNAP1 ...... ........ Snap format routine
```

Set Interval Timer:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSTMER ....... ........ Issue interval timer
```

Provide Time and Data:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSTM00 ....... ........ Return current time and date
```

Write to Operator:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSWTOR ....... ........ Write to operator
```

Write to Operator with Reply:

```
BLXSRTR0 ........ ........ BLX service router
 BLXSWTOR ....... ........ Write to operator
```

MODULE FLOWS FOR BLX DATA ACCESS SERVICES

All BLX Data Access Services functions are contained in the phase BLXCDAS0. It is loaded during BLX initialization and remains in storage until BLX termination is requested.

Allocate a Physical Resource:

```
BLXCALOC ........ ........ Data set allocation
 BLXCAINT ....... ........ Data set allocation initialization
 BLXCABLD ....... ........ Build dynamic text units
 BLXCAMER ....... ........ Merge model overrides
 BLXCALOD ....... ........ Obtain device type
  BLXCALOG ...... ........ Read data set label
  BLXCAMSG ...... ........ Allocation error message interface
```

Free a Physical Resource:

```
BLXCFREE ........ ........ Deallocate data set
 BLXCFREA ....... ........ Deallocate data set
 BLXCMSGS ....... ........ Issue error messages
```

Close a Data Set:

```
BLXCRQST ........ ........ DAS request interface
 BLXCCLSE ....... ........ Close data set
  BLXDDEQ0 ...... ........ Dequeue function
 BLXCANLE ....... ........ Issue error messages
```

Close and Free Resources for Current Task:

```
BLXCRQST ........ ........ DAS request interface
 BLXCFRDS ....... ........ Close and free resources
 BLXCANLE ....... ........ Issue error messages
```

Retrieve Record from Logical Resource:

```
BLXCRQST ........ ........ DAS request interface
 BLXCGETT ....... ........ Data set read
 BLXCANLE ....... ........ Issue error messages
```

Write Record to Logical Resource:

```
BLXCRQST ........ ........ DAS request interface
 BLXCPUTT ....... ........ Data set record write
 BLXCANLE ....... ........ Issue error messages
```

MODULE FLOWS FOR BLX MESSAGE SERVICES

BLX Message Services:

```
BLXSRTRO ........ ........ BLX service router
 BLXDMSGO ....... ........ Message router
```

DUMP ACCESS MODULE FLOW

This chapter contains information that can help in diagnosing problems within Dump Access:

- Module flow for dump availability
- Module flow for dump deletion
- Module flow for dump access driver
- Module flow for dump initialization
- Module flow for dump query
- Module flow for dump read
- Module flow for dump termination
- Module flow for dump write

MODULE FLOW FOR DUMP AVAILABILITY

This section shows the module flow for dump availability functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Dump Availability:

```
IJBXAAVL ........ ........ Dump access availability request entry point
 IJBXAGVS ....... ........ Get storage for shared buffer pool
 IJBXALNB ....... ........ Validity check library and dump name
 IJBXALCN ....... ........ Connect to library
 IJBXAGVS ....... ........ Get storage to process availability request
 IJBXABLD ....... ........ Use BLDL macro to check for existing member
 IJBXAMDC ....... ........ Disconnect from dump member
 IJBXALDC ....... ........ Disconnect from library
 IJBXAFVS ....... ........ Free storage used in process
```

MODULE FLOW FOR DUMP DELETE

This section shows the module flow for dump delete functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Dump Delete:

```
IJBXADEL ........ ........ Dump access delete request entry point
 IJBXAINT ....... ........ If dump access has not been initialized
                           call initialization process
 IJBXALNB ....... ........ Validity check library and dump name
 IJBXALCN ....... ........ Connect to library
 IJBXAMCN ....... ........ Connect to dump member
 IJBXAMDC ....... ........ Disconnect from dump member
 IJBXASTW ....... ........ Use STOW with delete option to delete member
 IJBXALDC ....... ........ Disconnect from library
 IJBXAFVS ....... ........ Free storage used in process
 IJBXAFND ....... ........ Check if mapping member exists in library
 IJBXAMDC ....... ........ Disconnect from mapping member
 IJBXASTW ....... ........ Use STOW with delete option to delete member
 IJBXALDC ....... ........ Disconnect from library
 IJBXAFVS ....... ........ Free storage used in process
 IJBXAFND ....... ........ Check if extension member exists in library
 IJBXAMDC ....... ........ Disconnect from extension member
 IJBXASTW ....... ........ Use STOW with delete option to delete member
 IJBXALDC ....... ........ Disconnect from library
 IJBXAFVS ....... ........ Free storage used in process
```

MODULE FLOW FOR DUMP ACCESS DRIVER

This section shows the module flow for dump access driver functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Dump Access Driver:

```
IJBXACTL ........ ........ Dump access entry point
 IJBXATRM ....... ........ Call termination processor if dump access has been
                           initialized on prior entry for certain request
 IJBXAGVS ....... ........ Get storage for DAPL control block
 IJBXAQDA ....... ........ Process query available dump address request
 IJBXARWR ....... ........ Process read request
 IJBXARWR ....... ........ Process write request
 IJBXADEL ....... ........ Process dump deletion request
 IJBXAAVL ....... ........ Process dump availability request
 IJBXATRM ....... ........ Process dump access termination request
 IJBXAINT ....... ........ Process initialization request
 IJBXAGMA ....... ........ Get maximum address of requested mode
```

MODULE FLOW FOR DUMP INITIALIZATION

This section shows the module flow for dump access driver functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Dump Initialization:

```
IJBXAINT ........ ........ Dump access initialization entry point
 IJBXAINS ....... ........ Get storage for feedback buffer, data buffers,
                           I/O buffers, private buffer pool and
                           shared buffer pool
  IJBXAGVS ...... ........ Get storage routine
 IJBXAIND ....... ........ Initialize dump member routine
  IJBXALNB ...... ........ Validity check input dump name
  IJBXALCN ...... ........ Library connect routine
   IJBXAGVS ..... ........ Get storage for LAMB, ACCB and LIBINFO block
   IJBXAFVS ..... ........ Free storage if error on connect
  IJBXAMCN ...... ........ Dump member connect routine
   IJBXAFND ..... ........ Check if dump member is in library
    IJBXAGVS .... ........ Get storage for INLMLRPL, STOW header,
                           directory entry
    IJBXAMDC .... ........ Disconnect from dump member
    IJBXALDC .... ........ Disconnect from library
    IJBXAFVS .... ........ Free storage if error in find process
   IJBXAGVS ..... ........ Get storage routine
 IJBXAINM ....... ........ Initialize dump mapping member
  IJBXAFND ...... ........ Check if mapping member is in library
   IJBXAGVS ..... ........ Get storage for INLMLRPL, STOW header,
                           directory entry
```

```
  IJBXAMDC ..... ........ Disconnect from mapping member
  IJBXALDC ..... ........ Disconnect from library
  IJBXAFVS ..... ........ Free storage if error in find process
 IJBXARMP ...... ........ Read mapping member records and create map table
  IJBXAGET ..... ........ Read mapping member record
   IJBXANOT .... ........ Perform librarian NOTE macro
  IJBXAGVS ..... ........ Get storage to hold map table record
 IJBXAAMP ...... ........ Allocate mapping member and create mapping record
  IJBXAALC ..... ........ Allocate new member in currently connected library
  IJBXAGET ..... ........ Read dump member record
   IJBXANOT .... ........ Perform librarian NOTE macro
  IJBXACMP ..... ........ Create mapping record entry using dump record
                          just read from member
   IJBXAMES .... ........ Locate next slot in map table for new entry
    IJBXAGMP ... ........ Get new map table storage
     IJBXAGVS .. ........ Get storage for map table
    IJBXASME ... ........ Sort map table
  IJBXAGET ..... ........ Read mapping member record
  IJBXAWMR ..... ........ Write mapping record to newly allocated mapping member
   IJBXASME .... ........ Sort map table entries
   IJBXAPSQ .... ........ Write record to library member
   IJBXAFVS .... ........ Free storage for map table entries
  IJBXAMDC ..... ........ Disconnect mapping member from library
  IJBXASTW ..... ........ STOW member directory entry to create new
                          member in library
  IJBXALDC ..... ........ Disconnect from library
  IJBXAFVS ..... ........ Free storage obtained in allocate process
  IJBXAFND ..... ........ Reconnect to newly created member
   IJBXAGVS .... ........ Get storage for INLMLRPL, STOW header,
                          directory entry
   IJBXAMDC .... ........ Disconnect from new member
   IJBXALDC .... ........ Disconnect from library
   IJBXAFVS .... ........ Free storage if error in find process
IJBXAINE ....... ........ Initialize extension member
 IJBXAFND ...... ........ Check if extension member is in library
  IJBXAGVS ..... ........ Get storage for INLMLRPL, STOW header,
                          directory entry
  IJBXAMDC ..... ........ Disconnect from extension member
  IJBXALDC ..... ........ Disconnect from library
  IJBXAFVS ..... ........ Free storage if error in find process
 IJBXAGVS ...... ........ Get storage member table entry
 IJBXAGET ...... ........ Read extension member record
 IJBXACMP ...... ........ Create mapping record entry using extension
                          member record
  IJBXAMES ..... ........ Build map entry and call sort routine
   IJBXAGMP .... ........ Get new map table storage
    IJBXAGVS ... ........ Get storage for map table
   IJBXASME .... ........ Sort map table
```

```
IJBXAINF ....... ........ Initialize LBDF member
 IJBXAPNT ...... ........ Connect to system library and LBDF member
  IJBXAGVS ..... ........ Get storage for INLMLRPL, STOW header,
                          directory entry
  IJBXAMDC ..... ........ Disconnect from extension member
  IJBXALDC ..... ........ Disconnect from library
  IJBXAFVS ..... ........ Free storage if error in find process
 IJBXAGVS ...... ........ Get storage member table entry
 IJBXAGET ...... ........ Read LBDF member record
 IJBXACMP ...... ........ Create mapping record entry using LBDF membe  record
  IJBXAMES ..... ........ Build map entry and call sort routine
   IJBXAGMP .... ........ Get new map table storage
    IJBXAGVS ... ........ Get storage for map table
   IJBXASME .... ........ Sort map table
```

MODULE FLOW FOR DUMP QUERY

This section shows the module flow for dump query functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Query Dump Address:

```
IJBXAQDA ........ ........ dump access query entry point
                          Validity check mode to see if valid
 IJBXAINT ....... ........ If dump access has not been initialized then
                          call initialization process
```

Loop through map tables till next available dump address is found or highest dump address is found depending on request.

MODULE FLOW FOR DUMP READ

This section shows the module flow for dump read functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Dump Read:

```
 IJBXARWR ....... ........ Dump access read request entry point
  IJBXAINT ...... ........ Initialization process
  IJBXARDR ...... ........ Select appropriate map tables for searching
   IJBXARDD ..... ........ Read virtual or real dump data record
    IJBXARDA .... ........ Select virtual or real map tables
     IJBXAMAP ... ........ Search map tables for dump address requested
     IJBXARBU ... ........ Find dump record in storage or retrieve from library
      IJBXARBS .. ........ Search storage buffers to see if record
                           is in main storage
      IJBXAGAC .. ........ Read record from library
      IJBXARBM .. ........ Move dump record from I/O buffer to data
                           buffer and update DAPL with contents data
   IJBXARSP ..... ........ Read specific record in dump member. Loop through
                           map tables till record number is located
    IJBXARBU .... ........ Find dump record in storage or retrieve from library
     IJBXARBS ... ........ Searth storage buffers to see if record
                           is in main storage
     IJBXAGAC ... ........ Read record from library
     IJBXARBM ... ........ Move dump record from I/O buffer to data
                           Buffer and update DAPL with contents data
   IJBXARSR ..... ........ Read symptom record, section 6 record or header
                           record. Loop through map tables until record is located
    IJBXARBU .... ........ Find dump record in storage or retrieve from library
     IJBXARBS ... ........ Search storage buffers to see if record
                           is in main storage
     IJBXAGAC ... ........ Read record from library
     IJBXARBM ... ........ Move dump record from I/O buffer to data
                           buffer and update DAPL with contents data
```

MODULE FLOW FOR DUMP TERMINATION

This section shows the module flow for dump termination functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Dump Termination:

```
IJBXATRM ........ ........ Dump access termination entry point
 IJBXATDD ....... ........ Dump member disposition routine
  IJBXAMDC ...... ........ Member disconnect routine
  IJBXASTW ...... ........ Create new member if this member is new
  IJBXAFVS ...... ........ Free storage used for LRPL and tables
  IJBXATMD ...... ........ Dump mapping member disposition routine
   IJBXAWMR ..... ........ If mapping member was created in this process
                           write map records to library
    IJBXASME .... ........ Sort map tables
    IJBXAPSQ .... ........ Write record to library
    IJBXAFVS .... ........ Free storage from map tables
   IJBXAFVS ..... ........ Free storage used for mapping tables
   IJBXAMDC ..... ........ Member disconnect routine
   IJBXASTW ..... ........ Create new member if this member is new
   IJBXALDC ..... ........ Disconnect from library
   IJBXAFVS ..... ........ Free storage used for mapping member
 IJBXATSR ....... ........ Storage release routine
  IJBXAFVS ...... ........ Free storage used for all buffers, library control
                           blocks and dump access control blocks
```

MODULE FLOW FOR DUMP WRITE

This section shows the module flow for dump write functions. The different levels of the calling modules are represented by indenting each lower level module name. A short description of each module in the module flow is shown.

Request Dump Write:

```
IJBXARWR ........ ........ Dump access write request entry point
 IJBXAINT ....... ........ If dump access has not been initialized
                           call initialization process
 IJBXAWAP ....... ........ Append new record to dump member
  IJBXAWNM ...... ........ Process writing of new dump member to library
   IJBXAALC ..... ........ Allocate new member in library
    IJBXAGVS .... ........ Get storage for LRPL and directory workarea
    IJBXAMDC .... ........ If error in allocate disconnect from
                           new member
    IJBXALDC .... ........ If error in allocate disconnect from library
    IJBXAFVS .... ........ If error in allocate free library control
                           block storage
   IJBXAGVS ..... ........ Get storage for new mapping member
   IJBXAALC ..... ........ Allocate new mapping member in library
    IJBXAGVS .... ........ Get storage for LRPL and directory workarea
    IJBXAMDC .... ........ If error in allocate disconnect from
                           new mapping member
    IJBXALDC .... ........ If error in allocate disconnect from library
    IJBXAFVS .... ........ If error in allocate free library control
                           block storage
   IJBXAPAC ..... ........ Write dump record to library member
   IJBXACMP ..... ........ Create mapping table entry for dump record
  IJBXAEMC ...... ........ Process writing record to a section 6 extension member
   IJBXAALC ..... ........ Allocate extension member in library
    IJBXAGVS .... ........ Get storage for LRPL and directory workarea
    IJBXAMDC .... ........ If error in allocate disconnect from
                           new mapping member
    IJBXALDC .... ........ If error in allocate disconnect from library
    IJBXAFVS .... ........ If error in allocate free library control
                           block storage
   IJBXAPAC ..... ........ Write extension record to library
   IJBXAMDC ..... ........ Disconnect member from library
   IJBXASTW ..... ........ Create new member in library with STOW macro
   IJBXALDC ..... ........ Disconnect from library
   IJBXAFVS ..... ........ Free storage used in extension process
   IJBXAINE ..... ........ Call extension member initialization
  IJBXAPAC ...... ........ Write dump record to an existing dump member
  IJBXACMP ...... ........ Create mapping table entry for dump record
```

```
IJBXAWUP ....... ........ Update existing dump record
 IJBXARDA ...... ........ Read dump data records into storage
  IJBXAMAP ..... ........ Search map tables for dump address requested
  IJBXARBU ..... ........ Find dump record in storage or retrieve from  library
   IJBXARBS .... ........ Search storage buffers to see if record
                          is in main storage
   IJBXAGAC .... ........ Read record from library
   IJBXARBM .... ........ Move dump record from I/O buffer to data
                          buffer and update DAPL with contents data
 IJBXAGVS ...... ........ Get storage for data just read
 IJBXAWDR ...... ........ Process write request to library member
  IJBXAFVS ..... ........ Free storage used to save dump data
  IJBXAPAC ..... ........ Write dump records back to library
```

Chapter 16. PARSER: Introduction

The parser (phase name IJPARSER), a table-driven routine, checks the validity of all items in a given command. It does this by first checking the verb against a command directory. If the verb is valid, the parser checks the specified operands against a syntax table ("Command Table") of all possible keyword or positional parameters associated with that verb. Any command syntax item that the parser finds to be valid is changed by the parser to a standardized internal format (SIF) for further processing by one of various VSE/Advanced Functions components, such as SDAID, which presented the command.

A syntactic error in any item of the command - a parameter, delimiter, or separator - results in an error message which is passed to the calling program's message routine. The operator may usually reenter the corrected command line.

The parser is capable of analyzing any command that conforms to syntax rules for parameters as follows:

- Use parameters that represent values, one per parameter.
  Examples:

```
      TRACE BR
      TRACE PGML
  BR and PGML represent values for further processing.
```

- Use keywords that indicate the type of value which is being specified.
  Example:

```
TRACE STORAGE PATT=E2C4C1C9C4
```

  PATT is a keyword which indicates a pattern value, and E2C4C1C9C4 is the value specified by the user.

- Indicate a list by using parentheses when more than one value is specified for a keyword.
  Example:

```
TRACE SVC=(0,1C 2A)
```

  Specification 0,1C 2A enclosed in parentheses is a list of values.

- Ensure that alternatives on the same parameter level are unique.
- Enter range specifications in the form value:value.
- Indicate a continuation on the subsequent line of input by a dash (-) preceded by a delimiter and followed by a blank in the current line.
- Use a semicolon as a command-end indicator if the next command is to be started on the same line.
- Separate parameters (those representing a value as well as those using a keyword) from each other by a valid delimiter: a comma, one or more blanks, or a comma plus one or more blanks.
- Separate values within a list from each other by a valid delimiter.
- Enclose alphameric character strings within a pair of apostrophes. Quoted strings may also have special characters.

Keyword prompting may be used for entering the operands of a command. The parser originated messages prompt the user for the required control information.

Environment and Invocation

For each command to be analyzed, the parser needs a command table (CT) in virtual storage. The CT must be available to the parser when it receives control. The program calling the parser must also furnish certain information (as well as receive output) via the parser control block; for example:

- Command input area pointer,
- Output area pointer (for messages),
- Command Directory pointer,
- (if provided), check exit and I/O interface routine addresses,
- SIF and value entries from processed commands.

The parser, although documented as an independent set of routines, functions only as part of the program which uses the parser's services. The complete parser control block contents are listed in the Data Area section, later in this chapter. For register conventions, see the following section.

Chapter 17. PARSER: Program Design and Organization Information

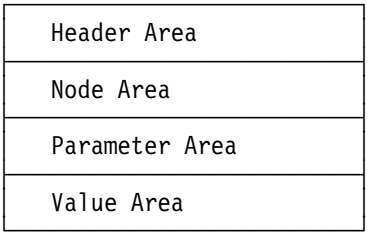
The general flow of control for command checking by the parser is illustrated in Figure 16 on page 336. In addition to the entry routine Parser (phase IJPARSER), subroutines are included in the following modules:

Module IJPANXTP: Subroutine NEXTPARM
IJPADIRS: DIRSERV
IJPAVALD: VALIDITY
IJPADEFT: DEFAULT
IJPASIFN: SIFENTRY
IJPAPMSG: PMSGLIB

The purpose of each is given in Figure 16 on page 336. A more detailed diagram of each is given later in this chapter.

Command Structure and the Command Table

The Command Table (CT) associated with each command verb indicates the required structure of the command; that is, the required sequence of keyword and positional parameters, in addition to other information. The command table formats the storage area (the size depends on each individual command) into four distinct areas:



- The Header area contains, for instance:
  - the command verb, abbreviation and number
  - environment requirements, mode flags, check exit information
  - pointers to associated Node, Parameter, and Value areas
- The Node area contains, for instance:
  - a block of pointers for each parameter that may occur (see below)
- Parameter area contains, for instance:
  - parameter number, flags, attributes, repetition range
  - pointers to prompting text, to default values, to error messages
- The Value area contains, for instance:
  - texts of prompting and error messages

For more details, see Figure 18 on page 339.

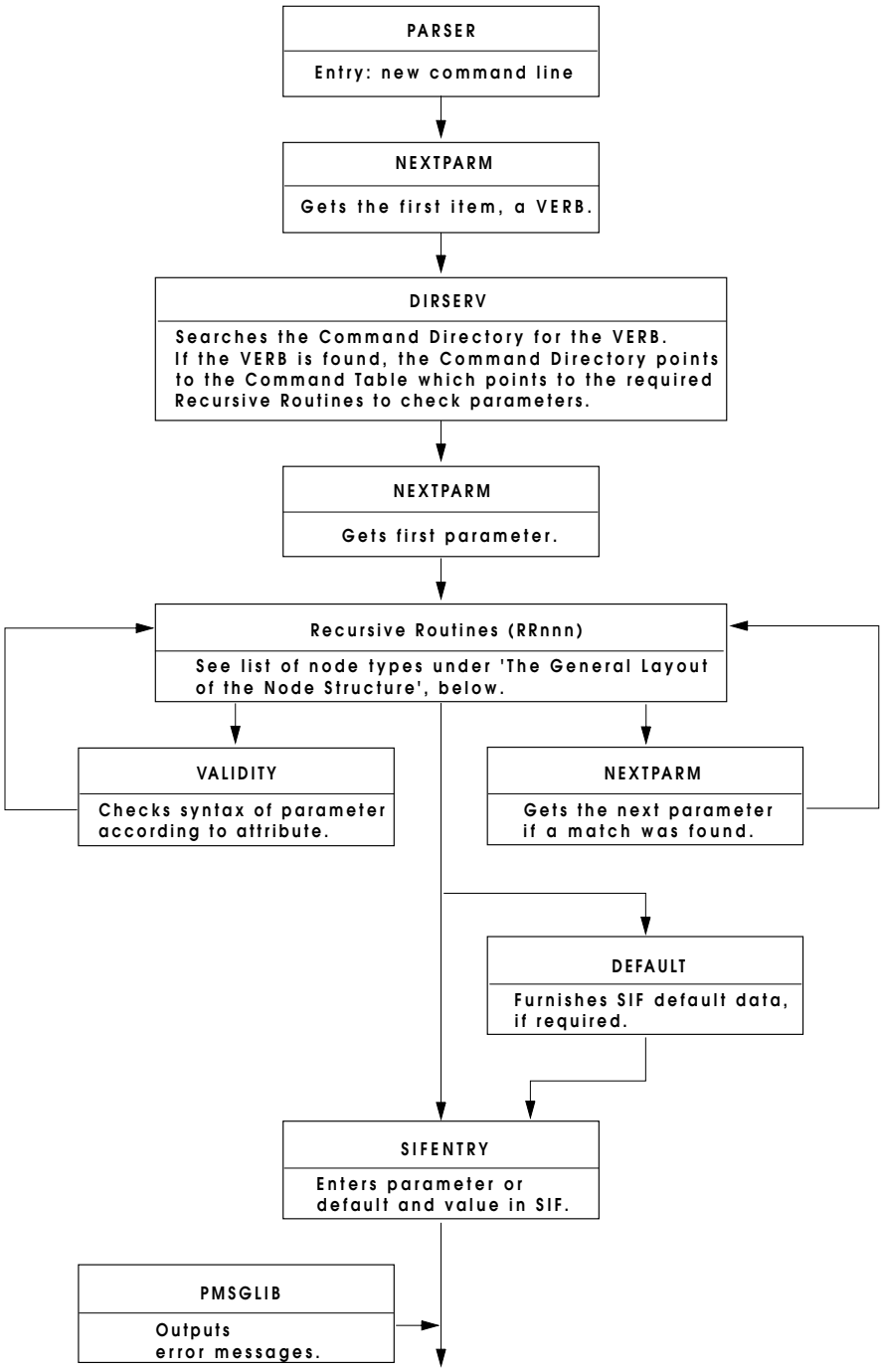
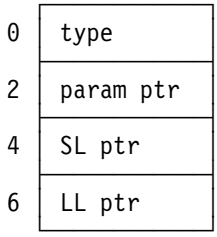


Figure 16. General Flow of Control for Command Checking

The General Layout of the Node Structure: The node structure part of the CT contains entries which represent the structure of the command. These entries (nodes) are used to establish the processing sequence of a given command. A node is of fixed format and contains the following fields:

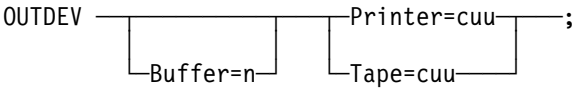
1. Node type
2. Offset for parameter description
3. Same level pointer
4. Lower level pointer



Node Type	*Abbreviation*	*SIF Entry*
Positional list	RRPLIST	no
Keyword list	RRKLIST	no
Command termination	RRTERM	no
Positional node	RRPOSN	yes
Positional - or	RRRPOR	no
Positional alternative	RRPALT	yes
Positional repetition	RRPREP	yes
Positional alternative with repetition	RRPAREP	yes
Keyword - or	RRKOR	no
Keyword alternative	RRKALT	yes
Keyword	RRKEYW	yes
Single keyword	RRSNGL	yes
Keyword repetition	RRKREP	no

The node type name; i.e., KALT, is the entry point name of the routine RRKALT handling this parameter (keyword alternative). The offset to the parameter; i.e., P1, (see Figure 18 on page 339) is the label of the parameter definition in the command table. P3, for example, describes the parameter PRINTER. The same level and lower level pointers define the logical connection between the various parameters. Alternatives are identified by an alternative number i.e. P3 A1 identifies the keyword printer, P3 A2 the keyword TAPE, and so on.

To illustrate the logic flow in the parser, the OUTDEV command from the SDAID program has the following syntax:



For an explanation of the syntax, see the manual *VSE/ESA Diagnosis Tools*.

The tables used by the parser for analyzing a command are made available through register 1, which contains a pointer to the Command Processor Control Block (CPCB). This is illustrated in Figure 19 on page 364. Figure 18 on page 339, an expansion of the command table shown in Figure 17 illustrates how the parser makes use of recursive routines to analyze the various parameters of a given command; in this case, the OUTDEV command from the SDAID program. This is further illustrated by examples in the following section.

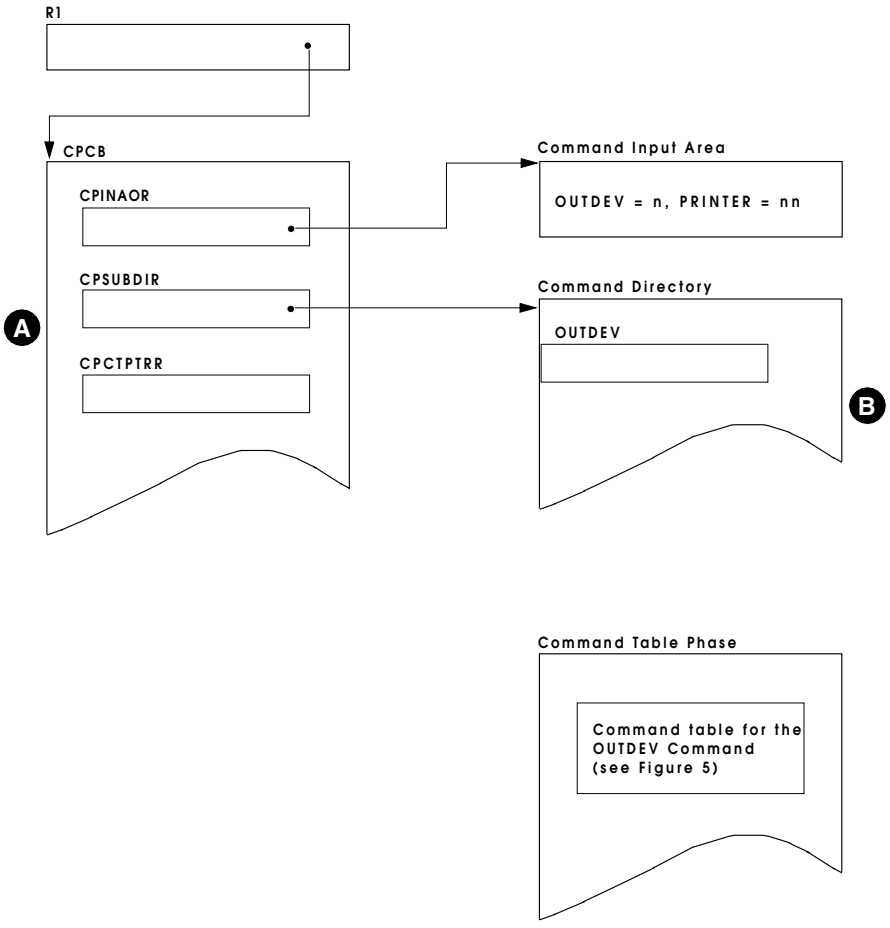
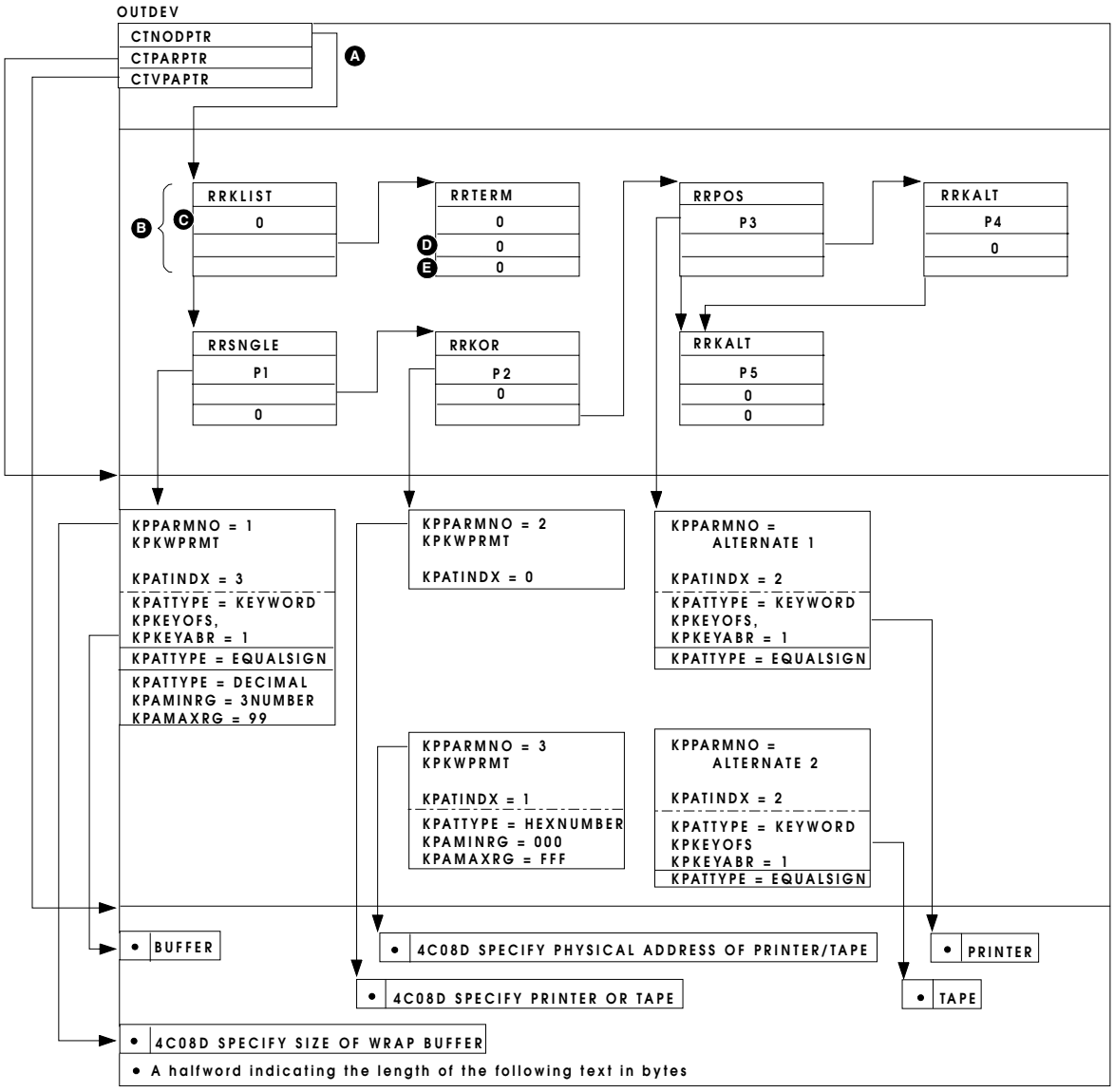


Figure 17. Table and Control Block Structure when the Parser Receives Control



A Fields within the command table header are interpreted via the dummy section CTHEADER.

B Parameter nodes. Fields of these nodes are interpreted via the dummy section CTSTRE.

C CTPARMOF - Parameter entry offset. Together with the address in the header field CTPARPTR.

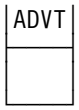
D CTSAMELV - Same level pointer. It points to the next parameter node of same level in the list of nodes for the command currently being processed.

E CTLOVELV - Lower level pointer. It points to the first or only parameter at the next or only lower level of nodes for the command currently being processed.

Figure 18. Control Flow Taken by the Parser for the OUTDEV (SDAID) Command

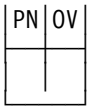
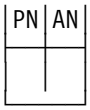
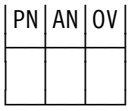
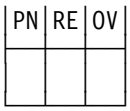
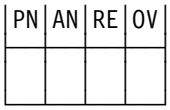
Logic Flow of the Parser

SIF Construction: The SIF consists of a fixed format string, called Standard Internal Text (SIT) and a related table with all variable length information, called Value Table (VT). The SIT contains an entry for every parameter in the command. An entry is composed of two or three fields. The following field types are used:

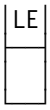
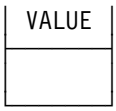
Field	*Content*	*Representation*
ADVT	Address of Value Table	
CN	Command Number	
PN	Parameter Number	
AN	Alternative Number	
OV	Offset into Value Table	
RE	Repetition Count	

The SIT starts with address of the value table and the command number. The command number is unique within the system. It consists of an identifier for the component (2 bytes) followed by a sequence number (2 bytes). The parameters of a command are identified by a parameter number. The parameter numbers are unique within one command only. They are entered into the PN field while the SIT is under construction. Alternatives of a parameter are identified by an alternative number. These numbers are unique within one parameter only. The fields contained in a SIT entry for a specific parameter depend on the parameter type.

The following entries are generated:

Parameter Type	*Fields*	*Representation*
– single keyword parameter (RRSNGL)	PN,OV	
– positional parm w/o alternative (RRPOS)		
– keyword alternatives (RRKALT)	PN,AN	
– positional alternatives (RRPALT)	PN,AN,OV	
– positional repetition (RRPREP)	PN,RE,OV	
– positional alternative with repetition (RRPAREP)	PN,AN,RE,OV	

SIT entries for keyword repetitions are built by repetitively using the entries defined above according to the parameter type. The VT is constructed from all variable information entered with the command. Variable information may be identifiers, strings, hexadecimal or decimal numbers; etc. To find the variable information belonging to a parameter, the address of the VT (ADVT field in the SIT) and the offset (OV field in the SIT) are added. The result is a pointer to the appropriate VT entry. A value table entry consists of the following fields:

Field	*Content*	*Representation*
LE	Length of Value	
VALUE	Variable Value	

After addressability to the first node (in the example RRKLIST) has been established, control is given up as defined by the type field; i.e., RRKLIST, RRSNGL. After one node is completed, the path defined by the lower level pointer is taken and after that the path defined by the same level pointer. (The only exceptions are the alternative nodes RRKALT, RRPALT, and RRPAREP. Here control is passed first to the same level, then to the lower level pointer.)

Examples:

These examples show the flow within the parser and the SIF produced from the given commands.

Note: The asterisk (*) and arrow (--->) indicate the nodes which enter information into the SIF and what is entered.

```
1.  OUTDEV   BUFFER=5   PRINTER=00E;
Parser ENTRY
RRKLIST
RRSNGL*---------> Parameter Number, Offset into Value Table
RRKOR
RRKALT*---------> Parameter Number, Alternate Number
RRPOS*----------> Parameter Number, Offset into Value Table
RRTERM
```

SIT

Addr of VT		PN	OV	PN	AN	PN	OV
	CMD NUM	P1	0	P2	A1	P3	3

VT

LE		Value	LE		Value		
0	1	5	0	3	0	0	E

```
2.  OUTDEV   BUF=22   T=230;
Parser ENTRY
RRKLIST
RRSNGL*---------> Parameter Number, Offset into VT
RRKOR
RRKALT
RRKALT*---------> Parameter Number, Alternate Number
RRPOS*----------> Parameter Number, Offset into VT
RRTERM
```

SIT

Addr of VT		PN	OV	PN	AN	PN	OV
	CMD NUM	P1	0	P2	A2	P3	4

VT

LE		Value		LE		Value		
0	2	2	2	0	3	2	3	0

```
3.  OUTDEV   PR=1FE;
    Parser ENTRY
    RRKLIST
    RRSNGL
    RRKOR
    RRKALT*-----------> Parameter Number, Alternate Number
    RRPOS*------------> Parameter Number, Offset into VT
    RRTERM
```

SIT

Addr of VT		PN	AN	PN	OV
	CMD NUM	P2	A1	P3	0

VT

LE		Value		
0	3	1	F	E

Note: P is a parameter number, A is an alternative number, and each square represents one byte.

Chapter 18. PARSER: Data Area Information

COMMAND TABLE HEADER STRUCTURE

```
2   CTNODPTR PTR(15),      00     ADDRESS OF NODE STRUCTURE
2   CTPARPTR PTR(15),      02     ADDRESS OF PARAMETERS
2   CTVPAPTR PTR(15),      04     ADDRESS OF VALUE PART

2   CTCOMAND CHAR(12),     06     COMMAND NAME
2   CTABBREV CHAR(6),      12     COMMAND NAME ABBREVIATION
2   CTCMDNR  CHAR(4),      18     COMMAND NUMBER

2   CTENVLST,              1C     ENVIRONMENT CHECK
 3   CTENVFLG   PTR(32),   1C     ENVIRONMENT ATTRIBUTE FLAG
 3   CTENVATR   PTR(32),   20     ENVIRONMENT CHECK ATTRIBUTE
  4   CTENVMTS  PTR(16),   20     ENVIRONMENT SPECIFICATION FLAGS

2   CTCEPEXI ,             24     Reserved for future use
 3   CTCEPFLG   PTR(32),   24
 3   CTCEPATR   PTR(31),   28
  4   CTCEPRTC  PTR(8),    28
  4   CTCEPID   PTR(8),    29
  4   CTCEPPHS  FIXED(15), 2A
   5   CTCEPADR FIXED(15), 2A
    6  CTCEPOFF FIXED(15), 2A
     8 CTCEPRID FIXED(8),  2A

2   CTCHKEXI,              2C     CHECK ROUTING INFO FOR COMMAND
 3   CTCHKFLG   PTR(32),   2C     CHECK ATTRIBUTE FLAG
 3   CTCHKATR   PTR(31),   30     CHECK ROUTING ATTRIBUTE
  4   CTCHKRTC  PTR(8),    30     ROUTE CODE
  4   CTCHKID   PTR(8),    31     IDENTIFICATION BYTE
  4   CTCHKPHS  FIXED(15), 32     OFFSET TO PHASE NAME
   5   CTCHKADR FIXED(15), 32     OFFSET TO BRANCH ADDRESS
    6  CTCHKOFF FIXED(15), 32     POINT OFFSET OF ROUTINE

2   CTMODE     BIT(16),    34     MODE SPECIFICATION
 4    CTNEWM   BIT(1),     34     ON IF NEW MODE
 4    CTCURM   BIT(1),     34     ON IF CURRENT MODE TERMINATES
 4    CTGDVAL  BIT(1),     34     ON IF COMMAND GENERAL VALID
 4    CTMDVAL  BIT(1),     34     ON IF COMMAND
                                  IN MODE DIRECTORY VALID
 4    CTSYNCH  BIT(1),     34     ON IF SYNTAX-CHECK
 4    CTNISUB  BIT(1),     34     ON IF NOT IN SUBSET
 4    CTSKIP   BIT(1),     34     ON IF SKIP
 4    CTDATA   BIT(1),     34     ON IF DATA
 4    CTCOLMN  BIT(1);     34     ON IF COLUMN
```

COMMAND TABLE NODE STRUCTURE ENTRY

```
  2   CTTYPE  BIT(16),       00     NODE TYPE
   4  CTCNTL  BIT(8),        00     CONTROL FIELD
    6 CTSIF   BIT(1),        00     ON FOR SIF ENTRY
    6 CTNOSUB BIT(1),        00     ON FOR NOSUBSET
   4  CTTYP   BIT(8),        01     NODE TYPE
  2   CTPARMOF PTR(15),      02     PARAMETER ENTRY OFFSET
  2   CTSAMELV PTR(15),      04     OFFSET SAME LEVEL
  2   CTLOWELV PTR(15);      06     OFFSET LOWER LEVEL

 DCL PLIST  BIT(8) CONSTANT('00'X),        POSITIONAL LIST
     KLIST  BIT(8) CONSTANT('04'X),        KEYWORD LIST
     TERM   BIT(8) CONSTANT('08'X),        COMMAND TERMINATION
     POSN   BIT(8) CONSTANT('0C'X),        POSIT. NODE
     POR    BIT(8) CONSTANT('10'X),        POSITIONAL - OR
     PALT   BIT(8) CONSTANT('14'X),        POSITIONAL ALTERNATIVE
     PREP   BIT(8) CONSTANT('18'X),        POSITIONAL REPETITION
     PAREP  BIT(8) CONSTANT('1C'X),        POS. ALT. WITH REPETITION
     KOR    BIT(8) CONSTANT('20'X),        KEYWORD - OR
     KALT   BIT(8) CONSTANT('24'X),        KEYWORD ALTERNATIVE
     KEYW   BIT(8) CONSTANT('28'X),        KEYWORD
     SNGL   BIT(8) CONSTANT('2C'X),        SINGLE KEYWORD
     KREP   BIT(8) CONSTANT('30'X);        KEYWORD REPETITION
```

THIS STRUCTURE DEFINES THE LAYOUT OF THE
KEYWORD AND POSITIONAL PARAMETERS OF THE PARSER

```
 2       KPPARMNO  FIXED(15),     00 PARAMETER NUMBER DEFINED BY CC
 2       KPEXCNTL  PTR  (16),     02 NODE CONTROL FLAGS
  4       KPDFLTCD PTR  (8),      02 DEFAULT ID
  4       KPCEP    BIT  (1),      02 ON IF CEP EXIT IN CURRENT COPY
  4       KPCHK    BIT  (1),      02 ON IF CHECK EXIT IN CURRENT COPY
  4       KPSTPOS  BIT  (1),      02 ON IF SIT POSITION IN CURRENT COPY
  4       KPERR    BIT  (1),      02 ON IF ERROR MSG DEFINED IN KPERRMSG
  4       KPPRMTFL BIT  (1),      02 ON IF PROMPT ANSWER DEFINED

 2       KPPROMPT  CHAR (6) BDY(HWORD),      PROMPT INFORMATION
  4       KPKWPRMT FIXED(15),     04 OFFSET TO KEYWORD PROMPT MESSAGE
  4       KPFLPRMT FIXED(15),     06 OFFSET TO FULL PROMPT MESSAGE
  4       KPPRMTOF FIXED(15),     08 OFFSET TO PROMPT ANSWER

 2        KPDEFOFS FIXED(15),     0A OFFSET TO DEFAULT VALUE

 2       KPREPRNG  FIXED(31),     0C REPETION RANGE
  4       KPMINRNG FIXED(15),     0C MINIMUM RANGE IF NOT PRESENT MIN=1
   6      KPREPID  PTR  (8),      0C DEFINES EXISTENCE OR *
  4       KPMAXRNG FIXED(15),     0E MAXIMUM RANGE MUST BE DEFINED

 2       KPERRMSG  FIXED(15),     10 OFFSET TO ERROR MESSAGE IF PRESENT

 2       KPATINDX  FIXED(15),     12 INDEX DEFINES MAX VECTOR ENTRIES

 2       KPCTPEND  CHAR (0);      14 END INDICATOR
```

STRUCTURE DEFINES THE ATTRIBUTE PART OF THE PARAMETER

```
 1    KPATRIBT(*) BASED(ADDR(KPCTPEND)),
                                  THE FOLLOWING DATA IS USED IN THE
                                  VALIDITY ROUTINE
  2     KPATRID  PTR  (32),       14 ATTRIBUTE IDENTIFICATION FLAGS
   4     KPATTYPE PTR(8),         14 ATTRIBUTE TYPE
   4     KPATFLGS PTR(8),         15 ATTRIBUTE FLAGS
    6     KPPAIR   BIT  (1),      15 ON IF PAIR
    6     KPCONCAT BIT  (1),      15 ON IF CONCAT
    6     KPCOMPNT BIT  (1),      15 ON IF MSHP COMPONENT FEATURE NUMBER
    6     *        BIT  (3),      15 RESERVED
    6     KPVPOF1  BIT  (1),      15 ON IF 1ST HW IN KPPARM CONTAINS
                                      OFFSET IN VALUE PART
    6     KPVPOF2  BIT  (1),      15 ON IF 2ND HW IN KPPARM CONTAINS
                                      OFFSET IN VALUE PART
   4     KPRESRVD PTR(15),        16 RESERVED
  2     KPPARM   PTR  (32),       18 OVERLAY AREA
  2     KPDUMEND CHAR (0);        1C END INDICATOR OF ATTRIBUTE LIST

 DCL  KPNULL   BIT (8) CONSTANT('00'X),     DUMMY ATTRIBUTE
      KPKEYWRD BIT (8) CONSTANT('04'X),     KEYWORD
      KPLITERL BIT (8) CONSTANT('08'X),     LITERAL
      KPEQUAL  BIT (8) CONSTANT('0C'X),     ITEM '=' SIGN
      KPIDENT  BIT (8) CONSTANT('10'X),     IDENTIFIER
      KPALPHA  BIT (8) CONSTANT('14'X),     ALPHA CHAR STRING
      KPSTRING BIT (8) CONSTANT('18'X),     STRING
      KPQUOTST BIT (8) CONSTANT('1C'X),     QUOTED STRING
      KPBITSTR BIT (8) CONSTANT('20'X),     BIT STRING
      KPDECNUM BIT (8) CONSTANT('24'X),     DECIMAL NUMBER
      KPHEXNUM BIT (8) CONSTANT('28'X),     HEX NUMBER
      KPCHKEXT BIT (8) CONSTANT('2C'X),     CHECK EXIT
      KPCEPEXT BIT (8) CONSTANT('30'X),     CEP EXIT
      KPENV    BIT (8) CONSTANT('34'X),     ENVIRONMENT ATTRIBUTE
      KPSEP    BIT (8) CONSTANT('38'X),     SEPARATOR
      KPLSKIP  BIT (8) CONSTANT('3C'X),     LINE SKIP
      KPSYSKIP BIT (8) CONSTANT('40'X),     SYSIN | SYSRDR SKIP
      KPCOLUMN BIT (8) CONSTANT('44'X),     COLUMN SPECIFIED
      KPSITPOS BIT (8) CONSTANT('48'X);     SIT POSITION SPECIFIED
```

USED IF ITEM IS A LITERAL OR FOR A RANGE DEFINITION

```
 1    KPATLIT(*) DEFINED(KPATRIBT),
  2     *        PTR,             14
  2     KPATRCHK FIXED(32),       18 IF KPLITERL = ON  THEN OFFSET TO
                                     LITERAL AND SHORTEST ABREV. ELSE
                                     TWO HALF WORDS FOR RANGE DEFINITION
   4      KPAMINRG FIXED(16),     18 MINIMUM RANGE. IF KPAMINR = 0 A
                                     MINIMUM RANGE OF 1 IS ASSUMED.
    6     KPLTOFFT FIXED(16),     18 OFFSET TO LITERAL.KPLITERL = ON
   4      KPAMAXRG FIXED(16),     1A MAXIMUM RANGE DEF. MUST EXIST
    6     KPLTRUNC FIXED(8);      1A NO.OF CHARS OF SHORTEST TRUNCATION
                                     USED ONLY IF KPLITERL = ON
```

USED FOR THE CEP EXIT DEFINITION

```
 1       KPATCEP(*)  DEFINED(KPATRIBT),
  2      *         PTR,           14
  2      KPCEPEXI  PTR  (32),     18 CEP  EXIT LOCATION ID
   3      KPCEPRTC PTR  (8),      18 ROUTE CODE
   3      KPCEPID  PTR  (8),      19 IDENTIFICATION BYTE
   3      KPCEPPHS FIXED(15),     1A OFFSET TO PHASE NAME OF ROUTINE
    4     KPCEPADR FIXED(15),     1A OFFSET TO BRANCH ADDRESS OF ROUTINE
     6    KPCEPOFF FIXED(15),     1A POINT OFFSET OF ROUTINE
      8   KPCEPRID FIXED(8);      1A ROUTE ID OR *
```

USED FOR THE CHECK EXIT INFORMATION

```
 1       KPATCHCK(*)  DEFINED(KPATRIBT),
  2      *         PTR,           14
  2      KPCHKEXI  PTR  (32),     18 CHECK EXIT LOCATION ID
   4      KPCHKRTC PTR  (8),      18 ROUTE CODE
   4      KPCHKID  PTR  (8),      19 IDENTIFICATION BYTE
   4      KPCHKPHS FIXED(15),     1A OFFSET TO PHASE NAME OF ROUTINE
     8    KPCHKADR FIXED(15),     1A OFFSET TO BRANCH ADDRESS OF ROUTINE
      9   KPCHKOFF FIXED(15);     1A POINT OFFSET OF ROUTINE
```

USED FOR THE KEYWORD ITEM

```
 1       KPATKWD(*)  DEFINED(KPATRIBT),
  2      *         PTR,           14
  2      KPKYWRD   FIXED(31),     18 KEYWORD IDENTIFIER
   4      KPKEYOFS FIXED(15),     18 OFFSET TO KEYWORD
   4      KPKEYABR PTR  (8);      18 NO.OF CHARS OF SHORTEST ABBREVIATION
```

USED FOR COLUMN INFO

```
 1       KPATCOL(*)  DEFINED(KPATRIBT),
  2      *         PTR,           14
  2      KPCOLMN   FIXED(31),     18
   4      KPCOLNUM FIXED(15),     18 COLUMN NUMBER ONLY FOR POSITIONAL
   4      KPCOLRST FIXED(15);     18 UNUSED
```

USED FOR THE LINE SKIP INFO

```
 1       KPATLSK(*)   DEFINED(KPATRIBT),
  2      *         PTR,           14
  2      KPLINSKP  FIXED(31),     18
   4      KPNOLIN  FIXED(15),     18 NUMBER OF LINES TO BE SKIPPED
   4      KPLRST   FIXED(15);     18 UNUSED
```

USED FOR THE SIT POSITION INFO

```
 1       KPATSITP(*)  DEFINED(KPATRIBT),
  2      *         PTR,           14
  2      KPSITP    FIXED(31),     18
   4      KPPOSIT  FIXED(15),     18 POSITION OF NEXT ENTRY IN SIT
   4      KPSITRST FIXED(15);     18 UNUSED
```

**THIS STRUCTURE DEFINES THE DATA TO BE WRITTEN INTO THE SIT
AND THE VALUE TABLE DEFINED WITHIN THE CPCB.**

```
THIS STRUCTURE MUST BE BASED ON REG1.

  2     SITENTRY    BDY(HWORD),     00
   4    SITINFO     PTR   (16),     00 SIT TYPE CONTROL INFORMATION
    6   SITCNTL     BIT   (8),      00 CONTROL INFO FOR THIS ENTRY
    6   SITTYPE     BIT   (8),      01 DEFINES THE NODE OF THE ITEM
   4    SITPNUM     PTR   (16),     02 PARAMETER NUMBER
    6   PNPARM      BIT   (1),      02 ON IF PARAMETER
    6   PNDEFLT     BIT   (1),      02 ON IF DEFAULT VALUE USED
    6   PNPAIR      BIT   (1),      02 ON IF PARAM VALUE HAS PAIR ATTRIB
   4    SITREST     BDY(HWORD),     04 REST OF SIF ENTRY ELEMENTS

 1      SNGLPOS     DEFINED (SITREST),  04 POSITIONAL & SINGLE NODES
  2     POSOV       FIXED (15),         04 OFFSET INTO VALUE TABLE

 1      KEYKALT     DEFINED (SITREST),  04 KEY &KEYWRD ALTERNAT NODES
  2     KEYAN       FIXED (15),         04 ALTERNATIVE NUMBER
   6    *           BIT(1),             04 ALWAYS ZERO
   6    ANDEFLT     BIT(1),             04 ON IF DEFAULT=@ AND PARM OMITTED

 1      PALTS       DEFINED  (SITREST), 04 POSITIONAL ALTERNATIVE
  2     PALAN       FIXED (15),         04 ALTERNATIVE NUMBER
  2     PALOV       FIXED (15),         06 OFFSET INTO VALUE TABLE

 1      PREPS       DEFINED (SITREST),  04 POSITIONAL WITH REPETITIONS
  2     PRERE       FIXED (15),         04 REPETITION COUNT
  2     PREOV       FIXED (15),         06 OFFSET INTO VALUE TABLE

 1      PAREPS      DEFINED (SITREST),  04 POS ALTERN WITH REPETITIONS
  2     PARAN       FIXED (15),         04 ALTERNATIVE NUMBER
  2     PARRE       FIXED (15),         06 REPETITION COUNT
  2     PAROV       FIXED (15);         08 OFFSET INTO VALUE TABLE
```

```
THE FOLLOWING CONSTANT DEFINITIONS WILL BE USED
TO INTERPRET THE POSITIONAL AND KEYWORD PARAMETERS

USED TO IDENTIFY CEP AND CHECK EXIT ROUTE CODES
AND HOW TO INTERPRET THE CEP AND CHECK ENTRY POINTS

 CDFETCH   BIT(8) CONSTANT('01'X),     EXIT IS VIA FETCH
 CDPOINT   BIT(8) CONSTANT('02'X),     EXIT IS VIA POINT
 CDBRANCH  BIT(8) CONSTANT('03'X),     EXIT IS VIA BRANCH
 CDRETURN  BIT(8) CONSTANT('04'X),     EXIT VIA RETURN ONLY IF CEP

USED TO INTERPRET THE DEFAULT DEFINITION

 CDDEFNO   BIT(8) CONSTANT('00'X),     DEFAULT = NO
 CDDEFSIF  BIT(8) CONSTANT('04'X),     DEFAULT = SIF (DEF = YES, SIF)
 CDDEF@    BIT(8) CONSTANT('08'X),     DEF = @ (DEF = YES, NOSIF)
 CDDEFYES  BIT(8) CONSTANT('0C'X),     DEFAULT = YES (SIF FROM CT)

USED TO INTERPRET THE REPETITION FIELD

 CDREPNO   BIT(8) CONSTANT('FF'X),     NO REPETITION DEFINED
 CDREPAST  BIT(8) CONSTANT('5C'X);     REP = *
```

IJPACPCB - Command Analyzer Control Block

PARAMETER PART

```
 2   CPCBGENP   BOUNDARY(WORD),          NOTE: LEVEL 2 RESTRICTED

  3   CPPARM     BOUNDARY(WORD),
```

CPCB IDENTIFIER

```
   4   CPID       CHAR(4),              CPCB IDENTIFIER   ('CPCB')
```

CPCB STATUS FLAGS

```
   4   CPSTFLGS,                        STATUS FLAGS AREA
    6  CPINSTAT   BIT(8),
     8 CPINCOMP   BIT(1),               BUFFER COMPLETELY PROCESSED
                                        IF ON. SET BY NEXTPARM
     8 CPLNEEND   BIT(1),               ON IF LINE END. COMMAND
                                        CONTINUATION REQUESTED.
     8 CPSIFFLG   BIT(1),               SIF IS READY. SET BY
                                        SIFENTRY.
     8 CPCMDC     BIT(1),               COMMAND CANCELED IF ON.
                                        SET BY NEXTPARM
     8 CPPRMPT1   BIT(1),               PROMPTING IN EFFECT IF ON.
                                        SET BY NEXTPARM. RESET BY
                                        RRTERM.
     8 CPNOPRMT   BIT(1),               NO PROMPTING ALLOWED IF ON
     8 CPCLUPLP   BIT(1),               CLEANUP LOOP FOR EXCESSIVE
    6  *          BIT(24),               RESERVED
```

INPUT BUFFER CONTROL

```
   4   CPIN,                            INPUT INFORMATION
    6  CPINADR    PTR,                  ADDRESS OF INPUT AREA
    6  CPINSIZ    FIXED(15),            SIZE OF INPUT AREA
    6  CPINLEN    FIXED(15),            LINE LENGTH OF INPUT
    6  CPINSEQ    PTR(15),              BEGIN OF SEQUENCE FIELD
    6  CPINBEG    PTR(15),              BEGIN OF COMMAND INPUT
```

MESSAGE BUFFER CONTROL

```
   4   CPOUT,                           MESSAGE INFORMATION
    6  CPOUTADR   PTR,                  ADDRESS OF MESSAGE AREA
    6  CPOUTSIZ   FIXED(15),            SIZE OF MESSAGE AREA
    6  CPOUTLEN   FIXED(15),            LENGTH OF MESSAGE
```

COMMAND NUMBER

```
   4   CPCMDNO    CHAR(4),              COMMAND/SIF NUMBER
```

SIF CONTROL

```
   4   CPSITVT,                         SIF (SIT/VT) INFORMATION
    6  CPSITADR   PTR,                  ADDRESS OF SIT
    6  CPVTADR    PTR,                  ADDRESS OF VALUE TABLE
    6  CPSITLEN   FIXED(15),            LENGTH OF SIT
    6  CPVTLEN    FIXED(15),            LENGTH OF VALUE TABLE
    6  CPXWRK     PTR,                  PTR TO EXTERNALLY DEFINED
                                        VALUE TABLE WORK AREA
    6  CPXWLEN    FIXED(15),            LENGTH OF EXTERNAL WORK AREA
   4   CPSITCNT   FIXED(15),            COUNTER FOR PERMUTED PARMS
```

ROUTING CONTROL

```
   4   CPROUTE    ,                     ROUTING INFORMATION
    5  CPCEPRTC   PTR(8),               ROUTING CODE
    5  *          PTR(8),               RESERVED
    5  *          FIXED(15),            RESERVED
    5  CPCEPPHN   CHAR(8) BDY(WORD),    PHASENAME OF CEP
     6  CPCEPADR  PTR(31),              ADDRESS OF CEP
      7  CPCEPOFF  FIXED(15),           OFFSET IN BRANCHTAB TO CEP
       8  CPCEPRID  BIT(8),             ROUTING ID OR *
```

I/O INTERFACE CONTROL

```
   4   CPIO,                            I/0 INTERFACE SPECIFICATION
    6  CPIOPTR    PTR,                  POINTER TO I/O INTERFACE.
                                        SET BY ENVIRONMENT.
    6  CPIOANC    PTR,                  I/O INTERFACE DYNAMIC WORK-
                                        AREA ANCHOR IN MULTITHREAD
    6  CPIOTYPE   BIT(8),               INDICATE READ AND/OR WRITE
     7  CPIOTYPW  BIT(1),                        WRITE
     7  CPIOTYPR  BIT(1),                        READ
     7  CPIOFSET  BIT(1),               ON: IOF CALLED
    6  CPIODEV    BIT(8),               DEVICE: SYSLOG, SYSLST,
                                        OR USER'S TERMINAL
     7  CPIODSYS  BIT(3),
      8 CPIODLOG  BIT(1),                       SYSLOG
      8 CPIODLST  BIT(1),                       SYSLST
      8 CPIODRDR  BIT(1),               ON: IF INPUT FROM READER
                                        OFF: INPUT FROM TERMINAL
    6  *          BIT(16),              RESERVED
    6  CPIOCNTL   PTR,                  SAVE AREA FOR RET NODE FOR
                                        CONT LINE PROCESSING
    6  * (1)      PTR,                  RESERVED
```

GENERAL COMMAND SET CONTROL

```
 4   CPGENCMD,                        GENERAL COMMAND SET
  6  CPGENDIR   PTR,                  ADDR OF GENERAL DIRECTORY
  6  CPGENOFS   PTR,                  CEP EXITS, POINT=OFFSET.
                                      ADDR OF CEP EXT ADDR LIST.
  6  CPGENANC   PTR,                  CEP DYNAMIC WORKAREA ANCHOR
                                      IN MULTITASKING/MULTITHREAD
  6  CPGENCOF   PTR,                  CHK EXITS, POINT = OFFSET
                                      ADDR OF CHK EXT ADDR LIST
```

SUBCOMMAND SET CONTROL

```
 4   CPSUBCMD,                        SUBCOMMAND SET FOR ONE MODE
  6  CPSUBDIR   PTR,                  ADDR OF MODE SUBDIRECTORY
  6  CPSUBOFS   PTR,                  CEP EXITS, POINT=OFFSET.
                                      ADDR OF CEP EXT ADDR LIST.
  6  CPSUBANC   PTR,                  CEP DYNAMIC WORKAREA ANCHOR
                                      IN MULTITASKING/MULTITHREAD
  6  CPSUBCH    BIT(8),               CHAIN TO GENERAL COMMANDS
  6  *          BIT(24),              RESERVED
```

CHECK EXIT CONTROL

```
 4   CPCHECK    CHAR(12) BDY(WORD),   CHECK EXIT INFO
  5  CPCHKRTC   PTR(8),               ROUTING CODE
  5  *          PTR(8),               RESERVED
  5  *          FIXED(15),            RESERVED
  5  CPCHKPHN   CHAR(8) BDY(WORD),    PHASENAME OF CHECK EXIT RT
   6  CPCHKADR  PTR(31),              ADDRESS OF CHECK EXIT RT
    7  CPCHKOFF  FIXED(15),          OFFSET IN BRANCHTAB TO CHK
 4   CPCHKSP    PTR,                  PTR TO PREPARED SIT ENTRY
 4   CPCHKVP    PTR,                  PTR TO PREPARED VT ENTRY
 4   CPCHKVL    PTR(15),              LENGTH OF VT ENTRY
 4   CPCHKFLG   PTR(16),              FLAGS
  6  CPCHKMSG   BIT(1),               ON IF CHK EXIT RETURNS MSG
  6  CPCHKSET   BIT(1),               ON: CHECK EXIT TAKEN
  6  CPCHKEC    BIT(1),               ON: SEMANTIC ERROR CORRECT
  6  CPCHKPE1   BIT(1),               ON: SEMANTIC ERROR CORRECT
                                          FOR FIRST PAIR ELEMENT.
  6  CPCHKPE2   BIT(1),               ON: SEMANTIC ERROR CORRECT
                                          FOR SECOND PAIR ELEM.
  6  CPCHKSVE   BIT(1),               SEVERE ERROR ISSUED IN A
                                      CHECK EXIT ROUTINE. COMMAND
                                      IS CANCELED.
```

OPTIONS CONTROL

```
   4   CPOPTION   BIT(32),              OPTIONS FOR CA
    6  CPINPUT    BIT(1),               ON IF INP MODE DURING EDIT
    6  CPSCAN     BIT(1),               ON IF SYNTAX CHK IN ED INPT
    6  CPDEBUG    BIT(1),               ON IF DBG MODE:MODULE TRACE
    6  *          BIT(29),
   4   * (1)      PTR,                  RESERVED
```

ENVIRONMENT INFORMATION

```
   4   CPENV,                           ENVIRONMENT INFORMATION
    6  CPENVID    BIT(16),              ENVIRONMENT ID
    6  CPENVPAR   BIT(16),              ENVIRONMENT PARAMETERS
```

IJPARSER INTERNAL CPCB PART

```
  3   CPINTRNL    BDY(WORD),
```

INTERNAL INTERFACE

```
   4   CPINTIF,
    6  CPCTPTR    PTR  (31),            CT-PTR
    6  CPPHASNM   CHAR (8),             PHASE NAME
    6  CPREGSAV,                        REGISTER SAVE-AREA
     8  CPREGSA1  PTR,
     8  CPNXTPIP  PTR,                  SAVE AREA POINTER TO ITEM
     8  CPNXTPIL  PTR,                  SAVE AREA LENGTH OF ITEM
    6  CPPARMNO   FIXED(31),            CURRENT PARAMETER NUMBER
    6  CPATTCNT   FIXED(15),            NOS OF ATTRIBUTES PROCESSED
    6  CPIFFLG    PTR(16),              INTERNAL FLAGS
     8  CPDSESW   BIT(1),               DIRSERV END SWITCH
     8  CPPRMTPA  BIT(1),               CURRENT PARAMETER PROVIDED
                                        AS PROMPT REPLY IF ON. NO
                                        PROMPT OR SUCCEEDING ITEMS
                                        IN REPLY IF OFF. SET AND
                                        RESET BY NEXTPARM.
     8  CPPRMTAC  BIT(1),               MORE ITEMS IN PROMPT REPLY
                                        IF ON. LAST ITEM IN REPLY
                                        IF OFF. SET/RESET NEXTPARM

     8  CPDFLTEN  BIT(1),               ON IF DEFAULT ENTRY MADE
                                        SET BY DEFAULT
                                        RESET BY NEXTPARM
```

```
    8  CPFLSHPM  BIT(1),                ON IF ERROR ROUTINE FLUSHES
                                        REST PARAMETERS.
                                        SET BY ERROR.
                                        RESET BY NEXTPARM.
    8  CPKREPER  BIT(1),                IF ON. ERROR IN A KEYWORD
                                        REPETITION LIST
    8  CPMDPOS   BIT(1),                ON IF CONT LINE IN POSPARM
    8  CPPRMLIL  BIT(1),                ON IF LAST ITEM IN PROMPT LN
    8  CPKREPMD  BIT(1),                ON IF PROCESSING IN KREP MOD
    8  CPCLCNT   BIT(1),                ON IF SUCCESSIVE CONT.LINES
    8  CPPRMMOD  BIT(1),                ON IF ANALYZER IS OPERATING
                                        IN PERMUTATION MODE.
    8  CPKRCNLN  BIT(1),                CONTINUATION LINE IN PROCESS
```

VALIDITY INTERNAL AREA

```
  4   CPVALAR,
   6  CPVALMIN   FIXED(15),            MINIMUM STRING LENGTH
   6  *          FIXED(15),
   6  CPVALMM    CHAR(32),             WORKFIELD FOR MIN AND MAX
   6  CPVALIT    CHAR(LENGTH(CPVALMM)),   WRKFLD FOR ITEM TRANSLATE
  4   *          PTR,                  RESERVED
```

STACK ENTRY PREPARATION AREA

```
  4   CPSTKENT,                        STACK ENTRIES
   6  CPPRMFLG   PTR  (32),            PERMUTATION CONTROL FLAGS
    8 CPPRMLOP   BIT  (1),             ON: PERMUTATION IN EFFECT
                                           IN CURRENT KLIST TREE
   6  CPCTSTRE   PTR  (31),            PTR TO CURRENT STRING ENTRY
   6  CPRETURN   PTR  (31),            RETURN ADDRESS
   6  CPREPCNT   FIXED(15),            REPETITION COUNTER
   6  CPNDTYPE   PTR  (16),            TYPE OF LAST PROCESSED NODE
    8 CPCNTL     BIT  (8),             TYPE CONTROL INFO
    8 CPTYPE     BIT  (8),             TYPE DEFINITION

   6  CPCURSIT   PTR  (31),            SIT POSITION DEFINED FOR
                                       THE CURRENT KLIST NODE.
   6  CPCURIP    PTR  (31),            POINTS TO THE CURRENT ITEM
                                       WHICH HAS A MISMATCH.
  4   CPRETCD2   PTR  (31),            SAVE AREA FOR RETURN CODE
                                       AFTER PROMPT
```

EXTENDED ATTRIBUTE INTERFACE AREA

```
4   CPEXATR,
 6  CPATFLGS   PTR  (16),            FLAG AREA
  8 CPPAIR     BIT  (1),             ON IF PAIR ATTR.PROCESSED
  8 CPCHKEXT   BIT  (1),             ON IF CHECK EXIT PROCESSED
  8 CPSECERR   BIT  (1),             ON IF ERROR IN 2ND PAIR ELE
  8 CPNOCLN    BIT  (1),             ON IF COLON MISSING |INCOMPL
  8 CPAIRERR   BIT  (1),             ERROR IN PAIR DETECTED.
  8 CPCOMPNT   BIT  (1),             ON IF MSHP COMPONENT FNUM
 6  CPINDX     PTR  (15),            INDEX INTO ATTRIB ARRAY
 6  CPOUTPTR   PTR  (15),            PTR INTO CPOUTBUF
 6  *          PTR  (15),
 6  CPRETCD    PTR  (31),            RETURN CODE SAVE AREA
 6  CPR7       PTR  (31),            POINTER TO CT PARAMETER PART
 6  CPR8       PTR  (31),            POINTER TO CT VALUE PART
 6  CPR9       PTR  (31),            POINTER TO CURRENT STRUCTURE
                                     ENTRY.
 6  CPMDINAR   PTR  (31),            SAVE AREA FOR CPINADR
 6  CPMDINLN   PTR  (31),            SAVE AREA FOR CPINLEN
 6  CPMDINSZ   PTR  (31),            SAVE AREA FOR CPINSIZ
 6  CPMDINSQ   PTR  (31),            SAVE AREA FOR CPINSEQ
 6  CPSECLNG   FIXED(15),            LENGTH OF SECOND PAIR ELEMT
 6  *          FIXED(15),            RESERVED
 6  CPSECPRM   PTR  (31),            PTR TO 2ND ELEMENT IN ERROR
 6  CPCLNPTR   PTR  (31),            PTR TO 1ST PAIR ELEMENT
 6  CPCLNLNG   PTR  (15),            LENGTH OF 1ST PAIR ELEMENT
4   *          PTR  (15),            DUMMY SPACE FOR FUTURE USE
```

NEXTPARM INTERNAL AREA

```
4   CPNEXTPA,
 6  CPSWITCH   PTR  (16),            AREA FOR LEVEL INFO ETC.
  8  CPCMDEND  BIT  (1),             ON IF COMMAND END
  8  CPVERBS   BIT  (1),             VERB SEARCH REQUESTED BY
                                     CAENTRY. RESET BY NEXTPARM.
  8  CPSBLEND  BIT  (1),             ON IF SUBLIST END
  8  CPNPIRES  BIT  (1),             ON IF NO NEW ITEM FROM NP
  8  CPRETPMK  BIT  (1),             ON IF '?' IS TO BE RETURNED
                                     OFF GET PROMPT TEXT
 6  CPLITEM    FIXED(15),            LENGTH OF ITEM
 6  CPPITEM    PTR  (31),            PTR TO ITEM IN I/O AREA
```

SIT INTERFACE AREA

```
4   CPSITIF   CHAR(10) BDY(WORD),    SIFDATA BASED ON THIS AREA
4   *         CHAR(2) BDY(HWORD),    PADDING
4   *         PTR,                   RESERVED
```

SIF ENTRY INTERNAL AREA

```
4   CPSIFENT,
 6  CPSITEF    PTR,                  NEXT FREE ENTRY IN SIT AREA
 6  CPSITEND   PTR,                  END OF SIT
 6  CPVTEF     PTR,                  NEXT FREE ENTRY IN VT AREA
 6  CPVTEND    PTR,                  END OF VT
 6  CPSENTFL   BIT(8),               STATUS FLAGS FOR SIFENTRY
  8 CPNXTCMD   BIT(1),               PROCESSING OF NEXT COMMAND
                                     BEGINS. SET BY CAENTRY.
                                     RESET BY SIFENTRY.
 6  *          BIT(24),              RESERVED


4   CPSTPCTL,                        SIT PREPARATION AREA CONTRL
 6  CPSTPBEG   PTR,                  BEGIN OF SIT PREP AREA
 6  CPSTPNXF   PTR,                  NEXT FREE ENTRY IN SIT PREP
 6  CPSTPEND   PTR,                  END OF SIT PREP AREA
```

LINE BEGIN STATUS SAVE AREA

```
4   CPSVELB,                         LINE BEGIN STATUS SAVE AREA
 6  CPLBCSTK   PTR,                  CURRENT STACK ENTRY
 6  CPLBCSTR   PTR,                  CURRENT STRUCTURE ENTRY
 6  CPLBSITE   PTR,                  NEXT FREE SIT ENTRY
 6  CPLBVTEF   PTR,                  NEXT FREE VT ENTRY
 6  CPLBRETN   PTR,                  RETURN POINT WITH FULL LINE
 6  CPLBFLGS   BIT(16),              STATUS FLAGS
  8 CPLBLNEE   BIT(1),               CONTINUATION LINE RECEIVED
 6  *          BIT(16),
4   CPSEMPPT   PTR,                  CONTAINS PTR TO PARM AREA
                                     OF ENTRY KALT AND PALT.
                                     SET IN KOR & POR NODES.
                                     RESET IN ERROR.
4   CPREGSA2   PTR,                  SAVE AREA FOR RET FROM PRMT
4   CPERRETR   PTR,                  CONTAINS THE RETURN POINT
                                     TO WHICH THE ERROR CORRECT.
                                     TRANSFER CONTROL.
```

ERROR INTERFACE AREA

```
4   CPERRIF,
 6  CPPBPTR   PTR,                   BEGIN OF PARAMETER TYPING
 6  CPERCLMN  FIXED(15),             COLUMN WHERE ERROR OCCURRED
 6  CPERSVRT  FIXED(15),             ERROR SEVERITY
 6  CPMSGID    CHAR(9) BDY(WORD),    MESSAGE ID FOR ERROR
  8 CPERMSG    CHAR(4) BDY(WORD),    CONTAINS ERROR CODE
  8 CPMSG      CHAR(5) BDY(WORD),    MESSAGE INDEX
 6  *          CHAR(3),              RESERVED
 6  CPRRBGIP  PTR,                   SAVE ERROR RESTART STATUS
 6  CPRRBGIL  PTR(15),               OF RECURSIVE ROUTINES
 6  *         PTR(15),               RESERVED
 6  CPERIRP   PTR,                   REST PARMS POINTER
 6  CPERREP   PTR,                   END OF REPLY POINTER
 6  CPERIRL   PTR(15),               LENGTH OF REST PARMS IN INP
 6  CPERRL    PTR(15),               LENGTH OF CORRECTION REPLY
 6  CPINSPTR  PTR,                   PTR TO PARMLIST FOR MSGLIB
```

STACK CONTROL AREA

```
  3  CPSTKCTL,
   4   CPBEGSTK   PTR  (31),             BEGIN OF STACK
   4   CPENDSTK   PTR  (31),             END OF STACK
   4   CPCURSTK   PTR  (31),             CURRENT STACK ENTRY
```

AREA FOR SIT PREPARATION

```
  3  CPSITPRP   CHAR(100),              SIT PREPARATION TABLE
```

AREA FOR SIF & STACK

```
  3  CPSIF      CHAR(200),              AREA FOR SIT ENTRIES

  3  CPVTBL     CHAR(300),              AREA FOR VALUE TABLE ENTRIES

  3  CPSTACK    CHAR(16*LENGTH(CPSTKENT)),    AREA FOR STACK
```

INTERNAL SAVE AREA

```
  3  CPINTSV    CHAR(72)  BDY(WORD),      INTERNAL SAVEAREA
  3  CPSAVE     CHAR(72),                 CA SAVEAREA
```

IJPARSER WORK AREA

```
  3  CPWORK     CHAR(100),              VT PREP AREA FOR PAREP, PREP
```

INTERFACE BETWEEN MESSAGELIB MODULE AND ERROR ROUTINE

```
  3  CPMESSGL   CHAR(50) BDY(WORD),
   4 CPLGMSG1  FIXED(31),               LENGTH OF FIRST INSERT VAL
    6 CPZERINS   CHAR(2)  BDY(WORD),    END INDICATOR
   4 CPINSRT1   CHAR(20) BDY(WORD),     FIRST INSERT VALUE
   4 CPLGMSG2  FIXED(31),               LENGTH OF SECOND INSERT VAL
    6 CPONEINS   CHAR(2)  BDY(WORD),    END INDICATOR
   4 CPINSRT2   CHAR(20) BDY(WORD),     SECOND INSERT VALUE
   4 CPTWOINS   CHAR(2),                INSERT END INDICATOR
 2  CPREST      CHAR(2048 - LENGTH(CPCBGENP)),
 2  CPEND       CHAR(0);                DUMMY END OF CPCB
```

END OF CPCB

Chapter 19. PARSER: Diagnostic Aids

Message Cross-Reference

The following table shows the parser messages and the modules which issue the messages. All messages are printed by the message module IJPAPMSG.

4D02I	IJPARS	IJPADIRS				
4D03I	IJPARS	IJPANXTP				
4D04I	IJPANXTP					
4D05I	IJPANXTP					
4D06I	IJPANXTP					
4D07A	IJPANXTP					
4D08I	IJPANXTP					
4D09I	IJPANXTP					
4D10I	IJPARS	IJPANXTP	IJPADEFT	IJPASIFN	IJPAVALD	
4D12I	IJPARS					
4D13I	IJPARS					
4D14I	IJPARS					
4D15I	IJPARS					
4D16I	IJPARS					
4D17I	IJPARS					
4D18I	IJPARS					
4D19I	IJPARS	IJPASIFN				
4D20I	IJPARS					
4D21I	IJPARS					
4D22I	IJPARS					
4D23I	IJPARS					
4D24I	IJPARS					
4D25I	IJPARS					

Chapter 20. LSERV

Introduction

The Label Service (LSERV) program consists of one module: IJBLSERV.

This program displays on SYSLST the contents of the label information area. This label information is written into that area by job control based on the following job control statements:

- For disk label information: DLBL, EXTENT
- For tape label information: TLBL

For detailed information about the internal organization of the label area, see the *Diagnosis Reference: Initial Program Load and Job Control.*

The format of the label information records is described in Figure 19 on page 364 and Figure 20 on page 365. LSERV assumes that format.

The label subarea in which the records are written depends on the user-specified job control OPTION statement. LSERV is executed by the // EXEC LSERV job control statement from either SYSLOG or SYSRDR. See the publication *VSE/ESA System Utilities* for a description of how to use LSERV. If LSERV runs real, the partition must have at least 8K bytes of real storage allocated.

Program Design and Organization Information

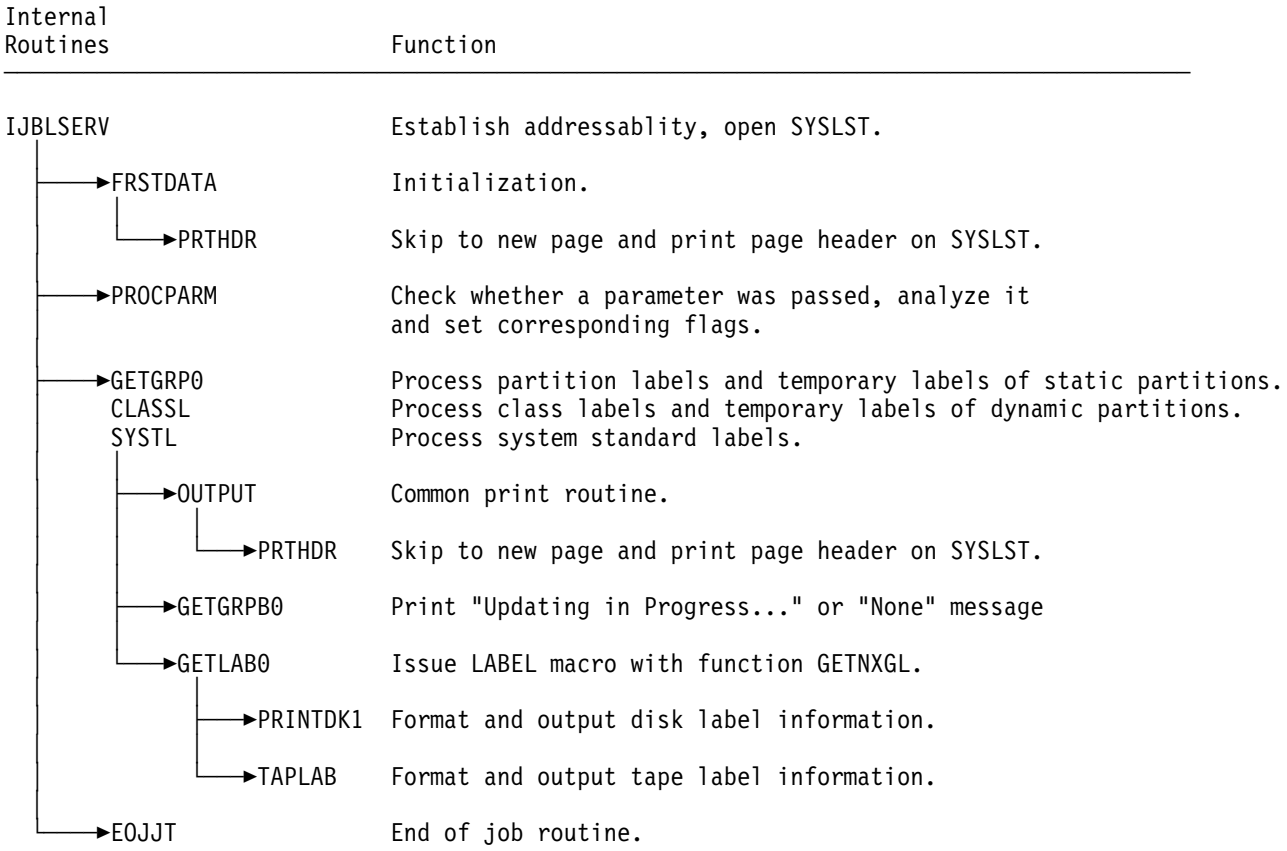
```
Phase name: LSERV

Called by: User (EXEC LSERV)

Phases called: $IJBSLA (via LABEL macro)

Data areas used: AVRADR    Volume information area
                 DSKLABLE  Disk label DSECT
                 LPLDCT    Label Parameter List
                 PCEADR    Partition related control blocks
                 SYSCOM    System Communication Region
                 TAPLABEL  Tape label DSECT

Messages issued: (on SYSLST): *** WARNING,WARNING: UPDATING IN PROGRESS FOR
                              *** THIS AREA, DATA NOT AVAILABLE OR OBSOLETE
                              1L90I, 1L91I, 1L92I

External input: Label information in SLA (System Label Area)

External output: Listing of all labels

Exits: EOJ (SVC 14)

Entry: IJBLSERV
```

Function Description: LSERV gets the label information from the symbolic label access routine $IJBSLA. The LABEL macro, which is described in the *Diagnosis Reference: IPL and Job Control,* is used for this purpose. Two functions may be performed by LSERV:

1. All label subareas are located sequentially to see if they contain any label information.
2. If there is label information, the label records are requested sequentially from symbolic label access routines.

All required control information for symbolic label access routines is contained in the LABEL PARAMETER LIST (LPL).

If a defined label subarea contains no label information records, a message is printed indicating that no records are present in this area; if a defined label subarea contains only data-secured file label information records, nothing is printed. Otherwise, the label information records (except for data-secured files) are formatted and printed on SYSLST. For examples of LSERV output, see *VSE/ESA System Utilities*.


Internal Routines
Function
IJBLSERV
Establish addressablity, open SYSLST.
FRSTDATA
Initialization.
PRTHDR
Skip to new page and print page header on SYSLST.
PROCPARM
Check whether a parameter was passed, analyze it and set corresponding flags.
GETGRP0
Process partition labels and temporary labels of static partitions.
CLASSL
Process class labels and temporary labels of dynamic partitions.
SYSTL
Process system standard labels.
OUTPUT
Common print routine.
PRTHDR
Skip to new page and print page header on SYSLST.
GETGRPB0
Print "Updating in Progress..." or "None" message
GETLAB0
Issue LABEL macro with function GETNXGL.
PRINTDK1
Format and output disk label information.
TAPLAB
Format and output tape label information.
EOJJT
End of job routine.

Data Area Information

Label Information Record Formats

TAPE	Field		Length decimal	Offset hex.
Key	1	Key Field	8	0
	2	Internal Sequence Number (Not Used)	1	8
Data	1	File Name	7	9
	2	Not Used	1	10
	3	File Identifier	17	11
	4	File Serial Number	6	22
	5	Volume Sequence Number	4	28
	6	File Sequence Number	4	2C
	7	Generation Number	4	30
	8	Version No. of Generation	2	34
	9	Creation Date	6	36
	10	Expiration Date	6	3C
	11	File Security Number	1	42
	12	Block Count	6	43
	13	System Code	13	49
	14	Flag Bytes	2	56

Figure 19. Format of the Label Information Records for Tape

DISK	Field		Length decimal	Offset hex
Key	1	File Name	7	0
	2	Internal Sequence Number	1	7
Data	1	DLBL—Extent Indicator	1	8
	2	File Name	7	9
	3	DA—IS Indicators	1	10
	4	File Identifier	44	11
	5	Format Identifier	1	3D
	6	File Serial Number	6	3E
	7	Volume Sequence Number	2	44
	8	Creation Date	3	46
	9	Expiration Date	3	49
	10	Retention Period	2	4C
	11	Open Code	1	4E
	12	System Code (see Note 1)	13	4F
	13	Volume Serial Number	6	5C
	14	Extent Type	1	62
	15	Extent Sequence Number	1	63
	16	Extent Lower Limit	4	64
	17	Extent Upper Limit	4	68
	18	Logical Unit Address	2	6C
	19	Flag Bytes/Reserved	2	6E
		Another extent if present for DA or IS files		

Figure 20. Format of the Label Information Records for Disk

Notes:

This field is processed only when VSAM is used or when the BLKSIZE parameter of the DLBL statement is used. For BAM BLKSIZE parameter the contents of field 12 are:

12	Not used	11	4F
	blocksize	2	5A

For sequential disk files, a complete block is repeated for each additional extent. For direct access or ISAM files, only fields 13 through 19 are repeated for each extent.

When VSAM is used, the contents of field 12 are:

12	Owning Catalog Filename Not used Buffer space to be allocated	7 2 4	4F 56 58

For VSAM files, there may be an additional Label Information Record following the VSAM Label Record:

DISK (VSAM only)	Field		Length (hex)	Offset
Key	1 2	File Name Int. Seq. Number	7 1	0 7
Data	1 2 3 4	File Name Disposition Records Record Size	1 1 8 4	7 9 C 14

Figure 21. Format of the Additional Label Information Record

Index

Special Characters

A

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

U

W

Communicating Your Comments to IBM

IBM VSE/Enterprise Systems Architecture
VSE Central Functions
Serviceability Aids
Diagnosis Reference
Version 6 Release 1

Publication No. SC33-6327-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF form and either send it postage-paid in the United States, or directly to:

  IBM Deutschland Entwicklung GmbH
  Department 3248
  Schoenaicher Strasse 220
  D-71032 Boeblingen
  Federal Republic of Germany

- If you prefer to send comments by FAX, use this number:
  - (Germany): 07031-16-3456
  - (Other countries): (+49)+7031-16-3456
- If you prefer to send comments electronically, use this network ID:

  INTERNET: s390id@de.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

IBM VSE/Enterprise Systems Architecture
VSE Central Functions
Serviceability Aids
Diagnosis Reference
Version 6 Release 1

Publication No. SC33-6327-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-6327-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

File Number: S370/S390-37
Program Number: 5686-066

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6327-00