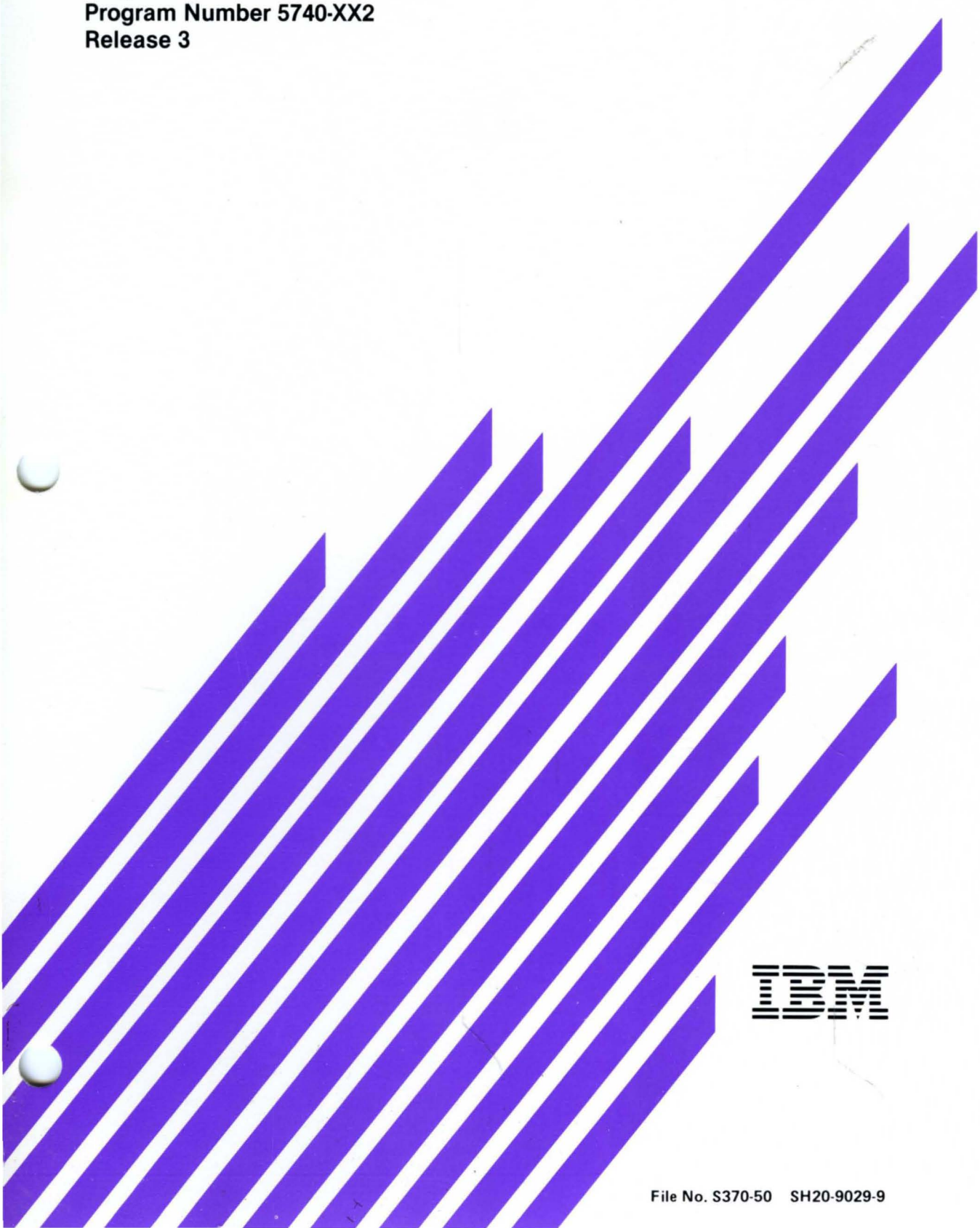
IMS/VS Version 1
Utilities Reference Manual

Program Product
Program Number 5740-XX2
Release 3



IBM

File No. S370-50 SH20-9029-9

IMS/VS Version 1
Utilities Reference Manual

Program Product
Program Number 5740-XX2
Release 3

File No. S370-50 SH20-9029-9

Tenth Edition (June 1986)

This is a major revision of, and makes obsolete, SH20-9029-8, and its technical newsletter, SN20-9406.

This edition applies to Version 1, Release 3 of IMS/VS, Program Product 5740-XX2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370, 30xx, and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

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 program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

UTILITY INDEX

* These utilities do not support CICS/VS.

HOW TO USE THIS PUBLICATION

This publication is designed for system programmers, application programmers, system analysts, and computer operators who require a knowledge of how to execute the Information Management System/Virtual Storage (IMS/VS) utility programs and IMS/VS Fast Path utility programs under the operating system.

Effective use of this publication requires an understanding of:

- Organization of this publication as a whole.
- Organization of each utility program description.
- Prerequisite publications.
- Associated publications.
- A description of IMS/VS publications, which appears in IMS/VS General Information Manual.

These topics are explained below.

Message Format Service utilities are described in IMS/VS Message Format Service User's Guide and the Security Maintenance utility is described in IMS/VS Installation Guide.

ORGANIZATION OF THE PUBLICATION

This publication contains six parts, each consisting of one or more chapters that explain related utilities.

Part 1, "Generation Utilities," has four chapters that describe the programs used to define a data base (DBDGEN); to define an application program (PSBGEN); to define the application control blocks library (ACBLIB) that contains data base and program descriptions; and to define the dynamic allocation parameter list (DFSMDA). The chapters are:

- Chapter 1, "Data Base Description (DBD) Generation"
- Chapter 2, "Program Specification Block (PSB) Generation"
- Chapter 3, "Application Control Blocks (ACB) Maintenance Utility"
- Chapter 4, "Dynamic Allocation Macro (DFSMDA)"

Part 2, "Online Change Processing," has one chapter that describes the utility provided to copy the contents of the staging libraries to the inactive libraries when adding, changing, or deleting IMS/VS resources online. This chapter is:

- Chapter 5, "Online Change Utility"

Part 3, "Data Base Utilities," has three chapters that describe the utilities used for reorganizing and recovering data bases and the facility that allows a user to implement these utilities in a specific manner. The chapters are:

- Chapter 6, "Data Base Reorganization/Load Processing"
- Chapter 7, "Data Base Recovery Utilities"
- Chapter 8, "Utility Control Facility"

Part 4, "IMS/VS System Log Utilities," has two chapters that describe the utilities used for analysis and recovery of the system log data. The chapters are:

- Chapter 9, "Log Maintenance Utilities"
- Chapter 10, "Log Data Formatting Utilities"

Part 5, "Performance Reporting and Service," has two chapters that describe the utilities used to produce performance-related summary reports; to copy messages onto a system output device; and to verify compatibility of system definitions for IMS/VS systems in a multisystem environment. The chapters are:

- Chapter 11, "Performance Reporting Utilities"
- Chapter 12, "System Service Utilities"

Part 6, "IMS/VS Fast Path," describes in two chapters the utilities used to assist in the creation, maintenance, and recovery of IMS/VS Fast Path main storage data bases (MSDBs) and data entry data bases (DEDBs). The chapters are:

- Chapter 13, "IMS/VS Fast Path MSDB Utilities"
- Chapter 14, "IMS/VS Fast Path DEDB Utilities"

ORGANIZATION OF UTILITY DESCRIPTIONS

Utility descriptions are organized, as much as possible, the same way to enable you to find information easily. Most utilities are described this way:

- Introduction to and description of the utility's functions.
- Job control statements and utility control statements. A brief explanation for the control statements used to execute the utility is included.
- Input and output (including return codes and significant output messages) used and produced by the utility.
- Examples of using the program, including the job control statements and utility control statements.

PREREQUISITE PUBLICATIONS

The reader should be familiar with OS/VS and its system generation, telecommunications, and the access methods used by IMS/VS. The prerequisite publications are:

- IMS/VS General Information Manual, GH20-1260
- IMS/VS Data Base Administration Guide, SH20-9025
- IMS/VS System Administration Guide, SH20-9178
- IMS/VS Application Programming, SH20-9026

ASSOCIATED PUBLICATIONS

The additional publications associated with this publication are:

- IMS/VS Primer Function Installation Guide, SH20-9208
- IMS/VS Operations and Recovery Guide, SH20-9209
- IMS/VS Release Guide for Version 1 Release 3, SH20-9207
- IMS/VS System Programming Reference Manual, SH20-9027
- IMS/VS Installation Guide, SH20-9081
- IMS/VS Operator's Reference Manual, SH20-9028
- IMS/VS Messages and Codes Reference Manual, SH20-9030
- Message Format Service User's Guide, SH20-9053
- IMS/VS Program Logic Manual, LY20-8069
- IMS/VS Failure Analysis Structure Tables (FAST) for Dump Analysis, LY20-8050
- IMS/VS Diagnostic Aids, LY20-8063
- OS Sort/Merge: Programmer's Guide—Program Number 5734-SM1, SC33-4007
- OS/VS Sort/Merge: Programmer's Guide—Program Number 5740-SM1, SC33-4035
- OS/VS Utilities, GC35-0005
- OS/VS1 Access Method Services, GC26-3840
- OS/VS2 Access Method Services, GC26-3841
- OS/VS Access Method Programmer's Guide, GC26-3838
- OS/VS JCL Reference, GC28-0618
- OS/VS2 JCL Reference Manual, GC28-0692
- OS/VS DOS/VS VM/370 Assembler Language, GC33-4010
- IMS/VS Data Base Recovery Control: General Information, GH35-0010
- IMS/VS Data Base Recovery Control: Guide and Reference, SH35-0027
- OS/VS2 System Programming Library: Service Aids, GC28-0674

In addition, the following microfiche is referred to:

- IMS/VS Pseudo Module Listing, LJB6-0004

CODING CONVENTIONS

This book uses the following conventions in showing coding formats and examples:

- Brackets [] indicate optional parameters.
- Underscored type indicates a default option. IMS/VS automatically assumes that the underlined item is the choice if none of the items is coded.
- Braces {} indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar represent alternative items. No more than one of the items may be selected.
- An ellipsis ... indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, spaces, periods, etc.) must be coded as shown.
- UPPERCASE TYPE indicates the exact characters to be entered. These items must be entered exactly as shown.
- Lowercase type indicates values that you supply.
- The lowercase b indicates one blank position.
- The control statements are free form. Operation codes must begin after column one. Operands must follow an operation code or prior operand. The first operand must be separated from the operation code by at least one blank column. Each operand should be separated from the previous operand by a comma. Operands may be continued on subsequent statements, but must start in column 16 on the continuation statement.

SUMMARY OF AMENDMENTS

RELEASE 3, JUNE 1986

PROGRAMMING CHANGES

- The Log Archive utility now supports CICS.
- A WAIT parameter has been added to the TYPE=RECON statement of the Dynamic Allocation macro.
- Starting and ending date specifications have been added to the Analysis Report of the Log Data Formatting Utilities chapter.

SERVICE CHANGES

Release 3 has been revised to reflect technical and editorial changes, especially the following:

- The Log Recovery section in Chapter 9
- Figure 77, Conditions that terminate the Batch Backout utility

RELEASE 3, FEBRUARY 1984

NEW PROGRAMMING FACILITIES

- During DBD generation for a Fast Path DEDB, the user can specify from 1 to 127 segment types and can optionally specify up to 8 subset pointers.
- The Log Archive utility is a new utility that copies the Online Log Data Set (OLDS) to the System Log Data Set (SLDS). It also copies batch DASD SLDSs to a tape SLDS, permitting release or reuse of the DASD space.
- The Online Change utility is a new utility that allows IMS/VS resources to be added, changed, or deleted without stopping IMS/VS. This utility copies the contents of one partitioned data set to another partitioned data set.
- The DEDB Area Data Set Create utility is a new utility that creates one or more copies from multiple DEDB area data sets during online operation.
- The DEDB Area Data Set Compare utility is a new utility that compares the identity of physical records of two or more data sets of an area during online operation.
- The System Log Recovery utility has been replaced with the Log Recovery utility (DFSULTR0). In addition to DUP and REP modes of operation available in the old utility, CLS mode of operation has been added to the Log Recovery utility. This mode closes the Online Log Data Set (OLDS), corrects DEDB incomplete log chains, and identifies PSBs requiring batch backout.
- The Batch Backout utility can now use one or more OLDSs and/or System Log Data Sets (SLDSs) as input.

- The DEDB Log Tape Check and Copy utility is no longer supported, but its logic is contained in the Log Recovery utility.
- The Log Tape Merge utility is now called Log Merge utility.
- The Fast Path Log Tape Analysis utility is now called Fast Path Log Analysis utility.
- The System Log Terminator utility has been eliminated.

SERVICE CHANGES

- The former Chapters 5 through 13 have been renumbered 6 through 14. Chapter 5, which covers the Online Change utility, is new.

RELEASE 2, SEPTEMBER 1982

NEW PROGRAMMING FACILITIES

- Several new execution-time parameters have been added to the specification of the DBRC= parameter on the following utilities:
  - Data Base Image Copy
  - Data Base Change Accumulation
  - Data Base Backout
  - System Log Recovery
  - System Log Terminator
  - DEDB Log Tape Check and Copy

RELEASE 2, MARCH 1981

NEW PROGRAMMING FACILITIES

- The operands that can be specified on the DEVICE= keyword in the DBD DATASET and AREA statements have been extended to support the 3375 and 3380 direct access storage devices.
- The PROCOPT=GO keyword in the PCB has two new options. These new options prevent IMS/VS from forcing read-only application programs to terminate abnormally.
- A new exit routine (DFSERA60) has been added. It formats the records produced when trace tables are written to the system log.

PROGRAMMING CHANGES

- DD statements for the DBRC RECON data sets have been added to JCL examples, as appropriate, throughout the manual.
- A new utility control statement (NOSEQCK) has been added to the Data Base Recovery utility. It allows you to specify that sequence checking on input log tapes not be performed.
- Changes have been made to the output of the program isolation trace record format and print module (DFSERA40).

- DBRC=NO can now be specified on the EXEC statement of the following utilities:
  - System Log Terminator
  - System Log Recovery
  - DB Change Accumulation
  - DEDB Log Tape Check and Copy
  - Batch Image Copy when PGM=DFSUDMP0 is specified on the EXEC statement

  DBRC=NO overrides the system definition inclusion of DBRC (Data Base Recovery Control feature) and specifies that these utilities are not to exit to DBRC.

VERSION 1, RELEASE 1.6, JULY 1980

- The Multiple Systems Verification utility verifies incompatibility between IMS/VS systems that use the VTAM link connection.
- The Multiple Systems Verification utility does not support MSC Directed Routing or Intersystem Communication (ISC).
- The STEPLIB and DFSRESLB DD statements have been added to the utility examples where necessary. The STEPLIB DD statement points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

VERSION 1, RELEASE 1.5, SEPTEMBER 1978

- For OS/VS2 MVS, the Dynamic Allocation and Deallocation macro (DFSMDA) builds a parameter list (for IMSVS.RESLIB) naming data base data sets that can participate in dynamic allocation, thereby eliminating the need to initially allocate these data sets through control region JCL. The DC Monitor log tape data set and Fast Path DEDB areas can also participate in dynamic allocation and deallocation.
- The Data Base Surveyor utility feature (DFSPRSUR), when run against all or part of an HDAM or HIDAM data base, produces a report that reveals areas needing reorganization or containing free space. PSBGEN supports this utility with the new OLIC= parameter, which provides authorization for this utility to be run against the specified data base.
- The Partial Data Base Reorganization utility (DFSPRCT1) reorganizes specific areas of an HDAM or HIDAM data base.
- The Online Data Base Image Copy utility (DFSUCIP0) can create an image copy of a data base while the data base is allocated to and being used by the online system. PSBGEN supports this utility with the new OLIC=parameter, which provides authorization for this utility to be run against the specified data base.
- For Fast Path Data Entry Data Bases (DEDBs), DBDGEN supports definition of up to 240 areas, 8 segment types, 255 fields per segment type and 1000 fields.
- For Fast Path PSBs, PSBGEN supports:
  - the inclusion of alternate PCBs and DL/I data base PCBs.
  - a special meaning for PROCOPT=P for DEDBs, which causes a new status code to indicate the best time for a SYNC

call and positioning to be maintained over the SYNC processing.

  - PROCOPT values of G, I, R, D, and A for dynamic terminal-related MSDBs.

  - multiple positioning for DEDBs with more than 2 dependent segments.

- The track recovery option of the Data Base Recovery utility (DFSURDB0) can be performed on a 3850 Mass Storage System.

- The use of the IMS/VS System Log Terminator utility (DFSFLOT0) may be decreased because the IMS/VS system log tape can be closed online by restart processing.

- The File Select and Formatting Print Program (DFSERA10) supports two functions, NEGOF and COPY, which allow log records to be selected by userid for printing or copying.

VERSION 1, RELEASE 1.4, JANUARY 1978

DC MONITOR REPORT PRINT PROGRAM

NEW PROGRAMMING FACILITY

Three new reports are described:

- Latch Conflict Counters

- Region Number—Job Number Cross-Reference

- Run Profile

Also, new IWAIT counts appear in two reports: IWAITs for PI enqueue in the Region IWAIT and Program I/O reports; wait-for-input IWAITs in the PROGRAM I/O report.

DB REORGANIZATION/LOAD PROCESSING

Specification Change

- The DFSVSAMP DD statement is required by several utilities for ISAM and OSAM data sets.

- The recommended and required values for LRECL= on DFSURWF1, DFSURWF2, and SORTIN DD statements are changed.

DBDGEN AND PSBGEN

Service

Many technical corrections and clarifications are made in Chapters 1 and 2. Especially:

- A maximum of 1000 FIELD statements for a secondary index data base

- The discussion of BLOCK= and SIZE= on the DATASET statement

- The operands unique to the AREA statement

- The maximum record lengths for BYTES= on the SEGM statement

- No default for PTR= on the LCHILD statement for an index target segment type
- The use of U and M on the FIELD statement of MSDBs and DEDBs
- Secondary indexing DBD examples
- The use of PROCOPT= values on the PCB and SENSEG statements

FILE SELECT AND FORMATTING PRINT PROGRAM

Specification Change

- A keyword is added to the OPTION statement: DDNAME= specifies the output data set to be used by DFSERA50.
- A keyword is deleted from the CONTROL statement: MOTION=.

MISCELLANEOUS SERVICE

- The introduction to the System Log Recovery Utility Program is clarified.
- The description of log record selection by the Log Tape Merge Utility is clarified.
- Descriptions of MSDB and DEDB utility return codes are added.

EDITORIAL SERVICE

- Cross-references to other IMS/VS publications are corrected for accuracy and ease of use.
- All message descriptions are deleted; they are described in IMS/VS Messages and Codes Reference Manual.

VERSION 1, RELEASE 1.4

NEW PROGRAMMING FEATURE

The Fast Path feature provides data base and data communication facilities for applications requiring high transaction rates but needing only simple data base structures. The Fast Path feature uses functions of the Data Communication feature and operates in existing telecommunication networks.

Fast Path provides two new types of data bases that are accessed with standard DL/I calls and, optionally, with Fast Path DL/I calls. The feature includes a message-handling facility to expedite the processing of Fast Path messages.

The IMS/VS Fast Path feature includes two main storage data base (MSDB) utilities; four data entry data base (DEDB) utilities; and a log data formatting utility:

- MSDB Dump Recovery utility
- MSDB Maintenance utility
- DEDB Initialization utility
- DEDB Sequential Dependent Scan utility
- DEDB Sequential Dependent Delete utility

- DEDB Direct Reorganization utility
- Fast Path Log Tape Analysis utility

The Fast Path feature includes new keywords and parameters for the IMS/VS DBD generation and PSB generation.

SERVICE CHANGES

- A procedure for recovering invalid logical relationship pointers in a data base has been added to "HD Reorganization/Unload Utility (DFSURGU0)" section.
- A description of an optional SNAP utility control statement for the DB Prefix Update utility has been added.
- A description of DB Recovery utilities has been rewritten. A new JCL example has been added.
- A clarification of the execution order of the Utility Control Facility (UCF) control statements has been added.
- Utility Control Facility (UCF) control statements have been modified and updated.
- The description of the "Log Format" section has been modified.
- A description of the JCL Requirements for the System Log Recovery utility and a new JCL example have been added.
- The description of the control statements for the System Log Recovery utility has been modified.
- Sample reports for the Statistical Analysis utility have been modified.
- The descriptions of control statements for the File Select and Formatting Print program have been added.
- A description of the Program Isolation Trace Record Format and Print Module has been rewritten and a sample output has been added.
- The description of DC Monitor reports has been modified.
- The description of the Program Isolation Trace Report utility has been rewritten.
- Several JCL errors in the examples on DB Reorganization/Load, DB Recovery, and UCF have been corrected.

VERSION 1, RELEASE 1.2

This publication has been revised to reflect technical and editorial changes made for Release 1.2.

TECHNICAL CHANGES

- Multiple Systems Coupling Feature

  The Multiple Systems Coupling feature allows a user to define a configuration consisting of up to 255 interconnected IMS/VS systems running on any combination of OS/VS1 and OS/VS2 systems.

  This feature includes two utilities and three reports for the DC Monitor Report Print utility:

  - Log Tape Merge utility

- Multiple Systems Verification utility
- MSC Traffic Report
- MSC Summaries Report
- MSC Queueing Summary Report

- 3350 Direct Access Storage Device

  The 3350 may now be specified for data base and message queue data set residence.

- Statistics Sort

  The Sort and Edit Passl module, DFSISTO, which is part of the Statistical Analysis utility program, includes a new option that shortens the time used to sort and produce statistical reports.

EDITORIAL CHANGES

This manual has been organized into four parts. See the "How to Use This Publication" section for organization of each part. Other organization changes are:

- Information on the Interactive Query Facility (formerly Chapter 10) has been moved to IMS/VS System Programming Reference Manual.
- Information on the DL/I Test program (formerly Appendix C) has been moved to IMS/VS Application Programming Reference Manual.
- Information on Insert, Delete, and Replace Rules in Chapter 1 has been deleted. Duplicate information is in IMS/VS System/Application Design Guide.

VERSION 1, MODIFICATION LEVEL 1.1

This release reflects technical changes to this publication in support of the following new functions:

- Utility Control Facility (UCF)
- Data Base Monitor Report Print Program

VERSION 1, MODIFICATION LEVEL 1

This release reflects technical changes to this publication in support of the following new and/or enhanced IMS/VS functions:

- Generalized Sequential Access Method (GSAM)

  This new access method allows batch programs to access OS BSAM and VSAM ESDS data sets. See the discussion of the DBD statement and the DATASET statement in Chapter 1, and the discussion of the GSAM PCB statement in Chapter 2.

- Enhancements to the Virtual Storage Access Method (VSAM)

  A new parameter has been added to the DBD statement to allow the user to specify VSAM password protection for a data base. A new parameter has been added to the PSBGEN statement to allow the user to specify a return code and abend option if an input/output error occurs on a data base during application program execution. See the discussion of the DBD statement and PSBGEN statement in Chapters 1 and 2, respectively.

- Response Alternate PCBs

Alternate PCBs may now be defined to meet the requirements of responding to terminals operating in response mode, conversational mode, or exclusive mode. Formerly, responses to these terminals had to be made to the I/O PCB. See the discussion of the alternate PCB statement in Chapter 2.

- Utility Control Facility

  Note: Information in this manual about the Utility Control Facility (UCF) is only for planning purposes until that facility is available.

- Partial Data Base/Data Set Recovery

  The new track recovery option of the Data Base Recovery program allows re-creation at the track level in the event of a permanent read/write error. (The data base access method must be enhanced VSAM.) See Chapter 5 for a discussion of the new track recovery option.

- DC Monitor Report Print Program

CONTENTS

FIGURES

PART 1. GENERATION UTILITIES

Part 1 has four chapters that describe the utilities used to assist in the creation and maintenance of IMS/VS data bases and application programs.

Chapter 1, "Data Base Description (DBD) Generation," describes the program (DBDGEN) that defines a data base to be used by an application program. Control statements used as input to DBDGEN and DBDGEN examples using HSAM, GSAM, HISAM, HDAM, HIDAM, MSDB, DEDB, INDEX, and LOGICAL are provided.

Chapter 2, "Program Specification Block (PSB) Generation," describes the program (PSBGEN) that defines an application program before execution. Control statements used as input to PSBGEN and PSBGEN examples are included.

Chapter 3, "Application Control Blocks (ACB) Maintenance Utility," describes the utility that defines the application control blocks library (ACBLIB), which contains data base and program descriptions. Control statement examples are included.

Chapter 4, "Dynamic Allocation Macro (DFSMDA)," describes the macro that defines data sets that are to participate in dynamic allocation and deallocation. Macro statements used as input to DFSMDA and DFSMDA examples are included.

CHAPTER 1. DATA BASE DESCRIPTION (DBD) GENERATION

OVERVIEW

This chapter describes the control statements used to create a Data Base Description (DBD). For a full understanding of this chapter, the reader must be familiar with the contents of IMS/VS Data Base Administration Guide.

This chapter contains three main sections. The first section is an overview of DBD generation control statements and their use in defining IMS/VS data bases. It also contains coding conventions and shows how a DBD generation is executed. The second section contains a detailed description of the control statements and their operands. The third section provides examples of different types of DBD generations. It also contains summaries of how different statements and operands apply for defining index and logical relationships.

INFORMATION SPECIFIED IN DBD GENERATION

All data bases must be defined through DBD generation prior to use by application programs. A DBD is the DL/I control block that contains all information used to describe a data base. Only one physical DBD should describe each physical data base, or user abend U850 or U853 may occur. At execution time, DL/I uses the DBD to create a set of internal control blocks. The DBD contains the following data base information:

- Segment types
- Physical and logical relationships between segment types
- Data base organization and access method
- Physical characteristics of the data base

OVERVIEW OF EACH TYPE OF DBD GENERATION

HSAM DBD Generation

During DBD generation for an HSAM data base, the user specifies:

- One data set group.
- The ddname of an input data set that is used when an application retrieves data from the data base.
- The ddname of an output data set used when loading the data base.
- From 1 to 255 segment types for the data base.
- From 0 to 255 fields within each segment type, with a maximum of 1000 fields within the data base.
- Optionally, the user can define a simple HSAM data base that can contain only one fixed-length segment type. When defined, no prefixes are built in occurrences of the segment type.

The user cannot specify for an HSAM data base:

- The use of hierarchic or physical child/physical twin pointers between segments in the data base.
- The use of logical or index relationships between segments.

GSAM DBD Generation

During DBD generation for a GSAM data base, the user specifies:

- One data set group.
- The ddname of an input data set that is used when an application retrieves data from the data base.
- The ddname of an output data set used when loading the data base.

The user cannot specify:

- SEGM and FIELD statements.
- The use of logical or index relationships between segments.

HISAM DBD Generation

During DBD generation for a HISAM data base, the user specifies:

- One to 10 data set groups.
- The ddname of one ISAM and one OSAM data set, or one VSAM key sequenced data set (KSDS) and one VSAM entry sequenced data set (ESDS) for each data set group.
- Optionally, the user can define a simple HISAM data base that can contain only one fixed-length segment type. When defined, no prefixes are built in occurrences of the segment type. The logical record length specified for a simple HISAM data base must be the same as the segment length specified.
- At least one segment type for each data set group, and a maximum of 255 segment types for the data base.
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the data base, one of which must be a unique sequence field in the root segment type for indexing root segment occurrences.
- A maximum of 32 secondary index relationships (optional) per segment type, and a maximum of 1000 for the data base.
- Logical relationships (optional) using symbolic pointer options when a segment in a HISAM data base points to another segment in a HISAM data base, and direct or symbolic pointer options when a segment in a HISAM data base points to a segment in an HDAM or HIDAM data base.
- Use of segment edit/compression exits to enable user supplied routines to manipulate occurrences of a segment type on their way to or from auxiliary storage (optional).

The user cannot specify:

- The use of hierarchic or physical child/physical twin pointers between segments in this HISAM data base.

HDAM DBD Generation

During DBD generation for an HDAM data base, the user specifies:

- The name of the user-supplied randomizing module used for placement of root segment occurrences.
- One to 10 data set groups.
- How free space is to be distributed in each data set group.
- The ddname of an OSAM or ESDS data set for each data set group defined.
- At least one segment type for each data set group, and a maximum of 255 segment types for the data base.
- Use of segment edit/compression exits (optional) to enable user-supplied routines to manipulate occurrences of a segment type on their way to or from auxiliary storage.
- The use of hierarchic or physical child/physical twin pointers between segments in the data base.
- Logical relationships (optional) between segments using direct address and/or symbolic pointer options.
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the data base.
- A maximum of 32 secondary index relationships (optional) per segment type and a maximum of 1000 for the data base.

HIDAM DBD Generation

During DBD generation for a HIDAM data base, the user specifies:

- One to 10 data set groups.
- How free space is to be distributed in each data set group.
- The ddname of an OSAM or ESDS data set for each data set group defined.
- At least one segment type for each data set group, and a maximum of 255 segment types for the data base.
- Use of segment edit/compression exits (optional) to enable user supplied routines to manipulate occurrences of a segment type on their way to or from auxiliary storage.
- A maximum of 32 secondary index relationships (optional) per segment type and a maximum of 1000 for the data base.
- The use of hierarchic or physical child/physical twin pointers between segments in the data base.
- Logical relationships (optional) between segments using direct address and/or symbolic pointer options.
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the data base, one of which must be a unique sequence field in the root segment type for indexing root segment occurrences.

MSDB DBD Generation

During DBD generation for a Fast Path MSDB, the user must specify:

- One data base name.

- One data set group.
- One segment type for the data base.
- From 0 to 255 fields within the data base.

The user cannot specify:

- A logical or index relationship between segments.
- Fields used with secondary indexes.

DEDB DBD Generation

During DBD generation for a Fast Path DEDB, the user must specify:

- One data base name.
- From 1 to 240 areas within a data base.
- From 1 to 127 segment types for the data base.
- From 0 to 255 fields for each segment type, with a maximum of 1000 fields within the data base, one of which must be a unique sequence field for the root segment type.
- The ddname or area name that is used to describe an area.

The user can optionally specify:

- Up to 8 subset pointers for each child type of the parent.

The user cannot specify:

- A logical or index relationship between segment types.
- Fields used with secondary indexes.

Index DBD Generation

Primary HIDAM Index DBD Generation

- Creates an index data base composed of one index segment type that indexes occurrences of the HIDAM root segment type.
- Contains one data set group. The user must specify the ddname of one ISAM and one OSAM data set, or the ddname of one KSDS.
- An index segment contains:
  - The sequence field key of the root segment occurrence it indexes.
  - In its prefix, a direct address pointer to the root segment occurrence.
- A primary HIDAM index DBD generation uses the following statements:

Quantity	Type
1	DBD
1	DATASET
1	SEGM
1	LCHILD
1	FIELD
1	DBDGEN
1	FINISH
1	END

Secondary Index DBD Generation

- Creates a secondary index data base comprised of from 1 to 16 index pointer segment types used to index target segment types in HISAM, HDAM, or HIDAM data bases.
- Contains one data set group. The user must specify the ddname of one KSDS only if all index pointer segment keys are unique, or the ddnames of one KSDS and one ESDS if index pointer segment keys are nonunique. Note that a secondary index must use VSAM.
- A secondary index DBD generation uses the following statements:

Quantity	Type
1	DBD
1	DATASET
1 to 16	SEGM
1 to 16	LCHILD
1 to 1000	FIELD
1	DBDGEN
1	FINISH
1	END

Logical DBD Generation: A logical DBD generation creates a logical data base comprised of logical segment types. A logical segment type is a segment type defined in a logical data base that represents a segment type or the concatenation of two segment types defined in a physical data base or data bases.

A logical DBD generation uses the following statements:

Quantity	Type
1	DBD
1	DATASET Defines a logical data set group
1-255	SEGM Each defines the name of a logical segment type, and the name of the segment type or types in physical data bases that are to be processed when a call is issued to process the logical segment type.
None	LCHILD The logical relationships used to create a logical data base must be defined in a physical data base or data bases.
None	FIELD All fields required for segments in a logical data base must have been defined in physical data bases.
1	DBDGEN
1	FINISH
1	END

DATA BASE DESCRIPTION RULES

Figure 1 shows the statement types used as input to the DBD generation utility to define a data base. Also included is the general use of each statement and the number of each type used for the nine types of DBD generations.

	Operation	General Use	Number used in each DBD generation								
			HSAM	GSAM	HISAM	HDAM	HIDAM	MSDB	DEDB	Index	Logical
	[PRINT] *	Controls printing of assembly listing if present	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1
	DBD	Defines data base name	1	1	1	1	1	1	1	1	1
[label]	DATASET	Defines a data set group within a data base	1	1	1-10	1-10	1-10	1	0	1	1
	AREA	Defines an area within a Fast Path data base	0	0	0	0	0	0	1-240	0	
	SEGM	Defines a segment type within a data set group or area	1-255	0	1-255	1-255	1-255	1	1-127	1**	1-255
	[LCHILD]	Defines a logical or index relationship between segment types	0	0	0-255	0-255	1-255	0	0	1**	0
	[FIELD] ****	Defines a field within a segment type	0-1000	0	1-1000	0-1000	1-1000	0-255	1-1000	1***	0
	[XDFLD] ****	Defines fields used with secondary indexes	0	0	0-1000	0-1000	0-1000			0	0
	DBDGEN	Indicates the end of DBD generation statements	1	1	1	1	1			1	1
	FINISH	Checks for successful DBD generation	1	1	1	1	1	1		1	1
	END	Indicates end of DBD generation input to the OS/VS assembler	1	1	1	1	1	1		1	1

* For operand information, see OS/VS and DOS/VS VM/370 Assembler Language, GC33-4010.

** Maximum of 16 for a secondary index data base.

*** Maximum of 1000 for a secondary index data base.

**** The maximum combined total of FIELD and XDFLD statements per DBD generation is 1000.

Figure 1. Summary of DBD Generation Statements

Figure 2 on page 8 summarizes the statement types in general form and identifies the operands that can be specified for each type of DBD generation.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
DBD	NAME=	(dbname1		R	R	R	R	R	R	R	R	R			
		,dbname2,...)							O						
	,ACCESS=	{HSAM / SHSAM}		R											
		(GSAM [,BSAM / VSAM])			R										
		{(HISAM [,ISAM / VSAM]) / (SHISAM[,VSAM])}				R									
		(HDAM [OSAM / ,VSAM])					R								
		(HIDAM [OSAM / ,VSAM])						R							
		(INDEX [ISAM / ,VSAM]	1						R						
		[,PROT / ,NOPROT]							O						
		,DOSCOMP)							O						
		LOGICAL								R					
		MSDB									R				
		DEDB										R			
	RMNAME=	(mod					R					R			
		,anch					O								
		,rbn					O								
		,bytes)					O								
	,PASSWD=	[YES / NO]			O	O	O	O	O						
DATASET or AREA	LOGICAL									R					
	or DD1=	ddname1		R	R	R	R	R	R			R			
	,DEVICE=	device		R		R	R	R	R			R			
	,MODEL=	model		O		O	O	O	O			O			

Figure 2 (Part 1 of 6). Summary of Statements and Operands Used by Data Base Type

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
DATASET or AREA (Cont.)	,DD2=	ddname2		R	O										
	,OVFLW=	ddname3	2			R			R						
	,BLOCK=	(blkfact1	3	O	O	O			O						
		,blkfact2)	3	O		O			O						
		— or —													
		size0	3				O	O							
	,SIZE=	(size1	3		O	O	O	O	O			R			
		,size2)	3			O			O						
	,RECORD=	(reclen1	3	O	O	O			O						
		,reclen2)	3	O	O	O			O						
	,RECFM=	,recfm1			R										
	,SCAN=	cyls					O	O							
	,FRSPC=	(fbff					O	O							
		,fspf)					O	O							
	,REL=	({NO / TERM[,fldnm] / FIXED[,fldnm] / DYNAMIC[,fldnm]})									R				
	,UOW=	(number1, overflow1)										R			
	,ROOT=	(number2, overflow2)										R			
SEGM	NAME=	segname1		R		R	R	R	R	R	R	R			
	,PARENT=	0	4	O		O	O	O	O	O		O			
		— or —													
		((segname2		R		R	R	R		R		R			
		,[SNGL / DBLE])					O	O				O			
		(Ipsegname											X	X	X
		,[VIRTUAL / PHYSICAL]											X	X	X
		,dbname1))											X	X	X

Figure 2 (Part 2 of 6). Summary of Statements and Operands Used by Data Base Type

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
SEGM (Cont.)	,BYTES=	(max bytes	5	R		R	R	R	R		R	R			
		,min bytes)	6			O	O	O				R			
	[,TYPE=	{DIR / SEQ}]	7									R			
	,FREQ=	frequency		O		O			O						
	[,POINTER= ,PTR=	({HIER / HIERBWD / TWIN / TWINBWD / NOTWIN}					O	O							
		{LTWIN / 'LTWINBWD}											X		X
		,LPARNT											X	X	X
		,CTR											X	X	X
		,PAIRED)											X	X	X
	,RULES=	([{P / L / V} {B / P / L / V} {P / L / V}]											X	X	X
		[FIRST ,LAST HERE])	8	O		O	O	O	O			O			
	,SOURCE=	((segname								R			X	X	X
		,[KEY DATA]								O			X	X	X
		,dbname)								R			X	X	X
		(,segname	9							R					
		,[KEY DATA]								O					
		,dbname))	9							R					
	,COMPRTN=	(routinename	6			O	O	O							
		[KEY DATA]	6			O	O	O							
		,INIT)	6			O	O	O							
	,SSPTR=	n										O			

Figure 2 (Part 3 of 6). Summary of Statements and Operands Used by Data Base Type

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
LCHILD	NAME=	(segname1	6			R	R	R	R				X	X	X
		,dbname)	6			R	R	R	R				X	X	X
	[,POINTER= ,PTR=]	SNGL	10						O				X	X	X
		DBLE											X	X	X
		NONE											X	X	X
		INDX	11				O	R							
		SYMB	12			O	O	O	O						
	,PAIR=	segname2											X	X	X
	,INDEX=	fldname							R						
	,RULES=	[FIRST LAST HERE]											X	X	X
FIELD	NAME=	(fldname1		R		R	R	R	R		R	R			
		,SEQ	13	O		O	O	O	O		O	O			
		[U M]) or	14	O		O	O	O	O		O	O			
		systrelfldname				O	O	O							
	,BYTES=	bytes		R		R	R	R	R		R	R			
	,START=	startpos		R		R	R	R	R		R	R			
	,TYPE=	X		O		O	O	O	O		O	O			
		C		O		O	O	O	O		O	O			
		P		O		O	O	O	O		O	O			
		H									O				
		F									O				
XDFLD	NAME=	fldname	6			R	R	R							
	,SEGMENT=	segname				O	O	O							
	,CONST=	char	15			O	O	O							

Figure 2 (Part 4 of 6). Summary of Statements and Operands Used by Data Base Type

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
XDFLD (Cont.)	,SRCH=	list1	15			R	R	R							
	,SUBSEQ=	list2	15			O	O	O							
	,DDATA=	list3				O	O	O							
	,NULLVAL=	value1				O	O	O							
	,EXTRTN=	name1				O	O	O							
DBDGEN				R	R	R	R	R	R	R	R	R			
FINISH				O	O	O	O	O	O	O	O	O			
END				R	R	R	R	R	R	R	R	R			

KEY
O = Optional
R = Required
X = Use for logical relationships

Notes:

1. A secondary index must use VSAM.
2. Required if ISAM/OSAM are the access methods. If VSAM is the access method for an index data base and the keys of all the index segments are unique, or if VSAM is the access method for a HISAM data base with only one segment type defined, OVFLW need not be specified. OVFLW is invalid for a simple HISAM data base.
3. When not specified by the user, DBDGEN generates value used.
4. The PARENT=keyword must be omitted, or PARENT=0 must be specified for the root segment type of a data base.
5. Maxbytes required for MSDB DBD; minbytes must not be specified.
6. Variable-length segments, segment edit/compression, logical relationships, and secondary indexing are invalid for simple HISAM data bases.
7. Type=SEQ is required on SEGM statements for the sequential dependent segment type.
8. Required when a segment type does not have a unique sequence field. The default value for DEDB dependent segments is HERE. For DEDB sequential dependent segments, FIRST is always used.

Figure 2 (Part 5 of 6). Summary of Statements and Operands Used by Data Base Type

9. Required when defining a concatenated segment type for a LOGICAL data base.

10. Required for primary index of HIDAM data base.

11. Required during a HIDAM DBD generation on the LCHILD statement that establishes the primary HIDAM index relationship. If PTR=INDX is specified for the target segment of a secondary index, the PTR= must be omitted or specified as PTR=SNGL on the LCHILD statement of the INDEX DBD.

12. If symbolic pointing is specified for the index target segment type when defining its physical data base, symbolic pointing must be specified in the secondary index for that segment type. If SYMB is specified for the target segment of a secondary index, then PTR=SYMB is specified on the LCHILD statement of the INDEX DBD also.

13. A unique sequence field is required for the root segment type of HISAM, HIDAM, SHISAM, DEDB, and nonterminal-related MSDB. A sequence field is not allowed for the sequential dependent segment of a DEDB. Root segment types in an HDAM data base must also have a key field defined, although the key need not be unique.

14. M not allowed for MSDB or DEDB DBDs.

15. The combined length of the constant, search and subsequence fields must not exceed 240 bytes.

Figure 2 (Part 6 of 6). Summary of Statements and Operands Used by Data Base Type

DBD Generation Input Deck Structure (Except for DEDB DBDs)

For all types of DBDs except DEDB DBDs, the DBDGEN program accepts ten types of control statements arranged in a specific order in the SYSIN input stream. Each control statement is described in detail in the following sections.

Figure 3 on page 14 shows the rules for structuring a DBD generation input deck. The order of statements shown is required for the nine types of DBD generation except DEDB. If included, the PRINT statement is the first statement in the input deck. When PRINT is not included, the DBD statement is first in the input deck. One or multiple DATASET statements follow the DBD statement and precede the DBDGEN statement. Each DATASET statement is followed by the SEGM, LCHILD, FIELD and XDFLD statements that define the segments, relationships between segments, and fields within segments in that data set group. Following each DATASET statement there must be at least one SEGM statement. When multiple SEGM statements follow a DATASET statement, they must be placed in hierarchic order.

FIELD statements follow the SEGM statement for the segment type within which they are defining fields. XDFLD statements follow a SEGM that defines a segment type that is an index target segment type for a secondary index. LCHILD statements follow a SEGM that defines a logical parent, HIDAM root, index target and index pointer segment types. LCHILD, XDFLD and FIELD statements need not be placed in any specific order behind a SEGM except when a FIELD statement defines a sequence field within a segment or when a secondary index relationship is being defined. When a FIELD statement defines a sequence field, that FIELD statement must precede any XDFLD statements or any other FIELD statements that follow a SEGM.

When a secondary index relationship is being defined, the LCHILD statement that establishes the relationship must be followed by the corresponding XDFLD statement with no intervening LCHILD statements between the two.

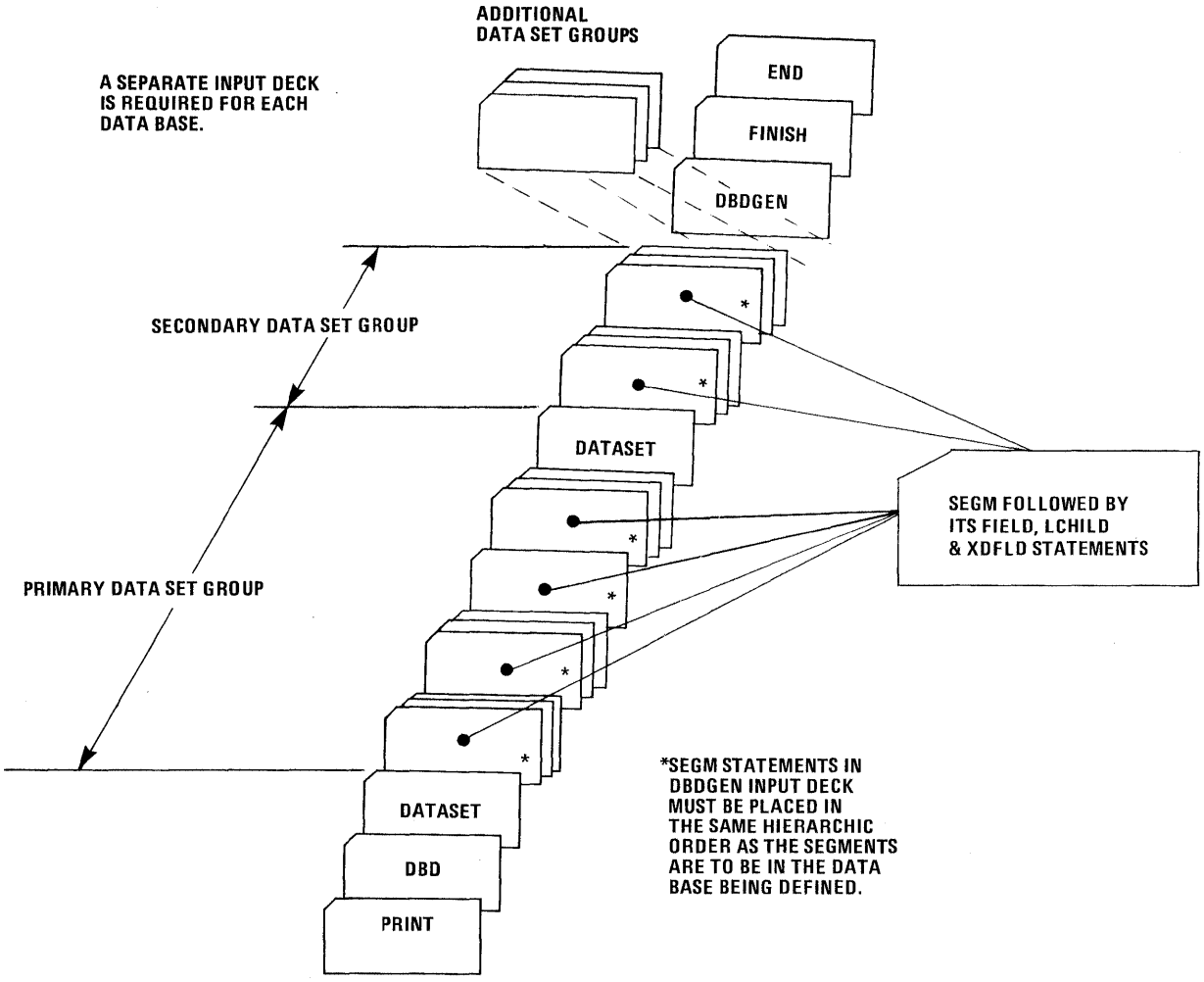


Figure 3. DBDGEN Input Deck Structure (Except DEDB)

DEDB DBD Generation Input Deck Structure

The input deck structure for a DEDB DBD generation is essentially the same as for the other types of DBD generation except that AREA statements are used instead of DATASET statements. All AREA statements must immediately follow the DBD statement. The SEGM statements and their associated FIELD statements follow the last AREA statement in hierarchic order.

Note that for DEDB DBD generation:

- The data set group concept does not apply.
- A secondary index is not permitted.
- Logical relationships between data bases are not permitted.
- LCHILD and XDFLD statements are not permitted.
- Sequential dependent segments cannot have dependents.

Figure 4 on page 15 shows the rules for structuring a DEDB DBD generation input deck.

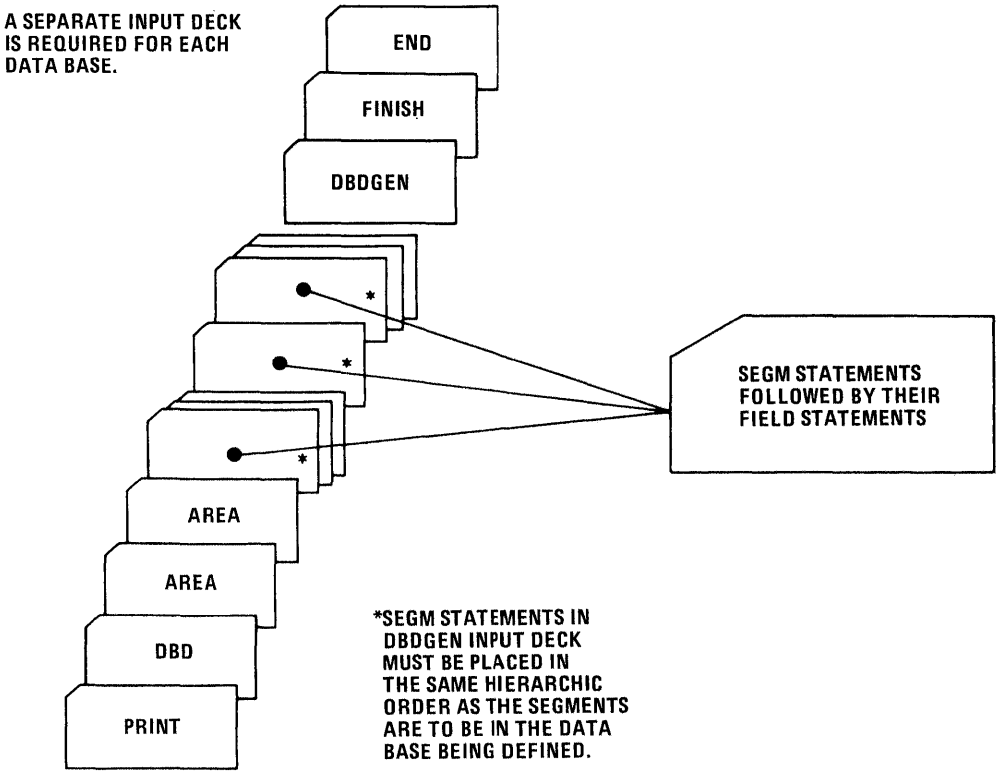


Figure 4. DEDB DBDGEN Input Deck Structure

DBD Generation Coding Conventions

DBD generation statements are assembler language macro instructions and therefore are subject to the rules contained in the publication OS/VS and DOS/VS VM/370 Assembler Language.

1. Each control statement must be identified by an operation code, called a "card-type code."

2. In the generalized format shown in the following descriptions of the control statements, the syntax conventions described in the Preface apply.

Execution of DBD Generation (JCL)

DBDGEN must be run as a normal operating system job after IMS/VS System definition. System definition causes the DBDGEN procedure to be placed in the IMSVS.PROCLIB library. To process a request for a DBDGEN, the DBD generation control statements must be created and appended to the following JCL (which invokes the DBDGEN procedure):

```
//DBDGEN  JOB  MSGLEVEL=1
//       EXEC    DBDGEN,MBR=
//C.SYSIN  DD  *

         DBD
         DATASET
         SEGM
         FIELD        DBD generation control cards
         LCHILD
         XDFLD
         DBDGEN
         FINISH
         END
/*
```

where:

keyword operand MBR=
is the name of the DBD to be generated. This name should be the same as the first name specified for the NAME= keyword on the DBD statement. The first data base name becomes the DBD member name and, in the case of a shared secondary index, the additional names are added as aliases. When a data base PCB relates to this DBD generation, one of the names specified in the NAME= keyword on the DBD statement must be the name used in the DBDNAME= keyword on the data base PCB statement. Except for a shared secondary index, the name used in the DBDNAME= keyword on the data base PCB statement must be the same as the name used in the MBR= keyword value.

The operating system start reader command used to read the preceding JCL must have access to IMSVS.PROCLIB. The IMS/VS procedure IMSRDR can be used in the following manner:

```
START IMSRDR,XXX,DCB=BLKSIZE=80
```

where:

XXX is the unit address of a card reader

Three types of printed output and a load module, which becomes a member of the partitioned data set named IMSVS.DBDLIB, are produced by a DBD generation. Each of these outputs is described in the following paragraphs.

CONTROL STATEMENT LISTING: This is a listing of the input statement images to this job step.

DIAGNOSTICS

Errors discovered during the processing of each control statement result in diagnostic messages. These messages are printed immediately following the image of the last control statement that is read. The message may reference either the control statement immediately preceding it or the preceding group of control statements. It is also possible that more than one message could be printed for each control statement. In this case, these messages follow each other on the output listing. After all the control statements have been read, a further check is made of the reasonableness of the entire deck. This may result in one or more additional diagnostic messages.

Any discovered error results in the diagnostic message(s) being printed, the control statements being listed, and the other outputs being suppressed. However, all the control statements are read and checked before the DBD generation execution is terminated. The link-edit step of DBD generation is not processed if a control statement error has been found.

ASSEMBLER LISTING

An OS/VS assembler language listing of the DBD macro expansion created by DBD generation execution is provided. The inclusion of an assembler language PRINT NOGEN statement can be used to eliminate a printout of this listing.

If the DBD generation is for a data base that uses VSAM as the operating system access method, a page in the assembler listing will provide recommended values for some of the parameters necessary to define the data sets of the data base to VSAM. Values other than those recommended may be desired for special reasons, such as performance improvement. If the control interval size is not specified (see SIZE= parameter later in this chapter), it defaults to the size recommended in this assembler listing. The following example shows the output produced for a HISAM data base. The parameters provided are in the format required for OS/VS Access Method Services control statements. The first DEFINE provides parameters for the key sequenced data set (KSDS) and the second DEFINE provides parameters for the entry sequenced data set (ESDS).

To provide a complete definition for a VSAM data set, the user must add parameters for data set name (NAME), space allocation (CYL), and volume assignment (VOLUMES) to those provided by DBD generation. Optional parameters such as FREESPACE and WRITECHECK may be included if desired.

If the /DBD command is used to allow an offline dump of a VSAM data base, SHARE OPTIONS(3) must be used in the VSAM DEFINE operation for the data sets of the data base.

Example of OS/VS Access Method Services parameters from DBD generation.

```
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,*
+*,      RECOMMENDED VSAM DEFINE CLUSTER PARAMETERS
+*,*
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,*                      *NOTE 1
+*,      DEFINE CLUSTER (NAME(DDI3I1) -
+*,*             INDEXED KEYS (10,6) -
+*,*             RECORDSIZE (680,680) -
+*,*             DATA (CONTROLINTERVALSIZE (4096))
+*,*  *NOTE 1: SHOULD SPECIFY DSNAME FOR DDI3I1
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,*                      *NOTE 2
+*,*    DEFINE CLUSTER (NAME(DDI301) NONINDEXED -
+*,*             RECORDSIZE (680,680) -
+*,*             CONTROLINTERVALSIZE (4096))
+*,*  *NOTE 2: SHOULD SPECIFY DSNAME FOR DDI301
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

The following example shows the DBDGEN input used to create the output in the previous example.

```
SEGM NAME=SEGB2,PARENT=((SEGA1)),BYTES=15,FREQ=3
FIELD NAME=(FLDB2,SEQ,U),BYTES=9,START=3,TYPE=C
SEGM NAME=SEGC1,PARENT=((SEGB2)),BYTES=20,FREQ=7
FIELD NAME=(FLDC1,SEQ,U),BYTES=10,START=4,TYPE=C
DBDGEN
FINISH
END
```

Segment flags are printed in DBD generation output to confirm what has been generated by that particular DBD generation. The flags, when interpreted, tell the user which pointer options were generated; the segment insert, delete, and replace rules specified; whether physical child pointers have been reserved in

this segment's prefix; and how many physical children are related to the segment. Segment flags appear in the output as an assembler language defined constant (DC) statement. The constant is defined as 8 hexadecimal digits followed by the comment, SEGMENT FLAGS. Each pair of digits in the constant is a hexadecimal byte. To interpret the constant, convert the first 6 digits to binary values, and the last 2 digits to decimal values as shown in Figure 5.

BYTE	CONVERTED VALUE	DESCRIPTION
0		POINTER POSITIONS GENERATED:
	1.......	CTR (Counter)
	.1......	Physical twin forward
	.11.....	Physical twin forward and backward
	...1....	Physical parent
	....1...	Logical twin forward
	....11..	Logical twin forward and backward
	......1.	Logical parent
	.1.....1	Hierarchic forward
	.11....1	Hierarchic forward and backward
1		SEGMENT PROCESSING RULES:
	10......	Insert physical
	01......	Insert virtual
	11......	Insert logical
	..10....	Insert nonsequential last
	..01....	Insert nonsequential first
	..11....	Insert nonsequential here at current position
	....10..	Replace physical
	....01..	Replace virtual
	....11..	Replace logical
	10	Delete physical
	01	Delete virtual
	11	Delete logical
	00	Bivirtual delete
2	..XX.XXX	Reserved
	1.......	Segment is paired
	.1......	Segment is a direct dependent in a FP DEDB
	....1...	Segment's parent has two physical child pointers; hierarchic pointers were not specified
3	0-254	Number of physical children of this segment pointed to by physical child pointers

Figure 5. Segment Flag Codes

SEGMENT PREFIX FORMAT DESCRIPTION

Output from DBD generation contains the statement:

```
DC  X'FEFD080A'  SEGMENT FLAGS
```

Convert the values to binary and decimal representations:

Byte 0	Byte 1	Byte2	Byte3
FE	FD	08	0A
11111110	11111101	00001000	10

Byte 0 Segment has counter, physical twin forward and backward, logical twin forward and backward, physical parent, and logical parent pointers.

Byte 1 The insert and replace rules specified are logical, and the delete rule specified is virtual. Nonsequenced inserts at current position.

Byte 2 Two 4-byte fields are reserved for physical child pointers in the parent of this segment.

Byte 3 This segment is the parent of 10 physical children.

LOAD MODULE

DBD generation is a two-step operating system job. Step 1 is a macro assembly execution which produces an object module that becomes a member of the IMSVS.DBDLIB library. Step 2 is a link-edit of the object module, which produces a load module that becomes a member of the IMSVS.DBDLIB library.

DBD GENERATION ERROR CONDITIONS

The DBD generation error messages are contained in IMS/VS Messages and Codes Reference Manual.

If operands or parameters other than those shown for each type of data base are coded, or if operands or parameters that are necessary are omitted, then one or more of the following conditions may occur:

- DBD generation may issue diagnostic messages that (a) flag operands or parameters that are not shown for the type of data base being defined, or (b) indicate that operands or parameters that are required for the type of data base being defined were omitted.
- DBD generation may complete, but DL/I may ignore the control information that was generated by the specification of operands or parameters that are not shown for the type of data base that was defined.
- DBD generation may complete, but DL/I may be unable to create and access the defined data base because (a) conflicting control information was specified when attempting to interrelate data bases, or (b) segment relationships describing the application program's view of the data base were not properly defined in the DBD generation.
- DBD generation may complete, and DL/I may create and access a data base. However, the results provided to the user may not be those desired. This condition may occur because the default actions taken by DL/I in response to finding missing or conflicting control information may be actions that the user had not considered during DBD generation.

DBD GENERATION CONTROL STATEMENT FORMATS

DBD STATEMENT

This statement names the data base being described and provides DL/I with information concerning its organization. There can be only one DBD control statement in the control statement input deck.

The format of the DBD macro instruction is shown in Figure 6 on page 21.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
DBD	NAME=	(dbname1		R	R	R	R	R	R	R	R	R			
		,dbname2,...)							O						
	,ACCESS=	{HSAM / SHSAM}		R											
		(GSAM [,BSAM / VSAM])			R										
		{((HISAM [,ISAM / VSAM])) / (SHISAM[,VSAM])}				R									
		(HDAM [,OSAM / VSAM])					R								
		(HIDAM [,OSAM / VSAM])						R							
		(INDEX [,ISAM / VSAM]							R						
		[,PROT / ,NOPROT]							O						
		,DOSCOMP)							O						
		LOGICAL								R					
		MSDB									R				
		DEDB										R			
	,RMNAME=	(mod					R					R			
		,anch					O								
		,rbn					O								
		,bytes)					O								
	,PASSWD=	{YES / NO}			O	O	O	O	O						

KEY
O = Optional
R = Required
X = Use for logical relationships

Figure 6. DBD Macro Instruction Format

Note: Optional operands, such as anch and rbn, may be required by certain randomizing modules. See the writeup for the randomizing module being used.

DBD
Identifies this statement as the DBD control statement.

NAME=
Specifies the name of the DBD for the data base being described. This name can be from one to eight alphameric characters and can be the same as that specified in the DD1= operand of the first DATASET control statement. For a shared secondary index data base, the names of up to 16 secondary index DBDs can be specified.

ACCESS=
Specifies the DL/I access method and the operating system access method to be used for this data base. The value of the operand has the following meaning.

HSAM
Means the Hierarchical Sequential Access Method (HSAM) is to be used for the data base described by this DBD. When HSAM is specified, and only one segment type is defined in the HSAM data base, this operand defaults to SHSAM.

SHSAM
Used to specify a simple HSAM data base that contains only one fixed length segment type. When a simple HSAM data base is defined, no prefix is required in occurrences of the segment type to enable IMS/VS to process the data base.

GSAM
Means the Generalized Sequential Access Method (GSAM) is to be used for the data base described by the DBD. BSAM or VSAM can be specified as the operating system access method. VSAM is the default. When GSAM is specified, no SEGM control statement is allowed in the DBD generation.

HISAM
Means the Hierarchical Index Sequential Access Method (HISAM) is to be used for the data base described by this DBD. ISAM or VSAM can be specified as the operating system access method. VSAM is the default.

SHISAM
Used to specify a simple HISAM data base that will contain only one fixed length segment type. A simple HISAM data base can only be specified when VSAM is specified as the operating system access method. When a simple HISAM data base is defined, no prefix is required in occurrences of the segment type to enable IMS/VS to process the data base.

HDAM
Means the Hierarchical Direct Access Method (HDAM) is to be used for the data base described by this DBD. OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

HIDAM
Means the Hierarchical Indexed Direct Access Method (HIDAM) is to be used for the data base described by the DBD. OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

INDEX
An index data base is defined to create the primary index to occurrences of the root segment type in a HIDAM data base, or to create a secondary index to a segment type in a HISAM, HDAM or HIDAM data base. For the primary index to a HIDAM data base, ISAM or VSAM can be specified as the operating system access method. For a secondary index, VSAM must be specified as the operating system access method. In both cases, VSAM is the default.

PROT or NOPROT
Applies only to secondary index data bases. The PROT operand on the DBD statement is an optional parameter that is used to ensure the integrity of all fields in index pointer segments that are used by IMS/VS. Use of this parameter prevents an application program from doing a replace operation on any field within an index pointer segment except for fields within the user data portion of index pointer segments. When PROT is specified, delete operations are still enabled for index pointer segments. If PROT is specified and a delete is issued for an index pointer segment, the index target segment pointer in the index pointer segment is deleted. However, the index source segment that caused the index pointer segment to be created originally is not deleted. If NOPROT is specified, an application program has replace and delete ability to all fields within an index pointer segment except the constant, search, and subsequence fields. Inserts to an index data base are invalid under all conditions. PROT is the default for this parameter.

DOSCOMP
Must be specified if the data base is an index, and it was created using DLI/DOS. DLI/DOS index data bases contain a segment code as part of the prefix. Selection of the DOSCOMP operand will cause IMS/VS to expect this code to be present in the defined data base, and to process so as to preserve this code. This includes providing a segment code for new segments being inserted. The DOSCOMP operand can only be specified for data bases that use VSAM.

LOGICAL
Means that the data base described by this DBD is a LOGICAL data base. A LOGICAL data base is composed of one or more physical data bases. A LOGICAL DBD generation is meaningful only when physical DBD generations exist that define the segment types that are referenced by SEGM statements in a LOGICAL DBD generation.

MSDB
Means a Fast Path main storage data base (MSDB) is described by the DBD.

DEDB
Means a Fast Path data entry data base (DEDB) is described by the DBD.

RMNAME=
Specifies information used to manage data stored in a Fast Path DEDB or in the primary data set group of an HDAM data base. This operand is only valid when ACCESS=HDAM or DEDB is specified. The parameters of this operand are defined below. A randomizing module controls root segment placement in or retrieval from the DEDB or HDAM data base. One or more modules, called randomizing modules, may be utilized within the IMS/VS system. A particular data base has only one randomizing module associated with it. A generalized module, which uses DBD generation-supplied parameters to perform randomizing for a particular data base, may be written to service several data bases. The

purpose of a randomizing module is to convert a value supplied by an application program for root segment placement in, or retrieval from, a DEDB or HDAM data base into a relative block number and anchor point number.

mod

Specifies the 1- to 8-character alphameric name of a user-supplied randomizing module used to store and access segments in this DEDB or HDAM data base. Examples of randomizing modules are given in "HDAM Randomizing Modules" and "DEDB Randomizing Module" in IMS/VS System Programming Reference Manual.

anch

Specifies the number of root anchor points desired in each control interval or block in the root addressable area of an HDAM data base. The default value of this parameter is one. The anch operand must be an unsigned decimal integer and must not exceed 255. Typical values are from 1 to 5.

When a user randomizing routine produces an anchor point number in excess of the number specified for this parameter, the anchor point used is the highest numbered one in the control interval or block. When a randomizing routine produces an anchor point number of zero for IMS/VS, IMS/VS uses anchor point one in the control interval or block.

The number of root anchor point for the DEDB is always 1.

rbn

Specifies the maximum relative block number value that the users wishes to allow a randomizing module to produce for this data base. This parameter is for HDAM data bases only. This value determines the number of control intervals or blocks in the root addressable area of an HDAM data base. The rbn operand must be an unsigned decimal integer whose value does not exceed $2^{24}-1$. If this parameter is omitted, no upper limit check is performed on the rbn created by the randomizing module. If this parameter is specified, but the user's randomizing module produces an rbn greater than this parameter, the highest control interval or block in the root addressable area is used by IMS/VS. If a user randomizing module produces a block number of zero, control interval or block one is used by IMS/VS.

In an HDAM or HIDAM OSAM data set, the first bit map is in the first block of the first extent of the data set. In an HDAM or HIDAM data base, the first control interval or block of the first extent of the data set specified for each data set group is used for a bit map. In a VSAM data set, the second control interval is used for the bit map and the first control interval is reserved. Note that IMS/VS adds one to the block calculated by the randomizer.

bytes

Specifies the maximum number of bytes of data base record that can be stored into the root addressable area in a series of inserts unbroken by a call to another data base record. This parameter is for HDAM data bases only. If this parameter is omitted, no limit is placed on the maximum number of bytes of a data base record that can be inserted into this data base's root segment addressable area. The bytes operand must be an unsigned decimal integer whose value does not exceed $2^{24}-1$. When the "rbn" parameter is omitted, the "bytes" parameter is ignored, which in turn, leaves no limit on the number of bytes of a data base record that can be inserted into the root addressable area.

If the "bytes" operand is specified for an HDAM data base and the length of the data base record is larger, the remainder of the record will be inserted into the overflow

area following the current end-of-file (EOF). This will require that enough space be available after the current EOF to contain the remainder of all data base records that exceed the "bytes" specification. If sufficient space is not available in the overflow area following the current EOF, the data base records are inserted randomly in the data base.

PASSWD=

YES

Causes DL/I open to use the DBDNAME for this DBD as the VSAM password when opening any data set for this data base. This parameter is only valid for DBDs that use VSAM as the access method. PASSWD=YES is invalid for ACCESS=LOGICAL, MSDB, or DEDB. When the user defines the VSAM data set(s) for this data base using the DEFINE statement of OS/VS Access Method Services, the control level (CONTROLPW) or master level (MASTERPW) password must be the same as the DBDNAME for this DBD. All data sets associated with this DBD must use the same password. For a description of the use and format of passwords for VSAM, see OS/VS Access Method Services.

For the IMS/VS DB/DC (online) system, all VSAM OPENs will bypass password checking and thus avoid operator password prompting. For the IMS/VS DB (batch) system, VSAM password checking is performed. In the batch environment, operator password prompting will occur if PASSWD=NO is specified and the data set is password protected at the control level (CONTROLPW) with passwords not equal to DBDNAME.

no

Specifies that the DBDNAME for this DBD should not be used as the VSAM password. NO is the default.

The intended use of this facility is to allow users to prevent accidental access of IMS/VS data bases by non-IMS/VS programs.

DATASET AND AREA STATEMENTS

A DATASET statement defines a data set group within a data base. An AREA statement defines an area within a data base (DEDB only). At least one DATASET or AREA statement is required for each DBD generation. The maximum number depends on the type of data base being defined. HSAM, SHSAM, GSAM, SHISAM, index, logical, and fast path main storage data bases can have only one data set group. Data Entry data bases can have 1 to 240 areas defined. HISAM, HDAM, and HIDAM data bases can be divided into 1 to 10 data set groups subject to rules that follow in the next section.

In the DBDGEN input deck, a DATASET statement precedes the SEGM statements for all segments that are to be placed in that data set group. The first DATASET statement of a DBD generation defines the primary data set group. Subsequent DATASET statements define secondary data set groups. The only exception to this is when the LABEL field of a DATASET statement is used. Refer to "Use of the Label Field" for this exception.

In the DBDGEN input deck for a DEDB, all AREA statements must be placed between the DBD statement and the first SEGM statement. At least one AREA statement is required, but as many as 240 AREA statements can be used to define multiple areas.

Rules for Dividing a Data Base into Multiple Data Set Groups

HISAM, HDAM and HIDAM data bases can be divided into a maximum of 10 data set groups according to the following restrictions. Each DATASET statement creates a separate data set group, except for the case explained in "Use of the Label Field." The first DATASET statement defines the primary data set group. Subsequent DATASET statements define secondary data set groups.

For HISAM data bases, secondary data set groups cannot be defined when VSAM is the access method used for the data base, or when ISAM/OSAM are used as the access methods for the data base, but the data base is indexed by a secondary index. For HISAM data bases using ISAM/OSAM as the access method and not indexed by a secondary index, all DATASET statements defining secondary data set groups must be immediately followed by a SEGM statement that defines a second level dependent of the root segment type. Thus a HISAM data base may be separated into multiple data set groups, but the division of the segment hierarchy between primary and secondary data set groups can only be performed between the first and second level of the hierarchy. In a HISAM data base, a primary data set group contains all segment types that hierarchically follow the root up to the segment type that starts a secondary data set group. A secondary data set group contains all segment types that hierarchically follow the dependent segment type up to the segment type that starts another secondary data set group, if any.

For HDAM or HIDAM data bases, DATASET statements may be used to divide the data base into multiple data set groups at any level of the data base hierarchy; however, the following restriction must be met. A physical parent and its physical children must be connected by physical child/physical twin pointers, as opposed to hierarchic pointers, when they are in different data set groups as shown in Figure 7 on page 27.

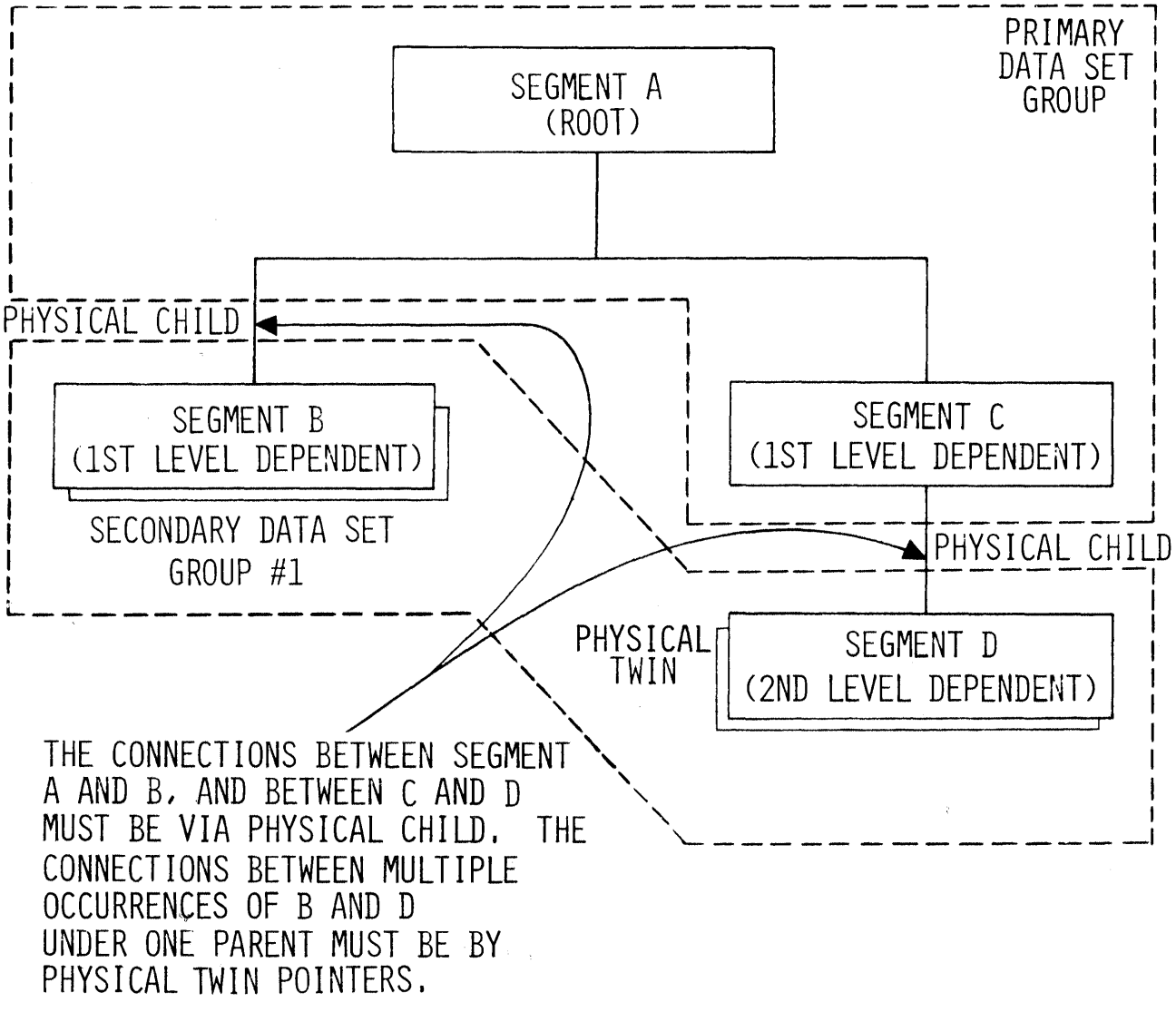


Figure 7. Connections through Physical Child and Physical Twin Pointers

Use of the Label Field

In HDAM or HIDAM data bases, it is sometimes desirable to place segments in data set groups according to segment size or frequency of access rather than according to their hierarchic position in the data structure. To achieve this while still observing the DBD generation rule that the SEGM statements defining segments must be arranged in hierarchic sequence, the LABEL field of the DATASET statement is used.

An identifying label coded on a DATASET statement is referenced by coding the same label on additional DATASET statements. Only the first DATASET statement with the common label can contain operands that define the physical characteristics of the data set group. All segments defined by SEGM statements that follow DATASET statements with the same label are placed in the data set group defined by the first DATASET statement with that label.

This capability can be used in much the same manner as the CSECT statement of OS/VS assembler language, with the following restrictions:

1. A label used in the label field of a DATASET statement containing operands cannot be used on another DATASET statement containing operands.
2. Labels must be alphameric and valid labels for an OS/VS assembler language statement.
3. Unlabeled DATASET statements must have operands.

Referring to Figure 7 on page 27, the following example illustrates use of the label field of the DATASET statement to group segment types of the same size in the same data set groups.

```
Label    Operation      Operand
         DBD            NAME=HDBASE,ACCESS=HDAM,
                        RMNAME=(RANDMODL,1,500,824)
DSG1     DATASET        DD1=PRIMARY,DEVICE=3350,BLOCK=1648
         SEGM           NAME=SEGMENTA,BYTES=100
DSG2     DATASET        DD1=SECOND,DEVICE=3350,BLOCK=3625
         SEGM           NAME=SEGMENTB,BYTES=50,PARENT=SEGMENTA
DSG1     DATASET
         SEGM           NAME=SEGMENTC,BYTES=100,PARENT=SEGMENTA
DSG2     DATASET
         SEGM           NAME=SEGMENTD,BYTES=50,PARENT=SEGMENTC
         DBDGEN
         FINISH
         END
```

The segments named SEGMENTA and SEGMENTC exist in the first data set group. The segments named SEGMENTB and SEGMENTD exist in the second data set group.

The formats of the DATASET and AREA statements are shown in Figure 8 on page 29.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
DATASET or AREA	LOGICAL									O					
	DD1=	ddname1		R	R	R	R	R	R			R			
	,DEVICE=	device	1	R		R	R	R	R			R			
	,MODEL=	model		O		O	O	O	O			O			
	,DD2=	ddname2		R	O										
	,OVFLW=	ddname3	2			R			R						
	,BLOCK=	(blkfact1	3	O	O	O			O						
		,blkfact2)	3	O		O			O						
		or size0	3				O	O							
	,SIZE=	(size1	3/4		O	O	O	O	O			R			
		,size2)	3			O			O						
	,RECORD=	(reclen1	3	O	O	O			O						
		,reclen2)	3	O	O	O			O						
	,RECFM=	,recfm1	5		R										
	,SCAN=	cyls					O	O							
	,FRSPC=	(fbff					O	O							
		,fspf)					O	O							
	,REL=	({NO | TERM[,fldnm] | FIXED[,fldnm] | DYNAMIC[,fldnm]})									R				
	,UOW=	(number1, overflow1)										R			
	,ROOT=	(number2, overflow2)										R			

KEY
O = Optional
R = Required
X = Use for logical relationships

Figure 8. DATASET and AREA Statement Format

Notes:

1. DEVICE is invalid for a GSAM data base.
2. Required if ISAM/OSAM are the access methods. If VSAM is the access method for an index data base and the keys of all the index segments are unique, or if VSAM is the access method for a HISAM data base with only one segment type defined, OVFLW need not be specified and will not be accessed. OVFLW is invalid for a simple HISAM data base.
3. When not specified by the user, DBDGEN generates value used.
4. The valid operand specifications for a DEDB SIZE keyword are 512, 1024, 2048, and 4096 bytes.
5. RECFM is only valid for a GSAM data base.

For information on using the AREA control statement, see "AREA" on page 38.

DATASET
Identifies this statement as a DATASET control statement.

LOGICAL or DD1=

LOGICAL
Specifies that a logical data base is being defined in this DBD generation. This operand must be specified if the ACCESS=LOGICAL operand is specified on this DBD generation's DBD statement. If this operand is specified, all other operands are invalid, and this must be the only DATASET statement for the DBD generation. The SEGM statements that follow this statement can have only the NAME= , PARENT= , and SOURCE= operands specified. No FIELD, XDFLD, or LCHILD statements may be used in a LOGICAL DBD generation.

DD1=
Specifies the ddname of the primary data set in this data set group. ddname1 must be a 1- to 8-character alphameric name. IMS/VS use of the data set indicated by this operand depends on the type of data base being defined as shown in Figure 9. For an HSAM or GSAM data base, this input data set is used when an application program retrieves data from the data base.

If a Fast Path data base is registered in DBRC, this parameter should specify the area name.

HSAM	GSAM	HISAM	HIDAM HDAM	MSDB	DEDB	INDEX	LOGICAL
ddname of input data set	ddname of input data set	ddname of primary data set in data set group	ddname of data set in data set group	operand is invalid	name of defined area	ddname of primary data set	operand is invalid

Figure 9. Use of DD1= Operand

DEVICE=
Specifies the physical storage device type on which the data sets in this data set group will be stored. This parameter is used during the DBD generation process to verify that the IMS/VS-calculated or user-specified block lengths do not exceed usable track length of the storage device specified. A list of the valid entries for this operand follows:

Device Type	Device=
DASD	2314,[1] 2305, 2319,[1] 3330, 3340, 3350, 3375, or 3380
Tape	2400,[1] 3400, or TAPE

The value 2400, 3400, or TAPE may be chosen only if an HSAM or simple HSAM data base is being defined. The largest LRECL that can be specified for 2400 and 3400 tape drives is 32760 bytes.

---

[1] Extended architecture (MVS/XA) does not support these devices. So that these devices are not specified, error checking is not invoked.

MODEL=
Is used when 2305 or 3330 is the device specified in the device operand. Following are the model numbers that can be specified for the 2305 and 3330:

For the 2305, MODEL=1 or MODEL=2. The default is MODEL=2. Extended architecture (MVS/XA) does not support the 2305 MODEL=1.

For the 3330, MODEL=1 or MODEL=11. The default is MODEL=1.

DD2=
Specifies the 1- to 8-character alphameric ddname of the output data set required for an HSAM or simple HSAM data base and optional for a GSAM data base. If it is omitted, ddnamel is assumed. This output data set is used by HSAM or GSAM when loading the data base.

OVFLW=
Specifies the 1- to 8-character alphameric ddname of the overflow data set in this data set group. This operand must be specified for (1) an INDEX data base that uses ISAM/OSAM, (2) an INDEX data base that uses VSAM and contains index pointer segments with nonunique keys, and (3) all data set groups of a HISAM data base except when only one segment type is defined in the HISAM data base and the access method used is VSAM.

The ddnames used in DD1, DD2, or OVFLW subparameters must be unique within an IMS/VS system or account. Nonunique ddnames in two or more DBDs might result in destruction of the data base. One situation that can result in destruction of a data base is if both ddnames were inadvertently used concurrently (both used in two different message regions of a data communications system or in two PCBs of one PSB used in a batch DL/I region of a data base only system).

Special note should be taken of three situations:

- The OVFLW operand is not allowed when a simple HISAM data base is defined.
- When a HISAM data base that contains only one segment type is defined, the OVFLW operand does not have to be specified if the operating system access method is VSAM.
- When VSAM is used for the primary index for a HIDAM data base, no OVFLW operand on the DATASET statement is required for the index DBD because all index segments are inserted in the key sequenced data set of the index.

BLOCK=
Is used to specify the blocking factors (blkfactl, blkfact2) to be used for data sets in a data set group for HSAM, SHSAM, GSAM, HISAM, SHISAM, and INDEX data bases, or is used to specify the block size or control interval size without overhead (sizeO) for the data set in a data set group for HDAM and HIDAM data bases. Figure 10 on page 32 explains the use of the BLOCK= and RECORD= operands.

HSAM	GSAM	HISAM	HIDAM HDAM	MSDB	DEDB	INDEX	LOGICAL
BLOCK= blkfact1 applies to input data set and should always be 1. blkfact2 applies to output data set and should always be 1. RECORD= reclen1 is input record length. reclen2 is output record length. HSAM is always unblocked; LRECL and BLKSIZE are equal.	BLOCK= blkfact1 applies to input/output data set. blkfact2 is invalid sub-parameter. RECORD= reclen1 is the size of an LRECL length or max. size for variable length record. recln2 is the min. size for variable length record. SIZE= size1 is BLKSIZE for input/output data set. size2 is invalid sub-parameter.	BLOCK= blkfact1 is primary data set blocking factor. blkfact2 is overflow data set blocking factor. RECORD= reclen1 is primary data set logical record length. reclen2 is overflow data set logical record length.	BLOCK= size0 is size without overhead of OSAM or VSAM data set in data set group RECORD= is ignored.	BLOCK= and RECORD= operands are invalid.	BLOCK= and RECORD= operands are invalid.	BLOCK= blkfact1 is primary data set blocking factor. blkfact2 is overflow data set blocking factor. RECORD= reclen 1 is primary data set logical record length. reclen2 is overflow data set logical record length.	BLOCK=, and RECORD= operands are invalid.

Note: When both reclen1 and reclen2 are specified in a DATASET statement, reclen2 must be equal to or greater than reclen1 except for GSAM.

Figure 10. Use of BLOCK= and RECORD= Operands

For HISAM and INDEX data bases that use ISAM/OSAM as the OS/VS access methods, the BLOCK=operand in conjunction with the RECORD= operand determines the block sizes of the ISAM and OSAM data sets in a data set group. The resulting block size for a data set is the record length specified times the blocking factor specified.

For HISAM, SHISAM, and INDEX data bases that use VSAM as the OS/VS access method, the SIZE= operand should be used to specify control interval size in place of the BLOCK= operand. If the SIZE= keyword is used for a HISAM, SHISAM, or INDEX data base, the BLOCK= keyword is invalid. In cases where the RECORD= and BLOCK= operands are used, the resulting control interval size must be a multiple of 512 when the resulting size is less than 8192 bytes. If the product of the record length specified times the blocking factor specified plus VSAM overhead is not a multiple of 512 and is less than 8192 bytes, the resulting control interval size is obtained by rounding the value up to the next higher multiple of 512. Control interval sizes in the range of 8192 to 30720 bytes (maximum allowed size) must be a multiple of 2048 bytes. When the product of the RECORD= and BLOCK= operands plus VSAM overhead is in the range of 8192 to 30720 bytes but is not a multiple of 2048, the

resulting control interval size is obtained by rounding the value up to the next higher multiple of 2048. The VSAM overhead is 7 bytes if the blocking factor is 1; otherwise, it is 10 bytes. The maximum block size for OSAM data sets is 18433 bytes for IMS/VS systems prior to Release 1.4 and 32K bytes for IMS/VS Release 1.4, and subsequent releases.

For HDAM and HIDAM data bases, the BLOCK= operand is used to enable the user to override DBDGEN's computation of control interval or block size. The user should note, however, that in addition to the value specified in the BLOCK= operand, DBDGEN adds space for root anchor points, a free space anchor point, and access method overhead. The block or control interval size that results can be determined by referring to the equations in the description of the SIZE= operand or by examining the output of DBDGEN.

SIZE=

Is used to override DBDGEN's computation of control interval or block size. If the value specified for SIZE= is different from the control interval size defined to VSAM using the Access Method Services, DL/I uses the value defined to VSAM. For DEDBs the DBDGEN SIZE= must match the control interval size defined to VSAM, because IMS uses this value in accessing the data set. If the control interval size is changed in the VSAM data set, the DBD for that area must be changed to the new SIZE= value. By redefining the control interval size to VSAM using the Access Method Services, the user can effectively modify the DBD without a DBDGEN. This allows the user to migrate data bases to new devices without a DBDGEN. When used, no overhead is added to the values specified and the value specified is not validated by IMS/VS. For VSAM data sets, when the values specified are less than 8192, they must be a multiple of 512. If not a multiple of 512, DBDGEN will round the value specified to the next higher multiple of 512 and issue a warning message. Values specified in the range of 8192 to 30720 bytes (maximum allowed size) must be a multiple of 2048. If not a multiple of 2048, DBDGEN will round the value specified to the next higher multiple of 2048 and issue a warning message. For HISAM, SHISAM, primary HIDAM index, and secondary index data bases, size1 specifies the control interval or block size of the primary data set in a data set group, and size2 specifies the control interval or block size of the overflow data set. For HDAM and HIDAM data bases, only the size1 operand is used. The size1 operand specifies the control interval or block size of the data set in the data set group. When SIZE is specified for a HISAM or INDEX data base where OS/VS access method is ISAM, the RECORD parameter must also be specified; the size value specified must be a multiple of the record parameter in order to allow QSAM or QISAM to open the data sets involved. Following are equations that show the minimum block or control interval size that can be specified by the user for data bases.

The maximum block size of OSAM data sets is 32K bytes.

For DEDBs, only the SIZE1 operand is valid. Valid operand specifications for a DEDB SIZE keyword are 512, 1024, 2048, and 4096 bytes.

HISAM Primary Data Set Group, Primary HIDAM Index, and Secondary Index Data Set Group:

For the primary data set group of a HISAM or INDEX data base, the minimum block or control interval size that can be specified for the primary data set is given by primary size and for the overflow data set by overflow size. Note that the overflow data set is not always required in the data set group.

primary size ≥ ROOTSEG + OVERHEAD + VSAM CONTROL

overflow size ≥ MAXSEG + OVERHEAD + VSAM CONTROL

where:

ROOTSEG=
Maximum root segment size including the segment prefix. Note that an INDEX VSAM root segment prefix does not include a segment code, unless it was created using DL/I DOS.

OVERHEAD=
7 bytes for ISAM/OSAM, if the data base has more then one physical segment type.

3 bytes for ISAM/OSAM, if the data base has only one physical segment type.

4 bytes for INDEX VSAM data bases with nonunique root segment keys.

0 bytes for INDEX VSAM data bases unique root segment keys, not created using DL/I DOS.

5 bytes for all other VSAM data bases.

VSAM CONTROL=
0 bytes for OSAM, 7 bytes for VSAM if the blocking factor is 1; otherwise it is 10 bytes.

MAXSEG=
The length in bytes of the longest segment in this data set group including the segment prefix.

HISAM Secondary Data Set Group:

For the secondary data set group of a HISAM data base, the minimum block size that can be specified for the primary data set is given by primary size and for the overflow data set by overflow size.

primary size ≥ MAX2ND + OVERHEAD + ROOT KEY LENGTH

overflow size ≥ MAXSEG + OVERHEAD

overflow size ≥ MAX2ND + OVERHEAD + ROOT KEY LENGTH

where:

MAX2ND=
The length in bytes of the longest second level segment type in this data set group including the segment prefix.

OVERHEAD=
7 bytes for ISAM/OSAM.

ROOT KEY LENGTH=
The length in bytes of the sequence field of the root segment.

MAXSEG=
The length in bytes of the longest segment in this data set group including the segment prefix.

HDAM Primary Data Set Group:

The minimum block or control interval size that can be specified for the primary data set group of an HDAM data base is dependent on whether or not the DBD statement rbn operand of the RMNAME parameter is specified.

- If rbn is specified, then the following two conditions must be met:

  size ≥ (RAPs*4) + FSEAP + 2 + VSAM CONTROL, and

  size ≥ MAXSEG + FSEAP + VSAM CONTROL

- If rbn is not specified, then the following condition must be met:

  size ≥ MAXSEG + (RAPs*4) + FSEAP + VSAM CONTROL

where:

RAPs=
The number of root anchor points specified for the root addressable area of the data base.

FSEAP=
4 bytes for a free space element anchor point.

VSAM CONTROL=
0 bytes for OSAM; 7 bytes for VSAM.

MAXSEG=
The length in bytes of the longest segment in this data set group including the segment prefix.

HDAM Secondary Data Set Groups:

size ≥ MAXSEG + FSEAP + VSAM CONTROL

where:

MAXSEG=
The length in bytes of the longest segment in this data set group including the segment.

FSEAP=
4 bytes for a free space element anchor point.

VSAM CONTROL=
0 bytes for OSAM; 7 bytes for VSAM.

HIDAM Data Set Groups:

The minimum block or control interval size that can be specified for data set groups in a HIDAM data base is dependent on the OS/VS access method specified. The block or control interval size of the primary data set group is also dependent on the type of pointers specified for the root segment type.

If forward-only hierarchic or physical twin pointers are specified for the root segment type of a HIDAM data base, the block or control interval size specified for the primary data set group must be:

size ≥ MAXSEG + FSEAP + RAP + VSAM CONTROL

Under any other conditions for primary or secondary data set groups, the block or control interval size specified must be:

size ≥ MAXSEG + FSEAP + VSAM CONTROL

where:

MAXSEG=
The length in bytes of the longest segment in this data set group including the segment prefix.

FSEAP=
4 bytes for a free space element anchor point.

VSAM CONTROL=
0 bytes for OSAM; 7 bytes for VSAM.

RAP=
4 bytes for one root anchor point.

RECORD=(reclen1,reclen2)
Specifies the data management logical record length(s) to be used for this data set group. This operand is optional and cannot be specified if ACCESS=LOGICAL is used on the DBD statement. reclen1 and reclen2 must be numeric values. The value of reclen2 must always be equal to or greater than the value of reclen1 except for GSAM. The meaning of each of the operand's parameters depends on the type of data base being defined as shown in Figure 10 on page 32. For a simple HISAM data base, the logical record length specified must be the same as the segment length specified. The minimum allowable logical record lengths for HISAM and INDEX DBDs are the same as the minimum block or control interval sizes described for the DATASET SIZE= operand, except that VSAM CONTROL should be ignored. In addition, for both the VSAM KSDS and ESDS for HISAM, and INDEX DBDs, the logical record length specified must also be an even value. For VSAM primary index (INDEX, VSAM) data bases, the overflow logical record length (reclen2) parameter should not be defined, because all index segments are inserted into the key sequence data set. For a GSAM data base, reclen1 specifies the size of a logical record for a fixed-length record or the maximum size for a variable-length undefined record. The value of reclen2 specifies the minimum size for a variable-length undefined record.

RECFM=
Specifies the format of the records in the data set. The record format is specified using the characters defined below:

F—the records are fixed-length.

FB—the records are fixed-length and blocked.

V—the records are variable-length.

VB—the records are variable-length and blocked.

U—the records are of undefined length.

This operand is only valid for a GSAM data base.

SCAN=
Specifies the number of direct access device cylinders to be scanned when searching for available storage space during segment insertion operations. This operand is optional. It is only used when this DBD generation defines a HIDAM or HDAM data base. If specified, cyls must be a decimal integer whose value does not exceed 255. Typical values are from 0 to 5. The default value is 3. If SCAN=0 is specified, only the current cylinder is scanned for space. Scanning is performed in both directions from the current cylinder position. If a scan limit value would cause scanning to include an area outside of the current extent, the scan limits are adjusted by IMS/VS such that

scanning will not exceed current extent boundaries. If space cannot be found for segment insertion within the cylinder bounds defined by this operand, space is used at the current end of the data set group for the data base.

FRSPC=
Specifies how free space is to be distributed in an HDAM or HIDAM data base. The fbff is the free block frequency factor, and it specifies that every nth control interval or block in this data set group will be left as free space during data base load or reorganization (where fbff=n). The range of fbff includes all integer values from 0 to 100 except fbff=1. The fspf is the free space percentage factor. It specifies the minimum percentage of each control interval or block that is to be left as free space in this data set group. The range of fspf is from 0 to 99. The default value for fbff and fspf is 0.

Note: If the total of the percentage of free space specified and any segment size exceeds the control interval or block size, a warning message that flags oversized segments is issued by DBDGEN. When loading oversized segments, the "fspf" specification is ignored and one control interval or block is used to load each oversized segment.

REL=
Defines whether an MSDB is a nonterminal-related (NO or TERM) or a terminal-related (FIXED and DYNAMIC) MSDB. There is no ownership of segments in nonterminal-related MSDBs.

With terminal-related MSDBs, each segment is assigned to a different LTERM. The LTERM name is the segment key but is not contained in the segment. Each LTERM owns no more than one segment per MSDB, and only the owner can alter a segment.

NO
Specifies a nonterminal-related MSDB without terminal-related keys. The key and the sequence field are part of the segment.

TERM
Specifies a nonterminal-related MSDB with terminal-related keys. The key is the LTERM name (not part of the segment) and there is no sequence field.

FIXED
Specifies a terminal-related fixed MSDB. The LTERM name is the segment key. Segment updates are allowed. Segment insertions and deletions are not allowed.

DYNAMIC
Specifies a terminal-related dynamic MSDB. The LTERM name is the segment key. Segments can be inserted and deleted. No more than one insertion or deletion may be made to the same MSDB from a single LTERM within one sync processing interval.

search field name
Specifies a 1- to 8-character alphameric name. The name must not be the same as any other field name defined in a FIELD statement.

Because a sequence field cannot be defined for an MSDB using an LTERM name as a segment key (REL=TERM, FIXED, or DYNAMIC), a search field name is provided to allow qualified calls. The only valid value in an SSA is an LTERM name. Therefore, the search field is treated as an 8-byte character field and no further definition is provided.

AREA
Identifies this statement as a Fast Path DEDB AREA control statement. One AREA statement is required for each area defined within a DEDB. As many as 240 AREA statements can be specified. Multiple AREA statements can be used to define multiple areas for a DEDB. The AREA statement must be placed between the DBD statement and the first SEGM statement in a DBDGEN input stream.

DD1=
Specifies the ddname of the defined area. ddname1 must be a 1- to 8-character alphameric name. This parameter may be an area name or a ddname for single area data sets but can be an area name only for multiple area data sets. If the data base is registered in DBRC, this parameter should specify the area name.

DEVICE=
Specifies the physical storage device type on which the data set in this area will be stored. A list of the valid entries for this operand follows:

Device Type	Device
DASD	2314,[2] 2305, 2319,[2] 3330, 3340, 3350, 3375, or 3380

MODEL=
Is used when 2305 or 3330 is the device specified in the device operand. Following are the model numbers that can be specified for the 2305 and 3330:

For the 2305, MODEL=1 or MODEL=2. The default is MODEL=2. Extended architecture (MVS/XA) does not support the 2305 MODEL=1.

For the 3330, MODEL=1 or MODEL=11. The default is MODEL=1.

SIZE=
Specifies the control interval. Size can be 512, 1024, 2048, or 4096 bytes only. No default value is allowed. With a 2314 or 2319 device, 4096 bytes cannot be specified.

UOW=

number1
Specifies the number of control intervals in a unit of work (UOW). Its value must be from 2 to 32767.

overflow1
Specifies the number of control intervals in the overflow section of a UOW. Overflow1 can be any value greater than or equal to one but at least one less than the specified value for number1.

Note that the total number of root anchor points (RAPs) within one UOW is given by number1 minus overflow1. Multiply the number of RAPs in one UOW by the number of UOWs in the root addressable part to find the total number of RAPs within an area.

---

[2] Extended architecture (MVS/XA) does not support these devices. So that these devices are not specified, error checking is not invoked.

ROOT=

number2
Specifies the total space to be allocated to the root addressable part of the area and to the part of the area reserved for independent overflow. It is expressed in UOWs. One UOW is reserved for online reorganization; the rest of the VSAM data set is reserved for sequential dependent data. The value must be greater than 2 and less than 32767, and cannot be larger than the amount of space actually in the VSAM data set.

overflow2
Specifies the space reserved for independent overflow in terms of UOWs. It must be at least one and must be less than the value specified for number2. Note that although independent overflow does not contain UOWs, the UOW size is used as the unit for space allocation.

The reorganization UOW is automatically allocated by the DEDB Initialization utility. VSAM space definition should include this additional UOW. That is, the total space required is the root addressable area, the independent overflow, and one additional UOW for reorganization.

EXAMPLE:

```
AREA DD1=XX,SIZE=2048,
    UOW=(64,14),
    ROOT=(936,36)
```

This example allocates 2048*64*936 bytes, and leaves the rest of the area for sequential dependent segments. Because there is only one root anchor point (RAP) per control interval, the total number of RAPs within the area is given by: (64-14)*(936-36) = 45000 RAPs.

Data Sets in IMS/VS Data Set Groups

The DD cards for the data sets in each IMS/VS data base must be provided with each job that accesses the data base. For data bases used by message or batch message processing programs, DD cards must be included in the JCL for the IMS/VS control region. For data bases which are used exclusively in the batch processing environment, DD statements must be included in the JCL for the batch processing region. In an MVS online environment, data bases can be dynamically allocated.

VSAM

When the operating system access method for a data base is VSAM, one DD statement is required for each KSDS and one for each ESDS. The parameters required on the DD statements have the following format:

```
//ddname     DD     DISP=SHR,DSNAME=
```

Since all VSAM data sets are cataloged, UNIT=, VOL=SER=, and SPACE= parameters are not required.

For a HISAM data base, two DD statements are required in general; one for the KSDS and one for the ESDS. If the HISAM data base has only one segment type defined, only the KSDS DD statement is required.

For an HDAM or HIDAM data base, one DD statement is required for each data set group. For the prime index of a HIDAM data base one DD statement is required for the KSDS.

For secondary index data bases with unique keys one DD statement is required for the KSDS.

For secondary index data bases with nonunique keys, two DD statements are required; one for the KSDS and one for the ESDS. In addition to the DD statements defining VSAM data sets, a DD statement specifying a data set containing parameters defining the IMS/VS VSAM buffer pool must be provided for batch regions. The DDNAME for this DD statement is DFSVSAMP. For online IMS/VS execution, this information is provided in a member of the IMSVS.PROCLIB data set with member name DFSVSMxx. See IMS/VS Installation Guide for more information on the IMSVS.PROCLIB data set.

ISAM/OSAM

Operating system procedures for execution of all IMS/VS region types are provided in Chapter 1 of IMS/VS System Programming Reference Manual. The appropriate DD statements must be appended to these procedures.

For HSAM, a DD statement for either input or output should be provided in the following format:

```
//ddname   DD  DSNAME= ,UNIT=     ,VOL=SER=    ,
//             DISP=   ,DCB=
```

Where the DD statement is for an HSAM output data set, the data set must be preallocated or the SPACE= operand must be present when a direct access storage device is used.

For HISAM, DD statements for ISAM and OSAM must be provided for each data set group.

The following is an example of the ISAM DD statements:

```
//ddl      DD  DSNAME=     (INDEX),UNIT=    ,VOL=SER=    ,
//           DISP=         ,DCB
//           DD  DSNAME=     (PRIME),UNIT=    ,VOL=SER=    ,
//           DISP=         ,DCB=
```

For more information on ISAM, see the appendix on "Creating and Retrieving Indexed Sequential Data Sets" in OS/VS JCL Reference.

The following is an example of the OSAM DD statements:

```
//ddovflw  DD  DSNAME=     ,UNIT=       ,VOL=SER=    ,
//           DISP=         ,DCB=(DSORG=PS)
```

The DCB parameter for an ISAM data set when a data base is being created should specify:

```
DCB=(DSORG=IS[,OPTCD=WM][,RECFM=FB])
```

where:

W = Write check (optional)

M = Master index creation (optional)

OPTCD=R indicates that in storage indexes are to be used for an ISAM data set allocated to IMS/VS. It must be specified for every IMS/VS execution in which in storage indexes are to be used. In storage indexes apply for BISAM, not to QISAM. IMS/VS will perform a conditional request for the amount of storage required to contain the highest level index. If enough storage is not available, in storage indexes will not be used but processing will continue without any error message notification. The required amount of storage can be determined by examining the DS2NOBYT field at displacement X'40' in the Format 2 DSCB control block.

Because the OPTCD field is empty in the DCB, any OPTCD options desired may be specified in the DCB field of the DD card.

The user must not specify OPTCD=L, which indicates the presence of a delete byte in the ISAM logical record. The user should not specify OPTCD=I for ISAM independent overflow, because ISAM is not used to make additions to the data base.

RECFM=FB is optional but if used, must be specified at load time..p is optional but if used, must be specified at load time. RECFM=F must not be specified.

For either an ISAM or an OSAM data set, the LRECL, BLKSIZE, and BUFL subparameters of the DCB parameter should be omitted. This information is obtained from the DBD and cannot be overridden.

When the HISAM data base is being created, the associated data sets must be preallocated or the SPACE= operand must be present.

For HDAM or HIDAM, a DD statement is required for the OSAM data set of each data set group. It should be in the following format:

```
//ddl      DD  DSNAME=          ,UNIT=      ,VOL=SER=    ,
//             DISP=       ,DCB=(DSORG=PS[,OPTCD=W])
```

When the HDAM or HIDAM data base is being created, the OSAM data set must be preallocated or the SPACE= operand must be present.

For an INDEX data base, the DD statements should be equivalent to that specified for HISAM.

Note: If a model DSCB is to be used to describe a generation data set, the LRECL, RECFM, and BLKSIZE parameters must be omitted from the model DSCB. This information is obtained from the DBD and cannot be overridden.

SEGM STATEMENT

The SEGM statement defines a segment type, the segment's position in a data base hierarchy, the physical characteristics of the segment, and how the segment is to be related to other segments. Except for GSAM data bases, at least one SEGM statement must immediately follow each DATASET statement; the segment defined by the SEGM statement is placed in the data set group defined by the DATASET statement. Except for Fast Path MSDBs and DEDBs, a maximum of 255 SEGM statements are allowed in a DBD generation. For a Fast Path MSDB, only one SEGM statement can be specified. For a Fast Path DEDB, at least one and up to 127 SEGM statements must immediately follow the last AREA statement; no other SEGM statements can be provided in the DBD generation. SEGM statements must be placed in the input deck in hierarchic sequence, and a maximum of 15 hierarchic levels may be defined.

The SEGM statement is used in conjunction with FIELD, XDFLD and LCHILD statements to totally define a segment to IMS/VS. The FIELD statement defines fields within segments, the XDFLD statement defines fields used for secondary indexing, and the LCHILD statement defines index or logical relationships between segments.

The format of the SEGM statement is shown in Figure 11 on page 42.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type								operands used for logical relationships		
				HSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
SEGM	NAME=	segname1		R	R	R	R	R	R	R	R			
	,PARENT=	0	1	O	O	O	O	O	O		O			
		or												
		((segname2		R	R	R	R		R		R			
		,[SNGL / DBLE])				O	O				O			
		,(Ipsegname										X	X	X
		,[VIRTUAL / PHYSICAL]										X	X	X
		,dbname1))										X	X	X
	,BYTES=	(max bytes	2	R	R	R	R	R		R	R			
		,min bytes)	3		O	O	O				R			
	[,TYPE=	{DIR / SEQ}]	4								R			
	,FREQ=	frequency		O	O			O						
	[,POINTER= / ,PTR=]	({HIER / HIERBWD / TWIN / TWINBWD / NOTWIN}				O	O							
		{LTWIN / 'LTWINBWD}										X		X
		,LPARNT										X	X	X
		,CTR										X	X	X
		,PAIRED)										X	X	X
	,RULES=	([{P / L / V} {B / P / L / V} {P / L / V}]										X	X	X
		[FIRST / LAST / 'HERE])	5	O	O	O	O	O			O			

Figure 11 (Part 1 of 2). SEGM Statement Format

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type								operands used for logical relationships		
				HSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
SEGM (Cont.)	,SOURCE=	((segname							R			X	X	X
		,[KEY / DATA]							O			X	X	X
		,dbname)							R			X	X	X
		(,segname	6						R					
		,[KEY / DATA]							O					
		,dbname))	6						R					
	,COMPRTN=	(routinename	3		O	O	O							
		,[KEY / DATA]	3		O	O	O							
		,INIT)	3		O	O	O							
	,SSPTR=	n									O			

KEY
O=Optional
R=Required
X=Use for logical relationships

Notes:

1. The PARENT=operand can be omitted, or PARENT=0 specified for the root segment type of a data base.

2. MSDB segment length must be defined as a multiple of 4 bytes not exceeding 32000 bytes.

3. Variable-length segments and segment edit/compression cannot be specified for a simple HISAM data base.

4. TYPE=SEQ is required on SEGM statements for the sequential dependent segment type.

5. Required when a segment type does not have a unique sequence field. LAST is the default except for Fast Path DEDBs.

   When using Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden. For DEDB direct dependent segment processing, HERE is the default.

6. Required when defining a concatenated segment type. Only allowed for a LOGICAL data base.

Figure 11 (Part 2 of 2). SEGM Statement Format

For the SEGM statement, the following abbreviations may be used in place of keywords specified in the macro definitions:

Keyword	Abbreviation
POINTER	PTR
FIRST	F
LAST	L
HERE	H
KEY	K
DATA	D
VIRTUAL	V
PHYSICAL	P

SEGM
Identifies this statement as a segment definition statement.

NAME=
Specifies the name of the segment type being defined. The specified name is used by DL/I and application programs in all references to this segment. Duplicate segment names are not allowed within a DBD generation. The segname1 operand must be a 1- to 8-character alphameric value. Each character must be in the range of A through Z, or 0 through 9, or be the character $, #, or @.

PARENT=
Specifies the name(s) of the physical and logical parents of the segment type being defined, if any.

0
For root segment types, the PARENT= keyword must be omitted or PARENT=0 specified.

segname2
For dependent segment types, specifies the name of this segment's physical parent.

SNGL or DBLE
Specifies the type of physical child pointers to be placed in all occurrences of the physical parent of the segment type being defined. SNGL/DBLE can be specified only for segments in HDAM, HIDAM, or DEDB data bases and are ignored if the physical parent specifies hierarchic pointers (PTR=HIER or HIERBWD).

SNGL causes a 4-byte physical child first pointer to be placed in all occurrences of the physical parent of the segment type being defined. SNGL is the default.

DBLE causes a 4-byte physical child first pointer and a 4-byte child last pointer to be placed in all occurrences of the physical parent of the segment type being defined.

lpsegname
Specifies the name of the logical parent of the segment type being defined, if any. This operand is used only during DBDGEN of a physical data base, and it must be specified on SEGM statements that define logical child segment types.

VIRTUAL or PHYSICAL
Specified for logical child segments only. They specify whether or not a symbolic pointer to the logical parent (logical parents concatenated key) is to be stored as a part of the logical child segment on the storage device

used. If PHYSICAL is specified, the concatenated key of the logical parent is stored with each logical child segment. If VIRTUAL is specified, only the intersection data portion of each logical child segment is stored. VIRTUAL is the default parameter. PHYSICAL must be specified for a logical child segment whose logical parent is in a HISAM data base, or for a logical child segment that will be sequenced on its physical twin chain through use of any part of the logical parents concatenated key.

dbname1
Specifies the name of the data base in which the logical parent is defined. If the logical parent is in the same data base as the logical child, dbname1 can be omitted.

SSPTR=
Specifies the number of subset pointers. You can specify from 0 to 8. When you specify 0 or if SSPTR is not specified, you are not using a subset pointer.

BYTES=
Specifies the length of the data portion of a segment type in bytes using an unsigned decimal integer(s).

Fixed-length segments

For fixed-length segments, "maxbytes" specifies the amount of storage used for the data portion of the segment. The minbytes operand cannot be specified for a fixed-length segment. This includes a fixed-length compressed segment. The maximum length specified for a segment type must not exceed the maximum record length of the storage device used minus any prefix or record overhead. For VSAM, the maximum record length is 30713 bytes; for tape, the maximum is 32760 bytes. The minimum length that can be specified for maxbytes must be large enough to contain all fields defined for the segment type. If the segment is a logical child segment type, the length must be sufficient to contain the concatenated key of the logical parent.

For a Fast Path MSDB, the maxbytes value specifies the length of the data portion of a fixed-length segment not to exceed 32000 bytes. The value specified must be a multiple of 4.

Variable-length segments

A segment type is defined as variable-length if the minbytes operand is included. The maxbytes field specifies the maximum length of any occurrence of this segment type. The maximum and minimum allowable values for the maxbytes operand are the same values as described for a fixed-length segment. The maximum value must be the larger of (1) the largest segment that appears in the user's application program I/O area or (2) the largest segment stored on disk.

The minbytes operand specifies the minimum amount of storage used by a variable-length segment. The maximum value for minbytes is the value specified for maxbytes. The minimum value for minbytes must be:

- For a segment type that is not processed by an edit/compression routine or is processed by an edit/compression routine but the key compression option has not been specified, minbytes must be large enough to contain the complete sequence field if a sequence field has been specified for the segment type.
- For a segment type that is processed by an edit/compression routine that includes the key compression option or a segment that is not sequenced, the minimum value is 4.

Because segments in a HSAM or simple HISAM data base cannot be variable-length, the minbytes operand is invalid for these data bases.

In a Fast Path DEDB, a segment starts with a 2-byte variable-length segment field, followed by user data specified by a FIELD statement. The value of minbytes can be specified from a minimum of 4 bytes to a maximum of maxbytes; however, the minbytes value must be large enough to contain this segment's sequence field (that is, minbytes ≥ START + BYTES - 1 of a sequence field following the SEGM statement). On any given DL/I call, the actual segment length may fall anywhere between the value of minbytes and the value of maxbytes. The value of maxbytes must not exceed the control interval size minus 120.

TYPE=

Describes the type of DEDB dependent segment. Must not be specified for root segments.

SEQ

Specifies that the segment is a sequential dependent segment type. Only one sequential dependent segment is permitted per DEDB, and, if specified, it must be the first dependent segment type.

DIR

Specifies that the segment is direct dependent segment type. DIR is the default.

FREQ=

Is only used for HSAM, HISAM, or INDEX data bases. It specifies the estimated number of times that this segment is likely to occur for each occurrence of its physical parent. The frequency operand must be an unsigned decimal number in the range 0.01 to $2^{24}-1$. If this is a root segment, "frequency" is the estimate of the maximum number of data base records that appear in the data base being defined. The frequency of occurrence of the root segment is used to determine the number of tracks required for cylinder index and prime ISAM extent allocation. The value of the FREQ= operand when applied to dependent segments is used to determine the logical record length and physical storage block sizes for each data set group of the data base.

Note: The IF0110 ARITHMETIC OVERFLOW or IEV103 MULTIPLICATION OVERFLOW assembler error messages may occur when the DBDGEN utility is attempting to calculate a recommended logical record length. If this should occur during a HSAM or HISAM DBD generation, the user may want to determine the logical record length and physical block size.

POINTER= or PTR=

Specifies the pointer fields to be reserved in the prefix area of occurrences of the segment type being defined. These fields are used to relate this segment to its immediate parent segment(s) and twin segments. The following table indicates the keyword options that may be specified for this operand. Each of these keyword options is subsequently described.

The use of the POINTER= operand is primarily for HDAM and HIDAM data bases. In addition, it can be used for segment types defined in HISAM data bases which participate in logical relationships with segment types in HDAM or HIDAM data bases. If a segment type is being defined in an HSAM data base, the POINTER= operand must be omitted. If the segment type being defined is in a HISAM data base and does not participate in a logical relationship, the POINTER= operand should be omitted.

Pointer Keyword Options and Abbreviations
HIER[H], HIERBWD[HB], TWIN[T], TWINBWD[TB], NOTWIN[NT]
LTWIN[LT], LTWINBWD[LTB], PAIRED
LPARNT[LP]
CTR[C]

Note that:

- Selected keyword options may be specified in any order, and must be separated by commas.
- A keyword option may be specified only once, and all keywords are optional.
- One keyword option may be selected from each (line) of the above table.
- A keyword option or its abbreviation (indicated in brackets) may be selected.

The keyword options of this operand have the following meanings:

HIER (H)
Reserves a 4-byte hierarchic forward pointer field in the prefix of occurrences of the the segment type being defined.

HIERBWD (HB)
Reserves a 4-byte hierarchic forward pointer field and a 4-byte hierarchic backward pointer field in the prefix of occurrences of the segment type being defined. Hierarchic backward pointers provide increased delete performance.

TWIN (T)
Reserves a 4-byte physical twin forward pointer field in the segment prefix being defined.

For HIDAM data base root segments, refer to "Use of RAPs in a HIDAM Data Base" in Chapter 4 of IMS/VS Data Base Administration Guide for a more detailed explanation of the use of PTR=TWIN.

TWINBWD (TB)
Reserves a 4-byte physical twin forward pointer field and a 4-byte physical twin backward pointer field in the segment prefix being defined. The twin backward pointers provide increased delete performance.

This option is recommended for HIDAM data base root segments. Refer to "Use of RAPs in a HIDAM Data Base" in Chapter 4 of IMS/VS Data Base Administration Guide.

NOTWIN (NT)
Used to prevent reserving space for a physical twin forward pointer in the prefix of occurrences of the segment type being defined. NOTWIN can be specified for a dependent segment type if the physical parent does not have hierarchic pointers specified, and no more than one occurrence of the dependent segment type will be stored as a physical child of any occurrence of the physical parent segment type. In addition, NOTWIN can be specified for the root segment type of a HIDAM data base. When NOTWIN is specified for a dependent segment type and an attempt is made to load or insert a second occurrence of the dependent segment as a physical child of a given physical parent

segment, an LB status code is returned when trying to insert the second occurrence during initial load, and an II status code is returned when trying to insert the second occurrence after initial load. The NOTWIN option may be specified for HDAM root segments but only when the randomizing module will not produce synonyms (keys with different values having the same block and anchor point). Any attempt to load or insert a synonym will be rejected with an LB or II status code.

LTWIN (LT)
Used for virtually paired logical relationships only when defining a real logical child. Reserves a 4-byte logical twin forward pointer field in the prefix of occurrences of the logical child segment type being defined. This parameter may only be specified if the segment type being defined is a logical child and is being defined in an HDAM or HIDAM data base. It should be noted that if PAIRED is specified, the LTWIN parameter is invalid.

LTWINBWD (LTB)
Used for virtually paired logical relationships only when defining a real logical child. Reserves a 4-byte logical twin forward-pointer field and a 4-byte logical twin backward field in the prefix of occurrences of the logical child segment type being defined. This parameter may only be specified if the segment being defined is a logical child and is being defined in an HDAM or HIDAM data base. It should be noted that if PAIRED is specified, the LTWIN parameter is invalid.

The use of LTWINBWD rather than LTWIN provides increased performance when deleting logical child segments.

LPARNT (LP)
Reserves a 4-byte logical parent pointer field in the prefix of occurrences of the segment type being defined. This parameter may only be specified when the segment type being defined is a logical child and the logical parent is in an HDAM or HIDAM data base. If the logical parent is in a HISAM data base, this parameter must be omitted, and the PARENT= operand for the segment being defined must specify PHYSICAL.

CTR (C)
Reserves a 4-byte counter field in the prefix of occurrences of the segment type being defined. A counter is required if a logical parent segment in a HISAM, HDAM, or HIDAM data base has logical child segments which are not connected to it by logical child pointers. Counters are placed in all segments requiring them automatically during DBD generation without the user specifying this parameter. To avoid a later DBD generation, however, the user can anticipate future requirements for counters and reserve a counter field in the prefix of occurrences of a segment type by using this parameter.

PAIRED
Indicates that this segment participates in a bidirectional logical relationship. This parameter is specified for 1) a virtual logical child segment type, or 2) both physically paired logical child segment types in a bidirectional logical relationship. If PAIRED is specified, the LTWIN and LTWINBWD parameters are invalid.

POINTER= Operand Default Values

The default option for the POINTER= operand in any HIDAM or HDAM DBD is:

PTR=(TWIN,LTWIN,LPARNT)

where:

LTWIN
is a default if the name of a logical parent (lpsegname) is specified, in the PARENT= operand of a SEGM statement.

LPARNT
is a default if VIRTUAL is selected in the PARENT= operand of a SEGM statement.

The default option for the POINTER= operand in an INDEX, HISAM, or HSAM DBD is no pointer fields.

If the POINTER= operand is explicitly stated on a SEGM statement, all parameters of the operand must be explicitly stated. The default values are only employed when the operand is omitted entirely.

Figure 12 on page 50 illustrates use of the POINTER= operand parameters for various types of DBD generations.

			Segment Definition				
			Physical Segments Contained in Data Base Type				
General Nomenclature	Keyword Parameter	Logical Segments GSAM MSDB DEDB	HSAM SHSAM SHISAM	HISAM	HDAM	HIDAM	INDEX
Reserved							
Ptr to next segment in hierarchy	HIER	INVALID	INVALID	IGN	VALID	VALID	IGN
Ptr to next and previous segments in hierarchy	HIERBWD	INVALID	INVALID	IGN	VALID	VALID	IGN
Ptr to next occurrence of physical twins	TWIN	INVALID	INVALID	IGN	VALID	VALID	IGN
Ptr to next and previous occurrence of physical twins	TWINBWD	INVALID	INVALID	IGN	VALID	VALID	IGN
Counter field in prefix	CTR	INVALID	INVALID	VALID	VALID	VALID	IGN

Legend:

INVALID—This parameter cannot be specified.

IGN—This parameter may be specified but it is ignored.

VALID—This parameter is valid.

Figure 12 (Part 1 of 2). Use of POINTER= Operand Parameters (No Logical Relationship)

General Nomenclature	Keyword Parameter	Logical Segments GSAM MSDB DEDB	Segment Definition: Physical Segments Contained in Data Base Type: HSAM SHSAM SHISAM	HISAM	HDAM	HIDAM	INDEX
Ptr to next occurrence of logical twin	LTWIN	INVALID	INVALID	IGN	VALID (1)	VALID (1)	IGN
Ptr to next and previous occurrence of logical twin	LTWINBWD	INVALID	INVALID	IGN	VALID (1)	VALID (1)	IGN
Ptr to logical parent segment	LPARNT	INVALID	INVALID	VALID (2)	VALID (3)	VALID (3)	IGN
Logical relationship between HS - HS or HS - HD or HD - HD	PAIRED	INVALID	INVALID	VALID (4)	VALID (5)	VALID (5)	IGN

Legend:

INVALID—This parameter cannot be specified.

IGN—This parameter may be specified but will be ignored.

VALID—This parameter is valid and used as indicated in the following notes.

Notes:

1. Used when a logical child segment being defined participates in a logical relationship. This should be specified if the segment exists within HDAM or HIDAM and the logical parent relates to the logical child with direct addresses (logical child pointers).

2. Can be used when a logical child segment is being defined in a HISAM data base and the logical parent is defined in an HDAM or HIDAM data base.

3. Can be used when a logical child segment is being defined in an HDAM or HIDAM data base and the logical parent is in an HDAM or HIDAM data base.

4. Can be used when a logical child segment is being defined in a HISAM data base and the logical parent is defined in a HISAM, HDAM, or HIDAM data base, and the logical relationship is bidirectional.

5. Used when a bidirectional logical relationship is being defined with two logical child segments both physically present or on the SEGM statement for a virtual logical child.

Figure 12 (Part 2 of 2). Use of POINTER= Operand Parameters (No Logical Relationship)

RULES=B or (PPP,FIRST) or LLL LAST or VVV HERE
Specifies the rules used for insertion, deletion, and replacement of occurrences of the segment type being defined. See "RULES Coding" in Chapter 4 of IMS/VS Data Base Administration Guide for a description of the various uses of this keyword.

B or PP or LLL or VVV
These operands specify the path type that must be used to insert, delete, or replace a segment: P specifies physical, L specifies logical, V specifies virtual, and B specifies bidirectional virtual.

The first column of the parameter applies to segment insertion, the second column applies to segment deletion, and the third column applies to segment replacement. Each of the three columns can contain the same or different characters. These parameters are specified for logical child segments and their physical and logical parent segments. They should be omitted for all segment types which do not participate in logical relationships.

FIRST or LAST or HERE
Specifies where new occurrences of the segment type defined by this SEGM statement are inserted into their physical data base (establishes the physical twin sequence). This value is only used when processing segments with no sequence field or with a nonunique sequence field. The value is ignored when specified for a segment type with a unique sequence field defined.

Except for HDAM roots, the rules of FIRST, LAST, or HERE do not apply to the initial loading of a data base and segments are loaded in the sequence presented in load mode. If a unique sequence field is not defined for the HDAM root on initial load or HD reload, the insert rules of FIRST, LAST, or HERE determine the sequence in which roots are chained. Thus the reload of an HDAM data base will reverse the order of the unsequenced roots when HERE or FIRST is used.

When processing HDAM roots without a unique sequence field, in update mode, the sample randomizing modules provided with IMS/VS (DFSHDC10 through DFSHDC40) will use the segment I/O area data in order to calculate a block/rap for an insert call. See "HDAM Randomizing Modules" in IMS/VS System Programming Reference Manual for additional information.

The rules of FIRST, LAST, or HERE are only valid for update mode after a data base has been loaded, except for the HDAM exceptions noted above. LAST is the default except for Fast Path DEDB segments.

For Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden. For direct dependent segment processing, you can specify FIRST, LAST or HERE. HERE is the default.

FIRST
For segments without a sequence field defined, a new occurrence is inserted before all existing physical twins. For segments with a nonunique sequence field defined, a new occurrence is inserted before all existing physical twins with the same sequence field value.

LAST
For segments without a sequence field defined, a new occurrence is inserted after all existing physical twins. For segments with a nonunique sequence field defined, a new occurrence is inserted after all existing physical twins with the same sequence field value.

HERE
For segments without a sequence field, a new occurrence is inserted immediately before the physical twin on which position was established. If a position was not established on a physical twin of the segment being inserted, then the new occurrence is inserted before all existing physical twins. For segments with a nonunique sequence field defined, a new occurrence is inserted immediately before the physical twin with the same sequence field value on which position was established. If a position was not established on a physical twin with the same sequence field value, then the new occurrence is inserted before all physical twins with the same sequence field value. The insert position is dependent on the position established by the previous DL/I call.

For insert calls issued against a physical path, a command code of L (last) takes precedence over the insert rule specified, causing a new occurrence to be inserted according to the insert rule of LAST.

SOURCE=
Is used for two purposes:

- To identify the real logical child segment type that is to be represented by the virtual logical child segment type that is being defined.
- To identify the segment type or types in physical data bases that are represented by the segment type being defined in a logical data base.

When defining a virtual logical child:

SOURCE= ((segname,DATA,dbname))

where:

- segname specifies the name of the real, logical child
- DATA must be specified indicating that both the key and the data portions of segname are to be used in constructing the segment.
- dbname is the name of the physical data base that contains the real logical child.

When defining a segment type in a logical data base.

```
SOURCE=((segname,KEY,dbname), (segname,KEY,dbname))
                 DATA                  DATA
```

where:
The first occurrence of (segname, KEY/DATA, dbname) refers to the segment in a physical data base that is being defined as a logical segment in this logical data base, or it refers to the logical child segment type in a physical data base that is used for the first portion of a concatenated segment type in this logical data base.

segname
Is the name of the segment type in the physical data base.

KEY
Specifies that the key portion of the segment specified in segname is to be placed in the key feedback area, but the segment is not to be placed in the user I/O area when a call is issued to process the logical segment type that represents segname.

DATA
Specifies that the key portion of the segment specified in segname is to be placed in the key feedback area, and the segment is to be placed in the user I/O area when a call is issued to process the logical segment type that represents segname.

dbname
Specifies the name of the physical data base that contains segname.

where:
The second occurrence of (segname, KEY/DATA, dbname) refers to the logical or physical parent segment type in a physical data base that is used for the destination parent part of a concatenated segment in this logical data base. The description of each operand for the second occurence is the same as described for the first occurrence.

When the first occurrence of (segname, KEY/DATA, dbname) refers to a virtual logical child, the second occurrence, if specified, must refer to the real logical child's physical parent.

When the source segment(s) is used to represent a concatenated segment, the KEY and DATA parameters are used to control which of the two segments or both are placed in the users I/O area on retrieval calls. If DATA is specified, the segment is placed in the users I/O area. If KEY is specified, the segment is not placed in the users I/O area, but the sequence field key, if one exists, is placed in the key feedback area of the PCB. The key of a concatenated segment is the key of the logical child, either the physical twin sequence field or the logical twin sequence field, depending on which path the logical child is accessed from. The KEY and DATA parameters apply to retrieval type calls only.

On insert calls, the users I/O area must always contain the logical child segment and, unless the insert rule is physical, the logical parent segment. Even if KEY is specified for a segment, the data base containing that segment must be available to IMS/VS when calls are issued against the logical data base containing the referenced segment. When the first occurrence of the SOURCE= segment specification references a logical child, the second occurrence referencing the destination parent for the concatenated segment should also be specified. If not explicitly specified it will be effectively included with the KEY parameter by default when the blocks are built.

The segments defined with a logical DBD generation must gain their physical definition from segments previously defined in one or more physical DBD generations.

If the SEGM statement defines a segment in an INDEX data set, the SOURCE= operand is invalid.

COMPRTN=
Is used to select the segment edit/compression exit option. This operand must not be specified if the SOURCE operand is used. The COMPRTN operand is invalid during DBDGEN for MSDB, DEDB, HSAM, simple HSAM, simple HISAM, INDEX, and logical data bases. When used for a HISAM data base, it must not change the sequence field offset for HISAM root segments. In addition, the minimum segment length that can be specified for a segment type where the segment edit/compression option is specified is 4 bytes.

routinename
Specifies the name of the user-supplied edit/compression exit routine. This name must be a 1- to 8-character alphameric value and must not be the same as any other name in IMSVS.RESLIB.

DATA
Specifies that the indicated exit routine will condense or modify data fields only. Sequence fields must not be modified, nor data fields that change the position of the sequence field in respect to the start of the segment. DATA is the default value if a compression routine is named but no option is selected.

KEY
Specifies that the user exit routine can condense or modify any and all fields within the named segment. This parameter is invalid for the root segment of a HISAM data base.

INIT
Indicates that initialization and termination processing control is required by the segment exit routine. When this parameter is specified, the edit/compression routine will gain control after data base open and after data base close.

LCHILD STATEMENT

The LCHILD statement is used as follows:

- Defines a logical relationship between two segment types in a HISAM, HIDAM or HDAM data base or a logical relationship between a segment type in any two of these data bases.
- Defines a primary HIDAM index or secondary index relationship between two segment types.

LOGICAL RELATIONSHIPS: Following any SEGM statement that defines a logical parent segment type in a DBDGEN input deck, there must be one LCHILD statement for each segment type that is a logical child of that logical parent, except for virtual logical child segment types. These LCHILD statements establish the relationships between the logical parent and its logical child segment types. The SOURCE= operand of a SEGM statement that defines a virtual logical child segment type establishes the same relationship between a logical parent and a virtual logical child segment type.

PRIMARY HIDAM INDEX RELATIONSHIP: Two LCHILD statements are used to establish the index relationship required between the primary HIDAM index data base and the root segment type of a HIDAM data base.

Following the SEGM statement that defines the root segment type in a HIDAM data base DBD generation, there must be an LCHILD statement that names the index pointer segment type in an index data base. Following the SEGM statement that defines the index pointer segment type in a primary HIDAM index data base DBD generation there must be an LCHILD statement that names the root segment type in a HIDAM data base.

SECONDARY INDEX RELATIONSHIPS: Two LCHILD statements are used to establish each secondary index relationship. Following a SEGM statement that defines an index target segment type, there must be one LCHILD statement for each index pointer segment type that points to that index target segment type. Each LCHILD statement following the SEGM for an index target segment type identifies the index pointer segment type that points to the index target.

Following a SEGM statement that defines an index pointer segment type in a secondary index data base, there must be an LCHILD statement that identifies its index target segment type.

A maximum of 255 LCHILD statements can occur in a single DBD generation. An LCHILD statement may follow only a SEGM statement, FIELD statement, XDFLD statement, or another LCHILD statement. Because logical relationships and index relationships must not be defined in an HSAM data base, LCHILD statements are invalid when ACCESS=HSAM.

The format of the LCHILD statement is shown in Figure 13.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type						operands used for logical relationships		
				HSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	HDAM	HISAM	HIDAM
LCHILD	NAME=	(segname1	1		R	R	R	R		X	X	X
		,dbname)	1		R	R	R	R		X	X	X
	[,POINTER= ,PTR=]	SNGL	2					O		X	X	X
		DBLE								X	X	X
		NONE								X	X	X
		INDX	3			O	R					
		SYMB	4		O	O	O	O				
	,PAIR=	segname2								X	X	X
	,INDEX=	fldname						R				
	,RULES=	[FIRST LAST HERE]								X	X	X

KEY
O = Optional
R = Required
X = Use for logical relationships

Notes:

1. Logical relationships and secondary indexing
2. Required for primary index of HIDAM data base.
3. Required during a HIDAM DBD generation on the LCHILD statement that establishes the primary HIDAM index relationship. If PTR=INDX is specified for the target segment of a secondary index, then PTR must be omitted or specified as PTR=SNGL on the LCHILD statement of the INDEX DBD.
4. If symbolic pointing is specified for the index target segment type when defining its physical data base, symbolic pointing should be specified in the secondary index for that segment type. If SYMB is specified for the target segment of a secondary index then PTR=SYMB is specified on the LCHILD statement of the INDEX DBD also.

Figure 13. LCHILD Statement Format

The following abbreviations may be used in place of keywords specified in the above macro definition:

Keyword	Abbreviation
POINTER	PTR
FIRST	F
LAST	L
HERE	H

NAME=
The segnamel operand specifies the name of the logical child, index pointer, index target or HIDAM root segment type that is to be associated with the segment type defined by the preceding SEGM statement in the DBD generation input deck. The dbname operand is the name of the data base that contains the segment type specified in segnamel. dbname can be omitted when segnamel is defined in this DBD generation. Both segnamel and dbname must be 1- to 8-character alphameric values.

POINTER= or PTR=
Used to specify the pointers used in logical or index relationships. When the POINTER= keyword is omitted from any index DBD generation, POINTER=SNGL is the default. POINTER=INDX or SYMB must be specified for any LCHILD statement following an index target segment type; no default is provided for this part of the index relationship. When the POINTER= keyword is omitted from an LCHILD statement which establishes a unidirectional or physically paired bidirectional logical relationship, POINTER=NONE is the default. When the POINTER= keyword is omitted or specified as NONE for an LCHILD statement which establishes a virtually paired bidirectional logical relationship, POINTER=SNGL is the default.

SNGL
Used for logical relationships, or index relationships implemented with direct address pointers. SNGL specifies that a logical child first pointer field is to be reserved in each occurrence of the segment type defined by the preceding SEGM statement in the DBDGEN input deck. When the preceding SEGM defines a logical parent, the pointer field contains a direct address pointer to the first occurrence of a logical child segment type. When the preceding SEGM defines the primary HIDAM index data base segment type, the pointer field contains a direct address pointer to a HIDAM data base root segment. When the preceding SEGM defines an index pointer segment type in a secondary index data base, the pointer field contains a direct address pointer to an index target segment.

DBLE
Used to specify two 4-byte pointer fields, logical child first and logical child last, reserved in the logical parent segment. The two pointers point to the first and last occurrences of logical child segment type under a logical parent. The logical child last pointer is of value when the logical child is not sequenced and the RULES= operand is LAST.

NONE
Should be used when the logical relationship from the logical parent to the logical child segment is not implemented or not implemented with direct address logical child pointers. In this case, the relationship from logical parent to logical child does not exist or is maintained by using physically paired segments. No pointer fields are reserved in the logical parent segment.

INDX
Is specified on the LCHILD statement in a HIDAM data base used to establish the index relationship between the HIDAM root segment type and the primary HIDAM index during a HIDAM data base DBD generation. INDX can also be specified on the LCHILD statement in the DBD for the target data base that establishes the index relationship between an index target segment type and a secondary index. In these cases, omit the PTR= operand or specify PTR=SNGL on the LCHILD statement of the primary or secondary index DBD. An LCHILD statement for a HIDAM primary index must precede the LCHILD statements for secondary indexes.

SYMB
Can be used in the DBD generation for the target data base of a secondary index to specify that the concatenated keys of the index target segments are to be placed in the index pointer segments in lieu of a direct pointer. SYMB must be specified when the index target segment type is in a HISAM data base. SYMB is optional when the index target segment type is in an HDAM or HIDAM data base.

An additional use of the SYMB operand in the INDEX DBDGEN is to prevent reserving space in the prefix of index pointer segments for the 4-byte direct address index target segment pointer that is not used when the index pointer is symbolic.

PAIR=
Is specified segname2 for bidirectional logical relationships only. The segname2 operand is the name of the logical child segment that is, physically or virtually, paired with the logical child segment specified in segname1. The segname2 operand must be a 1- to 8-character alphameric value.

INDEX=
Is specified on LCHILD statements for an Index DBD generation only. The fldname operand specifies the name of the sequence field of a HIDAM root segment type during DBD generation of the primary index for a HIDAM data base, or the name of an indexed field, defined through an XDFLD statement in an index target segment type during DBD generation of a secondary index data base (establishes the logical twin sequence).

RULES=
Is used for logical relationships when no sequence field or a nonunique sequence field has been defined for a virtual logical child. Under these conditions, the rule of FIRST, LAST, or HERE controls the sequence in which occurrences of the real logical child in the logical relationship are sequenced from the logical parent through logical child and logical twin pointers.

FIRST
States that, if no sequence field is specified for the logical child, a new occurrence is inserted before the first existing occurrence of the logical child. If a nonunique sequence field is specified for the logical child, a new occurrence is inserted before all existing occurrences with the same key.

LAST
States that, if no sequence field is specified for the logical child, a new occurrence is inserted after the last existing occurrence of the logical child. If a nonunique sequence field is specified for the logical child, a new occurrence is inserted after all existing occurrences with the same keys. LAST is the default option.

HERE

States that the insert is dependent on the position established by the previous DL/I call. If no sequence field is defined, the segment will be inserted before the logical twin that position was established on through the previous call. If no position was established by a previous call, the new twin is inserted before all existing logical twins. If a nonunique sequence field is defined, the segment is inserted before the logical twin with the same sequence field value on which position was established by a previous call. If no position was established on a logical twin with the same sequence field value, the segment will be inserted before all twins with the same sequence field value.

When a new occurrence of a logical child is inserted from its physical parent, no previous position exists for the logical child on its logical twin chain. Therefore, the new occurrence will be placed before all existing occurrences on the logical twin chain when no sequence field has been defined, or before all existing occurrences with the same sequence field value when a nonunique sequence field has been defined.

Note: For insert calls issued against a logical path, a command code of L (last) takes precedence over the insert rule specified, causing a new occurrence to be inserted according to the insert rule of LAST.

FIELD STATEMENT

FIELD—Define a Field

The FIELD statement defines a field within a segment type. Fields are referred to by PSBs when defining sensitivity to the fields or by an application program in a DL/I call segment search argument. A maximum of 1000 fields can be defined for all segments in a DBD generation, and a maximum of 255 fields can be defined for any segment type. A unique sequence field must be defined for the root segment type of HISAM, HIDAM, primary HIDAM INDEX, SHISAM, Fast Path DEDB and nonterminal-related MSDB data bases. Root segment types in a HDAM data base do not need a key field defined; but if it is defined, it need not be unique.

FIELD statements may appear in a DBD generation for three purposes:

1. They can be used to describe fields of a segment type as that segment type is seen when it is accessed from its physical parent segment.

2. They can be used to describe the fields of a real logical child segment type in a virtually paired logical relationship as seen when that segment type is accessed from its logical parent. The FIELD statements must immediately follow the SEGM statement defining the virtual logical child.

3. They can be used to describe system-related fields that are used for secondary indexing.

The format of the FIELD statement is shown in Figure 14 on page 60.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES*	operands used by data base type								operands used for logical relationships		
				HSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
FIELD	NAME=	(fldname1		R	R	R	R	R		R	R			
		,SEQ		O	O	O	O	O		O	O			
		[,U / ,M])	1	O	O	O	O	O		O	O			
		or												
		systrelfldname			O	O	O							
	,BYTES=	bytes		R	R	R	R	R		R	R			
	,START=	startpos		R	R	R	R	R		R	R			
	,TYPE=	X		O	O	O	O	O		O	O			
		P		O	O	O	O	O		O	O			
		C		O	O	O	O	O		O	O			
		H								O				
		F								O				

KEY
O = Optional
R = Required
X = Use for logical relationships

Note:

1. M is not allowed for MSDB or DEDB root segments.

Figure 14. FIELD Statement Format

NAME=

fldname1
Specifies the name of the field being defined within a segment type. The name specified can be referred to by an application program in a DL/I call SSA. Duplicate field names must not be defined for the same segment type. fldname1 must be a 1- to 8-character alphameric value.

SEQ
Identifies this field as a sequence field in the segment type. FIELD statements containing the keyword SEQ must be the first FIELD statements following a SEGM statement in a DBD generation input deck. If the sequence field of a real logical child segment consists of any part of the logical parent's concatenated key, the PHYSICAL parameter must be specified in the SEGM statement for the logical child to include the concatenated key of the logical parent with the logical child in storage.

As a general rule, a segment can have only one sequence field. However, in the case of virtually paired bidirectional logical relationships, multiple FIELD statements may be used to define a logical sequence field for the virtual logical child segment type, as described below.

A sequence field must be specified for a virtual logical child segment type if when accessing a logical child segment from its logical parent, one requires real logical child segments to be retrieved in an order determined by data in a field or fields of the real logical child segments. This sequence field can include any part of the segment as it appears when viewed from the logical parent (that is, the concatenated key of the real logical child's physical parent followed by any intersection data). Since it may be necessary to describe the sequence field of a logical child segment as accessed from its logical parent segment in discontiguous pieces, multiple FIELD statements with the SEQ parameter present are permitted. Each statement must contain a unique fldnamel parameter.

When using the sequence field as a qualification in an SSA, any sequence field defined may be used, but all succeeding sequence fields will be considered as a part of the named field. Therefore, the length of the field named in the SSA will be the concatenated length of the specified field plus all succeeding sequence fields. Note that this "scattered" sequence field is permitted only when specifying the sequence field for a virtual logical child segment. If the first sequence field is not included in a "scattered" sequence field in an SSA, DL/I treats the argument as a data field specification rather than a sequence field specification. DL/I must examine all segment instances on a twin chain when a data field specification is evaluated. When a sequence field specification is evaluated the search continues along the twin chain until a sequence field value that is higher than the SSA value is reached. The search stops at that point.

In a Fast Path MSDB, the keyword SEQ must be specified if the DATASET statement specifies REL=NO (a nonterminal-related MSDB without terminal-related keys); otherwise this keyword is invalid.

In a Fast Path DEDB, SEQ must be used in the root segment and can be specified in any direct dependent segment. SEQ cannot be specified for the sequential dependent segment.

U or M
Qualify the type of sequence (SEQ) field being specified. The parameter U indicates that only unique values are allowed in the sequence field of occurrences of the segment type. For a root segment type, the sequence field of each occurrence must contain a unique value, except in HDAM. The root segment type in an HDAM data base do not need a key field defined; but if it is defined, it need not be unique. For a dependent segment type, the sequence field of each occurrence under a given physical parent segment must contain a unique value. The parameter M indicates that duplicate values are allowed in the sequence field of occurrences of the segment type.

When no sequence field or a nonunique sequence field is defined for a segment, occurrences of the segment will be inserted according to the rule of FIRST, LAST, or HERE as specified on the SEGM or LCHILD statement for that segment.

It is highly recommended that all segments which participate in a logical relationship have unique sequence fields. This includes physical and logical parents as well as physical and logical child segments. Multiple sequence fields for a virtual logical child segment type must be uniformly defined as either unique or nonunique.

In a nonterminal-related Fast Path MSDB without terminal-related keys, unique (U) values must be specified for the root sequence field. In a Fast Path DEDB, unique (U) values must be specified for the sequence field of the root segment. A dependent segment in a fast path DEDB does not require a key. However, if a key is defined, it must be unique.

systrelfldname
Defines a system related field which can only be used for secondary indexing. There are two types of system-related fields:

- All of or a portion of the concatenated key of an index source segment type defined by the preceding SEGM statement. The name for this type of system-related field can be up to 8 characters long, and must begin with the three characters '/CK'. The fourth through eighth characters permit unique identification of the field being defined whose name must be unique among all other fields defined in the segment type. This type of system-related field is defined to enable using the concatenated key of an index source segment, or portions of the concatenated key in the subsequence or duplicate data fields of index pointer segments. Assume the following concatenated key:

Root key (10 bytes)	Dependent key (3 bytes)	Dependent key (7 bytes)	Dependent key (8 bytes)

If three system-related fields were to consist of bytes 2 through 8 of the root key, byte 1 of the second key and bytes 5 and 6 of the fourth key, the FIELD statements specifying this could be as follows:

```
NAME=/CK1
BYTES=7
START=2

NAME=/CK2
BYTES=1
START=11

NAME=/CK3
BYTES=2
START=25
```

The three system-related fields defined can then be specified for use in the subsequence or duplicate data fields of index pointer segments by including the names of the system related fields in lists for the subsequence or duplicate data fields on an XDFLD statement.

- The second type of system-related field is defined within an index source segment type to ensure uniqueness of sequence field keys in a secondary index. The name specified for this type of system-related field must begin with the characters /SX, and the name specified can be up to 8 characters in length. When this type of system-related field is defined in an index source segment type, IMS/VS generates a unique 4-byte value, and places it in the subsequence field of the index pointer segment generated from an index source segment.

On an XDFLD statement, a /CK field can be included in the list of fields specified for either the subsequence or DDATA fields or both of an index pointer segment. A /SX field can only be included in the list of fields specified for the subsequence field of index pointer segments.

BYTES=
Specifies the length of the field being defined in bytes. For fields other than system-related fields, BYTES must be a valid self-defining term whose value does not exceed 255. If a concatenated key or a portion of a concatenated key of an index source segment type is defined as a system-related field, the value specified can be greater than 255, but must not exceed the length of the concatenated key of the index source segment. The length of a /SX system-related field is always 4 bytes; therefore, when specified, the BYTES operand is disregarded. For the sequence field of a Fast Path MSDB segment, BYTES must not exceed 240. For the sequence field of a Fast Path DEDB segment, BYTES must not exceed the value of minbytes specified for the segment.

START=
Specifies the starting position (startpos) of the field being defined in terms of bytes relative to the beginning of the segment. Startpos must be a numeric term whose value does not exceed 32767. Startpos for the first byte of a segment is one. For variable-length segments, the first 2 bytes contain the length of the segment. Therefore the first actual user data field starts in byte 3. Overlapping fields are permitted. When a SEGM statement defines a logical child segment, the first n bytes of the segment type is the logical or physical parent's concatenated key. A field starting in position one would define all or a portion of this field. A field starting in position n+1 would start with intersection data.

If used for a system-related field to describe a portion of the concatenated key as a field in an index source segment type, START= specifies the starting position of this portion of the concatenated key relative to the beginning of the concatenated key, the first byte of which is considered to have a position of one. In this case, it must be a numeric term whose value does not exceed the length of the concatenated key plus 1 minus the value specified in the BYTES operand. The startpos operand for the /SX system-related field is disregarded.

TYPE=
Specifies the type of data that is to be contained in this field. The value of the parameter specified for this operand indicates that one of the following types of data will be contained in this field:

X = hexadecimal data
P = packed decimal data
C = alphameric data or a combination of types of data
F = binary fullword data
H = binary halfword data

It should be noted that all DL/I calls perform field comparisons on a byte-by-byte binary basis. No check is made by IMS/VS to ensure that the data contained within a field is of the type specified by this operand, except when the defined field is indexed or is in a Fast Path MSDB (see rules below).

Types X, C, P, H, and F are valid in a Fast Path MSDB, with the following rules applying:

- Only a C or X field can contain another field.
- A single field can have multiple definitions as long as no more than one definition is arithmetic (types P, H, and F).

- If a field contains any part of an arithmetic field, it must contain the entire field.
- The sequence field must be TYPE=C or X.
- The sequence field cannot be part of any other field.
- SSA and FSA comparisons of arithmetic fields use arithmetic rather than logical compare operations.
- Initial loading and call processing routines test for valid digits and X and P type fields.
- The following rules apply to the Fast Path MSDB field length:
  - TYPE=X: BYTES=1 to 256
  - TYPE=P: BYTES=1 to 16
  - TYPE=C: BYTES=1 to 256
  - TYPE=F: BYTES=4
  - TYPE=H: BYTES=2
  - Field types F and H must have explicit length specifications as shown above.
  - Fields should be aligned on appropriate boundaries for performance optimization if they are involved in compare or arithmetic operations and are a fullword or halfword long. The beginning of the segment will be aligned on a fullword boundary.

When sensitivity to a field has been defined, the field is filled with a value under these conditions:

1. When the application program is not sensitive to this field on an insert call.
2. When:
   - The application program replaces a variable-length segment with a segment that is longer than the existing segment,
   - This field is in the added portion of the segment, and
   - The application program is not sensitive to this field.
3. When the application program retrieves a variable-length segment that does not contain this field.

The TYPE parameter determines the value to be used, as follows:

TYPE	Value Used
X	binary zeros
P	packed decimal zero
C	blanks

If an alphameric field (TYPE=C) is partially present in the physical segment, the data is moved to the field in the user's I/O area and padded on the right with blanks. Partially present hexadecimal or packed decimal fields are replaced with the fill value when presented to the user.

XDFLD—Define an Indexed Field

The XDFLD statement is used only for secondary index relationships. Its purpose is to define the name of an indexed field that is associated to an index target segment type, identify the index source segment type, and identify the index source segment fields that are used in creating a secondary index. In addition, information regarding suppressing the creation of index pointer segments is provided through this control statement. This statement may not be used to reference a segment in a DBD where ACCESS=INDEX, SHSAM, SHISAM, HSAM, MSDB, or DEDB has been specified.

A maximum of 32 XDFLD statements are allowed per SEGM statement. The number of XDFLD and FIELD statements combined must not exceed 255 per SEGM statement, and must not exceed 1000 per DBD generation.

One XDFLD statement is required for each secondary index relationship. It must appear in the DBD generation input deck for the indexed data base after the LCHILD statement that references the index pointer segment. Only FIELD statements for the index target segment may appear between the LCHILD statement and the associated XDFLD statement in the input deck. The index target segment, which is the segment defined by the preceding SEGM statement in the DBD generation input deck must not be either a logical child segment type or a dependent of a logical child segment type.

The format of the XDFLD statement is shown in Figure 15 on page 66.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type						operands used for logical relationships		
				HSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	HDAM	HISAM	HIDAM
XDFLD	NAME=	fldname	1		R	R	R					
	,SEGMENT=	segname			O	O	O					
	,CONST=	char	2		O	O	O					
	,SRCH=	list1	2		R	R	R					
	,SUBSEQ=	list2	2		O	O	O					
	,DDATA=	list3			O	O	O					
	,NULLVAL=	value1			O	O	O					
	,EXTRTN=	name1			O	O	O					

KEY

O = Optional
R = Required
X = Use for logical relationships

Notes:

1. An XDFLD statement is not allowed during DBD generation of a simple HISAM data base.

2. The combined length of the constant, search, and subsequence fields must not exceed 240 bytes.

Figure 15. XDFLD Statement Format

NAME=
Specifies the name of the indexed data field of an index target segment. The name specified actually represents the search field of an index pointer segment type as being a field in the index target segment type. The name specified can be used to qualify SSAs of calls for an index target segment type through the search field keys of index pointer segments. This enables accessing occurrences of an index target segment type through a primary or secondary processing sequence based on data contained in a secondary index. fldname must be a 1- to 8-character alphameric value. Since the name specified is used to access occurrences of the index target segment type based on the content of a secondary index, the name specified must be unique among all field names specified for the index target segment type.

SEGMENT=
Specifies the index source segment type for this secondary index relationship. segname must be the name of a subsequently defined segment type, which is hierarchically below the index target segment type or it can be the name of the index target segment type itself. The segment name specified must not be a logical child segment. If this operand is omitted, the index target segment type is assumed to be the index source segment.

CONST=
Specifies a character with which every index pointer segment in a particular secondary index is identified. This operand is optional. The purpose of this operand is to identify all index pointer segments associated with each secondary index when multiple secondary indexes reside in the same secondary index data base. Char must be a 1-byte self-defining term.

SRCH=
Specifies which field or fields of the index source segment are to be used as the search field of a secondary index. List1 must be a list of one to five field names defined in the index source segment type by FIELD statements. If two or more names are included, they must be separated by commas and enclosed in parentheses. The sequence of names in the list is the sequence in which the field values will be concatenated in the index pointer segment search field. The sum of the lengths of the participating fields constitutes the index target segment indexed field length which must be reflected in segment search arguments.

SUBSEQ=
Specifies which, if any, fields of the index source segment are to be used as the subsequence field of a secondary index. List2 must be a list of one to five field names defined in the index source segment by FIELD statements. If two or more names are included, they must be separated by commas and enclosed in parentheses. The sequence of names in the list is the sequence in which field values will be concatenated in the index pointer segment subsequence field. This operand is optional.

DDATA=
Specifies which, if any, fields of the index source segment are to be used as the duplicate data field of a secondary index. List3 must be a list of one to five field names defined in the index source segment by FIELD statements. If two or more names are included, they must be separated by commas and enclosed in parentheses. The sequence of names in the list is the sequence in which field values will be concatenated in the index pointer segment duplicate data field. This operand is optional.

NULLVAL=
Enables suppressing the creation of index pointer segments when the index source segment data used in the search field of an index pointer segment contains the specified value.

The value1 operand must be a 1-byte self-defining term (X'10',C'Z', 5, or B'00101101') or the words BLANK or ZERO. BLANK is equivalent to C' ' or X'40'. ZERO is equivalent to X'00' or 0, but not C'0'. If a packed decimal value is required, it must be specified as a hexadecimal term with a valid number digit and zone or sign digit (X'3F' for a packed positive 3 or X'9D' for negative 9).

No indexing is performed when each field of the index source segment specified in the SRCH= operand has the value of this operand in every byte. For example, if the NULLVAL=C'9' were specified, then the associated index would have no entries indexed on the value C'9999...9'.

There is a slight difference in the case of packed fields. For packed fields, each field which composes the search field is considered to be a separate packed value. For example, if the NULLVAL=X'9F' were specified in a case where the search field was composed of three 2-byte packed source fields, there would be no index entries with the search field value of X'999F999F999F' because these and only these would be suppressed.

Also, with the same NULLVAL=X'9F', if the search field were one 6-byte field, then no indexing would be performed whenever the value of the search field was X'99999999999F'.

The only form of the sign that will be checked is the form specified. For example, if X'9C' is specified, X'9F' will not cause suppression.

EXTRTN=
Specifies the name of a user-supplied index maintenance exit routine that is used to suppress the creation of selected index pointer segments. The operand (name1) must be the name of a user-supplied routine which will get control whenever DL/I attempts to insert, delete or replace an index entry because of changes occurring in the indexed data base. This exit routine can inspect the affected index source segment and decide whether or not an index pointer segment should be generated.

If both the NULLVAL= and the EXTRTN= operands are specified, indexing of a segment will be performed only if neither causes suppression.

DBDGEN STATEMENT

The DBDGEN statement indicates the end of DBD generation control statements used to define the DBD. This statement must be included.

The format of the DBDGEN statement is shown in Figure 16.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
DBDGEN				R	R	R	R	R	R	R	R	R			

KEY
O = Optional
R = Required
X = Use for logical relationships

Figure 16. DBDGEN Statement Format

FINISH STATEMENT

The FINISH statement is optional and is retained for compatibility.

The format of the FINISH statement is shown in Figure 17.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
FINISH				O	O	O	O	O	O	O	O	O			

KEY
O = Optional
R = Required
X = Use for logical relationships

Figure 17. FINISH Statement Format

END STATEMENT

The END statement indicates the end of input statements to the OS/VS assembler. This statement must be included.

The format of the END statement is shown in Figure 18.

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type									operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	MSDB	DEDB	HDAM	HISAM	HIDAM
END				R	R	R	R	R	R	R	R	R			

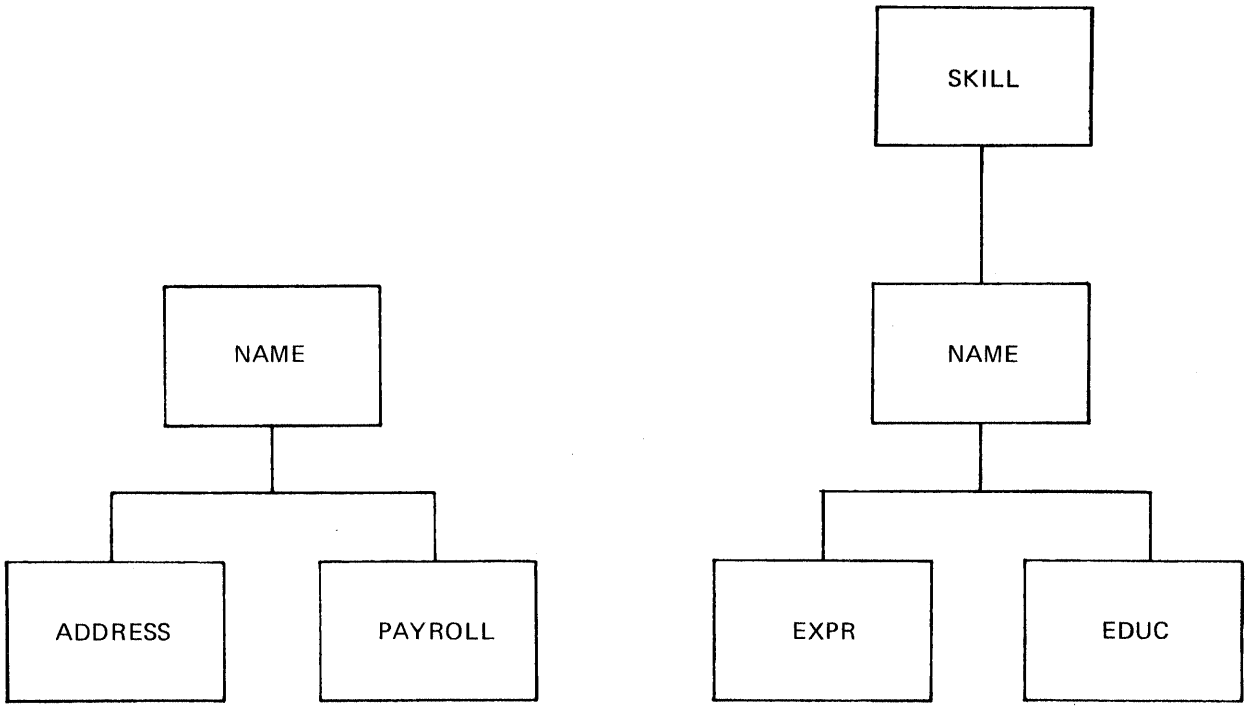
KEY
O = Optional
R = Required
X = Use for logical relationships

Figure 18. END Statement Format

DBD GENERATION EXAMPLES

EXAMPLES WITHOUT SECONDARY INDEX OR LOGICAL RELATIONSHIPS

The DBD generation examples provided in the following section show the statements that are required to define HSAM, HISAM, HDAM, HIDAM, primary HIDAM Index, GSAM, and Fast Path main storage and data entry data bases without secondary indexes or logical relationships. The examples in Figure 20 on page 71 through Figure 24 on page 75 (excludes the GSAM, MSDB, and DEDB examples) are based on either one or both of the data structures depicted below in Figure 19.



Payroll data structure

Skills inventory data structure

Figure 19. Payroll and Skills Inventory Data Structures

HSAM DBD GENERATION

The examples in Figure 20 (below) show the DBD generation control statements that define the skills inventory and payroll data structures as HSAM data bases.

HSAM DBD Generation of Skills Inventory Data Base

```
DBD    NAME=SKILLINV,ACCESS=HSAM
DATASET DD1=SKILHSAM,DD2=HSAMOUT,DEVICE=TAPE,BLOCK=1,
       RECORD=3000

SEGM   NAME=SKILL,BYTES=31,FREQ=100
FIELD  NAME=TYPE,BYTES=21,START=1,TYPE=C
FIELD  NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEGM   NAME=NAME,BYTES=20,FREQ=500,PARENT=SKILL
FIELD  NAME=STDCLEVL,BYTES=20,START=1,TYPE=C

SEGM   NAME=EXPR,BYTES=20,FREQ=10,PARENT=NAME
FIELD  NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD  NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM   NAME=EDUC,BYTES=75,FREQ=5,PARENT=NAME
FIELD  NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD  NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

HSAM DBD Generation of Payroll Data Base

```
DBD    NAME=PAYROLDB,ACCESS=HSAM
DATASET DD1=PAYROLL,DD2=PAYOUT,BLOCK=1,RECORD=1000,DEVICE=TAPE

SEGM   NAME=NAME,BYTES=150,FREQ=1000,PARENT=0
FIELD  NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1,TYPE=C
FIELD  NAME=MANNBR,BYTES=15,START=61,TYPE=C
FIELD  NAME=ADDR,BYTES=75,START=76,TYPE=C

SEGM   NAME=ADDRESS,BYTES=200,FREQ=2,PARENT=NAME
FIELD  NAME=HOMEADDR,BYTES=100,START=1,TYPE=C
FIELD  NAME=COMAILOC,BYTES=100,START=101,TYPE=C

SEGM   NAME=PAYROLL,BYTES=100,FREQ=1,PARENT=NAME
FIELD  NAME=HOURS,BYTES=15,START=51,TYPE=P
FIELD  NAME=BASICPAY,BYTES=15,START=1,TYPE=P

DBDGEN
FINISH
END
```

Figure 20. HSAM DBD Generation

HISAM DBD GENERATION

The examples in Figure 21 show the DBD generation control statements that define the skills inventory and payroll data structures as HISAM data bases.

HISAM DBD Generation of Skills Inventory SKILLINV Data Base

```
DBD    NAME=SKILLINV,ACCESS=HISAM
DATASET DD1=SKLHISAM,OVFLW=HISAMOVF,DEVICE=3350

SEGM   NAME=SKILL,BYTES=31,FREQ=100
FIELD  NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD  NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEGM   NAME=NAME,BYTES=20,FREQ=500,PARENT=SKILL
FIELD  NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM   NAME=EXPR,BYTES=20,FREQ=10,PARENT=NAME
FIELD  NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD  NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM   NAME=EDUC,BYTES=75,FREQ=5,PARENT=NAME
FIELD  NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD  NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

HISAM DBD Generation of Payroll Data Base

```
DBD    NAME=PAYROLDB,ACCESS=HISAM
DATASET DD1=PAYROLL,OVFLW=PAYROLOV,DEVICE=3350

SEGM   NAME=NAME,BYTES=150,FREQ=1000,PARENT=0
FIELD  NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1,TYPE=C
FIELD  NAME=MANNBR,BYTES=15,START=61,TYPE=C
FIELD  NAME=ADDR,BYTES=75,START=76,TYPE=C

SEGM   NAME=ADDRESS,BYTES=200,FREQ=2,PARENT=NAME
FIELD  NAME=HOMEADDR,BYTES=100,START=1,TYPE=C
FIELD  NAME=COMAILOC,BYTES=100,START=101,TYPE=C

SEGM   NAME=PAYROLL,BYTES=100,FREQ=1,PARENT=NAME
FIELD  NAME=HOURS,BYTES=15,START=51,TYPE=P
FIELD  NAME=BASICPAY,BYTES=15,START=1,TYPE=P

DBDGEN
FINISH
END
```

Figure 21. HISAM DBD Generations

HDAM DBD GENERATION

The examples in Figure 22 show the control statements required to define the skills inventory data structure as HDAM data bases. The first example defines a data base that uses heirarchic pointers, and the second example defines a data base that uses physical child and physical twin pointers.

HDAM DBD Generation of Skills Inventory KSILLINV Data Base with Hierarchic Pointers

```
DBD     NAME=SKILLINV,ACCESS=HDAM,RMNAME=(RAMDMODL,1,500,824)
DATASET     DD1=SKILHDAM,DEVICE=3350,BLOCK=1648,SCAN=5

SEGM    NAME=SKILL,BYTES=31,PTR=H,PARENT=0
FIELD   NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD   NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEGM    NAME=NAME,BYTES=20,PTR=H,PARENT=SKILL
FIELD   NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM    NAME=EXPR,BYTES=20,PTR=H,PARENT=NAME
FIELD   NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD   NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM    NAME=EDUC,BYTES=75,PTR=H,PARENT=NAME
FIELD   NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD   NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

HDAM DBD Generation of Skills Inventory Data Base with Physical Child and Physical Twin Pointers

```
DBD     NAME=SKILLINV,ACCESS=HDAM,RMNAME=(RAMDMODL,1,500,824)
DATASET     DD1=SKILHDAM,DEVICE=3350,BLOCK=1648,SCAN=5

SEGM    NAME=SKILL,BYTES=31,PTR=T,PARENT=0
FIELD   NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD   NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEGM    NAME=NAME,BYTES=20,PTR=T,PARENT=((SKILL,SNGL))
FIELD   NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM    NAME=EXPR,BYTES=20,PTR=T,PARENT=((NAME,SNGL))
FIELD   NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD   NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM    NAME=EDUC,BYTES=75,PTR=T,PARENT=((NAME,SNGL))
FIELD   NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD   NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

Figure 22. HDAM DBD Generations

HIDAM DBD GENERATION

A HIDAM data base is indexed through the sequence field of its root segment type. In defining the HIDAM and primary HIDAM index data bases, an index relationship is established between the HIDAM root segment type and the segment type defined in the primary HIDAM index data base. Figure 23 summarizes the statements required to establish the index relationship between the HIDAM root segment type and the index segment type in the primary HIDAM index data base. Only those operands pertinent to the index relationship are shown.

Primary HIDAM Index Relationship

```
HIDAM:                              INDEX:

DBD NAME=dbd1,ACCESS=HIDAM          DBD NAME=dbd2,ACCESS=INDEX

SEGM NAME=seg1,BYTES=,              SEGM NAME=seg2,BYTES=
     POINTER=

LCHILD NAME=(seg2,dbd2),            LCHILD NAME=(seg1,dbd1),
       PTR=INDX                            INDEX=fld1

FIELD NAME=(fld1,SEQ,U),            FIELD NAME=(fld2,SEQ,U),
      BYTES=,START=                       BYTES=,START=
```

Figure 23. Summary of Statements for the Primary HIDAM Index Relationship

The next two examples show the control statements that define the skills inventory data structure as two HIDAM data bases. The first is defined with heirarchic pointers, and the second is defined with physical child and physical twin pointers. Since a HIDAM data base is indexed on the sequence field of its root segment type, an INDEX DBD generation is required. Figure 24 shows the control statements for the two HIDAM DBD generations and the index DBD generation.

HIDAM DBD Generation of Skills Inventory Data Base with Hierarchic Pointers

```
DBD     NAME=SKILLINV,ACCESS=HIDAM
DATASET    DD1=SKLHIDAM,DEVICE=3350,BLOCK=1648,SCAN=5

SEGM  NAME=SKILL,BYTES=31,PTR=H,PARENT=0
FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C
LCHILD NAME=(INDEX,INDEXDB),PTR=INDX

SEGM  NAME=NAME,BYTES=20,PTR=H,PARENT=SKILL
FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM  NAME=EXPR,BYTES=20,PTR=H,PARENT=NAME
FIELD NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM  NAME=EDUC,BYTES=75,PTR=H,PARENT=NAME
FIELD NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

HIDAM DBD Generation of Skills Inventory SKILLINV Data Base with Physical Child and Physical Twin Pointers

```
DBD     NAME=SKILLINV,ACCESS=HIDAM
DATASET    DD1=SKLHIDAM,DEVICE=3350,BLOCK=1648,SCAN=5

SEGM  NAME=SKILL,BYTES=31,PTR=T,PARENT=0
LCHILD NAME=(INDEX,INDEXDB),PTR=INDX

FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEGM   NAME=NAME,BYTES=20,PTR=T,PARENT=((SKILL,SNGL))
FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM  NAME=EXPR,BYTES=20,PTR=T,PARENT=((NAME,SNGL))
FIELD NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM  NAME=EDUC,BYTES=75,PTR=T,PARENT=((NAME,SNGL))
FIELD NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

INDEX DBD Generation for HIDAM Data Base SKILLINV

```
DBD   NAME=INDEXDB,ACCESS=INDEX
DATASET DD1=INDXDB1,DEVICE=3350
SEGM  NAME=INDEX,BYTES=21,FREQ=10000
LCHILD NAME=(SKILL,SKILLINV),INDEX=TYPE
FIELD NAME=(INDXSEQ,SEQ,U),BYTES=21,START=1
DBDGEN
FINISH
END
```

Figure 24. HIDAM and Primary HIDAM Index DBD Generations

GSAM DBD GENERATION

Figure 25 shows the DBD generation control statements that define input and output data sets for a GSAM data base.

```
DBD   NAME=CARDS,ACCESS=(GSAM,BSAM)
DATASET  DD1=ICARDS,DD2=OCARDS,RECFM=F,RECORD=80
DBDGEN
FINISH
END
```

Figure 25. GSAM DBD Generations

MSDB DBD GENERATION

Figure 26 shows the DBD generation statements necessary to define the three types of main storage data base DBDs.

DBD Generation for a Nonterminal-Related MSDB without LTERM Keys

```
DBD     NAME=MSDBLM02,ACCESS=MSDB
DATASET REL=NO
SEGM    NAME=LDM,BYTES=4
FIELD   NAME=(FIELDSEQ,SEQ,U),BYTES=1,START=1,TYPE=X
DBDGEN
FINISH
END
```

DBD Generation for a Nonterminal-Related MSDB with LTERM Keys

```
DBD     NAME=MSDBLM04,ACCESS=MSDB
DATASET REL=(TERM,FIELDLDM)
SEGM    NAME=LDM,BYTES=52
FIELD   NAME=FIELDSEQ,BYTES=4,START=1,TYPE=C
FIELD   NAME=FIELDX01,BYTES=2,START=5,TYPE=X
FIELD   NAME=FIELDC01,BYTES=2,START=5,TYPE=C
FIELD   NAME=FIELDH01,BYTES=2,START=7,TYPE=H
FIELD   NAME=FIELDF01,BYTES=4,START=9,TYPE=F
FIELD   NAME=FIELDC03,BYTES=2,START=13,TYPE=C
FIELD   NAME=FIELDP01,BYTES=2,START=13,TYPE=P
FIELD   NAME=FIELDP02,BYTES=1,START=15,TYPE=P
FIELD   NAME=FIELDP03,BYTES=16,START=16,TYPE=P
FIELD   NAME=FIELDH02,BYTES=2,START=32,TYPE=H
FIELD   NAME=FIELDF02,BYTES=4,START=34,TYPE=F
FIELD   NAME=FIELDX03,BYTES=12,START=38,TYPE=X
FIELD   NAME=FIELDH03,BYTES=2,START=50,TYPE=H
DBDGEN
FINISH
END
```

DBD Generation for a Fixed Terminal-Related MSDB

```
DBD     NAME=MSDBLM05,ACCESS=MSDB
DATASET REL=(FIXED,FIELDLDM)
SEGM    NAME=LDM,BYTES=52
FIELD   NAME=FIELDSEQ,BYTES=4,START=1,TYPE=C
FIELD   NAME=FIELDX01,BYTES=2,START=5,TYPE=X
FIELD   NAME=FIELDC01,BYTES=2,START=5,TYPE=C
FIELD   NAME=FIELDH01,BYTES=2,START=7,TYPE=H
FIELD   NAME=FIELDF01,BYTES=4,START=9,TYPE=F
FIELD   NAME=FIELDC03,BYTES=2,START=13,TYPE=C
FIELD   NAME=FIELDP01,BYTES=2,START=13,TYPE=P
FIELD   NAME=FIELDP02,BYTES=1,START=15,TYPE=P
FIELD   NAME=FIELDP03,BYTES=16,START=16,TYPE=P
FIELD   NAME=FIELDH02,BYTES=2,START=32,TYPE=H
FIELD   NAME=FIELDF02,BYTES=4,START=34,TYPE=F
FIELD   NAME=FIELDX03,BYTES=12,START=38,TYPE=X
FIELD   NAME=FIELDH03,BYTES=2,START=50,TYPE=H
DBDGEN
FINISH
END
```

Figure 26 (Part 1 of 2). Main Storage Data Base DBD Generations

DBD Generation for a Dynamic Terminal-Related MSDB

```
DBD     NAME=MSDBLM06,ACCESS=MSDB
DATASET REL=(DYNAMIC,FIELDLDM)
SEGM    NAME=LDM,BYTES=52
FIELD   NAME=FIELDSEQ,BYTES=4,START=1,TYPE=C
FIELD   NAME=FIELDX01,BYTES=2,START=5,TYPE=X
FIELD   NAME=FIELDC01,BYTES=2,START=5,TYPE=C
FIELD   NAME=FIELDH01,BYTES=2,START=7,TYPE=H
FIELD   NAME=FIELDF01,BYTES=4,START=9,TYPE=F
FIELD   NAME=FIELDC03,BYTES=2,START=13,TYPE=C
FIELD   NAME=FIELDP01,BYTES=2,START=13,TYPE=P
FIELD   NAME=FIELDP02,BYTES=1,START=15,TYPE=P
FIELD   NAME=FIELDP03,BYTES=16,START=16,TYPE=P
FIELD   NAME=FIELDH02,BYTES=2,START=32,TYPE=H
FIELD   NAME=FIELDF02,BYTES=4,START=34,TYPE=F
FIELD   NAME=FIELDX03,BYTES=12,START=38,TYPE=X
FIELD   NAME=FIELDH03,BYTES=2,START=50,TYPE=H
DBDGEN
FINISH
END
```

Figure 26 (Part 2 of 2). Main Storage Data Base DBD Generations

DEDB DBD GENERATION

Figure 27 shows the DBD generation statements necessary to define a data entry data base DBD.

DBD Generation for a DEDB

```
DEDB1       DBD     NAME=DEDB0001,ACCESS=DEDB,RMNAME=RMOD1
AREA0       AREA    DD1=DB1AREA0,          AREA 0
                    DEVICE=3330,MODEL=1,SIZE=1024,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA1       AREA    DD1=DB1AREA1,          AREA 1
                    DEVICE=3330,MODEL=11,SIZE=1024,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA2       AREA    DD1=DB1AREA2,          AREA 2
                    DEVICE=3340,SIZE=1024,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA3       AREA    DD1=DB1AREA3,          AREA 3
                    DEVICE=3340,SIZE=4096,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA4       AREA    DD1=DB1AREA4,          AREA 4
                    DEVICE=2314,SIZE=1024,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA5       AREA    DD1=DB1AREA5,          AREA 5
                    DEVICE=2314,SIZE=512,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA6       AREA    DD1=DB1AREA6,          AREA 6
                    DEVICE=2305,MODEL=1,SIZE=2048,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA7       AREA    DD1=DB1AREA7,          AREA 7
                    DEVICE=2305,MODEL=2,SIZE=4096,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA8       AREA    DD1=DB1AREA8,          AREA 8
                    DEVICE=3350,SIZE=1024,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
AREA9       AREA    DD1=DB1AREA9,          AREA 9
                    DEVICE=3350,SIZE=2048,
                    ROOT=(10,5),           5 UOW'S/AREA
                    UOW=(15,10)            5 A.P.'S + 10 DEP. OFLOW.
ROOTSEG     SEGM    NAME=ROOTSEG1,PARENT=0,BYTES=(300,50)
ROOTLFLD    FIELD   NAME=(ROOTKEY1,SEQ,U),BYTES=8,START=3,TYPE=C
SDSEG       SEGM    NAME=SDSEGNM1,PARENT=ROOTSEG1,BYTES=(300,50),
                    TYPE=SEQ
SDFLD       FIELD   NAME=SDSCFLD1,BYTES=10,START=3,TYPE=C
DDSEG       SEGM    NAME=DDSEGNM1,PARENT=ROOTSEG1,
                    BYTES=(40,15),TYPE=DIR
DDFLD1      FIELD   NAME=(DD1FLD1,SEQ,U),BYTES=4,START=6
DDFLD2      FIELD   NAME=DD1FLD2,BYTES=5,START=10,TYPE=P
            DBDGEN
            FINISH
            END
```

Figure 27. Data Entry Data Base DBD Generations

Figure 28 shows the DBD generation statements necessary to define a DEDB with subset pointers.

DBD Generation for DEDB Subset Pointers

```
DBD NAME=DEDBDB,ACCESS=DEDB,RMNAME=DBFHD040
AREA  DD1=DEDBDD,DEVICE=3330,MODEL=1,SIZE=1024,
            ROOT=(10,5),UOW=(15,10)
SEGM  NAME=A,BYTES=(48,27),PARENT=0
FIELD NAME=(A1,SEQ,U),BYTES=10,START=3,TYPE=C
SEGM  NAME=B,BYTES=(24,11),PARENT=((A,SNGL)),TYPE=DIR,SSPTR=5
FIELD NAME=(B1,SEQ,U),BYTES=5,START=3,TYPE=C
FIELD NAME=B2,BYTES=5,START=10,TYPE=C
SEGM  NAME=C,BYTES=(34,32),PARENT=((B,DBLE)),RULES=(,HERE),TYPE=DIR
FIELD NAME=(C1,SEQ,U),BYTES=20,START=3,TYPE=C
SEGM  NAME=D,BYTES=(52,33),PARENT=((A,DBLE)),TYPE=DIR,SSPTR=3
FIELD NAME=(D1,SEQ,U),BYTES=2,START=3,TYPE=C
SEGM  NAME=B,BYTES=(52,33),PARENT=((A,DBLE)),RULES=(,FIRST),TYPE=DIR
FIELD NAME=(B1,SEQ,U),BYTES=2,START=3,TYPE=C
DBDGEN
FINISH
END
```

Note: SSPTR=n, where n indicates the number of subset pointers

Figure 28. DBD Generation of DEDB Subset Pointers Sample

SUMMARY OF PHYSICAL DATA BASE DESCRIPTION EXAMPLES

An application program through a data base PCB may operate on any of the data bases previously described. The value of the DBDNAME= operand on the data base PCB control statement should equal the value of the NAME= operand on a DBD control card statement of DBD generation. The SENSEG statements following the data base PCB statements in PSB generation should reference segments defined by SEGM statements in the named DBD generation.

When a HIDAM data base is used by an application program, the value of the DBDNAME= operand on the PCB statement should equal the value of the NAME= operand on the DBD statement for the HIDAM DBD generation. The LCHILD statement in the HIDAM DBD provides IMS/VS with the relationship to the necessary INDEX DBD and index data base. The INDEX DBD name should not be specified in the DBDNAME= operand of a data base PCB.

EXAMPLES WITH LOGICAL RELATIONSHIPS

Figure 29 shows the three types of logical relationships that can be defined in IMS/VS data bases. Also in the figure are the statements required to define each type of relationship. Only the operands pertinent to the relationship are shown, and it is assumed that each type of relationship is defined between segments in two data bases named DBD1 and DBD2.

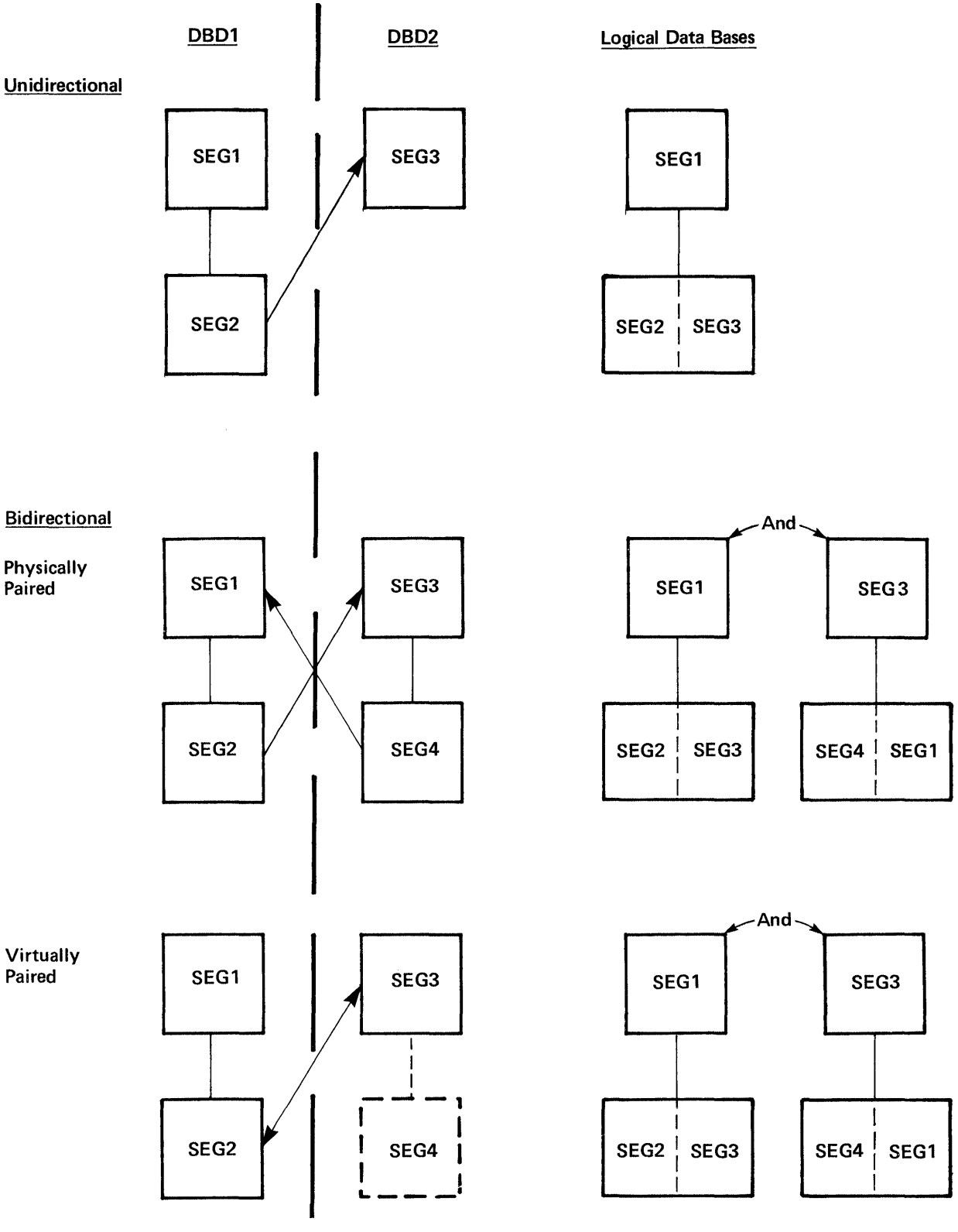
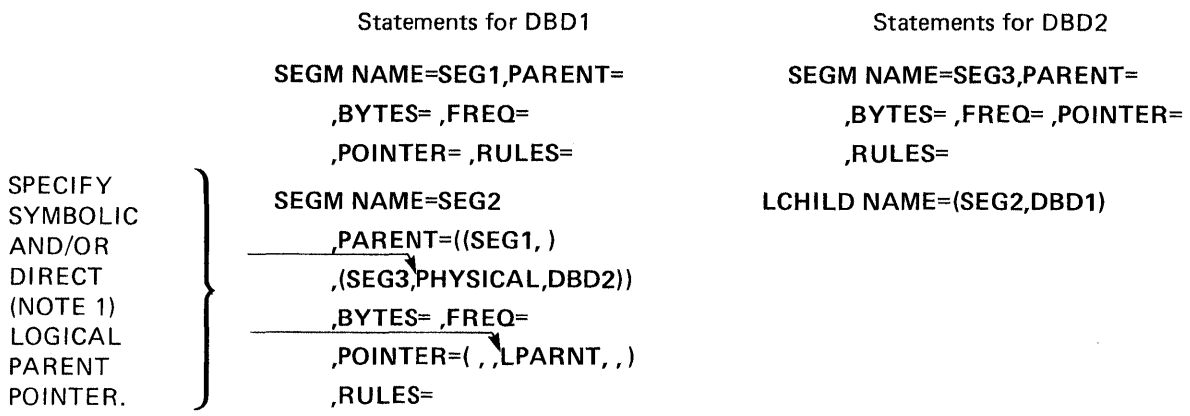


Figure 29 (Part 1 of 3). Summary of Logical Relationships

Unidirectional Logical Relationship



Physically Paired Bidirectional Logical Relationship

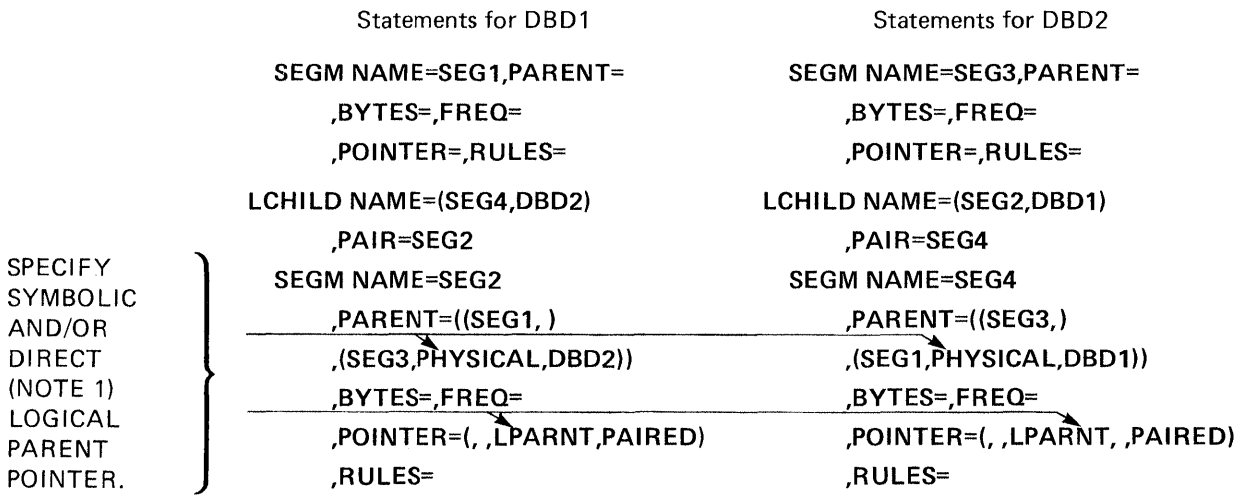
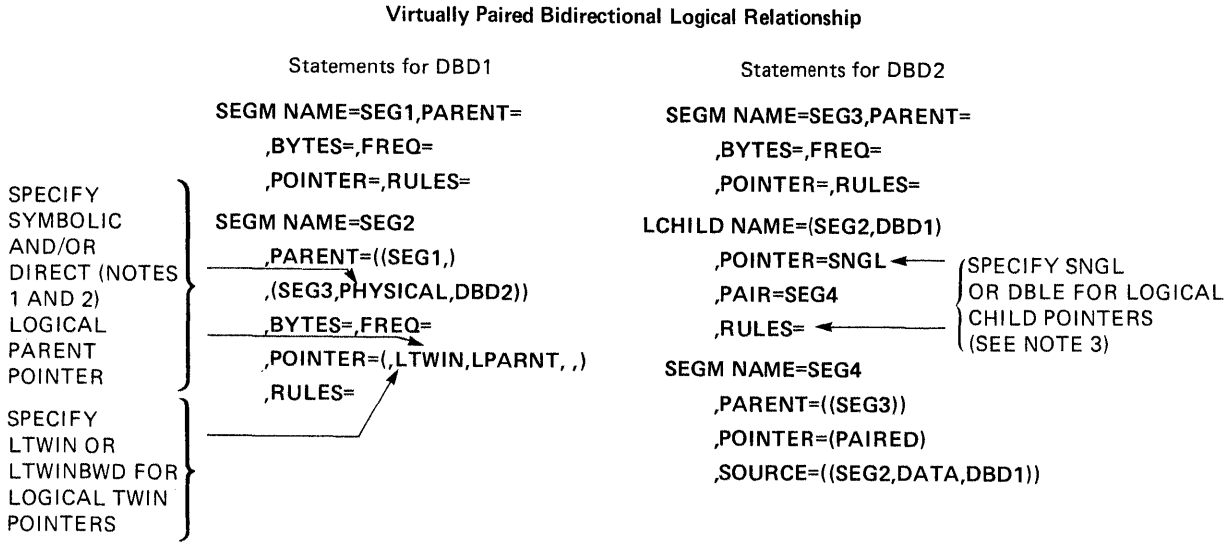


Figure 29 (Part 2 of 3). Summary of Logical Relationships



Notes:

1. The direct address pointer can be specified only when the logical parent is in an HDAM or HIDAM data base.
2. A HISAM data base can participate in a virtually paired logical relationship only when the real logical child is in an HDAM or HIDAM data base and its logical parent is in the HISAM data base.
3. The LCHILD RULES= operand is used when either no sequence field or a nonunique sequence field has been defined for the virtual logical child or when the virtual logical child segment does not exist.

Figure 29 (Part 3 of 3). Summary of Logical Relationships

Figure 30 illustrates how logical relationships and logical data bases are defined. Part 1 depicts the physical data structures, Part 2, the logical relationship between the physical data structures, and Part 3, the logical data bases that can be defined as a result of the logical relationships. Examples of DBD generation statements follow Figure 30.

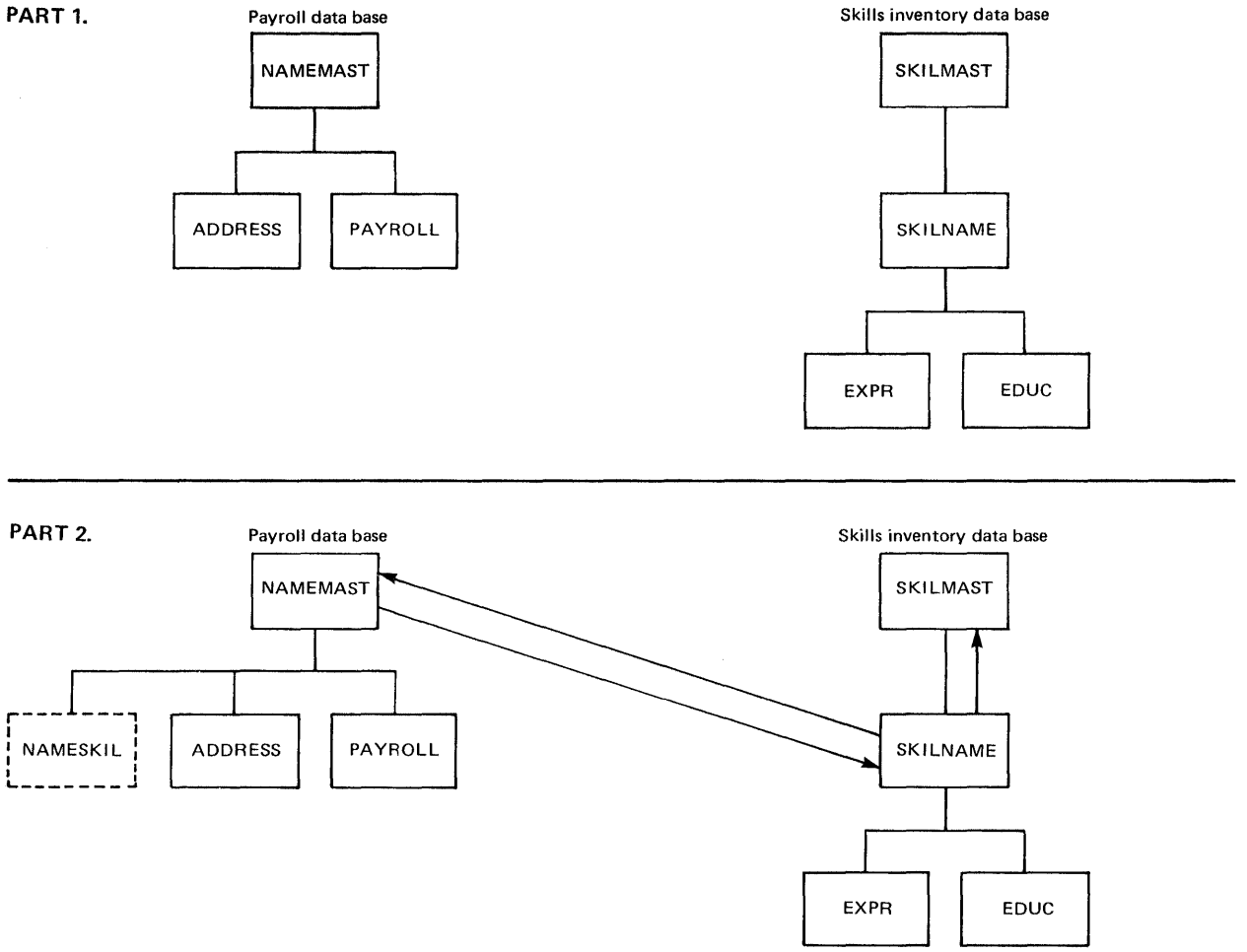


Figure 30. Logical Relationship between Physical Data Bases and the Resulting Logical Data Bases That Can Be Defined

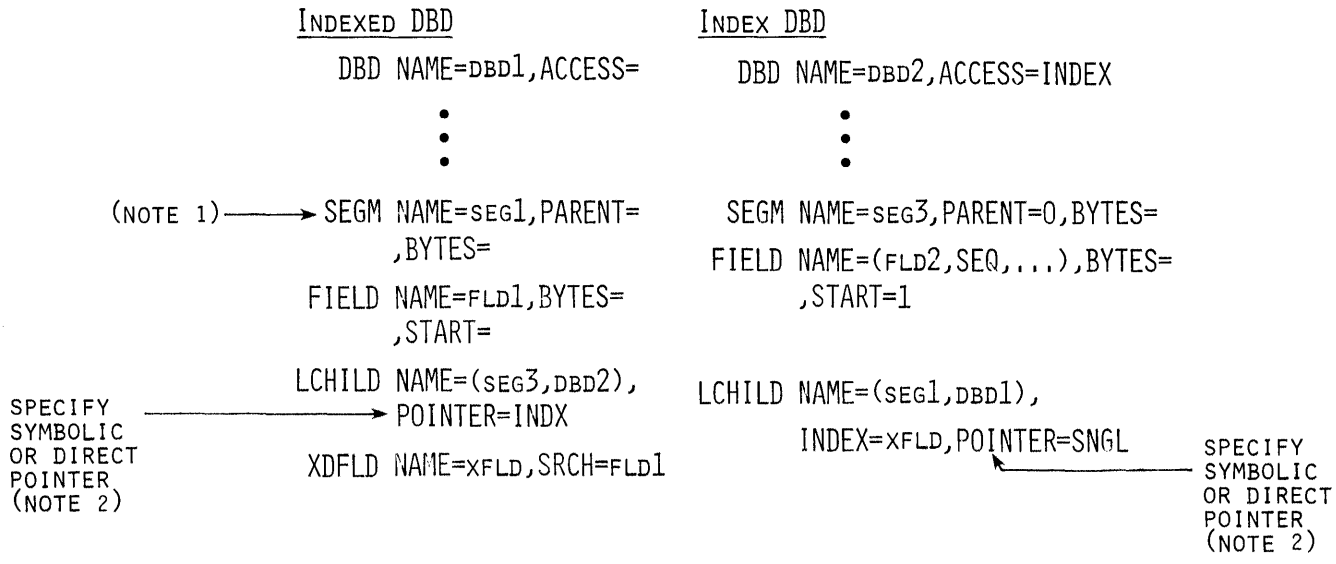
The following example shows the DBD generation statements necessary to define: 1) the payroll and skills inventory data structures depicted in Part 2 of Figure 30 as a HIDAM and HDAM data base with a virtually paired bidirectional logical relationship between the two data bases, and 2) the logical data structures depicted in Part 3 of Figure 30 as logical data bases.

```
DBD     NAME=PAYROLDB,ACCESS=HIDAM
DATASET DD1=PAYHIDAM,DEVICE=3350,BLOCK=1648,SCAN=3

SEGM    NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV),                    X
        BYTES=150
LCHILD  NAME=(INDEX,INDEXDB),PTR=INDX
LCHILD   NAME=(SKILNAME,SKILLINV),PAIR=NAMESKIL,PTR=DBLE
FIELD NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1,TYPE=C
FIELD NAME=MANNBR,BYTES=15,START=61,TYPE=C
FIELD NAME=ADDR,BYTES=75,START=76,TYPE=C

SEGM NAME=NAMESKIL,PARENT=NAMEMAST,PTR=PAIRED,                    X
      SOURCE=((SKILNAME,DATA,SKILLINV))
FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD NAME=STDLEVL,BYTES=20,START=22,TYPE=C

SEGM    NAME=ADDRESS,BYTES=200,PARENT=NAMEMAST
FIELD NAME=(HOMEADDR,SEQ,U),BYTES=100,START=1,TYPE=C
FIELD NAME=COMAILOC,BYTES=100,START=101,TYPE=C

SEGM    NAME=PAYROLL,BYTES=100,PARENT=NAMEMAST
FIELD NAME=(BASICPAY,SEQ,U),BYTES=15,START=1,TYPE=P
FIELD NAME=HOURS,BYTES=15,START=51,TYPE=P

DBDGEN
FINISH
END

DBD     NAME=SKILLINV,ACCESS=HDAM,RMNAME=(RAMDMODL,1,500,824)
DATASET     DD1=SKILHDAM,DEVICE=3350,BLOCK=1648,SCAN=5

SEGM  NAME=SKILMAST,BYTES=31,PTR=TWINBWD
FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEGM  NAME=SKILNAME,                                              X
      PARENT=((SKILMAST,DBLE),(NAMEMAST,P,PAYROLDB)),             X
      BYTES=80,PTR=(LPARNT,LTWINBWD,TWINBWD),                     X
      RULES=(VVV)
FIELD NAME=(EMPLOYEE,SEQ,U),START=1,BYTES=60,TYPE=C
FIELD NAME=(STDLEVL),BYTES=20,START=61,TYPE=C

SEGM  NAME=EXPR,BYTES=20,PTR=T,                                   X
      PARENT=((SKILNAME,SNGL))
FIELD NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD NAME=CLASSIF,BYTES=10,START=11,TYPE=C

SEGM  NAME=EDUC,BYTES=75,PTR=T,                                   X
      PARENT=((SKILNAME,SNGL))
FIELD NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C
DBDGEN
FINISH
END
```

```
DBD    NAME=LOGICDB,ACCESS=LOGICAL
DATASET LOGICAL

SEGM   NAME=SKILL,SOURCE=((SKILMAST,,SKILLINV))

SEGM NAME=NAME,PARENT=SKILL,                                          X
       SOURCE=((SKILNAME,,SKILLINV),(NAMEMAST,,PAYROLDB))

SEGM NAME=ADDRESS,PARENT=NAME,SOURCE=((ADDRESS,,PAYROLDB))
SEGM NAME=PAYROLL,PARENT=NAME,SOURCE=((PAYROLL,,PAYROLDB))
SEGM NAME=EXPR,PARENT=NAME,SOURCE=((EXPR,,SKILLINV))
SEGM NAME=EDUC,PARENT=NAME,SOURCE=((EDUC,,SKILLINV))

DBDGEN
FINISH
END

DBD    NAME=LOGIC1,ACCESS=LOGICAL
DATASET   LOGICAL

SEGM  NAME=NAME,SOURCE=((NAMEMAST,,PAYROLDB))

SEGM NAME=ADDRESS,PARENT=NAME,SOURCE=((ADDRESS,,PAYROLDB))
SEGM NAME=PAYROLL,PARENT=NAME,SOURCE=((PAYROLL,,PAYROLDB))

SEGM   NAME=SKILL,PARENT=NAME,                                        X
       SOURCE=((NAMESKIL,,PAYROLDB),(SKILMAST,,SKILLINV))
SEGM   NAME=EXPR,SOURCE=((EXPR,,SKILLINV)),PARENT=SKILL
SEGM   NAME=EDUC,SOURCE=((EDUC,,SKILLINV)),PARENT=SKILL

DBDGEN
FINISH
END
```

EXAMPLES WITH SECONDARY INDEXES

Figure 31, Figure 32, and Figure 33 summarize the statements required to establish a secondary index relationship between a segment type in an indexed data base and a segment type in a secondary index data base. Figure 31 shows the statements required when the index target and index source segment types are the same. In Figure 32, the index target and index source segment types are different. Figure 33 shows the statements required for a shared secondary index DBD generation. In all three figures, only those operands pertinent to the secondary index relationships are shown.

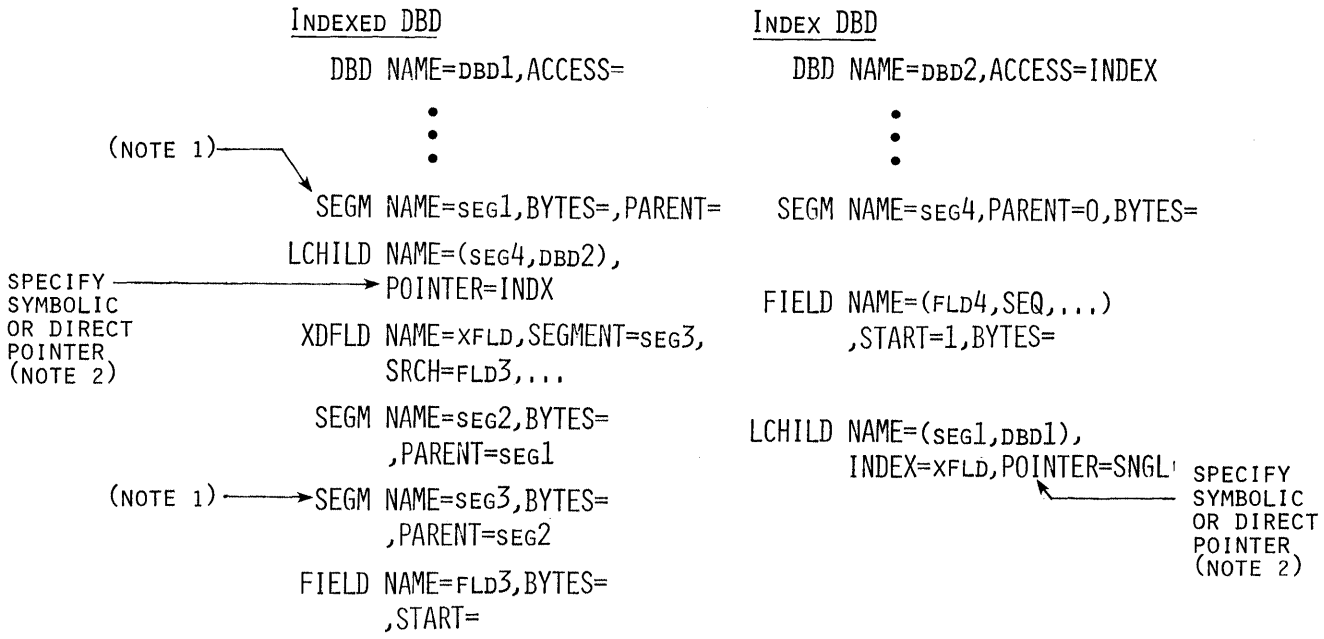


Notes:

1. The index target segment type may be a root or a dependent segment type; it must not be either a logical child segment type or a dependent of a logical child segment type. The index source segment type must not be a logical child segment type.

2. The example is shown with direct pointers for the index pointer segment types in the index DBD. If symbolic pointing is desired, POINTER=SYMB should be specified on both LCHILD statements; symbolic pointing is required when the index target segment type is in a HISAM data base.

Figure 31. Same Index Source and Target Segment Types



Notes:

1. The index target segment type may be a root or a dependent segment type, it must not be either a logical child segment type or a dependent of a logical child segment type. The index source segment type must not be a logical child segment type.

2. The example is shown with direct pointers for the index pointer segment types in the index DBD. If symbolic pointing is desired, POINTER=SYMB should be specified on both LCHILD statements; symbolic pointing is required when the index target segment type is in a HISAM data base.

Figure 32. Different Index Source and Target Segment Types

INDEXED DBD

```
   DBD NAME=DBD1,ACCESS=
         .
         .
         .
  SEGM NAME=SEG1,BYTES=,PARENT=
 FIELD NAME=FLD1,BYTES=
      ,START=
 FIELD NAME=FLD2,BYTES=
       ,START=
LCHILD NAME=(SEG3,DBD2),
       POINTER=INDX
 XDFLD NAME=XFLD1,SRCH=FLD2,
       CONST=C'2',...
         .
         .
         .
  SEGM NAME=SEG2,BYTES=,PARENT=
 FIELD NAME=FLD4,BYTES=
       ,START=
LCHILD NAME=(SEG5,DBD3),
       POINTER=INDX
XDFLD NAME=XFLD2,SRCH=FLD4,
       CONST=C'1',...
```

INDEX DBD

```
   DBD NAME=(DBD2,DBD3),ACCESS=INDEX
         .
         .
         .
  SEGM NAME=SEG3,PARENT=0,BYTES=1
 FIELD NAME=(FLD3,SEQ,...),
       START=1,BYTES=
LCHILD NAME=(SEG1,DBD1),
       INDEX=XFLD1


  SEGM NAME=SEG5,PARENT=0,BYTES=
 FIELD NAME=(FLD10,SEQ,...),
       START=1,BYTES=
LCHILD NAME=(SEG2,DBD1),
       INDEX=XFLD2
```

Note: This example is shown with direct pointers for the index pointer segment types, and with the index source segment type, and the index target segment type the same. Symbolic pointing or differing index source and target segments types may be used; however, all secondary index data bases in the shared index must uniformly specify either symbolic pointers or direct pointers; a mixture of symbolic and direct pointing is not allowed in a shared secondary index data base.

Figure 33. Shared Secondary Index Data Base DBD Generation

DBDGEN FOR SECONDARY INDEX DATA BASES

Figure 34 shows a data base indexed by two secondary indexes. The first secondary index, X1, uses the same segment for its index target segment and index source segment; the second secondary index, X2, has an index target segment that is different from its index source segment type.

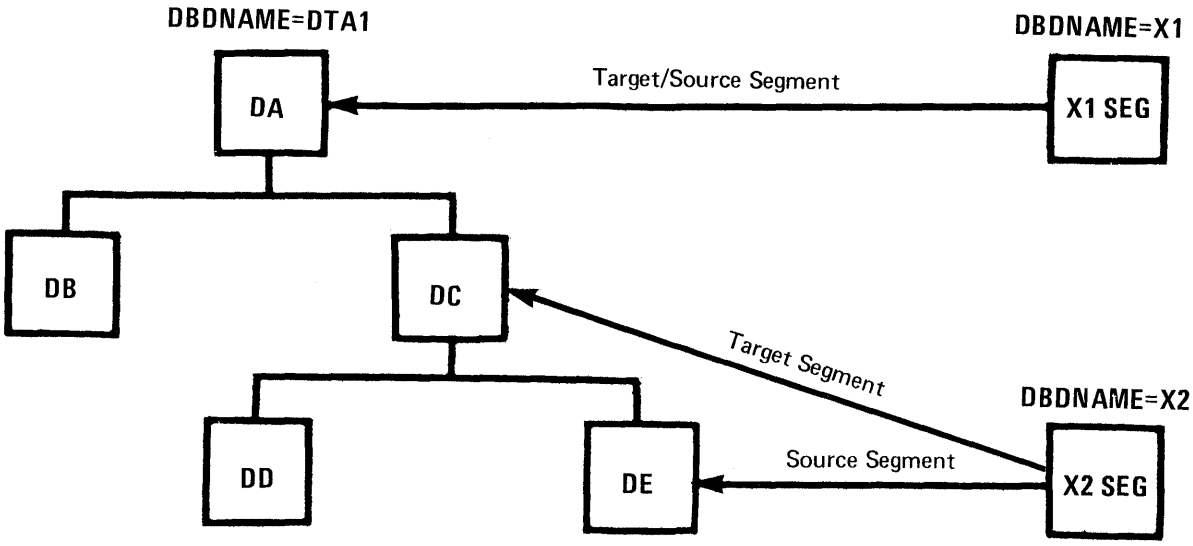


Figure 34. Data Base Indexed by Two Secondary Indexes

Figure 35 shows the DBD generation statements that define the indexed data base and the secondary index data bases.

DBDGEN for Indexed Data Base

```
DBD     NAME=DTA1,ACCESS=HDAM,RMNAME=(RANDMODL,1,500,824)
DATASET DD1=D1,DEVICE=3330,MODEL=1
SEGM    NAME=DA,PARENT=0,BYTES=15
FIELD   NAME=(DAF1,SEQ),BYTES=5,START=1
LCHILD  NAME=(X1SEG,X1),PTR=INDX
XDFLD   NAME=DAF1X,SRCH=DAF1
SEGM    NAME=DB,PARENT=DA,BYTES=20
FIELD   NAME=(DBF1,SEQ),BYTES=5,START=1
SEGM    NAME=DC,PARENT=DA,BYTES=20
FIELD   NAME=(DCF1,SEQ),BYTES=5,START=1
LCHILD  NAME=(X2SEG,X2),PTR=SYMB
XDFLD   NAME=DCF1X,SRCH=DEF1,SEGMENT=DE
SEGM    NAME=DD,PARENT=DC,BYTES=25
FIELD   NAME=(DDF1,SEQ),BYTES=5,START=1
SEGM    NAME=DE,PARENT=DC,BYTES=25
FIELD   NAME=(DEF1,SEQ),BYTES=5,START=1
DBDGEN
FINISH
END
```

DBDGEN for Secondary Index X1

```
DBD     NAME=X1,ACCESS=INDEX
DATASET DD1=X1P,DEVICE=3330,MODEL=1
SEGM    NAME=X1SEG,BYTES=5,PARENT=0
FIELD   NAME=(X1F1,SEG,U),START=1,BYTES=5
LCHILD  NAME=(DA,DTA1),INDEX=DAF1X,POINTER=SNGL
DBDGEN
FINISH
END
```

DBDGEN for Secondary Index X2

```
DBD     NAME=X2,ACCESS=INDEX
DATASET DD1=X2P,DEVICE=3330,MODEL=1
SEGM    NAME=X2SEG,BYTES=5,PARENT=0
FIELD   NAME=(X2F1,SEQ,U),START=1,BYTES=5
LCHILD  NAME=(DC,DTA1),INDEX=DCF1X,POINTER=SYMB
DBDGEN
FINISH
END
```

Figure 35. Indexed Data Base and Secondary Index Data Base

DBDGEN FOR A SHARED SECONDARY INDEX DATA BASE

Figure 36 shows a data base indexed by three secondary indexes in a shared secondary index data base. Each secondary index uses the same segment as the index target segment and the index source segment. Figure 37 on page 93 shows the DBD generation statements that define the indexed data base, the primary index data base, and the shared secondary index data base.

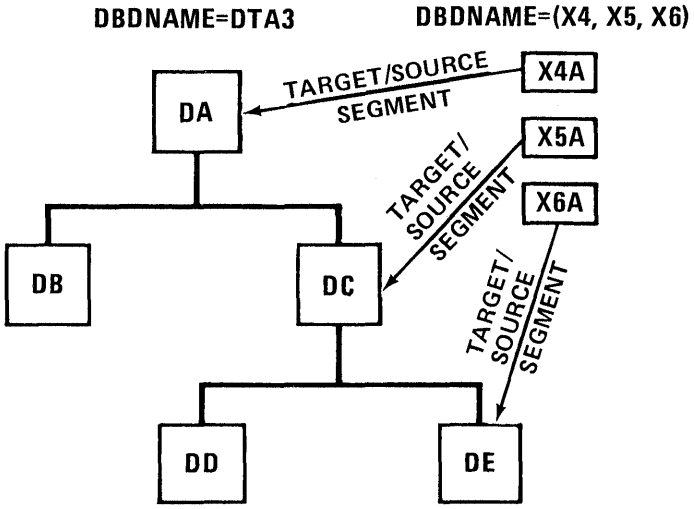


Figure 36. Data Base Indexed by Three Secondary Indexes in a Shared Secondary Index Data Base

DBDGEN for Indexed Data Base

```
DBD     NAME=DTA3,ACCESS=HIDAM
DATASET DD1=D1,DEVICE=3350
SEGM  NAME=DA,PARENT=0,BYTES=15
LCHILD NAME=(INDEX,X2),PTR=INDX
FIELD NAME=(DAF1,SEQ),BYTES=5,START=1
LCHILD NAME=(X4A,X4),PTR=INDX
XDFLD NAME=DAF1X,SRCH=DAF1,CONST=C'1'
SEGM  NAME=DB,PARENT=DA,BYTES=20
FIELD NAME=(DBF1,SEQ),BYTES=5,START=1
SEGM  NAME=DC,PARENT=DA,BYTES=20
FIELD NAME=(DCF1,SEQ),BYTES=5,START=1
LCHILD NAME=(X5A,X5),PTR=INDX
XDFLD NAME=DCF1X,SRCH=DCF1,CONST=C'2'
SEGM  NAME=DD,PARENT=DC,BYTES=25
FIELD NAME=(DDF1,SEQ),BYTES=5,START=1
SEGM  NAME=DE,PARENT=DC,BYTES=25
FIELD NAME=(DEF1,SEQ),BYTES=5,START=1
LCHILD NAME=(X6A,X6),PTR=INDX
XDFLD NAME=DEF1X,SRCH=DEF1,CONST=C'3'
DBDGEN
FINISH
END
```

DBDGEN for Primary Index Data Base

```
DBD   NAME=X2,ACCESS=INDEX
DATASET DD1=X2P,DEVICE=3350
SEGM  NAME=INDEX,BYTES=5
LCHILD NAME=(DA,DTA3),INDEX=DAF1
FIELD NAME=(INDXSEQ,SEQ,U),BYTES=5,START=1
DBDGEN
FINISH
END
```

DBDGEN for Shared Secondary Index Data Base

```
DBD   NAME=(X4,X5,X6),ACCESS=INDEX
DATASET DD1=X4P,DEVICE=3350,OVFLW=X40
SEGM  NAME=X4A,BYTES=6,PARENT=0
FIELD NAME=(X4F1,SEQ,U),START=1,BYTES=6
LCHILD NAME=(DA,DTA3),INDEX=DAF1X
SEGM  NAME=X5A,BYTES=6,PARENT=0
FIELD NAME=(X5F1,SEQ,M),START=1,BYTES=6
LCHILD NAME=(DC,DTA3),INDEX=DCF1X
SEGM  NAME=X6A,BYTES=6,PARENT=0
FIELD NAME=(X6F1,SEQ,M),START=1,BYTES=6
LCHILD NAME=(DE,DTA3),INDEX=DEF1X
DBDGEN
FINISH
END
```

Figure 37. Indexed Data Base, Primary Index Data Base, and Shared Secondary Index Data Base DBD Generations

CHAPTER 2. PROGRAM SPECIFICATION BLOCK (PSB) GENERATION

OVERVIEW

Before an application program can be executed under IMS/VS, it is necessary to describe that program and its use of logical terminals and logical data structures through a PSB generation. The PSB generation statements supply the identification and characteristics of the IMS/VS resources to be used. These program communication blocks (PCBs) represent message destinations and data bases to be used by the application program. In addition, there must be a statement supplying characteristics of the application program itself. There must be one PSB for each message, batch, or Fast Path program. The name of the PSB and its associated application program must be the same in a telecommunications system.

PSB generation places the created PSB in the PSB library. Each PSB is a member of the operating system partitioned data set IMSVS.PSBLIB. For IMS/VS batch execution (DLI region type), the necessary PSB is loaded from PSBLIB and the expanded PSB needed for DL/I processing is built from it. Before online execution, ACBGEN must be performed to prebuild the expanded PSBs into ACBLIB. PSBLIB is used as input to the ACBGEN process. Batch executions may also use prebuilt blocks from ACBLIB by specifying region type 'DBB' on the JCL execute statement.

PSB RULES

There are six basic types of statements used for a PSB generation:

- PCB statements for output message destinations other than the source of the input message. These are called alternate PCBs, and they are used in message processing, batch message processing, and Fast Path programs which interface with the IMS/VS message queues.
- PCB statements for DL/I and Fast Path data bases. These are used by message, batch, and Fast Path processing programs to define interfaces to a data base.
- SENSEG statements for segments within a data base to which the application program is sensitive. These are used with message, batch, and Fast Path processing programs to define logical data structures.
- SENFLD statements for fields within a segment to which the application program is sensitive.
- PSBGEN statement for each PSB. This statement is used to indicate the characteristics of the associated application program.
- An assembler language END statement is required for each PSBGEN statement.

Note that the above list does not include a PCB for the input message source. I/O PCBs exist within the IMS/VS online control program nucleus for this purpose. Upon entry to the application program used for message processing, a PCB pointer to the source of the input message is provided as the first entry in a list of PCB address pointers. The remainder of the PCB list has a direct relationship to the PCBs as defined within the associated PSB and must be defined in the application program in the same

order as was defined during PSB generation. All PCBs may be used by the application program when making DL/I message and data base calls. Only one PCB is used in a particular DL/I call.

In the case of a batch program, there is no I/O PCB unless requested with the CMPAT option on the PSBGEN statement. Therefore, if CMPAT=YES is not specified, the PCB list provided to the program has a direct relationship to the PCBs within the PSB. No TP PCBs should be contained in a PSB for batch processing in a batch processing region.

Alternate PCBs may be specified in a PSB associated with a batch program operative in an IMS/VS batch message processing region. These PCBs are available for output message queuing. A batch program operative in batch message processing regions may access messages from the input message queue. An I/O PCB is always provided as in the case of a message processing program.

Alternate and modifiable alternate PCBs can be specified in a PSB associated with a Fast Path program executing in a Fast Path region. A response alternate PCB with the same PTERM can be used to send a Fast Path output message back to the original PTERM with a different component attached to the terminal. An alternate PCB with no alternate response specification can be used to send an output message to any terminal or IMS/VS message queue.

To test message processing or batch message processing programs in a batch processing region, the CMPAT option of the PSBGEN statement should be used. When CMPAT=YES is specified, IMS/VS provides PCBs to the application as if it were executing in a message processing region. Using CMPAT eliminates the need to recompile the program between batch and online executions.

The PCB list passed to the application program upon entry should be referenced within the application program by the names defined within the application program for making DL/I calls and interrogating PCB information (that is, status codes and feedback information). The address of a PCB is normally the second parameter in a DL/I call from an application program to IMS/VS. The PCB address may represent the source of the input message, the destination for an output message, or a data base. Upon completion of a DL/I call, the PCB contains status and feedback information pertinent to the call. For greater detail, refer to "DB PCB Masks" in IMS/VS Application Programming.

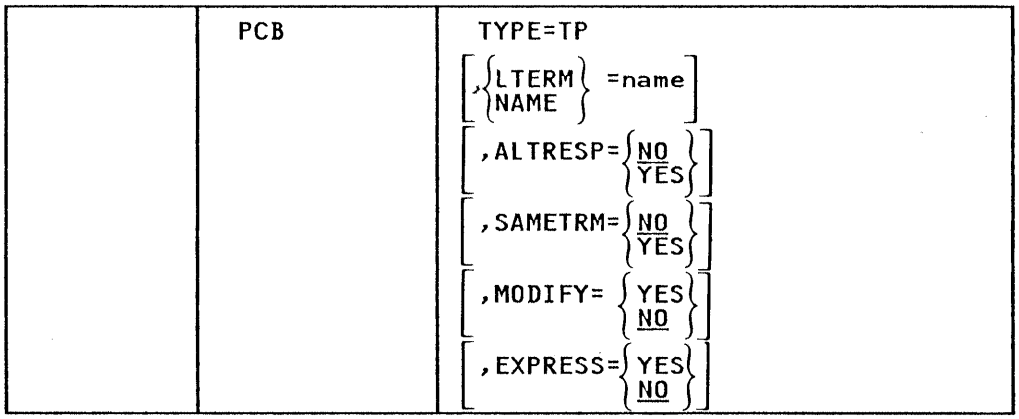
PSB CONTROL STATEMENT FORMAT

To reiterate, no PCB statement is needed in PSB generation for the I/O PCB. IMS/VS builds it automatically. This is true for message processing application programs, batch processing application programs that operate in IMS/VS batch message processing regions and wish to obtain input messages from the IMS/VS message queues, and Fast Path application programs that operate in an IMS/VS Fast Path dependent region. Batch processing application programs that operate in IMS/VS batch processing regions never have an I/O PCB, unless specifically requested in the PSBGEN macro statement.

ALTERNATE PCB STATEMENT

The alternate PCB describes a destination other than the source of the current input message. This statement allows the application program to send output messages to a destination other than the source of an input message. A PCB statement is required for each destination to which output is to be sent. These messages may be sent to either an output terminal or an input transaction queue to be processed by another program. There must be a separate alternate PCB for every output message destination. If the input source terminal is all that is required to respond with output, do not include any PCB statements of this type. Message processing programs, batch message processing programs, and Fast Path programs may have alternate PCB statements in their associated PSBs. An alternate PCB cannot be used to send a message to a Fast Path transaction; however, Fast Path application programs can use an alternate PCB to route messages to any terminal or IMS/VS transaction.

Alternate PCB statements must be first in the PSB generation control card deck, followed by the statements identifying PCBs associated with DL/I data bases.



where:

TYPE=TP
Is a required keyword parameter for all alternate PCBs.

LTERM= or NAME=
Is the parameter keyword for the output message destination. The "name" is the actual destination of the message and is either a logical terminal name (LTERM=) or a transaction-code name (NAME=). When the name is a transaction-code name, output messages to this PCB are enqueued for input to the program used to process the transaction code named by the NAME operand. The name must be from 1- to 8-alphameric characters in length, and must be specified in the user's IMS/VS system definition as a logical terminal name or transaction code. The LTERM= or NAME= operand is required except when MODIFY=YES is specified.

ALTRESP=
Specifies whether (YES) this alternate PCB can be used instead of the I/O PCB for responding to terminals in response mode, conversational mode, or exclusive mode. The default value is NO. ALTRESP=YES is only valid for alternate TP PCBs.

SAMETRM=
Specifies whether (YES) IMS/VS should verify that the logical terminal named in the response alternate PCB is assigned to the same physical terminal as the logical terminal that originated the input message. The default value is NO. SAMETRM=YES must be specified for response alternate PCBs used by conversational programs and programs operating with terminals in response mode. SAMETRM=NO should be specified if alternate response PCBs are used to send messages to output-only devices that are in exclusive mode.

MODIFY=
Specifies whether the alternate PCB is modifiable (YES). This feature allows for the dynamic modification of the destination name associated with this PCB. Default value is NO. If this operand is specified as YES, the NAME= or LTERM= operand must be omitted.

EXPRESS=
Specifies whether messages from this alternate PCB are to be sent (YES) or are to be backed out (NO) if the application program should terminate abnormally.

YES
When YES is specified, EXPRESS messages are sent except when an IMS/VS control region abends or a deadlock condition occurs. Under these conditions, a message is pending in this PCB if at least one message insert for this PCB has been successfully completed, and no message Get Unique has been issued for a multiple mode transaction or no purge has been issued. In either case, the message will not be sent during an emergency restart or deadlock termination because the transaction that caused the message will be reprocessed.

If the PSB is defined as a Fast Path application in the IMS/VS system definition, EXPRESS=YES, if specified, will be ignored at execution time for a response alternate PCB.

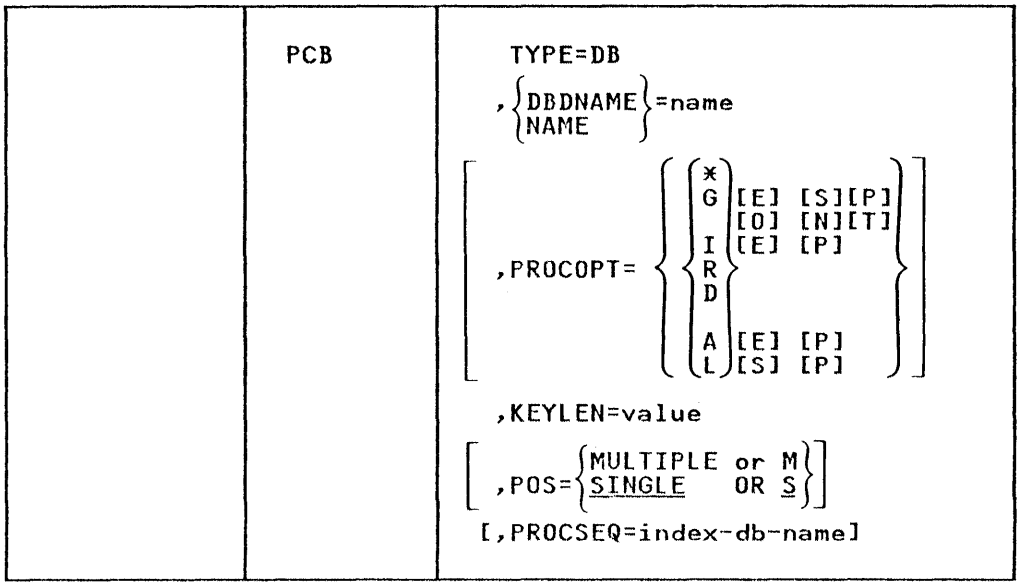
NO
When NO is specified, messages are backed out if the application program abnormally terminates. NO is the default.

DL/I DATA BASE PCB STATEMENT

The second type of statement in a PSB generation deck is one that specifies a description of a PCB for a DL/I data base. Although one or more data base PCBs are usually included in a PSB, this second type of statement may not be required. For example, a message switching program or conversational message program may not require access to a DL/I data base. Therefore, a data base PCB is not required.

The maximum number of data base PCBs that can be defined in a PSBGEN is 255 minus the number of alternate TP PCBs defined. This is the maximum value for application programs executing in all IMS region types (MSG, DL/I, etc.).

The format for the DL/I data base PCB statement is:



* These can be any combination; if all G, I, R, and D are chosen, then replace with A (A=GIRD).

where:

TYPE=DB
Is a required keyword parameter for all DL/I data base PCBs.

DBDNAME= or NAME=
Is the parameter keyword for the name which specifies the physical or logical DBD to be used as primary source of data base segments for this logical data structure. The logical structure, which is defined under this PCB with one or more SENSEG statements, is the hierarchical set of data segments to which the associated application program is sensitive. This logical hierarchy of data segments may or may not exist as a physical hierarchy. This will depend upon the relationship of segments defined by SENSEG statements and the existence of these segments in one or more data bases as defined by their data base descriptions (DBDs). All SENSEG statements which follow this PCB statement and precede the next PCB or PSBGEN statement must refer to segments defined in the DBD named in the DBDNAME= or NAME= operand of this PCB. (Refer to the SENSEG statement description later in this chapter.)

The keywords DBDNAME and NAME are synonymous. DBDNAME was added because it is more descriptive, and NAME was kept for compatibility with earlier releases.

PROCOPT=
Is the parameter keyword for the processing options on sensitive segments declared in this PCB that may be used in an associated application program. This operand allows for a maximum of four characters. The letters in the operand have the following meaning:

G—Get function.

I—Insert function.

R—Replace function. Includes G.

D—Delete function. Includes G.

A—All, includes the above four functions.

P—Required if command code D is to be used, except for ISRT calls in a batch program that is not sensitive to fields. PROCOPT=P is not required if command code D is used when processing DEDBs. Refer to the next section, "Use of PROCOPT= with Fast Path," for more information on how to use PROCOPT=P with DEDBs.

O—Read only; do not enqueue to check availability. Must be specified as GO, GON, GONP, GOT, GOTP, or GOP only.

N—Reduces the number of abnormal terminations read-only application programs are subject to. Read-only application programs can reference data being updated by another application program. When this happens, it's possible that an invalid pointer to the data exists. If an invalid pointer is detected, the read-only application program abnormally terminates. By specifying N, this type of abnormal termination is avoided and a GG status code is returned to the application program instead. When the application program receives a GG status code, it must decide whether to terminate processing or continue processing by accessing the data using a different path. N must be specified as GON or GONP.

T—Has the same purpose and works like the N operand, except that T causes DL/I to automatically retry the operation. If the retry fails, as with N, a GG status code is returned to the application program. T must be specified as GOT or GOTP.

E—Enables exclusive use of the data base or segment by online programs. To be used in conjunction with G, I, D, R, and A.

L—Load function for data base loading (except HIDAM).

GS—Get segments in ascending sequence only (HSAM only).

LS—Segments loaded in ascending sequence only (HIDAM, HDAM). This load option is required for HIDAM.

If GS is specified for HSAM data bases, they will be read using the Queued Sequential Access Method (QSAM) instead of the Basic Sequential Access Method (BSAM) in a DL/I IMS/VS region. Because LS is specified for HIDAM data bases, the index for the root segment sequence field will be created at the time the data base is loaded. If the PROCOPT operand is omitted, it will default to PROCOPT=A. The replace and delete functions also imply the Get function.

CAUTION: If the 'O' option (read-only) is used for a PCB, the data that is read should not be used as a basis for updating records in any data base. With this option, IMS/VS does not check the ownership of the segments returned; this means that the read-only user might get a segment that had been updated by another user. If the updating user should then abend and be backed out, the read-only user would have a segment that did not (and never did) exist in the data base. Therefore, the 'O' option user should not update based on data read with that option.

A user ABEND from the retrieve module (U8xx) can occur with PROCOPT=GO if another program updates pointers when this program is following the pointers. Pointers are updated during the insert and delete functions and during replacement of a variable-length segment. To reduce the number of abnormal terminations of this type, you can code the PROCOPT= operand with an N or a T.

Notes:

1. If any PCBs in the PSB have a PROCOPT of L or LS and either explicitly reference HISAM or HIDAM data bases, or implicitly reference INDEX data bases, then no other PCB in the same PSB may reference any of the above data bases, either explicitly or implicitly, with a PROCOPT other than L or LS. The SENSEG statements within that PCB should not contain INDICES= operands.

2. Edit compression routines should be designed to ensure that a segment will not be decompressed beyond its maximum length. Expansion of a segment beyond the maximum length may overlay IMS control blocks, resulting in an ABEND0C4 (ABEND0CX) in the IMS control region. When using PROCOPT=GO, the segment data is not protected by program isolation and may change concurrently with segment decompression from the buffer by the edit compression routine. This should be taken into consideration when coding an edit compression routine to be used with PROCOPT=GO. For more information about using edit compression routines see the IMS/VS System Programming Reference Manual.

3. If L is specified for a PCB which references a data base with multiple data set groups, the PCB should include at least one SENSEG statement for each data set group in the data base.

4. If the 'O' option is used for a PCB, the SENSEG statement must not specify a PROCOPT of I, R, D, or A.

5. An online application program always has exclusive use of the SHSAM or HSAM data bases, which are referenced by PCBs in its PSB. No other application programs will be concurrently scheduled to access those same SHSAM or HSAM data bases in an online environment.

6. If the Online Data Base Image Copy utility refers to this PCB, the value of PROCOPT= L or LS is not allowed. If the data base to be copied is the index portion of a HIDAM data base, only PROCOPT=G and PROCOPT=GO are valid. If PROCOPT=E is specified, the Online Image Copy utility will execute with exclusive control of the data base, even though the utility does not require such control.

7. If the Data Base Surveyor utility feature refers to this PCB, PROCOPT=G must be specified.

8. In the case of concatenated segments, the PROCOPT= operand governs the logical child segment of the concatenated segment. The logical parent of the concatenated segment is governed by the RULES= operand of the SEGM statement.

USE OF PROCOPT= WITH FAST PATH: In a Fast Path nonterminal-related or fixed terminal-related MSDB, only the processing options G and R are valid.

G—Get function.

R—Replace function. Includes G.

In a Fast Path dynamic terminal-related MSDB, the processing options G, I, R, D, and A, or any combination of G, I, R, and D are valid.

G—Get function.

I—Insert function.

R—Replace function. Includes G.

D—Delete function. Includes G.

A—All. Includes above four functions.

In a Fast Path DEDB, the processing options G, I, R, D, A, P, N, T and O are valid.

G—Get function.

I—Insert function.

R—Replace function. Includes G.

D—Delete function. Includes G.

A—All, includes above four functions.

P—Position function.

O—Read only; do not enqueue to check availability. Must be specified as GO, GON, or GOT. Selecting PROCOPT=GO|GON|GOT for DEDBs at PSB generation time indicates that read without integrity is in effect. As for DL/I data bases, no locking mechanism is used to maintain the integrity of the DEDB data being retrieved. Unlike DL/I data bases, PROCOPT=GO|GON|GOT is effective only if the IMS/VS host subsystem has an access intent of RO (read only) against the DEDB. With other ACCESS values the processing intent attribute is dynamically modified at scheduling time and set as if PROCOPT=G had been specified.

N—Reduces the number of abnormal terminations read-only application programs are subject to. Read-only application programs can reference data being updated by another application program. When this happens, it's possible that an invalid pointer to the data exists. If an invalid pointer is detected, the read-only application program abnormally terminates. By specifying N, this type of abnormal termination is avoided and a GG status code is returned to the application program instead. When the application program receives a GG status code, it must decide whether to terminate processing or continue processing by accessing the data using a different path. N must be specified as GON or GONP.

T—Has the same purpose and works like the N operand, except that T causes DL/I to automatically retry the operation. If the retry fails, as with N, a GG status code is returned to the application program. T must be specified as GOT or GOTP.

A DLET or ISRT call to a terminal-related dynamic MSDB from a program with no input LTERM present, for example, a batch-oriented BMP, will result in a status code of 'AM', regardless of the processing option specification.

The replace function also implies the get function. If the referenced segment is a root or direct dependent segment, A implies G, I, R, and D. Only processing options of G, I, and GI are valid for sequential dependent segments.

The processing option of P is valid only when specified for a root segment to be used by an IMS/VS online batch message application program. With this option, a 'GC' status code is returned when a UOW boundary is crossed during a G(H)U, G(H)N, or ISRT on a root segment. Also, data base positioning is maintained across a valid SYNC call (blank status code returned), when the sync is issued immediately after receiving a 'GC' status code. In the case of a sync process failure, position is set to the last valid sync point or, if no valid sync point exists, to the start of the data base. A SYNC call without a preceding 'GC' status will also cause position to be set to the start of the data base. For more information on the P processing option or the UOW for DEDBs, see IMS/VS Data Base Administration Guide.

If the D command code is used in a call to a DEDB, the P processing option need not be specified in the PCB for the program; the P processing option has a different meaning for DEDBs than for other data bases.

Note: PSBGEN accepts invalid processing options if specified, but the Application Program Control Blocks Maintenance utility fails. The error does not appear in the PSBGEN but appears in the ACBGEN.

KEYLEN=

Is the value specified in bytes of the longest concatenated key for a hierarchic path of sensitive segments used by the application program in the logical data structure. Figure 38 explains the definition of KEYLEN.

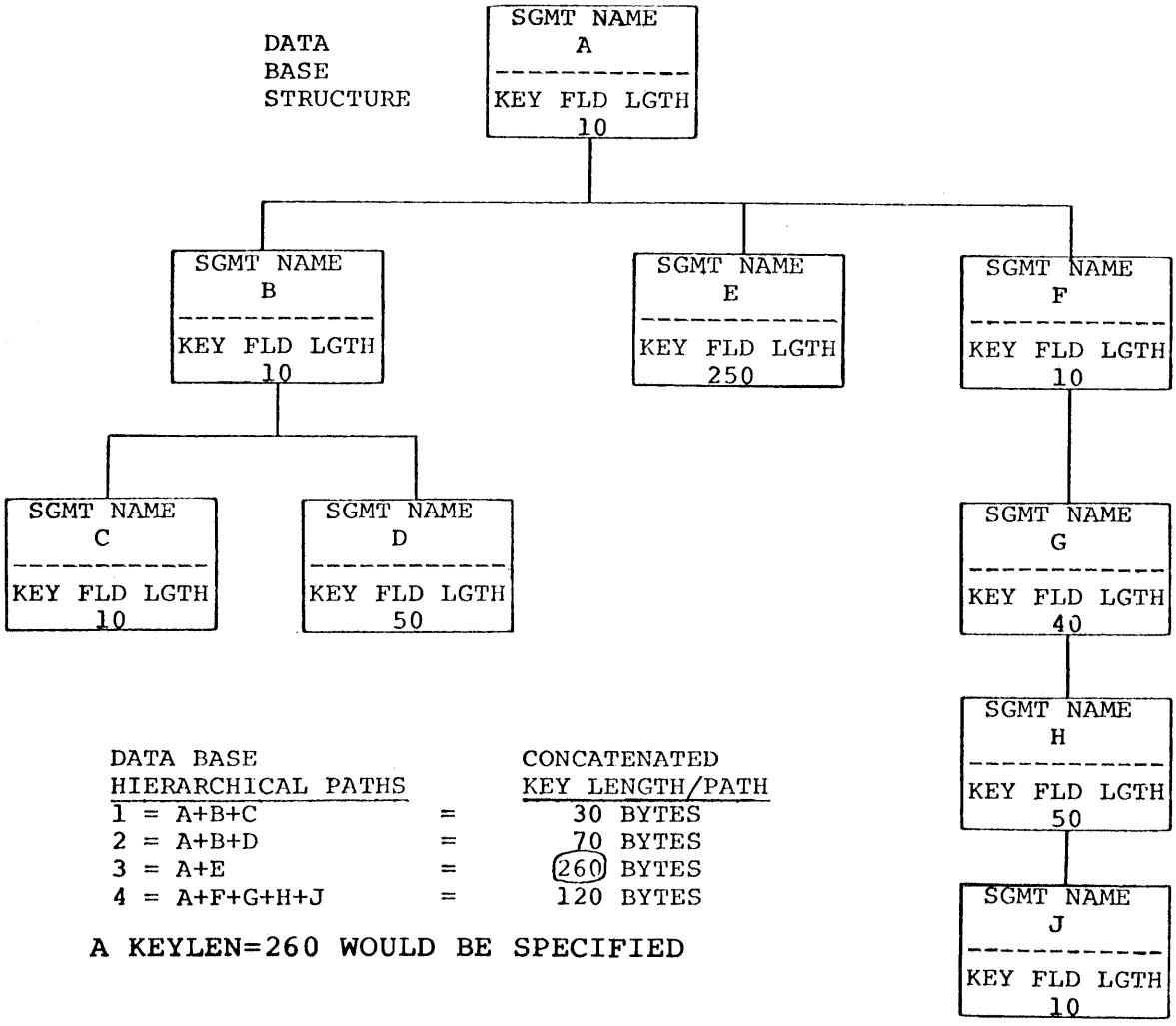


Figure 38. KEYLEN Definition

For a nonterminal-related MSDB without terminal-related keys, the value has to be greater than or equal to the value of the BYTES parameter of the sequence field in the DBD generation and be within the range of 1 through 240 bytes.

For a terminal-related MSDB (using the LTERM name as a key), this value must be 8.

POS=

Specifies whether single or multiple positioning is desired for the logical data structure. Single or multiple positioning provides a functional variation in the call. This functional difference is documented in "Using Multiple Positioning" in IMS/VS Application Programming. The performance variation between single and multiple positioning is insignificant. Multiple positioning is not supported by HSAM.

POS=S or SINGLE is the default, except for DEDBs having more than two dependent segments, in which case POS=M or MULTIPLE is used.

Note: Coding a POS value on the PCB statement for a DEDB will not override the default that is selected based on the number of dependent segments.

PROCSEQ=

Specifies the name of a secondary index that is used to process the data base named in the DBDNAME operand through a secondary processing sequence. The operand is optional. It is valid only if a secondary index exists for this data base. If this operand is used, subsequent SENSEG statements have to reflect the secondary processing sequence hierarchy of segment types in the indexed data base. Thus, for example, the first SENSEG statement must name the indexed segment with a PARENT=0 operand.

index-db-name must be the name of a secondary index DBD. Note that, when using a secondary processing sequence, processing options 'L' and 'LS' are invalid. Inserting and deleting the index target segment and any of its inverted parents are not allowed. When the blocks are built, if the processing option for these segments includes 'I' or 'D', a warning message indicates that the processing option has been changed to reflect this restriction.

GSAM PCB Statement

The format for the GSAM data base PCB statement is:

```
|        | PCB | TYPE=GSAM                                  |
|        |     | ,{DBDNAME}=name                            |
|        |     |  {NAME   }                                 |
|        |     | , PROCOPT={G[S]}                           |
|        |     |           {L[S]}                           |
```

where:

TYPE=GSAM

Is a required keyword parameter for all GSAM data base PCBs that will be allocated and processed in the dependent region.

DBDNAME= or NAME=

Is a required keyword parameter for the name that specifies the GSAM DBD to be used as the primary source of data set description. SENSEG statements must not follow this PCB statement.

PROCOPT=

Is a required parameter for the processing options on the data set declared in this PCB that can be used in an associated application program. The operand is specified using the following characters:

G—Get function

L—Load function
S—Large-scale sequential activity. Use GSAM multiple-buffering option (BUFFIO).

The PCB statement must follow the PCB statements with TYPE=TP or DB if any exist in the PSB generation. The rule is:

TP PCBs—first
DB PCBs—second
GSAM PCBs—last

SENSEG STATEMENT

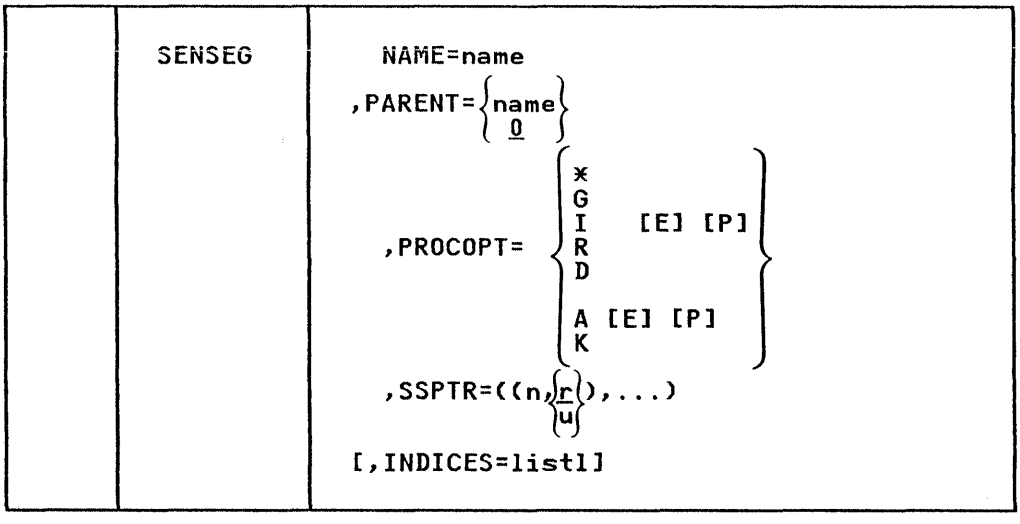
The SENSEG statement is used in conjunction with the data base PCB statement for defining a hierarchically related set of data segments. This set represents segments to which a program through this PCB is sensitive. This segment set may physically exist in one data base or may be derived from several physical data bases. One or more SENSEG statements can be included; each statement must immediately follow the PCB statement to which it is related. There must be one SENSEG statement for each segment to which the application program is sensitive. All segments in the hierarchic path to any required segment must be specified. A maximum of 1000 SENSEG statements can be defined in a single PSB generation.

The order in which SENSEG statements are sequenced after a PCB statement determines the logical access order for the segments. The SENSEG statement sequence must follow the physical sequence of the segments as defined in DBDGEN, with the following exceptions:

1. The logical order of dependent segments whose parent segment does not use hierarchic pointing may differ from the physical sequence.

2. If the PROCSEQ parameter is used in the PCB statement, the SENSEG statement sequence reflects the secondary processing sequence specified by the PROCSEQ parameter.

The format of the SENSEG statement is shown in Figure 39.



* These can be any combination; if all four are chosen, then replace with A (A=GIRD).

Figure 39. SENSEG Statement Format

where:

NAME=
Is the name of the segment type as defined through a SEGM statement during DBD generation. The field is from 1 to 8 alphameric characters.

PARENT=
Is the segment type name of this segment's parent. This operand is required for all dependent segments. The field is from 1 to 8 alphameric characters or 0. If this SENSEG statement defines a root segment type as being sensitive, this operand must equal zero; if omitted, PARENT=0 is the default.

PROCOPT=
Indicates the processing options allowable for use of this sensitive segment by an associated application program. This operand has the same meaning as the PROCOPT= operand on the PCB statement. In addition to the valid options for this operand listed in Figure 39 on page 104, an option can be used on the SENSEG statement which does not apply to the PCB statement. A PROCOPT of K indicates key sensitivity only. A GN call with no SSAs can only access data-sensitive segments. If a key-sensitive segment is designated for retrieval in an SSA, the segment is not moved to the user's I/O area; but the key is placed at the appropriate offset in the key feedback area of the PCB. If this PROCOPT= operand is not specified, the PCB PROCOPT operand is used as default. If there is a difference in the processing options specified on the PCB and SENSEG statements and the options are compatible, SENSEG PROCOPT overrides the PCB PROCOPT. If PROCOPT= L or LS is specified on the preceding PCB statement, this operand must be omitted.

Do not specify a SENSEG statement for a virtual logical child segment type if PROCOPT= L or LS is specified. The replace and delete function also imply the get function.

If a segment has PROCOPT=K specified, an unqualified get next call (GN) skips to the next sensitive segment with a PROCOPT other than K.

The SENSEG PROCOPT overrides the PCB PROCOPT. If PROCOPT=E is specified in the PCB, the SENSEG PROCOPT must also specify E if it is intended to schedule exclusively for that SENSEG.

Note that it's not valid to code the N or T processing option in the SENSEG statement. They can only be coded in the PCB statement.

Notes:

1. The processing option for a DEDB sequential dependent segment must be either G or I. If one of these values is not specified on the PCB statement, then PROCOPT=G or I must be specified on the SENSEG statement.

2. In the case of concatenated segments, the PROCOPT= operand governs the logical child segment of the concatenated segment. The logical parent of the concatenated segment is governed by the RULES= operand of the SEGM statement.

SSPTR=
Specifies the subset pointer number and the sensitivity for the pointer. Up to 8 subset pointers may be defined. The subset pointer number (the first operand) must be 1 through 8. The sensitivity for the pointer (the second operand) must be R (read sensitive) or U (update). If the first operand and the second operand are not specified, the

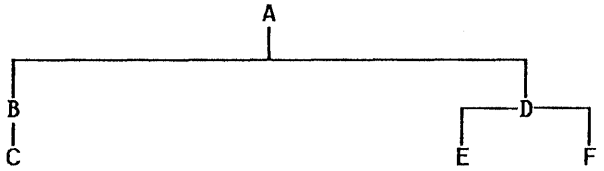
pointer has no sensitivity. If only n is specified, the pointer is read sensitive (R is the default).

U (update sensitivity) cannot be used if GO has been specified on the PCB statement.

INDICES=
Specifies which secondary indexes contain search fields that are used to qualify SSAs for an indexed segment type. The INDICES= operand can be specified for indexed segment types only. It enables SSAs of calls for the indexed segment type to be qualified on the search field of the index segment type contained in each secondary index specified. An SSA of a call for an indexed segment type cannot be qualified on the search field of a secondary index unless that secondary index was specified in the INDICES= operand of the SENSEG statement for the indexed segment type or in the PROCSEQ= operand of the PCB statement. For listl, up to 32 DBD names of secondary indexes can be specified. If two or more names are specified, these names must be separated by commas and the list enclosed in parentheses.

Note: Do not specify INDICES= on a SENSEG statement if PROCOPT=L, LS, I, or D was specified on the preceding PCB statement.

EXAMPLE OF SEGMENT DEFINITION: The data structure is:



Note: All of the above segments are defined within one DBD.

The complete PCB and SENSEG statements for the above data structure might be written as follows:

```
Col.  10    Col.  16                              Col.  72

PCB         TYPE=DB,DBDNAME=DATABASE,             X
            PROCOPT=A,KEYLEN=22
SENSEG      NAME=A,PARENT=0,PROCOPT=G
SENSEG      NAME=B,PARENT=A,PROCOPT=G
SENSEG      NAME=C,PARENT=B,PROCOPT=I
SENSEG      NAME=D,PARENT=A,PROCOPT=A
SENSEG      NAME=E,PARENT=D,PROCOPT=G
SENSEG      NAME=F,PARENT=D,PROCOPT=A
```

SENFLD STATEMENT

The SENFLD statement is used with the SENSEG statement to indicate those fields within a segment to which an application program is sensitive. One or more SENFLD statements can be included; each statement must follow the SENSEG statement to which it is related. A maximum of 255 SENFLD statements can be defined for a given SENSEG statement. A maximum of 10 000 SENFLD statements can be defined in a single PSB generation. The format of the SENFLD statement is shown in Figure 40. The same field may be referenced in more than one SENFLD statement within a SENSEG.If the duplicate field names participate in a concatenated segment and the same field name appears in both portions of the concatenation, the first reference will be to the logical child, and all subsequent references will be to the logical parent. This referencing sequence determines the order in which fields will be moved to the user's I/O area.

The length field of a variable-length segment may not be referenced through a SENFLD statement.

A SENFLD statement may not appear within a SENSEG with PROCOPT=K.

A SENFLD statement may not appear within a SENSEG with PROCOPT=I or L, if the SENSEG refers to a logical child segment.

If SENFLD statements are used within a SENSEG with PROCOPT=I or L, a SENFLD statement must be included for the segment sequence field, if it exists.

For retrieve-only processing a user may request, via the SENFLD statement, that the same data be moved to multiple locations in the user's I/O area, provided no overlap occurs.

This statement is not supported for MSDB and DEDB.

	SENFLD	NAME=name , START=startpos [,{REPLACE / REPL} = {YES or Y / NO or N}]

Figure 40. SENFLD Statement Format

where:

NAME=
Is the name of this field as defined through a FIELD statement during DBD generation. The field is from 1 to 8 alphameric characters.

START=
Specifies the starting position of this field relative to the beginning of the segment within the user's I/O area. startpos for the first byte of a segment is one. startpos must be a decimal number whose value does not exceed 32767.

REPLACE= OR REPL=
Specifies whether or not this field may be altered on a replace call. If omitted, REPLACE=YES is the default.

PSBGEN STATEMENT

The fifth type of statement required for a PSB generation is one that specifies characteristics of the application program. The format for the PSBGEN statement is shown in Figure 41 on page 108.

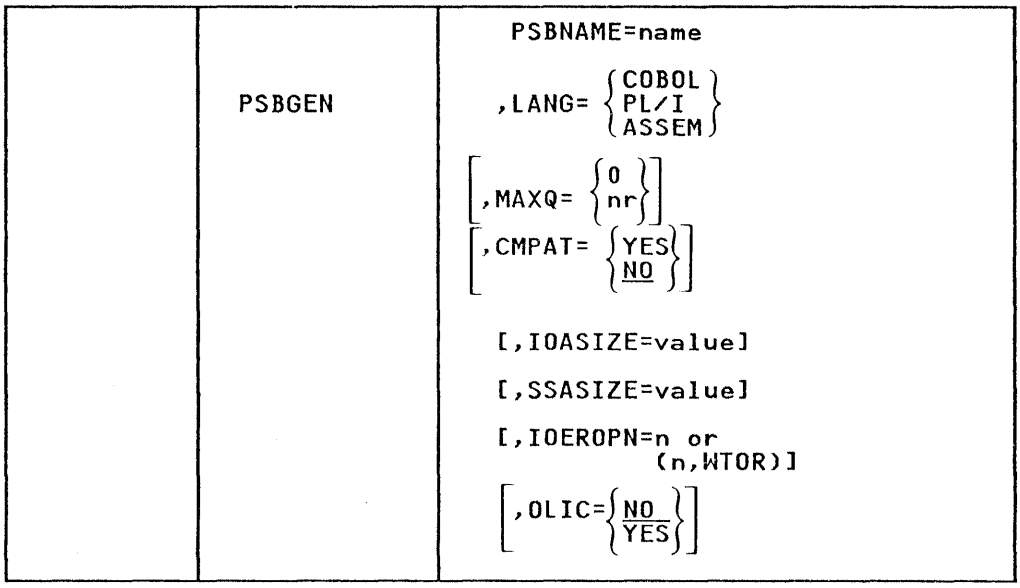


Figure 41. PSBGEN Statement Format

where:

PSBNAME=
Is the parameter keyword for the alphameric name of this PSB. The name value for the PSBNAME must be 8 characters or fewer in length. This name becomes the load module name for the PSB in the library IMSVS.PSBLIB. This name must be the same as the program load module name in the program library called IMSVS.PGMLIB if the program is to run in a message processing region. No special characters may be used in the name.

LANG=
Is the parameter keyword for the compiler language in which this message processing or batch processing application program is written. The value for this parameter must be either COBOL, PL/I, or ASSEM, with no trailing blanks.

MAXQ=nr
Is the maximum number of data base calls with Qx command codes which may be issued between synchronization points. If this number is exceeded, the application program will abend. The default value is zero.

CMPAT=
Provides for compatibility between BMP or MSG and Batch-DL/I parameter lists. If CMPAT=YES, the PSB is always treated as if there were an I/O PCB, no matter how it is used. If CMPAT=NO, the PSB has an I/O PCB added only when run in a BMP or MSG region. The default is NO.

IOASIZE=
Allows the user to specify the size of the largest I/O area to be used by the application program. The size specification is used to determine the amount of main storage reserved in the PSB pool to hold the control region's copy of the user's I/O area data during scheduling of this application program. If this value is not specified, the ACB utility program calculates a maximum I/O area size to be used as a default. The size calculated is the total length of all sensitive segments in the longest possible path call. (The total length of the segment must be used, even if the application program is not sensitive to all fields in a segment.) The value specified is in bytes, with a maximum of 256000. It should be noted,

however, that the combined length of all concatenated segments to be returned to the application on a single path call may not exceed 65535 bytes.

If the PSB contains any field sensitive segment, and IOASIZE is specified, the specified value is used only if it is larger than the IOASIZE calculated by the ACBGEN utility. The value of IOASIZE which will be used is indicated in message DFS0593I issued by ACBGEN.

If STAT calls or the test program (DFSDDLT0) is to be used with this PSB, then IOASIZE must be greater than 360 bytes.

If CMD or GCMD calls (from automated operator interface application programs) are to be used with this PSB, then IOASIZE must be at least 132 bytes.

If GSAM checkpoint/restart is going to be used, this parameter should be set to a value equal to or greater than the larger of the:

- I/O area necessary to receive data from a GU call issued during restart, while repositioning DL/I data bases that were checkpointed.
- Largest LRECL used in a GSAM data set that is checkpointed (this parameter is required for GSAM/VSAM data sets).

Note: Either the value pointed to by the third parameter (I/O AREA LEN) of the XRST CALL or the value of this parameter will be used, depending on which value is larger.

SSASIZE=
Allows the user to specify the maximum total length of all SSAs to be used by the application program. The size specification is used to determine the amount of main storage reserved in the PSB pool to hold the control region's copy of the user's SSA strings during scheduling of this application program. If this value is not specified, the ACB utility program calculates a maximum SSA size to be used as a default. The size calculated is the maximum number of levels in any PCB within this PSB times 280. The value specified is in bytes, with a maximum of 256000.

IOEROPN=
In applicable only in batch type regions (DLI or DBB). The n subparameter is the condition code returned to the operating system when IMS/VS terminates normally and one or more input or output errors occurred on any data base during the application program execution. The n subparameter is a number from 0 to 4095.

If n=451, IMS/VS terminates with a U451 abend rather than pass a condition code to the operating system. If IMS/VS or the application program abends (other than U451), and an I/O error has also occurred, and n=451, a write-to-programmer of message DFS0426I is issued. This message indicates that an I/O error has occurred during execution and that a U451 abend would have occurred if the actual abend had not.

If the WTOR subparameter is specified, a WTOR for the DFS0451A I/O error message is issued, and DL/I waits for the operator to respond before continuing. If the operator responds "ABEND," IMS/VS terminates with a U451 abend. A response of "CONT" causes IMS/VS to continue. Any other response causes the DFS0451A message to be reissued.

By using the IOEROPN parameter, the user can set a unique JCL condition code when an I/O error occurs and test the condition code in subsequent job steps. If this parameter is not specified, the return code passed from the

application program is passed to the operating system and status codes and console messages are the only indications of data base I/O errors.

If you code the WTOR subparameter, you must code the n subparameter and parentheses are required. If you code only IOEROPN=n, parentheses are not required.

OLIC=
Specifies whether the user of this PSB is authorized to execute the Online Data Base Image Copy utility or the Surveyor utility feature run as a BMP against a data base named in this PSB. YES allows the Online Image Copy and the Surveyor utility feature; NO prohibits the Online Image Copy and the Surveyor utility feature. NO is the default. This operand is invalid if any PCB specifies PROCOPT=L or LS.

Note: This operand is not applicable to CICS/VS.

There may be several PCB statements for message output and several PCB statements for data bases, but only one PSBGEN statement in a PSB generation statement deck. The PSBGEN statement must be the last statement in the deck preceding the END statement.

END STATEMENT

The five types of PSB generation statements must be followed by an END statement. The END statement is required by the macro assembler to indicate the end of the assembly data.

EXECUTION OF PSB GENERATION—JCL

PSBGEN is run as a normal operating system job after IMS/VS system definition. IMS/VS system definition causes the procedure named PSBGEN to be placed in the IMSVS.PROCLIB procedure library. The following JCL statements are used to invoke the PSBGEN procedure.

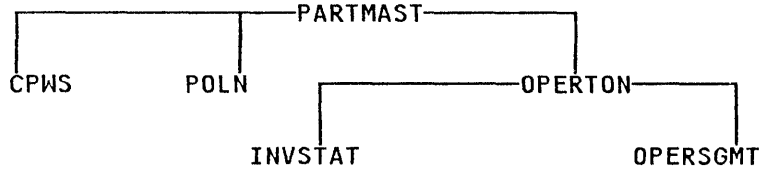
```
//PSBGEN   JOB  MSGLEVEL=1
//         EXEC    PSBGEN,MBR=
//C.SYSIN  DD   *

            PCB
         SENSEG       The control statements for PSB generation
         PSBGEN
         END
/*
```

where:

MBR=
Is the name of the PSB to be generated. This name must be the same as the name specified on the PSBNAME= operand of the PSBGEN statement. If this precaution is not followed, a user ABEND 929 can occur during execution, or message DFS929I ("BLDL FAILED FOR MEMBER") can be received during an ACBGEN "BUILD PSB" operation.

PSB GENERATION EXAMPLES

A PSB generation is to be done for a message processing program to process the following hierarchic data structure. Output messages are to be transmitted to logical terminals OUTPUT1 and OUTPUT2 in addition to the terminal representing the source of input.



EXAMPLE 1:

```
PCB     TYPE=TP,NAME=OUTPUT1
PCB     TYPE=TP,NAME=OUTPUT2
PCB     TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
SENSEG  NAME=PARTMAST,PARENT=0,PROCOPT=A
SENSEG  NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
SENSEG  NAME=POLN,PARENT=PARTMAST,PROCOPT=A
SENSEG  NAME=OPERTON,PARENT=PARTMAST,PROCOPT=A
SENSEG  NAME=INVSTAT,PARENT=OPERTON,PROCOPT=A
SENSEG  NAME=OPERSGMT,PARENT=OPERTON
PSBGEN  LANG=COBOL,PSBNAME=APPLPGM1
END
```

Example 2 shows these statements being used for a batch program, where programs using this PSB do not reference the telecommunications PCBs in the batch environment.

EXAMPLE 2:

```
PCB     TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
SENSEG  NAME=PARTMAST,PARENT=0,PROCOPT=A
SENSEG  NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
SENSEG  NAME=POLN,PARENT=PARTMAST,PROCOPT=A
SENSEG  NAME=OPERTON,PARENT=PARTMAST,PROCOPT=A
SENSEG  NAME=INVSTAT,PARENT=OPERTON,PROCOPT=A
SENSEG  NAME=OPERSGMT,PARENT=OPERTON
PSBGEN  LANG=COBOL,PSBNAME=APPLPGM1
END
```

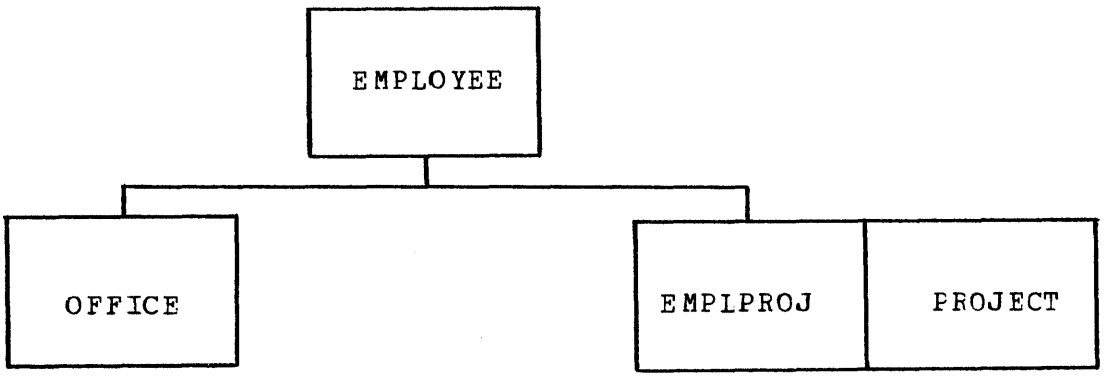
Example 3 shows that a PSB generation is done for a batch message processing program. The GSAM PCB is used by the application program to generate a report file.

EXAMPLE 3:

```
PCB     TYPE=TP,NAME=OUTPUT1
PCB     TYPE=TP,NAME=OUTPUT2
PCB     TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
SENSEG  NAME=PARTMAST,PARENT=0,PROCOPT=A
SENSEG  NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
PCB     TYPE=GSAM,DBDNAME=REPORT,PROCOPT=LS
PSBGEN  LANG=COBOL,PSBNAME=APPLPGM3
END
```

FIELD LEVEL SENSITIVITY PSB GENERATION EXAMPLE

Example 1 shows a PCB for a batch program using field level sensitivity.



SEGMENT NAME	FIELD NAME	START LOCATION	LENGTH
EMPLOYEE	EMPSSN	1	9
	EMPLNAME	10	10
	EMPFNAME	20	9
	EMPMI	29	1
	EMPADDR	30	30
OFFICE	OFNUMBER	1	5
	OFPHONE	6	7
EMPLPROJ	EPFUNCTN	1	20
	EPTIMEST	21	5
	EPTIMCUR	26	5
PROJECT	PROJNUM	1	8
	PROJTTLE	9	20
	PROJSTRT	29	8
	PROJEND	37	8
	PROJSTAT	45	1

```
PCB      TYPE=DB,NAME=FISDBD1,PROCOPT=GRP,KEYLEN=20
SENSEG   NAME=EMPLOYEE,PARENT=0
SENFLD   NAME=EMPLNAME,START=13,REPL=NO
SENFLD   NAME=EMPFNAME,START=1,REPL=NO
SENFLD   NAME=EMPMI,START=11
SENSEG   NAME=OFFICE,PARENT=EMPLOYEE
SENSEG   NAME=EMPLPROJ,PARENT=EMPLOYEE
SENFLD   NAME=PROJNUM,START=1
SENFLD   NAME=PROJTITLE,START=10
SENFLD   NAME=EPFUNCTN,START=35
SENFLD   NAME=EPTIMEST,START=60
SENFLD   NAME=EPTIMCUR,START =70
PSBGEN   LANG=ASSEM,PSBNAME=FISPCB1
END
```

FAST PATH PSB GENERATION EXAMPLES

Example 1 shows the statements for an MSDB PSB containing eight PCBs.

EXAMPLE 1:

```
PCB     TYPE=DB,DBDNAME=MSDBLM01,PROCOPT=R,      NONTERMINAL-RELATED      X
        KEYLEN=4                                 END OF PCB STATEMENT
SENSEG  NAME=LDM,PARENT=0                        (DEFAULT)
PCB     TYPE=DB,DBDNAME=MSDBLM02,PROCOPT=R,      NONTERMINAL-RELATED      X
        KEYLEN=1
SENSEG  NAME=LDM,PARENT=0
PCB     TYPE=DB,DBDNAME=MSDBLM03,PROCOPT=R,      NONTERMINAL-RELATED      X
        KEYLEN=2
SENSEG  NAME=LDM,PARENT=0
PCB     TYPE=DB,DBDNAME=MSDBLM04,PROCOPT=R,      NONTERMINAL-RELATED      X
        KEYLEN=8                                 TERM KEYS
SENSEG  NAME=LDM,PARENT=0
PCB     TYPE=DB,DBDNAME=MSDBLM05,PROCOPT=R,      FIXED RELATED            X
        KEYLEN=8
SENSEG  NAME=LDM,PARENT=0
PCB     TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=A,      DYNAMIC RELATED          X
        KEYLEN=8
SENSEG  NAME=LDM,PARENT=0
PCB     TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=R,      DYNAMIC RELATED          X
        KEYLEN=8
SENSEG  NAME=LDM,PARENT=0
PCB     TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=G,      DYNAMIC RELATED          X
        KEYLEN=8
SENSEG  NAME=LDM,PARENT=0
PSBGEN  LANG=ASSEM,PSBNAME=DDLTM01               END OF PSBGEN MACRO
END     END OF PSB GEN
```

Example 2 shows the statements for DEDB subset pointers.

EXAMPLE 2:

```
PCB TYPE=DB,DBDNAME=X,PROCOPT=A,KEYLEN=100
SENSEG NAME=A,PARENT=C
SENSEG NAME=B,PARENT=A,SSPTR=((1,R),(2,U),(5))
SENSEG NAME=C,PARENT=B
SENSEG NAME=D,PARENT=A,SSPTR=((2,R))
PSBGEN LANG=COBOL,PSBNAME=APPI01
END
```

Notes:

1. SSPTR=((n,r))
   n: Subset pointer number in this SENSEG
   r: Sensitivity for the pointer (R:read, U:update)
2. If n and r are not specified, the pointer has no sensitivity
3. If n is specified but r is not specified, the default is R (read sensitive)

DESCRIPTION OF PSB GENERATION OUTPUT

PSB generation produces three types of printed output and one load module, which becomes a member of the partitioned data set, IMSVS.PSBLIB. Each of these outputs is described in the following paragraphs.

CONTROL STATEMENT LISTING

This is a listing of the input statement images to this job step.

DIAGNOSTICS

Errors discovered during the processing of each control statement result in diagnostic messages being printed immediately following the image of the last control statement read before the error was discovered. The message may either refer to the control statement immediately preceding it or the preceding group of control statements. It is also possible that more than one message could be printed for each control statement. In this case, they follow each other on the output listing. After all the control statements have been read, a further check is made of the logic of the entire deck. This may result in one or more additional diagnostic messages.

Any discovered error results in the diagnostic message(s) being printed, the control statements being listed, and the other outputs being suppressed. However, all the control statements are read and checked before the PSB generation execution is terminated. The link-edit step of PSB generation is not executed if a control statement error has been found.

ASSEMBLER LISTING

Except when PRINT NOGEN is specified, an operating system assembler language listing of the PSB created by PSB generation execution if provided.

LOAD MODULE

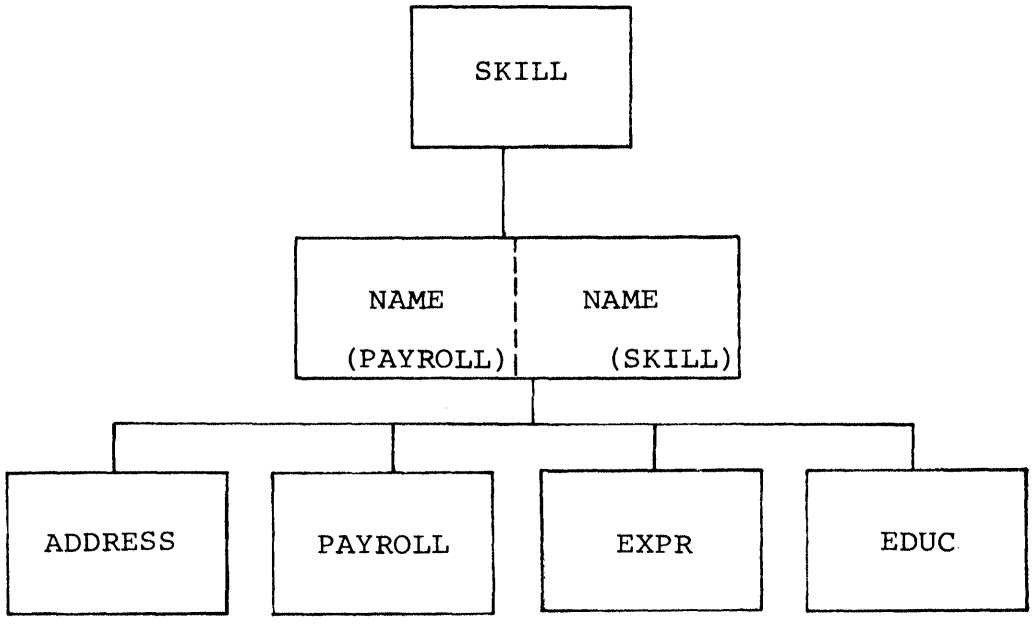
PSB generation is a two-step operating system job. Step 1 is a macro assembly execution that produces an object module. Step 2 is a link-edit of the object module, which produces a load module that becomes a member of IMSVS.PSBLIB.

PSB GENERATION ERROR CONDITIONS

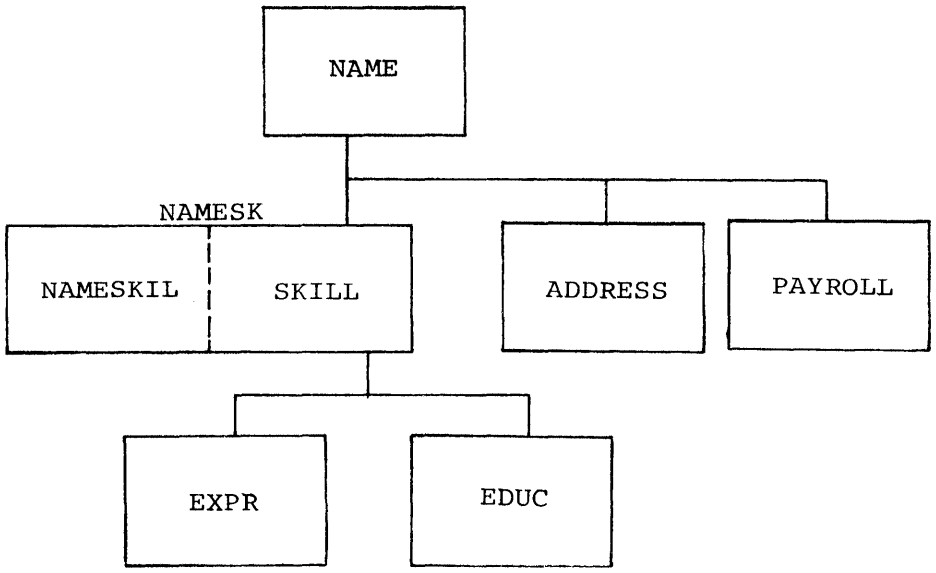
See Chapter 6, "Program Specification Block Generation (PSBGEN) Messages," in IMS/VS Messages and Codes Reference Manual for a complete description of the IMS/VS messages that indicate PSB errors.

ADDITIONAL PSB GENERATION EXAMPLES

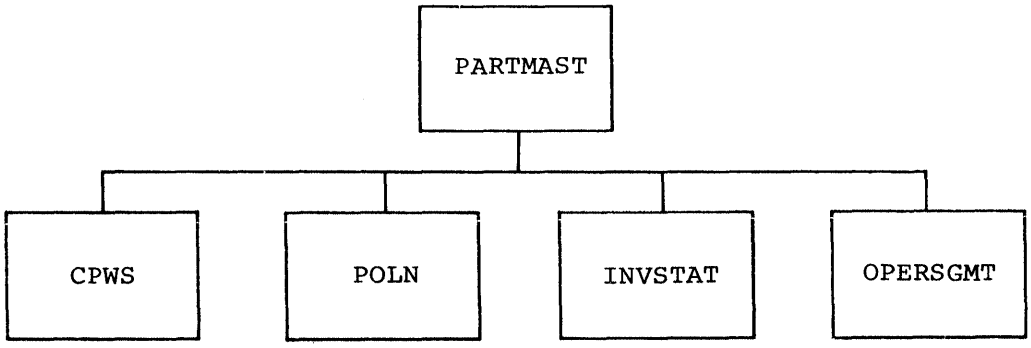
LOGICAL DATA BASE




```
PCB         TYPE=DB,DBDNAME=LOGIC1;PROCOPT=G,KEYLEN=151,POS=M
SENSEG      NAME=SKILL,PARENT=0,PROCOPT=A
SENSEG      NAME=NAME,PARENT=SKILL,PROCOPT=A
SENSEG      NAME=ADDRESS,PARENT=NAME,PROCOPT=A
SENSEG      NAME=PAYROLL,PARENT=NAME,PROCOPT=A
SENSEG      NAME=EXPR,PARENT=NAME,PROCOPT=A
SENSEG      NAME=EDUC,PARENT=NAME,PROCOPT=A
PSBGEN      LANG=COBOL,PSBNAME=PGMX
END
```

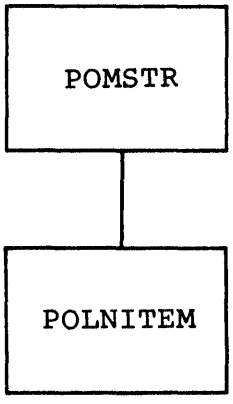



```
PCB         TYPE=DB,DBDNAME=LOGICDB,PROCOPT=A,KEYLEN=241,POS=M
SENSEG      NAME=NAME,PARENT=0,PROCOPT=G
SENSEG      NAME=NAMESK,PARENT=NAME,PROCOPT=G
SENSEG      NAME=EXPR,PARENT=NAMESK,PROCOPT=G
SENSEG      NAME=EDUC,PARENT=NAMESK,PROCOPT=G
SENSEG      NAME=ADDRESS,PARENT=NAME,PROCOPT=G
SENSEG      NAME=PAYROLL,PARENT=NAME,PROCOPT=G
PSBGEN      LANG=PL/I,PSBNAME=PGMY
END
```



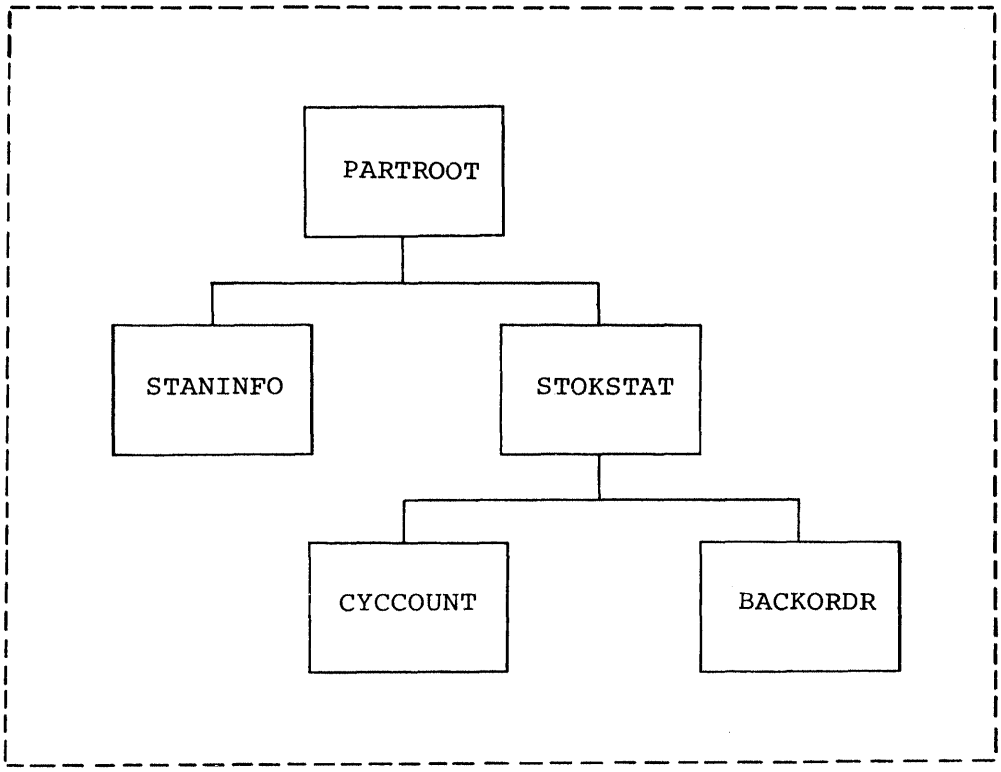

```
PCB         TYPE=TP,LTERM=OUTPUT
PCB         TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=GIDR,KEYLEN=100
SENSEG      NAME=PARTMAST,PARENT=0,PROCOPT=A
SENSEG      NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
SENSEG      NAME=POLN,PARENT=PARTMAST,PROCOPT=A
SENSEG      NAME=INVSTAT,PARENT=PARTMAST,PROCOPT=A
SENSEG      NAME=OPERSGMT,PARENT=PARTMAST
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
```

EXAMPLE: PODR (PURCHASE ORDER) DATA BASE



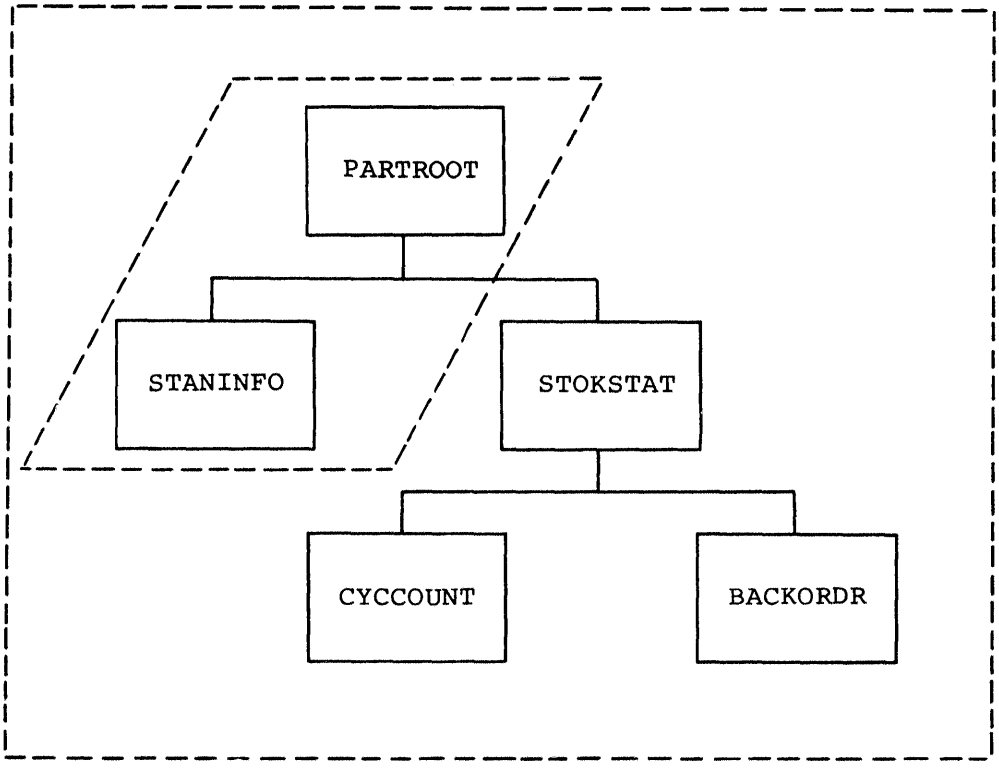

```
PCB         TYPE=TP,NAME=OUT1
PCB         TYPE=TP,NAME=OUT2
PCB         TYPE=DB,DBDNAME=PODB,PROCOPT=GID,KEYLEN=200
SENSEG      NAME=POMSTR
SENSEG      NAME=POLNITEM,PARENT=POMSTR
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM3
END
```

EXAMPLE 1: SAMPLE PROBLEM—APPLICATION DATA BASE (DBDNAME=DI21PART)



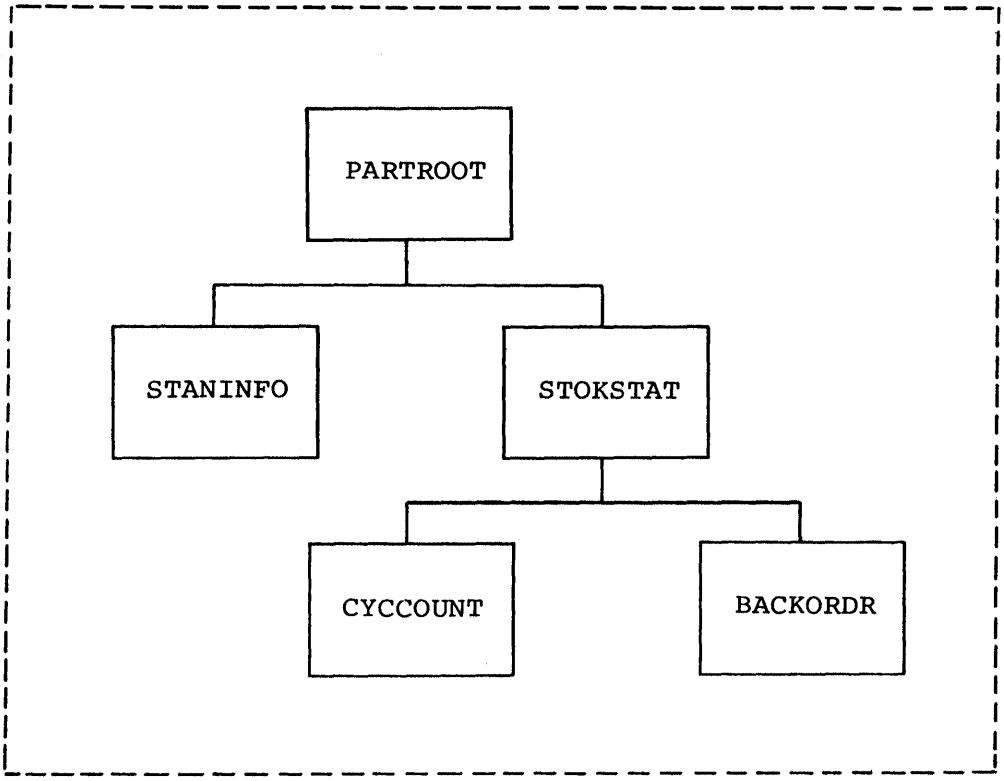

```
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM01
END
```

EXAMPLE 2: SAMPLE PROBLEM—APPLICATION DATA BASE (DBDNAME=DI21PART)



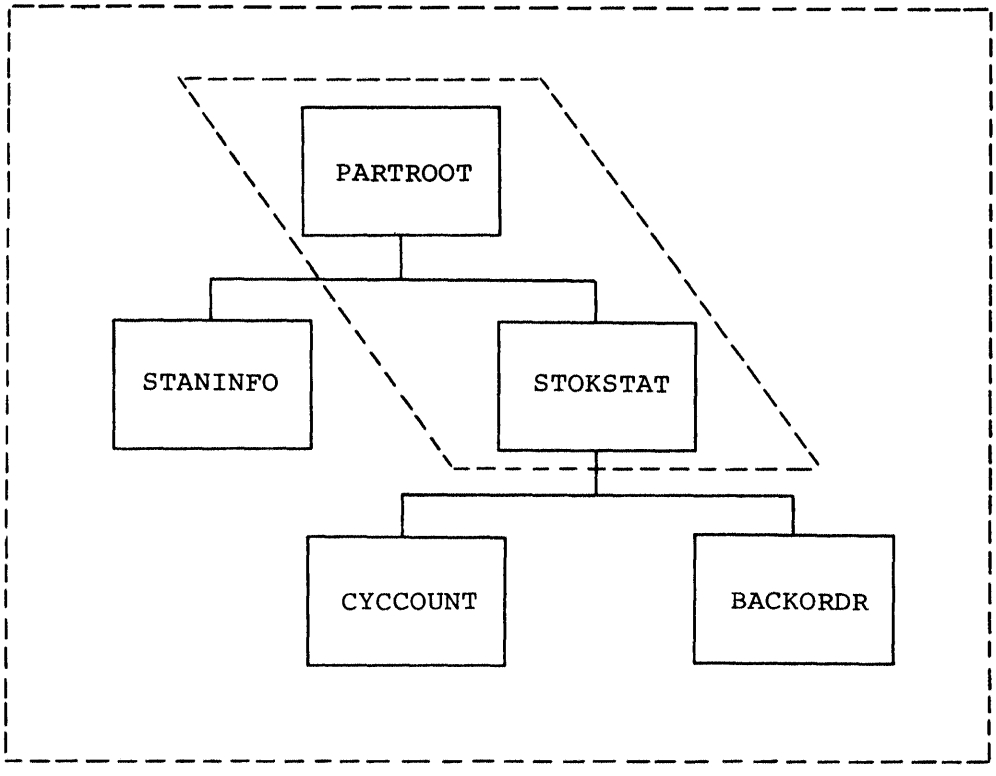

```
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=19
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM02
END
```

EXAMPLE 3: SAMPLE PROBLEM—APPLICATION DATA BASE (DBDNAME=DI21PART)



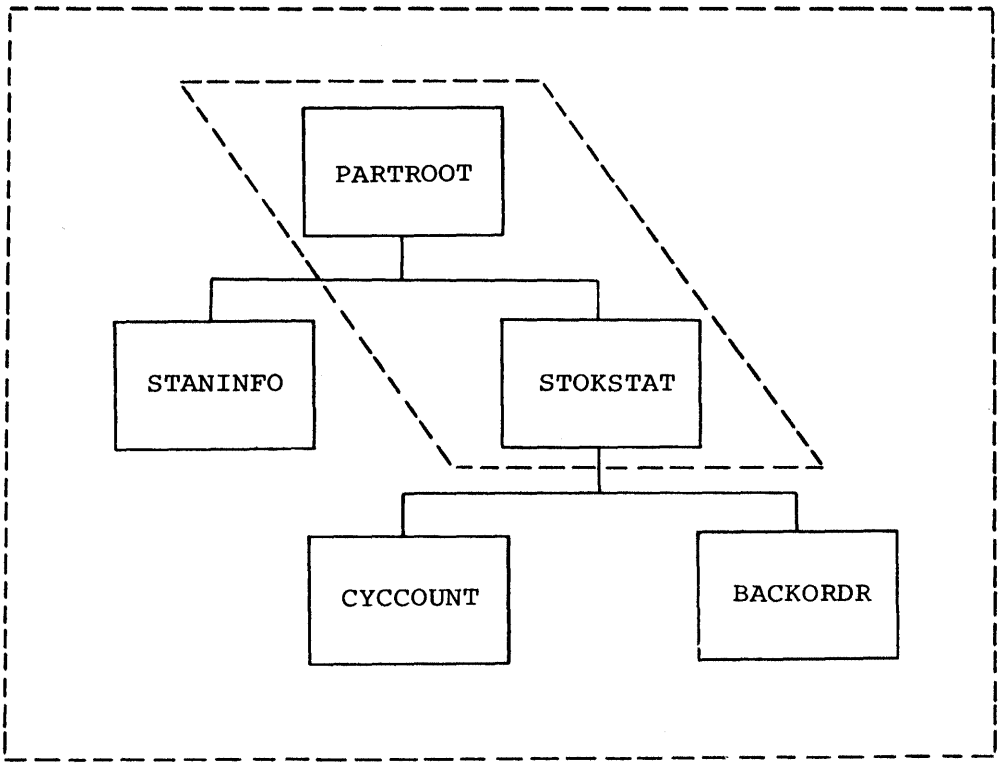

```
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=43
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
SENSEG      NAME=CYCCOUNT,PARENT=STOKSTAT
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM03
END
```

EXAMPLE 4: SAMPLE PROBLEM—APPLICATION DATA BASE (DBDNAME=DI21PART)



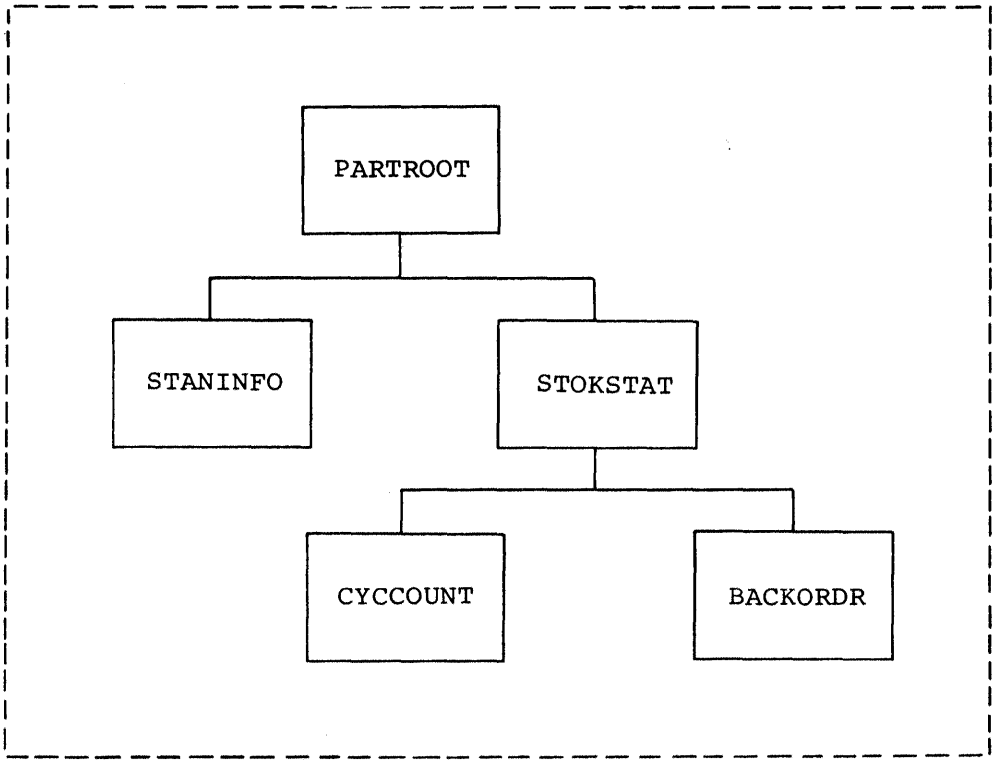

```
PCB         TYPE=TP,LTERM=HOWARD
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=A,KEYLEN=33
SENSEG      NAME=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM05
END
```

EXAMPLE 5: SAMPLE PROBLEM—APPLICATION DATA BASE (DBDNAME=DI21PART)




```
PCB         TYPE=TP,LTERM=HOWARD
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=A,KEYLEN=33
SENSEG      NAME=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM06
END
```

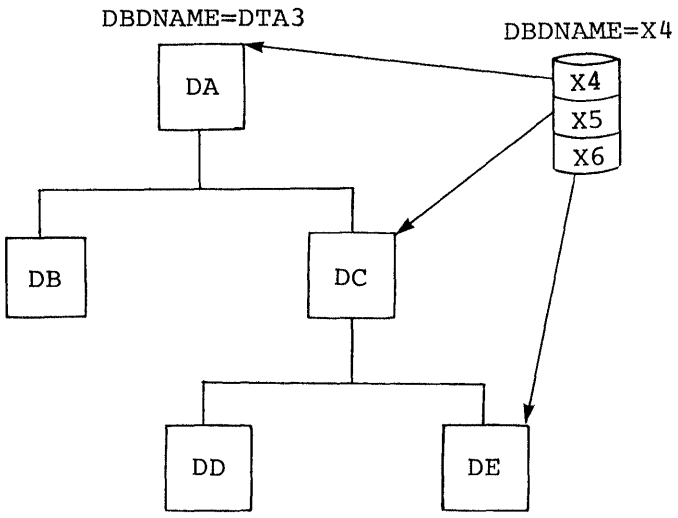
EXAMPLE 6: SAMPLE PROBLEM—APPLICATION DATA BASE (DBDNAME=DI21PART)



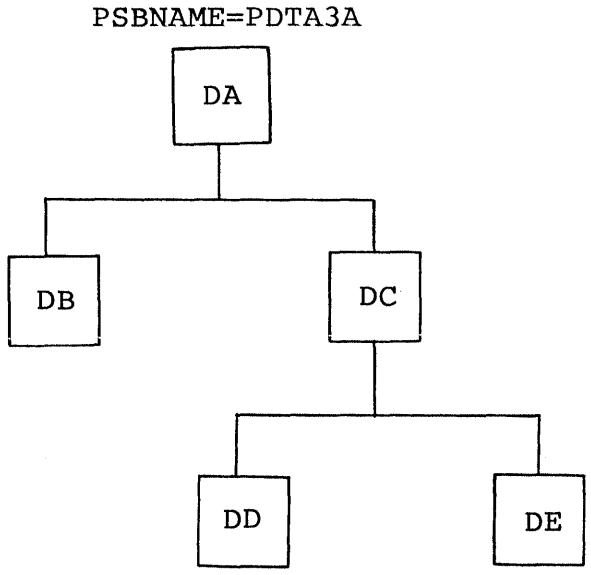

```
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=43
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
SENSEG      NAME=CYCCOUNT,PARENT=STOKSTAT
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM07
END
```

EXAMPLE 7: SHARED SECONDARY INDEX

A. Data base structure

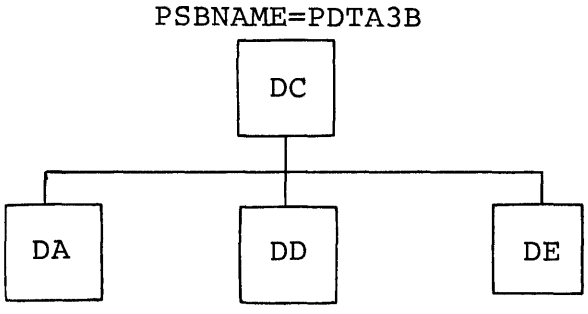


B. Data base structure for index through DA




```
PCB         TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X4
SENSEG      NAME=DA,PARENT=0
SENSEG      NAME=DB,PARENT=DA
SENSEG      NAME=DC,PARENT=DA,INDICES=X5
SENSEG      NAME=DD,PARENT=DC
SENSEG      NAME=DE,PARENT=DC,INDICES=X6
PSBGEN      LANG=COBOL,PSBNAME=PDTA3A
END
```

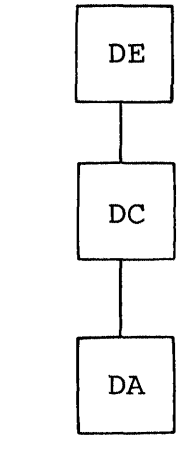
C. Data base structure for index through DC




```
PCB         TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X5
SENSEG      NAME=DC,PARENT=0
SENSEG      NAME=DA,PARENT=DC,INDICES=X4
SENSEG      NAME=DD,PARENT=DC
SENSEG      NAME=DE,PARENT=DC,INDICES=X6
PSBGEN      LANG=COBOL,PSBNAME=PDTA3B
END
```

Note: This data base structure may also include, as a substructure, the data base structure for index through DA.

D. Data base structure for index through DE




```
PCB         TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X6
SENSEG      NAME=DE,PARENT=0
SENSEG      NAME=DC,PARENT=DE,INDICES=X5
SENSEG      NAME=DA,PARENT=DC,INDICES=X4
PSBGEN      LANG=COBOL,PSBNAME=PDTA3C
END
```

Note: This data base structure may also include, as substructures, the data base structures for indexes through DA and DC.

E. PCB for INDEX data base

```
PCB         TYPE=DB,DBDNAME=X4,PROCOPT=A,KEYLEN=5
SENSEG      NAME=X4A,PARENT=0
PCB         TYPE=DB,DBDNAME=X5,PROCOPT=A,KEYLEN=5
SENSEG      NAME=X5A,PARENT=0
PCB         TYPE=DB,DBDNAME=X6,PROCOPT=A,KEYLEN=5
SENSEG      NAME=X6A,PARENT=0
PSBGEN      LANG=COBOL,PSBNAME=PX4
END
```

CHAPTER 3. APPLICATION CONTROL BLOCKS (ACB) MAINTENANCE UTILITY

OVERVIEW

When an application program is scheduled for execution, IMS/VS must first have available the DBD and PSB control blocks previously created by the DBDGEN and PSBGEN procedures. These control blocks must then be merged and expanded into an IMS/VS internal format called "application control blocks" (ACBs). The merge and expansion process is called "block building."

The purpose of the Application Control Blocks Maintenance utility is to save instruction execution and direct access wait time and to improve performance in application scheduling. It provides a facility for prebuilding the required application control blocks offline; hence when the application is scheduled its ACBs can be read in directly, and control can be passed promptly to the application program.

The prebuilding of application control blocks is required for the data base/data communication system. It is optional for the data base system. The use of IMSVS.ACBLIB in a data base system requires less virtual storage than building the ACBs dynamically from PSBLIB and DBDLIB.

GSAM PSBs and DBDs cannot be predefined using ACBGEN because the control blocks for GSAM are different from the standard IMS/VS data set control blocks. PSBs that reference GSAM as well as non-GSAM data bases can be predefined using ACBGEN to build the control block for the non-GSAM data bases.

IMS/VS conforms to MVS rules for data set authorization. If an IMS/VS job step is authorized, all libraries used in that job step must be authorized. To run an IMS/VS batch region as unauthorized, a nonauthorized library must be concatenated to IMSVS.RESLIB.

FUNCTIONAL DESCRIPTION

The Application Control Blocks Maintenance utility maintains the prebuilt blocks (ACB) library (IMSVS.ACBLIB). The ACB library is a consolidated library of program (PSB) and data base (DBD) descriptions. Through control statements, the maintenance utility may be directed to build all control blocks for all PSBs, for a specific PSB, or for all PSBs that reference a specific DBD. This utility does not change the PSB in IMSVS.PSBLIB or the DBD in IMSVS.DBDLIB. If changes are made in either PSBs or DBDs that require that the associated PSB or DBD be changed, these changes must be made before the utility is run. Additions, changes, and deletions to IMSVS.ACBLIB can be made without stopping IMS/VS, by using the Online Change utility and the Online Change function. For more information on using the Online Change utility, see Chapter 5.

It should be noted that changes in PSBs may require modifications to the affected application programs as well. For example, if a DBD has a segment name changed, all PSBs which are sensitive to that segment must have their SENSEG statements changed. Application programs which use this data base may need to be modified as well.

The Application Control Blocks Maintenance utility requires certain resources of the IMS/VS system but not the total system. IMSVS.PSBLIB and IMSVS.DBDLIB are shared data sets. IMSVS.ACBLIB must be used exclusively. The utility may therefore only be executed using an ACB library which is not concurrently allocated to an active IMS/VS system.

IMSVS.ACBLIB is modified and cannot be used for any other purpose during execution of this program. IMSVS.ACBLIB is a partitioned data set and carries required linkage information in the directory. This allows the operating system (IEHMOVE) and data set (IEBCOPY) utilities to be used for maintenance purposes.

Figure 42 shows the functional relationship of the input/output data sets and their naming requirements.

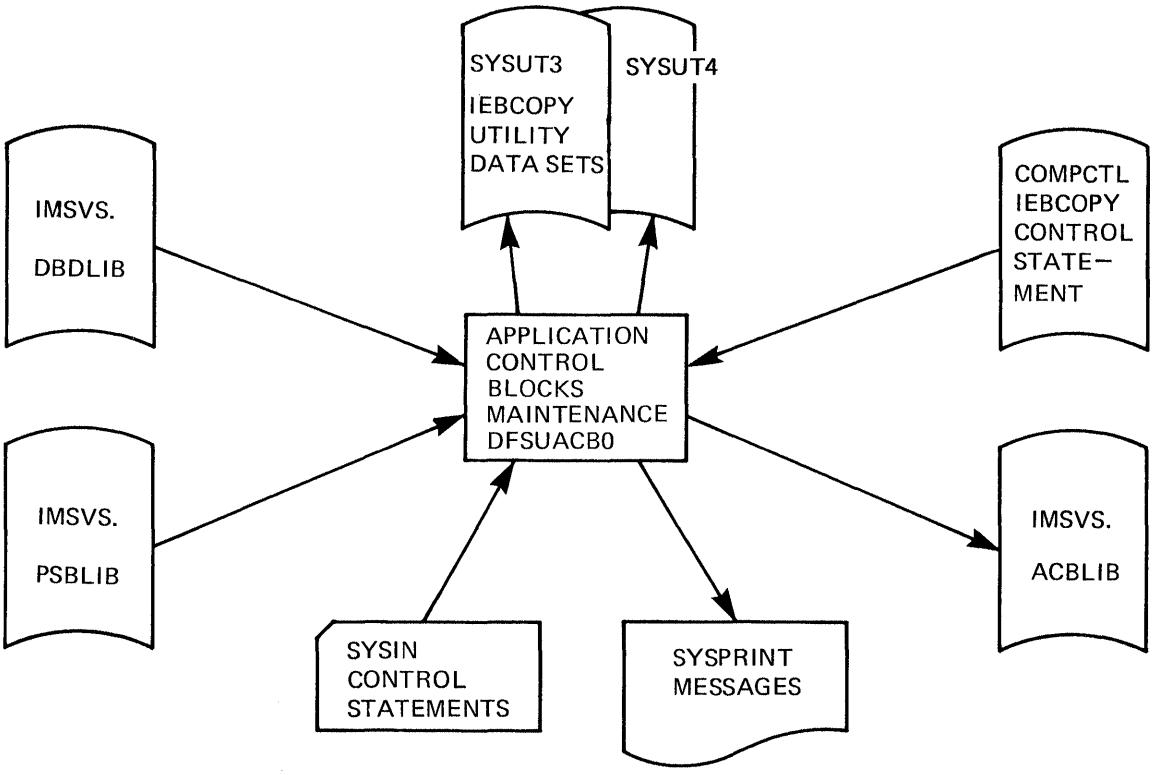


Figure 42. Application Control Blocks Maintenance Utility

JCL REQUIREMENTS

The Application Control Blocks Maintenance utility is executed as a standard OS/VS job. A JOB statement defined by the using installation, an EXEC statement, and DD statements that define input and output data sets are required.

The region size for execution of the ACB utility depends on the size of the blocks to be generated and typically varies from 100 to 150K bytes.

EXEC

This statement may be in the form PGM=DFSRRC00 or may invoke a procedure which contains the required JOB control statements. A parm field must be in the form:

```
PARM='UPB,PRECOMP,POSTCOMP'
```

where PRECOMP requests the IMSVS.ACBLIB data set be compressed before blocks are built and POSTCOMP requests compression after the blocks are built. 'UPB' indicates that the block maintenance utility is to receive control. This parameter is required. PRECOMP and POSTCOMP are optional and may be used in any combination.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library which contains the IMS/VS SVC modules.

SYSPRINT DD
Defines the output message data set. It can be a printer or be routed through the output stream. If DISP=(MOD,...) is specified, it can be a tape volume or a direct access device. The output can be blocked a multiple of 121.

IMS DD
Defines the IMSVS.PSBLIB and IMSVS.DBDLIB data sets. The data sets must be concatenated and contain all PSBs and DBDs to be used. The data sets may be shared with other jobs for read only. IMSVS.PSBLIB must be the first data set in the concatenation.

IMSACB DD
Defines the IMSVS.ACBLIB data set. This data set is modified and may not be shared with other jobs.

SYSUT3 DD
Defines a work data set that is required if either 'PRECOMP' or 'POSTCOMP' is specified on the EXEC statement. See the IEBCOPY chapter of OS/VS Utilities, for space allocation requirements.

SYSUT4 DD
Same function as SYSUT3.

COMPCTL DD
Defines the control input data set to be used by IEBCOPY if 'PRECOMP' or 'POSTCOMP' is specified. It should refer to a data set containing the following record only:

```
'COPY INDD=IMSACB,OUTDD=IMSACB'
```

Note: If both PRECOMP and POSTCOMP are requested on the EXEC statement parms, this data set must be capable of being closed with a reread option. It is suggested that this data set reference a member of IMSVS.PROCLIB which contains the required control statement:

```
//COMPCTL DD DSN=IMSVS.PROCLIB(DFSACBCP),DISP=SHR
```

SYSIN DD
Defines the control statement data sets. It may reside on a tape volume, direct access device, card reader, or be routed through the input stream. The input may be blocked a multiple of 80. As many control statements as are required may be processed during one execution of this utility.

CONTROL STATEMENT REQUIREMENTS

The utility control statements for this program are free-form. A statement is coded as a card image and is contained in columns 1 to 71. The control statement may optionally contain a name starting in column 1. The operation field must be preceded by and followed by one or more blanks. The operand is composed of one or more PSB/DBD names and must be preceded by and followed by one or more blanks. Commas, parentheses, and blanks can be used only as delimiting characters. Comments may be written following the last operand of a control statement, separated from the operand by one or more blanks. A control statement may be continued by inserting a comma after the last operand of the statement, inserting a non-blank character in column 72 and

continuing the statement in column 16 of the next control statement. Columns 1 to 15 of the continuing statement must be blank.

Col 1	Operation	Operand
	BUILD	ALL PSB= (psbname,...) [DBD=(dbdname,...)]
	DELETE	[PSB=(psbname,...)] [DBD=(dbdname,...)]

where:

BUILD
Indicates that blocks are to be built for the named PSBs which refer to the named DBDs.

DELETE
Indicates that blocks are to be deleted from ACBLIB. The named PSBs and all PSBs that refer to the named DBDs are deleted.

PSB=ALL
Means blocks are to be built for all PSBs that currently reside in IMSVS.PSBLIB. This function causes all PSBs in IMSVS.ACBLIB to be deleted and their space made available for reuse. If this option is selected, all other statements are ignored. This operand may not be used with a DELETE statement. This operand is normally used to create an initial IMSVS.ACBLIB.

PSB=(psbname,...)
Means blocks are to be built or deleted for all PSBs named on this control statement. As many of this type of control statement as required may be submitted. This operand is normally used to add a new PSB to IMSVS.ACBLIB or delete a PSB no longer in use. Parentheses may be omitted if a single operand is supplied.

DBD=(dbdname,...)
Means blocks are to be built or deleted for this DBD and all PSBs which reference this DBD either directly or indirectly through logical relationships. The DBD to be built must already exist in IMSVS.ACBLIB. The referencing PSBs must already exist in IMSVS.ACBLIB. PSBs newly added to IMSVS.PSBLIB must be referenced by PSB= operands. Because deleting a PSB does not delete any DBDs referenced by the PSB, this operand may be used to delete specific DBDs. However, deleting or building a DBD causes every PSB in IMSVS.ACBLIB that references the named DBD to be rebuilt or deleted based on the request type. Parentheses may be omitted if a single operand is supplied.

Notes:

1. Every PSB processed by this program generates a member in the IMSVS.ACBLIB data set. DBDs referenced by PSBs generate a member the first time the specific DBD is processed or any time a DBD name appears on a control statement. All PSBs that reference the same DBD carry information in their directory entries to connect the PSB to the referenced DBDs.

2. Logical DBDs do not generate a member in IMSVS.ACBLIB and cannot be referenced on BUILD or DELETE control statements.

CAUTION: When a DBD is replaced in IMSVS.DBDLIB, it must also be included in a BUILD DBD control statement. This is the only valid way the DBD can be replaced in IMSVS.ACBLIB without doing a BUILD PSB=ALL.

The control statement, BUILD PSB, always gets all DBDs from DBDLIB. If a BUILD PSB is performed that references a modified DBD on DBDLIB, the PSB replaced on ACBLIB will contain the updated version of the DBD. If this BUILD PSB occurs before a BUILD DBD for the changed DBD, ACBLIB will contain PSBs with different versions of the DBD. The PSBs specified in the BUILD PSB will contain the updated DBD, while those not built will reference the old DBD. When a DBD for a PSB on ACBLIB does not match the accessed data base, the results will be unpredictable. (For example, U852 abend occurs because segment codes have been added or deleted in the changed DBD). Therefore, when DBDGEN is run for later use, do not build a PSB that refers to the changed DBD unless the data base reflects the change.

When a physical DBD is changed and is referenced in a BUILD DBD statement, all physical DBDs that are logically related to the one that was changed must be referenced also in a BUILD DBD statement. However, DBDs that are logically related to these DBDs do not need to be rebuilt.

Figure 43 is an example of physical data bases, where A is the changed DBD. B and C are logically related to A; D is logically related to B; E is logically related to C; but D and E are not referenced in the BUILD DBD statement because they are not logically related to A.

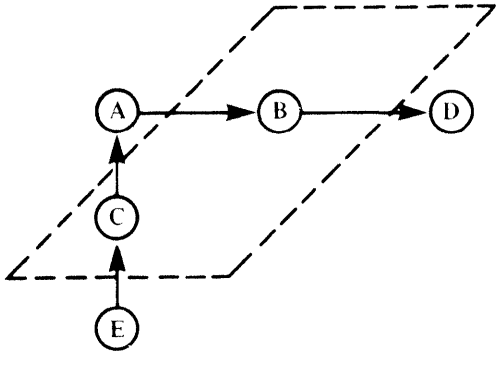


Figure 43. Example of Logically Related Physical Data Bases

RETURN CODES

The following return codes are provided.

Code	Meaning
0	Successful completion of all operations.
4	One or more warning messages issued.
8	One or more blocks could not be built.
16	Program terminated due to severe errors.

EXAMPLES OF ACB MAINTENANCE UTILITY PROGRAM

EXAMPLE 1

This example creates blocks for all PSBs that currently reside in IMSVS.PSBLIB. All blocks currently existing in IMSVS.ACBLIB are deleted and their space is reused. This option will normally be used for initial creation of the IMSVS.ACBLIB data set. If space is not yet allocated for ACBLIB, there should be a space parameter and a DISP=NEW on the IMSACB DD statement.

```
//BLDBLKS   JOB  1,1,MSGLEVEL=(1,1)
//STEP     EXEC  PGM=DFSRRC00,REGION=120K,PARM='UPB'
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//IMS        DD  DSN=IMSVS.PSBLIB,DISP=SHR
//           DD  DSN=IMSVS.DBDLIB,DISP=SHR
//IMSACB     DD  DSN=IMSVS.ACBLIB,DISP=OLD
//SYSIN      DD  *
               BUILD PSB=ALL
/*
```

EXAMPLE 2

This example creates blocks for PSB1, PSB2, and PSB3. All other PSBs in IMSVS.ACBLIB remain unchanged. If any DBDs referenced by these PSBs do not exist in IMSVS.ACBLIB, they are added. In addition, DBD5 and DBD6 are deleted from ACBLIB. IMSVS.ACBLIB is compressed after the blocks are built, and deletions are performed.

```
//BLDBLKS   JOB  1,1,MSGLEVEL=(1,1)
//STEP     EXEC  PGM=DFSRRC00,REGION=120K,
//          PARM='UPB,POSTCOMP'
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//IMS        DD  DSN=IMSVS.PSBLIB,DISP=SHR
//           DD  DSN=IMSVS.DBDLIB,DISP=SHR
//COMPCTL    DD  DSN=IMSVS.PROCLIB(DFSACBCP),DISP=SHR
//IMSACB     DD  DSN=IMSVS.ACBLIB,DISP=OLD
//SYSUT3     DD  DSN=&SYSUT3,DISP=(,DELETE),UNIT=SYSDA,
//               SPACE=(80,(150,50))
//SYSUT4     DD  DSN=&SYSUT4,DISP=(,DELETE),UNIT=SYSDA,
//               SPACE=(256,(20,10))
//SYSIN      DD  *
              BUILD  PSB=(PSB1,PSB2,PSB3)
              DELETE DBD=(DBD5,DBD6)
/*
```

Note: The COMPCTL data set contains one 80-byte record in the form bbCOPY INDD=IMSACB,OUTDD=IMSACB

EXAMPLE 3

This example deletes PSB1 from the IMSVS.ACBLIB data set and causes all PSBs in the IMSVS.ACBLIB data set that reference DBD4 to have their blocks rebuilt. If PSB1 referenced DBD4, it will not be rebuilt, since PSB1 had just been deleted from IMSVS.ACBLIB. Note that PSB1 is <u>not</u> deleted from IMSVS.PSBLIB. The IMSVS.ACBLIB is compressed before and after the blocks have been built.

```
//BLDBLKS    JOB 1,1,MSGLEVEL=(1,1)
//STEP      EXEC PGM=DFSRRC00,REGION=120K,
// PARM='UPB,PRECOMP,POSTCOMP'
//STEPLIB    DD  DSN=IMS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//IMS        DD  DSN=IMSVS.PSBLIB,DISP=SHR
//           DD  DSN=IMSVS.DBDLIB,DISP=SHR
//IMSACB     DD  DSN=IMSVS.ACBLIB,DISP=SHR
//COMPCTL    DD  DSN=IMSVS.PROCLIB(DFSACBCP),DISP=SHR
//SYSUT3     DD  DSN=&SYSUT3,DISP=(,DELETE),UNIT=SYSDA,
//           SPACE=(80,(150,50))
//SYSUT4     DD  DSN=&SYSUT4,DISP=(,DELETE),UNIT=SYSDA,
//           SPACE=(256,(20,10)),DCB=KEYLEN=8
//SYSIN      DD  *
     DELETE PSB=PSB1
     BUILD DBD=DBD4
/*
```

CHAPTER 4. DYNAMIC ALLOCATION MACRO (DFSMDA)

OVERVIEW

For the IMS/VS DC feature running on OS/VS2 (MVS), the Dynamic Allocation macro (DFSMDA) builds a parameter list on IMSVS.RESLIB naming IMS/VS data sets that can participate in dynamic allocation and deallocation. The use of DFSMDA requires that all specified data base data sets be cataloged, but eliminates the need to initially allocate them through control region JCL. If a data base data set is specified in DFSMDA, it will be allocated by IMS/VS either when a /START command is issued for the data base or when an IMS/VS application program is scheduled. The data set will be deallocated by the /DBR command. If a data base data set is specified in the JCL, it will be allocated during control region initialization and may be deallocated with the /DBR command and reallocated with the /START command.

Multivolume indexed sequential data sets must be cataloged in the master catalog or a private VSAM catalog using Access Method Services. See "Creating and Retrieving Indexed Sequential Data Sets"in OS/VS2 JCL Reference Manual for further information.

Because dynamic allocation cannot resolve logical relationships between DBDs, a dynamic allocation member must be defined for each DBD in a logically related data base. For example, a HIDAM data base is composed of two logically related DBDs, the index DBD (with or without overflow) and the data area DBD. Each DBD in this example must have a dynamic allocation member with the same name as the DBD.

Unless it is a dynamic allocation member, no member that has the same name as a data base should be link-edited into IMSVS.RESLIB.

If you are going to dynamically allocate a data base, all DD cards referenced in the DMB for the data base must be defined in the TYPE=DATASET, DDNAME= parameter. A data base cannot be partially allocated by JCL and partially allocated by a dynamic allocate member.

The DC Monitor data set can also participate in dynamic allocation and deallocation, if named in the DFSMDA macro, and the data set is on tape. The DC Monitor data set will be allocated when it is started with the /TRACE ON command and deallocated when it is stopped with the /TRACE OFF command. It need not be initially allocated through JCL. Also, it must not be cataloged, regardless of the allocation method.

If the multiple DEDB area data set facility is used, it is recommended that all data sets belonging to that area be registered in DFSMDA. The specified areas will be allocated by IMS/VS either when a /START command is issued for the area or when an application program attempts to use the area. The area will be deallocated by /STOP AREA. Multiple areas can be deallocated by /STOP ADS.

In MVS, online log data sets (OLDS) and system log data sets (SLDS) can be dynamically allocated if named in the DFSMDA macro. The DFSMDA macro must be defined to permit SLDS input to IMS/VS restart in MVS.

See the /START, /STOP, and /DBR commands in IMS/VS Operator's Reference Manual for descriptions of how the data sets specified in the DFSMDA macro are treated by these commands.

The input to the DFSMDA macro consists of statements as explained in "Macro Statements" later in this chapter.

The output from the DFSMDA macro consists of text decks and linkage editor control cards that will be used to create load modules in IMSVS.RESLIB.

The IMSVS.RESLIB load modules for dynamic allocation can be changed simply by regenerating the parameter list with new parameters.

With CICS/VS, the DFSMDA macro can only be used for allocation of RECON data sets.

JCL REQUIREMENTS

The dynamic allocation macro statements are supplied as input to the IMSDALOC procedure and executed as an OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define the input and output data sets are required.

EXEC
This statement should be in this form:

```
IMSDALOC
```

See "Member Name IMSDALOC" in IMS/VS System Programming Reference Manual for a description of the contents of this procedure.

SYSIN DD
Defines the input data set containing the DFSMDA macro statements.

MACRO STATEMENTS

The DFSMDA macro is coded as an OS/VS macro: A statement label is optional, the macro "DFSMDA" is coded after one or more blanks, and additional parameters are separated by blanks. OS/VS continuation rules apply.

The DFSMDA macro has several statement types (as indicated by the TYPE= parameter), each of which uses different additional parameters. Code one TYPE=INITIAL statement to start the parameter list build, then as many TYPE=DATABASE, TYPE=DATASET, and TYPE=FPDEDB statements as necessary, then a TYPE=DFSDCMON if the DC Monitor data set is to be included, and finally a TYPE=FINAL to end the list. Explanations of all DFSMDA statement types follow. The maximum number of TYPE=DATABASE statements is 250.

The TYPE=INITIAL Statement

This statement indicates the start of a parameter list build and is required. No other parameters are valid on a TYPE=INITIAL statement. The format of this statement is:

	DFSMDA	TYPE=INITIAL

```
DFSMDA TYPE=INITIAL
```

The TYPE=DATABASE Statement

This statement specifies the start of the definition for a data base to participate in dynamic allocation and deallocation: one or more TYPE=DATASET statements should follow. (This statement is not used for a Fast Path DEDB area.) The format of the statement is:

	DFSMDA	TYPE=DATABASE, DBNAME=databasename

where:

DBNAME=
Specifies the DBD name of a data base whose data sets are to be dynamically allocated. This name is used as a member name in IMSVS.RESLIB to identify this data base parameter list. Care should be taken to ensure this name does not conflict with already existing members in IMSVS.RESLIB. This includes, but is not limited to, IMS/VS modules and user-supplied exit routines.

The TYPE=FPDEDB Statement

This statement defines an area within a Fast Path Data Entry Data Base (DEDB). One TYPE=FPDEDB statement is required for each area to be specified. The format of this statement is:

	DFSMDA	TYPE=FPDEDB [,DBNAME=areaname]

where:

DBNAME=
Specifies the DBD name of the DEDB in which the specified area resides. This parameter is optional, and is used for documentation purposes only. For DEDB areas, the IMSVS.RESLIB parameter list is not named with the data base name, but rather with the area's ddname.

The TYPE=DATASET Statement

This statement defines either a data set within the data base specified in the previous TYPE=DATABASE statement or a Fast Path DEDB area. One complete TYPE=DATASET is used for each data set or area data set defined. Every data set within a data base to be dynamically allocated and deallocated must be named in a TYPE=DATASET statement. When defining Fast Path DEDB areas, a TYPE=FPDEDB statement must precede each TYPE=DATASET statement. The format of this statement is:

	DFSMDA	TYPE=DATASET, DSNAME=datasetname, DDNAME=ddname ,DISP={OLD / SHR} ,INDEX={NINCORE / INCORE}

where:

DSNAME=
Specifies the name of the data set. The name may be any combination of simple and compound names valid in JCL, except the name cannot contain special characters.

DDNAME=
Specifies the name of the DD statement defining this data set. This name is the same as that used in the DATASET or AREA statement of the DBDGEN.

For multiple ADSs, this name is the same as the ADDN name registered in the ADS RECON data set.

DISP=
Specifies the disposition of this data set when allocated. The default is OLD.

INDEX=
Specifies, for an ISAM data set only, whether the high level index is to reside in virtual storage. The default is for the index not to reside in virtual storage (INDEX=NINCORE).

The TYPE=DFSDCMON Statement

This statement defines the dynamic allocation parameter list for the DC Monitor data set. The format of this statement is:

	DFSMDA	TYPE=DFSDCMON, DDNAME=IMSMON, DSNAME=datasetname, UNIT={TAPE | unit}, BUFNO={4 | n}, BLKSIZE={4096 | nnnn}

where:

DSNAME=
Specifies the name of the data set, which must not be cataloged. The name may be any combination of simple and compound names valid in JCL, except the name cannot contain special characters.

UNIT=
Specifies the unit for the DC Monitor data set. The value of UNIT= can be the name of any tape device valid to the installation. The default is TAPE.

BUFNO=
Specifies the number of buffers for the DC Monitor data set. Valid numbers are from 2 to 9, inclusive; the default is 4.

BLKSIZE=
Specifies the block size for the DC Monitor data set. The default is 4096. This value will be replaced with a larger value at initialization time if the value is not large enough for the DC monitor log records. In addition, the block size will be rounded up to a doubleword boundary.

The TYPE=RECON Statement

This statement defines the dynamic allocation parameter list for data base recovery control (DBRC).

The format of this statement is:

	DFSMDA	TYPE=RECON, DSNAME=datasetname, DDNAME=RECONn, WAIT=YES/NO

where:

DSNAME=
Specifies the name of the data set. The name may be any combination of simple and compound names valid in JCL, except that it may not contain special characters.

DDNAME=
Specifies the name of the DD statement defining this data set. This name must be RECON1, RECON2, or RECON3.

WAIT=
If YES is specified on any of the TYPE=RECON statements (RECON1, RECON2, RECON3), a wait will be issued for any of the RECONs found to be offline during DBRC initialization. WAIT=NO is the default. Omitting the WAIT= parameter or specifying WAIT=NO will cause dynamic allocation to fail in the event that a RECON data set is offline during DBRC initialization.

The TYPE=OLDS Statement

This statement defines the dynamic allocation parameter list for the online log data set (OLDS).

There must be as many DFSMDA macros as there are OLDS data sets.

The DFSMDA member name must be the same as the DDNAME of the OLDS it defines.

The format of this statement is:

	DFSMDA	TYPE=OLDS, DSNAME=datasetname, DDNAME=DFSOLxnn

where:

DSNAME=
Specifies the name of the data set. The name may be any combination of simple and compound names valid in JCL, except that it may not contain special characters.

DDNAME=
Specifies the OLDSs to be allocated. If the OLDSs are dual, there must be a pair of macros, one with the ddname of the primary OLDS and the other with the ddname of the secondary OLDS (for example, DFSOLP01 and DFSOLS01). The data set must be cataloged. Substitute P for x when declaring a primary data set. Substitute S for x when declaring a secondary data set. Values from 00 through 99 may be specified for nn.

The TYPE=SLDS Statement

This statement defines the dynamic allocation parameter list for the system log data set (SLDS). SLDSs are dynamically allocated when required as input for restart. A single DFSMDA member with name IMSLOGR must be created to specify the UNIT information required for allocation. All SLDSs to be used as input to restart must reside on the same device type.

The format of this statement is:

	DFSMDA	TYPE=SLDS, UNIT=device type, DDNAME=IMSLOGR

where:

UNIT=
Specifies the device required for allocation. All SLDSs used as input for restart must reside on the same device type. This applies to both the primary and secondary data sets when dual logging is used. The device type can be tape or DASD.

DDNAME=IMSLOGR
This is the required value for DDNAME.

The TYPE=FINAL Statement

This statement indicates the end of a parameter list build and is required. No other parameters are valid on a TYPE=FINAL statement. The format of this statement is:

	DFSMDA	TYPE=FINAL

ERROR MESSAGES

See Chapter 7, "Dynamic Allocation (IMSDALOC) Messages," in IMS/VS Messages and Codes Reference Manual for a complete description of the IMS/VS messages that indicate DFSMDA errors.

EXAMPLES

EXAMPLE 1

Example 1 shows the JCL and macro statements to specify 3 data bases and the DC Monitor data set to participate in dynamic allocation and deallocation.

```
//DALOC JOB 1,1,MSGLEVEL=(1,1)
//STEP EXEC IMSDALOC
//SYSIN DD *
    DFSMDA  TYPE=INITIAL
    DFSMDA  TYPE=DATABASE,DBNAME=DI41M101
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.M1I3I1,DDNAME=M1I3I1
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.M1I301,DDNAME=M1I301
    DFSMDA  TYPE=DATABASE,DBNAME=DX41SK03
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.DB5H111,DDNAME=DXSK0301,          X
            DISP=SHR,INDEX=INCORE
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.DB5H222,DDNAME=DXSK0302,          X
            DISP=SHR,INDEX=INCORE
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.DB5H333,DDNAME=DHSK0301,          X
            DISP=SHR,INDEX=NINCORE
    DFSMDA  TYPE=DATABASE,DBNAME=DH41SK03
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.DB4D111,DDNAME=DDSK0101,          X
            DISP=SHR
    DFSMDA  TYPE=DATASET,DSNAME=IMSQA.DB4D222,DDNAME=DDSK0102,          X
            DISP=SHR
    DFSMDA  TYPE=DFSDCMON,DDNAME=IMSMON,DSNAME=I115T237.IMSMON
    DFSMDA  TYPE=FINAL
    END
/*
```

EXAMPLE 2

Example 2 shows the JCL and macro statements to specify 3 Fast Path DEDB areas to participate in dynamic allocation and deallocation.

```
//DALOC JOB   1,1,MSGLEVEL=(1,1)
//STEP  EXEC  IMSDALOC
//SYSIN DD   *
       DFSMDA TYPE=INITIAL
       DFSMDA TYPE=FPDEDB,DD41SK02
       DFSMDA TYPE=DATASET,DSNAME=DB9AREA0,DDNAME=DB9AREA0
       DFSMDA TYPE=FPDEDB
       DFSMDA TYPE=DATASET,DSNAME=DB22AR0,DDNAME=DB22AR0,             X
              DISP=SHR
       DFSMDA TYPE=FPDEDB,DEDBJN22
       DFSMDA TYPE=DATASET,DSNAME=DB22AR1,DDNAME=DB22AR1,             X
              DISP=OLD
       DFSMDA TYPE=FINAL
       END
/*
```

EXAMPLE 3

In MVS, SLDSs are dynamically allocated when required as input for restart. Example 3 shows the JCL and macro statements for SLDS to participate in dynamic allocation and reallocation.

```
//ASSMBLY  EXEC   IMSDALOC
//ASSEM.SYSLIB DD DSN=RNC.MACLIB,DISP=SHR
//             DD DSN=I13OTS13.MACLIB, DISP=SHR
//             DD DSN=SYS1.MACLIB,DISP=SHR
//ASSEM.SYSIN  DD*
        DFSMDA  TYPE=INITIAL
        DFSMDA  TYPE=SLDS,UNIT=3330-1,DDNAME=IMSLOGR
        DFSMDA  TYPE=FINAL
        END
/*
//LNKEDT.SYSLMOD DD DSN=IMSQA.TNUC2,
//                  DISP=SHR,VOL=SER=USER01,
//                  UNIT=SYSDA
```

PART 2. ONLINE CHANGE PROCESSING

Part 2 has one chapter, Chapter 5, "Online Change Utility," which describes the utility provided to copy one partitioned data set to another partitioned data set.

CHAPTER 5. ONLINE CHANGE UTILITY

Changes to some IMS/VS system resources can be made without stopping the system. This chapter describes the utility used to bring these changes online. This utility is also used in the initial installation of IMS/VS. This utility does not support CICS/VS.

Three copies of the following libraries are required for online change:

ACBLIB Data base and program descriptors such as DMBs and PSBs

FORMAT Control blocks produced by the MFS language utility and service utility

MATRIX Control blocks for the system security tables

MODBLKS A subset of the control blocks for the resources to be modified

One copy is used exclusively for offline functions. This library has no suffix and is called the staging library. The other two copies have a suffix of A or B. Only one of these libraries is used by the IMS/VS online system at any one time. This is referred to as the active library. The other is called the inactive library.

The Online Change utility can copy the contents of the staging library to the inactive library based on the information in the status data set, MODSTAT. Issuing the Online Change Command, /MODIFY, causes the inactive library to become the active library.

The Online Change utility can copy the contents of the staging libraries to the active libraries during the installation of IMS/VS, prior to the first cold start. To do this, the A parameter for the output ddname must be specified when invoking the utility, because the initial contents of IMSVS.MODSTAT will specify the active libraries. See IMS/VS Installation Guide for more information on initializing IMSVS.MODSTAT.

The utility performs an OS/VS ENQ on the data set name/volume serial number with a scope of SYSTEMS on the input and output data sets. IMS/VS online performs the same OS/VS ENQ on the active libraries.

RESTRICTIONS

If any of the ACBLIB, FORMAT, or MODBLKS libraries are shared among IMS/VS systems, you must ensure that all systems are using the same libraries during execution of this utility. Additions or changes that require new IMS/VS modules to be added to the IMSVS.RESLIB data set cannot be done using the Online Change utility. Also, you cannot add MSDBs and DEDBs using this utility.

JCL REQUIREMENTS

The Online Change utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

This utility operates by clearing the target library data set and then invoking IEBCOPY to move the source library contents. If IEBCOPY abends because of insufficient space, the contents of the target library are unpredictable. To avoid this, allocate equal amounts of space for source and target libraries.

EXEC

The EXEC statement determines which copy will be made and which data sets will be used for input and output. The format of this statement is:

```
PGM=DFSUOCU0,PARM=(copy_type,input_library,target_library)
```

copy_type

Specifies the library to be copied. It can be the ACB, FORMAT, MATRIX, or MODBLKS library.

input_library

Defines the library ddnames to be used as input.

Parameter	Meaning
S	IMS staging library (IMSACB, FORMAT MATRIX, or MODBLKS)
I	User input library (IMSACBI, FORMATI, or MODBLKSI)

The use of the I parameter will permit an input library other than the staging library.

target_library

Defines the library ddnames to be used for output.

Parameter	Meaning
A	IMS A library (IMSACBA, FORMATA, MATRIXA, or MODBLKSA)
B	IMS B library (IMSACBB, FORMATB, MATRIXB, or MODBLKSB)
O	User output library (IMSACBO, FORMATO, MATRIXO, or MODBLKSO)
U	Target library (inactive) determined by the utility, using the MODSTAT data set. The target will be the library not currently in use by the IMS/VS online system (inactive).

During online operation, avoid using the A or B parameter for the output library because an incorrect choice could cause IMS/VS to overlay the active library.

The use of the O parameter will permit a target data set other than the active or inactive data set.

Use of the U parameter provides automatic protection against overlaying an active library, and is recommended.

IMSACB DD
IMSACBA DD
IMSACBB DD
Defines the staging, active, or inactive ACBLIB.

FORMAT DD
FORMATA DD
FORMATB DD
Defines the staging, active, or inactive MFS format library.

MATRIX DD
MATRIXA DD
MATRIXB DD
Defines the staging, active, or inactive library containing the security tables.

MODBLKS DD
MODBLKSA DD
MODBLKSB DD
Defines the staging, active, or inactive system definition library.

MODSTAT DD
Defines the inactive or active data sets that online IMS/VS should use at initialization.

SYSUDUMP DD
Defines the dump data set for this program. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream.

SYSPRINT DD
Defines the message output data set. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream. This DD statement must always be included.

SYSUT3 DD
Defines a work data set that is required.

SYSUT4 DD
Same function as SYSUT3.

COPYCTL DD
Defines a copy control data set to be built prior to calling IEBCOPY.

Example

This example shows the JCL for the procedure in IMSVS.PROCLIB, OLCUTL, which can be used to invoke the Online Change utility.

```
//        PROC  TYPE=,IN=,OUT=,SOUT=A
//S       EXEC  PGM=DFSUOCU0,PARM=(&TYPE,&IN,&OUT)
//MODBLKS  DD   DSN=IMSVS.MODBLKS,DISP=SHR
//MODBLKSA DD   DSN=IMSVS.MODBLKSA,DISP=SHR
//MODBLKSB DD   DSN=IMSVS.MODBLKSB,DISP=SHR
//IMSACB   DD   DSN=IMSVS.ACBLIB,DISP=SHR
//IMSACBA  DD   DSN=IMSVS.ACBLIBA,DISP=SHR
//IMSACBB  DD   DSN=IMSVS.ACBLIBB,DISP=SHR
//FORMAT   DD   DSN=IMSVS.FORMAT,DISP=SHR
//FORMATA  DD   DSN=IMSVS.FORMATA,DISP=SHR
//FORMATB  DD   DSN=IMSVS.FORMATB,DISP=SHR
//MATRIX   DD   DSN=IMSVS.MATRIX,DISP=SHR
//MATRIXA  DD   DSN=IMSVS.MATRIXA,DISP=SHR
//MATRIXB  DD   DSN=IMSVS.MATRIXB,DISP=SHR
//MODSTAT  DD   DSN=IMSVS.MODSTAT,DISP=SHR
//SYSUDUMP DD   SYSOUT=A
//SYSPRINT DD   SYSOUT=A
//SYSUT3   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//COPYCTL  DD   DSN=&&COPYCTL,DISP=(NEW,DELETE),
//              UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=160)
```

Note: DD cards with a suffix of 'I' or 'O' will have to be added for any user input or output data sets required (for example, IMSACBI or IMSACBO for an ACB library copy). If the automatic option (parameter 'U') is not used for determination of the target data set, the MODSTAT DD statement is not required.

PART 3. DATA BASE UTILITIES

Part 3 has three chapters that describe the utilities used for reorganizing and recovering data bases and the facility that allows a user to implement these utilities in a specific manner.

Chapter 6, "Data Base Reorganization/Load Processing," describes the utilities used for scanning and reorganizing data bases. Included are control statement requirements and examples; output messages provided by the HISAM Unload/Reload utilities and the HD Unload/Reload utilities; and samples of commonly used data base operations that can be performed by many of these utilities.

Chapter 7, "Data Base Recovery Utilities," describes the utilities used for rapid recovery of a data base data set rendered unusable because of read and/or write errors. Control statements and examples are included.

Chapter 8, "Utility Control Facility," describes the facility that allows a user to accomplish most data base utility operations and maintenance functions under the control of one step and in one scheduling. Control statements required to execute UCF and examples are included.

CHAPTER 6. DATA BASE REORGANIZATION/LOAD PROCESSING

This chapter describes the ten IMS/VS utility programs that are used for reorganization/load processing of data bases. They are:

1. HISAM Reorganization Unload (DFSURUL0)
2. HISAM Reorganization Reload (DFSURRL0)
3. HD Reorganization Unload (DFSURGU0)
4. HD Reorganization Reload (DFSURGL0)
5. Data Base Surveyor Utility Feature (DFSPRSUR)
6. Partial Data Base Reorganization Utility (DFSPRCT1)
7. Data Base Prereorganization (DFSURPR0)
8. Data Base Scan (DFSURGS0)
9. Data Base Prefix Resolution (DFSURG10)
10. Data Base Prefix Update (DFSURGP0)

An overview is presented initially to provide a description of each utility.

These utilities are of two types—physical reorganization utilities and logical relationship resolution utilities.

Following the overview of each of these utilities, these topics are also presented:

- Initial Data Base Load Considerations
- Data Base Physical Reorganization Considerations
- Data Base Reorganization/Load Flowchart
- Loading a Secondary Index

The remainder of the chapter describes each utility program separately and in greater detail (including JCL requirements and numerous examples).

The messages and statistics provided by the HISAM Unload/Reload and HD Unload/Reload utilities are contained in this chapter along with sample procedures and JCL examples of commonly used data base operations that can be performed by a number of the utilities described.

Note: When unloading and reloading a DEDB containing subset pointers, IMS/VS does not automatically retain the position of the subset pointers.

Information on reorganizing data bases that are shared is contained in IMS/VS System Administration Guide.

PHYSICAL REORGANIZATION UTILITY PROGRAMS

There are six physical reorganization utility programs: (1) HISAM Reorganization Unload, (2) HISAM Reorganization Reload, (3) HD Reorganization Unload, (4) HD Reorganization Reload, (5) Data Base Surveyor feature, and (6) Partial Data Base Reorganization. A brief description of each utility is provided in the following sections.

HISAM Reorganization Unload Utility

This utility can be used to: (a) unload a HISAM or HIDAM primary index data base to a QSAM formatted unload data set, and (b) create or reorganize secondary indexes from the work data sets created during initial load or reorganization. The output from this utility can be used as input to either the Data Base Recovery utility or the HISAM Reload utility. (The Data Base Recovery utility is described in the "Data Base Recovery Utilities" chapter of this manual.)

HISAM Reorganization Reload Utility

This utility can be used to reload a HISAM or HIDAM primary index data base from a QSAM formatted data set created by the HISAM Reorganization Unload utility. Item (b) above for the HISAM Unload utility applies equally to the HISAM Reload utility.

Reorganization of HISAM data bases is significantly faster using the HISAM Unload/Reload utilities instead of the HD Unload/Reload utilities.

The HISAM Unload/Reload utilities cannot be used to make structural changes other than changes to logical record length and block size. The HD Unload/Reload utilities, however, allow structural changes to be made to a data base.

Note: The HISAM Reorganization Unload/Reload utilities cannot be used to reorganize HISAM data bases that are indexed by a secondary index or that contain segments with direct address pointers used in logical relationships. The HD Reorganization Unload/Reload utilities must be used instead.

HD Reorganization Unload Utility

This utility can be used to unload an HDAM, HIDAM, or HISAM data base to a QSAM formatted data set.

Note that this utility cannot be used to reorganize a HISAM data base that is an index to a HIDAM data base. Instead, the HISAM Reorganization Unload utility must be used.

Note: The DFSURWF1 DD card should not be included with this utility program.

HD Reorganization Reload Utility

This utility can be used to: (a) reload an HDAM, HIDAM, or HISAM data base from a QSAM formatted data set created by the HD Unload utility, and (b) create work data sets (if the data base that is reloaded includes logical relationships or secondary indexes) that are used as input to the logical relationship resolution utilities.

Note that this utility cannot be used to reorganize a HISAM data base that is an index to a HIDAM data base. Instead, the HISAM Reorganization Reload utility must be used.

Use of the HD Unload/Reload utilities in making structural changes to a data base is discussed later in this chapter under "Restrictions" for the HD Reorganization Unload utility.

Data Base Surveyor Utility Feature

This utility feature reports on the organization of a data base as a preliminary to performing a partial data base reorganization. The Surveyor utility feature reports on chain lengths and free space to point out portions of a data base that need reorganization and areas that can receive reloaded records.

Partial Data Base Reorganization Utility

This utility reorganizes selected parts of an HDAM or HIDAM data base. Two steps are required to execute this utility. The first step performs prereorganization initialization, such as building control tables and work records. The second step performs the reorganization functions, including unload, reload, scan, sort, and prefix resolution.

LOGICAL RELATIONSHIP RESOLUTION UTILITY PROGRAMS

There are four logical relationship resolution utility programs: (1) Data Base Prereorganization, (2) Data Base Scan, (3) Data Base Prefix Resolution, and (4) Data Base Prefix Update.

A brief description of each logical relationship resolution utility is provided in the following sections.

Data Base Prereorganization Utility

This utility creates a control data set that is used by the other logical relationship resolution utilities. It also indicates which data bases and segments, if any, must be scanned by the Data Base Scan utility. If secondary indexes exist when initially loading or reorganizing an indexed data base, the Prereorganization utility must be executed against the indexed data base to create a control data set used in the creation of a secondary index.

Data Base Scan Utility

This utility scans any nonreorganized data bases not being loaded or reorganized that contain logical relationships that are affected by loading and/or reorganizing other data bases. It also generates output work data sets that will be used by the Data Base Prefix Resolution utility.

Data Base Prefix Resolution Utility

This utility combines and sorts all work data sets generated by the HD Reload utility, Data Base Scan utility, or by initial loading of a data base. This utility generates an output work data set that contains the prefix information needed to complete the loading and/or reorganization of data bases that contain logical relationships. If secondary indexes are present, a separate output data set is also generated.

Data Base Prefix Update Utility

This utility uses the output data set generated by the Data Base Prefix Resolution utility to update the prefix of each segment whose prefix information is affected by a data base load and/or reorganization.

INITIAL DATA BASE LOAD CONSIDERATIONS

The loading of a physical data base is the responsibility of the user. The user must provide a program for that purpose and appropriate PSBs to indicate that the data base is being loaded. Refer to the "PSB Generation" chapter in this manual for information on the PSB operands required for initial load.

The load program JCL provided by the user must include a DD statement with the ddname of DFSURWF1. Data created by IMS/VS as a result of the user's DL/I ISRT calls is placed into the DFSURWF1 data set during initial load.

The DFSURWF1 data set is used to resolve logical or secondary index relationships. The DCB parameters for this DD statement must include RECFM=VB and the BLKSIZE must be the same for each step of the process. The recommended BLKSIZE value is 1008. A value of LRECL=900 is recommended, but a smaller value (as small as 300) can be used if no secondary indexes are present. This work file cannot be a VSAM data set, although the user may have a VSAM data base. If the data base that is to be loaded contains logical relationships or is to be indexed by a secondary index, the logical relationship utilities must be executed. For further information on work file space requirements, see Appendix C, "Initial Load Program" in IMS/VS Data Base Administration Guide. See also the sample DFSLRLOD procedure at the end of this chapter.

The following additional restrictions and considerations apply:

1. When initially loading segments involved in logical relationships, a number of rules must be observed.

   - If the segment is a logical parent, the segment must be loaded by a separate DL/I insert call.
   - Prior to inserting a logical child segment, both the logical parent's concatenated key and the logical child intersection data (if any) must be placed in the user I/O area.
   - Data bases that are related through logical relationships can be loaded independently of each other. Before running the logical relationship resolution utilities, however, the user must ensure that for any logical child segment loaded, the corresponding logical parent segment is also (or has already been) loaded.
   - For a virtually-paired-bidirectional logical relationship, the real logical child segment must be loaded into the data base of its physical parent.
   - If two segment types are physically paired, the paired segments must be inserted at initial data base load.

2. When a data base indexed by a secondary index is initially loaded, several of the reorganization and logical relationship utilities must be executed in addition to the user's load program. See Figure 46 on page 160 for the utilities that must be executed and the order in which they must be run.

3. DFSURWF1 work data sets created during initial loads, scans, and reloads must be concatenated as input to the Prefix Resolution utility.

4. The use of nonunique sequence fields for one or more segments in a data base permits the occurrence of nonunique concatenated keys for one or more segment types. If the user defines nonunique sequence fields for segments in a data base, and if occurrences of a given segment type are allowed to have the same concatenated key, certain ambiguities can arise during processing of the data base. In particular, for a segment type that has multiple occurrences of the same concatenated key and that is also involved in a logical relationship, the following must be observed:

   - At initial data base load time, if logical parent segments with nonunique concatenated keys exist in a data base, the logical relationship resolution utilities attach all logical child segments that contain the same concatenated key to the first logical parent segment in a data base that has that concatenated key. Thus, one

of the logical parents with a nonunique concatenated key will own all logical children pointing to that concatenated key, while other logical parents with that concatenated key will own no logical children.

- If a user has some mechanism other than a concatenated key for distinguishing logical parents with nonunique concatenated keys, he may insert the logical children at other than initial data base load time. In this manner, logical parents with nonunique concatenated keys can each own logical children. However, if logical children that have a logical parent with a nonunique concatenated key do not point to that logical parent with a logical parent pointer, the results provided by the logical relationship resolution utilities at data base reorganization time are undefined. It should be noted that if logical parents with nonunique concatenated keys do not actually occur, the logical relationship resolution utilities will correctly maintain logical relationships at data base reorganization time.

DATA BASE PHYSICAL REORGANIZATION CONSIDERATIONS

Because the reorganization of a large data base often requires considerable time, it is important to determine when reorganization is actually indicated. A number of criteria can be used to determine when it is expedient to reorganize a data base.

A data base should be reorganized periodically, for instance, when segments within a data base record are no longer physically adjacent in auxiliary storage or when deleted segments continue to occupy storage space. Reorganizing the data base would significantly improve processing time.

The number of additions and deletions made to a data base is another criterion for physical reorganization of a data base. The number of additions made to a data base is shown on the Application Accounting Report produced from the IMS/VS Statistical Analysis utility.

The Transaction Response Report is another helpful means of determining when to reorganize a data base. For instance, when many of the segments being referenced within a data base are in the overflow data set of a HISAM data base and/or the segment chains in the overflow data set become long, response times increase because of the amount of direct access arm movement required to respond to calls. This type of information would be reflected on the Transaction Response Report.

For data bases that use ISAM/OSAM as the access method, use of the IEHLIST utility program (see LISTVTOC control statement) can also be instrumental in determining when to reorganize a data base. Care must be taken, for instance, to ensure that there is always unused space in an overflow data set so that additions can be made at any time. The IEHLIST program would be valuable in this instance, because it lists the data set control blocks to monitor the amount of unused space available for overflow data set additions. For data bases that use VSAM as the access method, use of the OS/VS Access Method Services (see LISTCAT command) can provide the same type of information.

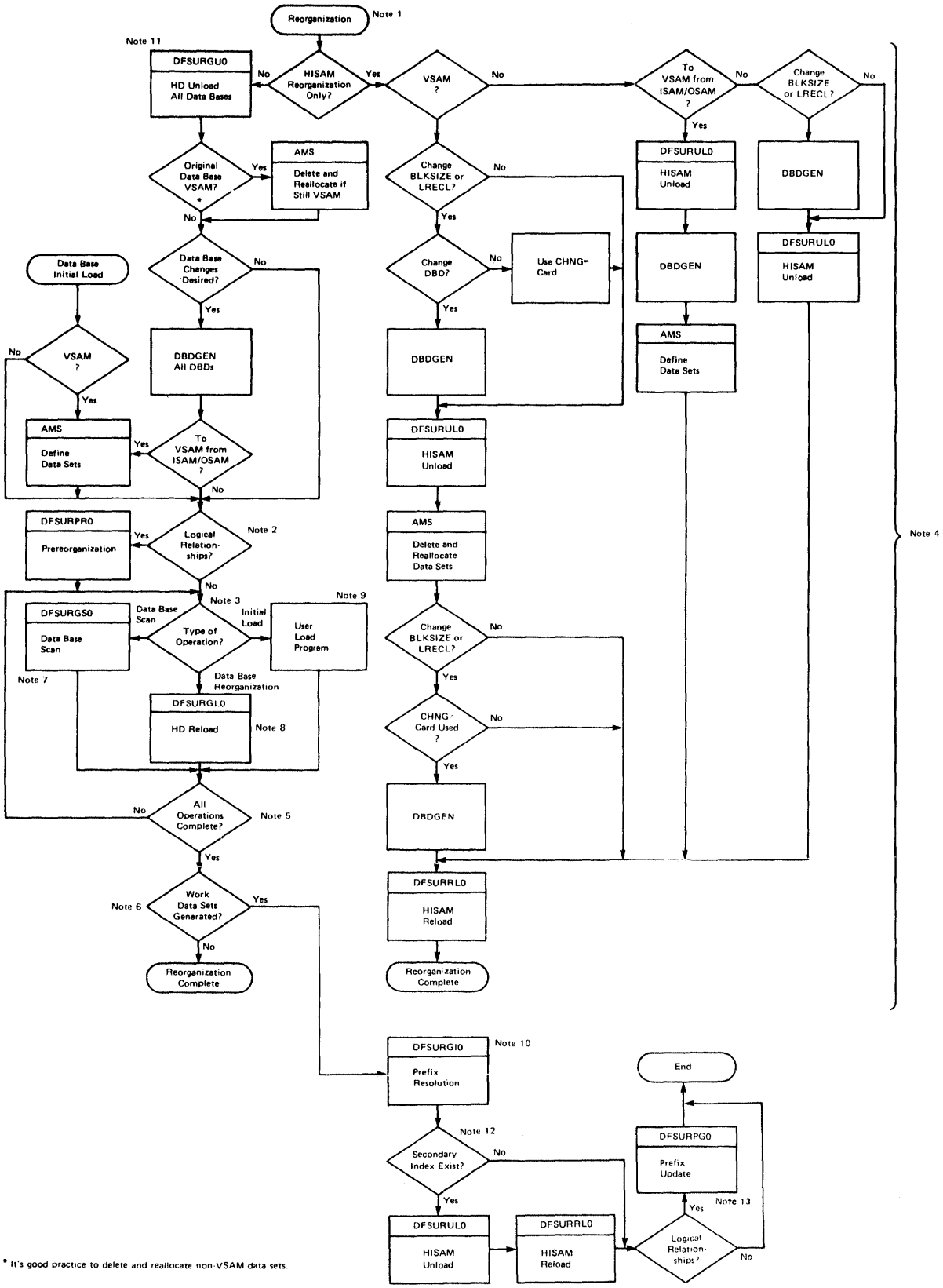
The Data Base Surveyor utility feature can aid the user in determining whether to reorganize all or part of an HDAM or HIDAM data base. The output from this utility can point out portions of the data base that have excessive chain lengths due to significant insert and delete processing. The output also reports on contiguous free space that could be used as target areas during a partial data base reorganization.

The Partial Data Base Reorganization utility can be used when only a part of an HDAM or HIDAM data base is poorly organized. Sufficiently large free areas, including the areas freed during the unload phase of execution, must exist within the data base to receive the reorganized records.

If a VSAM data set has been defined in a user catalog, it is also necessary to identify the user catalog by means of either a JOBCAT or a STEPCAT DD statement. See OS/VS Virtual Storage Access Method Programmer's Guide for more details.

DATA BASE REORGANIZATION/LOAD FLOWCHART

Figure 44 on page 156 and the accompanying notes explain how the user may determine the necessary utility programs required for reorganization and/or load processing of a data base.



* It's good practice to delete and reallocate non-VSAM data sets.

Figure 44. Data Base Reorganization/Load Flowchart

Notes:

1. The data base reorganization/load processing utilities (that is, the HISAM Unload/Reload, HD Unload/Reload, Prefix Resolution and Prefix Update utilities) can be used to operate on one or more data bases concurrently. For example, one or more existing data bases can be reorganized at the same time that other data bases are being initially loaded. Any or all of the data bases being operated on can be logically interrelated. A data base operation is defined to be an initial data base load, a data base unload/reload (reorganization), or a data base scan.

2. If one or more segments in any or all of the data bases being operated upon is involved in either a logical relationship or a secondary index relationship, the YES branch must be taken. The Prereorganization utility can also be used to determine which data base operations must be performed.

3. Based upon the information given to it on control statements, the data base Prereorganization utility provides a list of data bases that must be initially loaded, reorganized, or scanned. The number and sequence of data bases specified on the prereorganization control statement must not be changed between reload and prefix resolution.

4. This area of the flowchart must be followed once for each data base to be operated upon, whether the operation consists of an initial load, reorganization, or scan. The operations may be done for all data bases concurrently, or one data base at a time may be operated upon. It should be noted that if the various data base operations are performed sequentially, work data set storage space can be saved and processing efficiency increased if DISP=(MOD,KEEP) is specified for the DFSURWF1 DD statement associated with each data base operation. The work data set attributes for the data base initial load, reorganization, and scan programs must be identical.

   When using the HD Reload utility, it is necessary to first do all unloads and scans of logically related data bases if logical parent concatenated keys are defined as virtual in the logical child.

5. The user must ensure that all operations indicated by the Prereorganization utility (if it was executed) are completed prior to taking the YES branch.

6. If any work data sets were generated during any of the data base operations that were executed by the user, the YES branch must be taken. It should be noted that the presence of a logical relationship in a data base does not guarantee that work data sets will be generated during a data base operation. The reorganization/load processing utilities determine the need for work data sets dynamically, based upon the actual segments presented during a data base operation. If any segments that participate in a logical relationship are loaded, work data sets will be generated and the YES branch must be taken.

   If for any specific data base operation no work data set was generated for the data base, processing of that data base is complete, and it is ready for use.

   When a HIDAM data base is initially loaded or reorganized, its primary index will be generated at data base load time.

7. It is required that the DB Scan utility be run before a data base is unloaded when logical parent concatenated keys are defined as virtual in the logical child data base to be unloaded.

This program should be executed against each data base listed in the output of the Prereorganization utility. A work data set can be generated for each data base scanned by this utility. Data bases to be scanned are listed after the characters "DBS=" in one or more output messages of the Prereorganization utility.

8. The HD Reorganization Reload utility may cause the generation of a work data set to be later used by the Prefix Resolution utility. Data bases to be reorganized using the HD Unload/Reload utilities are listed after the character "DBR=" in one or more output messages of the Prereorganization utility.

9. The user-provided initial data base load program may automatically cause the generation of a work data set to be later used by the Prefix Resolution utility. The user need not add code to the initial load program to accomplish work data set generation. This will be done automatically by IMS/VS through the user program issuing ISRT requests. The user must, however, provide a DD statement for this data set along with the other JCL necessary to execute the initial load program. Data bases to be initially loaded are listed after the characters DBIL= in one or more output messages of the Prereorganization utility.

10. The data base Prefix Resolution utility combines the workfile output from the Data Base Scan utility, the HD Reorganization Reload utility, and the user's initial data base load execution to create an output data set to be used by the Prefix Update utility. The Prefix Update utility then completes all logical relationships defined for the data bases that were operated upon.

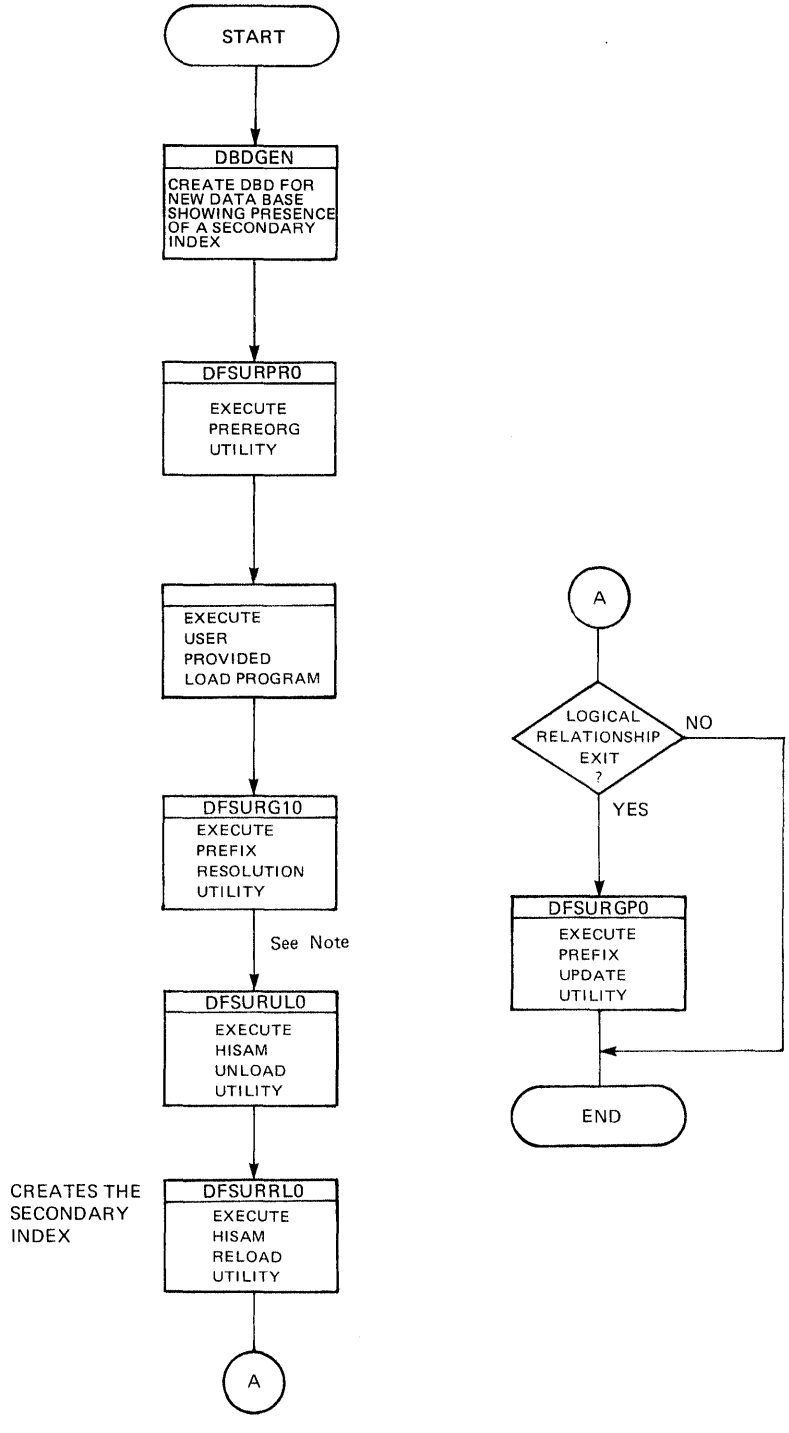
11. This path must be taken for HISAM data bases with logical relationships or if structural changes are required (for example, HISAM to HDAM, pointer changes, additional segments, or adding a secondary index).

12. If a secondary index is to be created or if two secondary indexes are to be combined, the HISAM Unload/Reload utilities must be run. After the HISAM Unload/Reload utilities are run, if there are logical relationships in the data base, the Prefix Update utility must be executed before the reorganization or load process is considered to be complete.

13. For information on scratching and allocating ISAM/OSAM data sets, see "Allocation for OSAM Data Sets Using IEFBR14" under "Designing the IMS/VS Online System" in IMS/VS System Administration Guide.

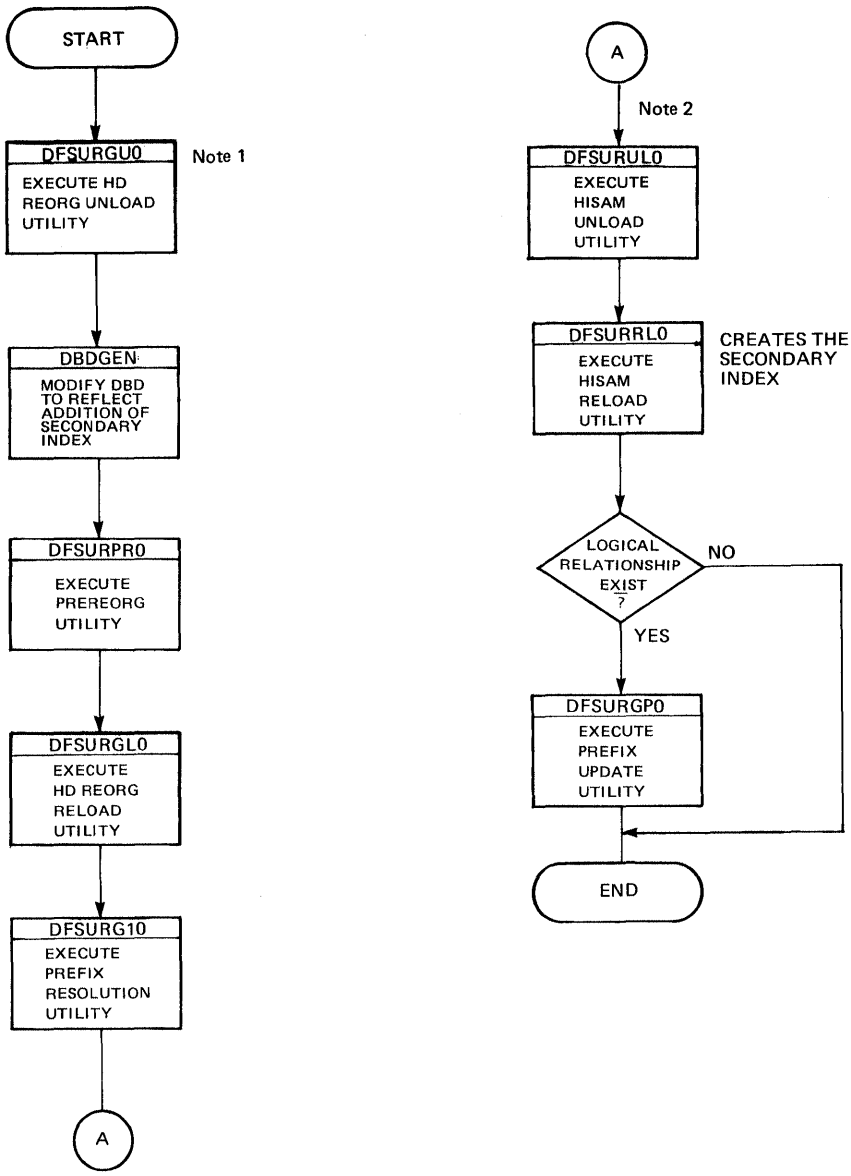
LOADING A SECONDARY INDEX

Figure 45 on page 159 and Figure 46 on page 160 depict the utility flow of two separate cases of loading a secondary index.



Note: The input data set that contains the secondary index information is created in the Prefix Resolution utility with the ddname of DFSURIDX. It is used to create a new index, to merge the new information into an existing shared secondary index, or to replace a member in an existing shared secondary index. A status code of GB or GE will be issued when an attempt is made to read the secondary index, if no segments were inserted when it was loaded.

Figure 45. Initial Load of a Data Base with Secondary Indexes



Notes:

1. Adding a secondary index to an existing data base is considered a structural modification to the data base. For this reason, the data base must be reorganized, and the HD Reorganization Unload/Reload utilities must be used even with HISAM data bases.

2. The input data set that contains the secondary index information is created in the Prefix Resolution utility with the ddname of DFSURIDX. It is used to create a new index, to merge the new information into an existing shared secondary index, or to replace a member in an existing shared secondary index.

Figure 46. Adding a Secondary Index to an Existing Data Base or Reorganizing a Data Base That Has a Secondary Index

HISAM REORGANIZATION UNLOAD UTILITY (DFSURUL0)

The HISAM Reorganization Unload utility provides a means of unloading a HISAM data base and creating a reorganized output that can be used as input to either the Data Base Recovery utility or the HISAM Reorganization Reload utility. In addition, this utility performs the formatting of index work data sets created by the Prefix Resolution utility to a form that can be used by the HISAM Reload utility to create a secondary index or to merge records from the index work data set with a shared secondary index, if one exists.

The output is blocked to the block size of the output device, up to a maximum of 16K bytes. Because the blocking factor is determined at execution time, standard labels must be used on all output volumes. Although performance of this utility is somewhat dependent on the number of random retrievals necessary in the OSAM data set, performance can usually be enhanced by providing additional buffers through the JCL for the ISAM input data set. Increasing the number of buffers for the output data sets beyond the default value of 2, however, will not improve performance.

Note: The functions of this utility can be performed by the Utility Control Facility. Refer to the chapter entitled "Utility Control Facility" in this publication for a description of its operation.

RESTRICTIONS

The following restrictions apply to use of the HISAM Reorganization Unload utility:

1. This utility cannot be used to unload a SHISAM data base.
2. If this utility is to be used for recovery purposes, the user must reload the data base with the HISAM Reorganization Reload utility prior to applying changes to the data base. If the user does not immediately reload but waits until recovery time, the segments will be reloaded into a different location. The logs created between unload and reload will refer to the old location and recovery is impossible.
3. This utility should not be used to unload a HISAM data base if the data base contains logical child segments that contain direct address logical parent pointers. This utility cannot be used to make structural changes or to change data base organizations. The HD Reorganization Unload/Reload utility should be used for these purposes.
4. Features of this utility allow the data base logical record length and blocking factors to be changed to create a more efficient storage organization. To change block size or record length of ISAM/OSAM formats, the DBD must be reassembled with the new logical record length and blocking factor specified prior to executing this utility. (The old specifications are taken from the data set labels and are converted to the new specifications during the unload.)
5. To change block or logical record lengths of VSAM data sets, a new DBD can be generated with new sizes and the data base must be unloaded using the new DBD. The OS/VS Access Method Services utility must then be run to delete the old data sets and to define new data sets containing the new block sizes and logical record lengths (or, rather, their logical equivalents in VSAM, that is, control interval size and LRECLs). The data bases can then be reloaded using the HISAM Reload utility. A CHANGE control statement must be included to specify control interval or record sizes.

6. The new DBD must have the same name as the old DBD. Both old and new DBDs can exist in the system if two separate libraries are used and the appropriate library is referenced at unload and reload time.

7. To convert ISAM/OSAM HISAM data sets to VSAM data sets, the data base must be unloaded using the existing DBD and a new DBD generated in which VSAM is specified as the OS/VS access method. The OS/VS Access Method Services utility must be run to define the new data sets, after which the HISAM Reload utility is run.

   - The logical record length of the VSAM data set must be equal to or greater than the ISAM/OSAM LRECL minus 2 bytes.

8. If the output data set of HISAM Unload is used to make the conversion from ISAM/OSAM to VSAM, the output data set cannot be used as input to the Data Base Recovery utility. An output image copy of the data base must be made immediately after the reload and before any changes are logged. (See the description of the Data Base Image Copy utility in this publication.)

9. For information on scratching and allocating ISAM/OSAM data sets, see "Allocation for OSAM Data Sets Using IEFBR14" in "Defining the IMS/VS Online System," in IMS/VS System Administration Guide.

10. If this utility is used to unload or reload a shared secondary index that has alias names not registered to DBRC, message DFS194W will be issued. Because alias names are not required to be registered in the RECON data set, this message can be ignored.

Figure 47 on page 163 is a flow diagram of the HISAM Reorganization Unload utility.

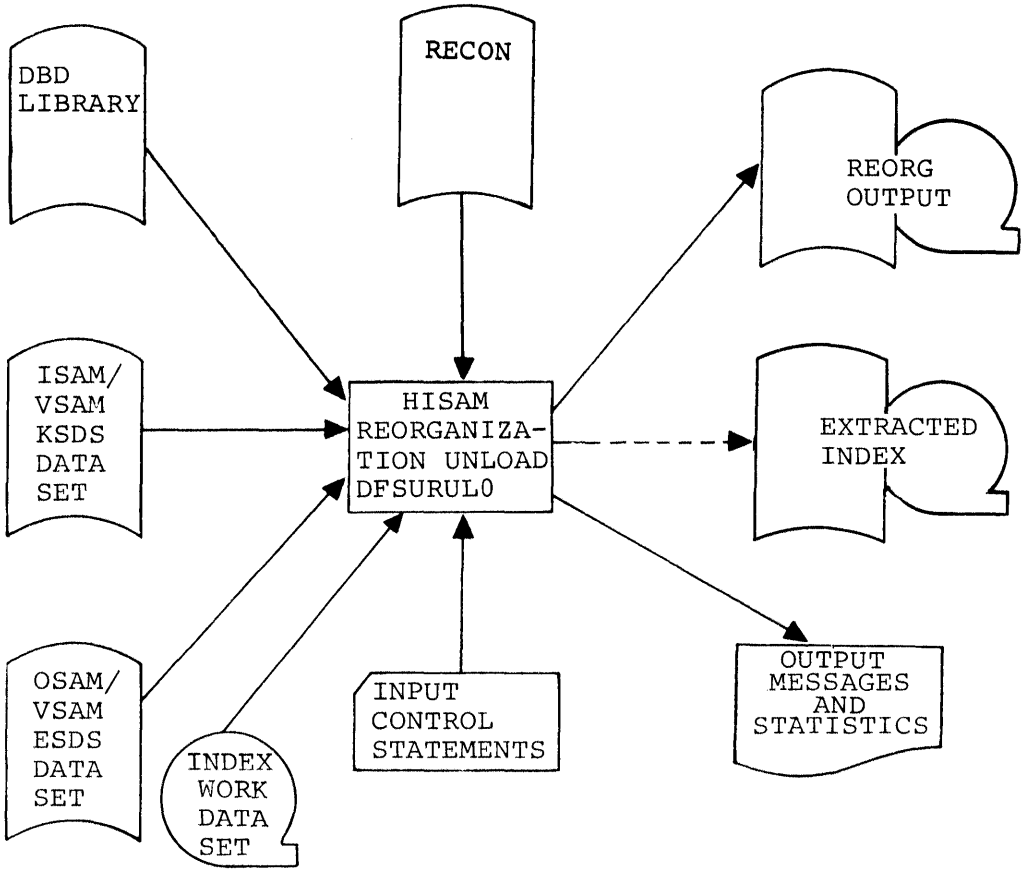


Figure 47. HISAM Reorganization Unload Utiltiy

JCL REQUIREMENTS

The HISAM Reorganization Unload utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURUL0,DBNAME'
```

The normal IMS/VS positional parameters such as SPIE, BUF, and DBRC can follow the program name in the PARM field. See the DBBBATCH or DLIBATCH procedures in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD

Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD

Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBD that describes the data base to be reorganized (that is, DSN=IMSVS.DBDLIB,DISP=SHR). This data set must reside on a direct access device.

SYSPRINT DD
Defines the output message and statistics data set. The data set can reside on a tape, direct access device, or printer, or be routed to the output stream. DCB parameters specified for this data set are RECFM=FBA and LRECL=133. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 133.

SYSIN DD
Defines the input control statement data set. This data set can reside on a tape, direct access device, or card reader, or be routed through the input stream.

isamin DD
Defines the VSAM KSDS or ISAM data set to be reorganized. The ddname must be the same as the name in the DBD that describes this data set. It must also appear on the utility control statement in the SYSIN data set of this job step. One DD statement of this type must be present for each VSAM KSDS or ISAM data set to be reorganized.

This DD statement represents the primary data set of the HISAM data base. In case of secondary index creation, one or more DD statements should be provided: each DD statement represents a secondary index data set to be reorganized.

osamin DD
Defines the VSAM ESDS or OSAM data set to be reorganized. The ddname must be the same as the name in the DBD which describes this data set. One DD statement of this type must be present for each VSAM ESDS or OSAM data set to be reorganized. If HISAM data bases consist of ISAM/OSAM groups, DD statements for each group are required for a single reorganization.

dataout1 DD
Defines the first copy of the reorganized output data set. One DD statement of this type is required for each ISAM/OSAM or VSAM data set group to be reorganized. It can be any name, but the name must appear in the associated utility control statement. The data set must reside on either tape or a direct access device.

dataout2 DD
Defines the second copy of the reorganized output data set. This optional statement is required only if two copies of the output are requested. Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job. Although performance would be somewhat diminished, a total rerun would not be required.

The same requirements described for the dataout1 DD statement apply.

indexwrkds DD
Describes the output data set (DFSURIDX) from the Prefix Resolution program which contains secondary index information. This statement is required if the utility control statement is type "X"; otherwise, it is optional. The ddname must be the same as the name starting in position 40 of the control statement.

DFSEXTDS DD
This optional DD statement is required only if "E" is specified in position 3 of the utility control statement (see below), and is used to write out an unloaded version of the records that have been split out from a shared

secondary index as specified in control statements. The DCB attributes are determined dynamically, depending on the output device type and the VSAM LRECLs used. Standard labels must be used.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.)

The data set can reside on a tape, direct access device, or card reader, or be routed through the input stream.

RECON1 DD
Defines the first data base recovery control (DBRC) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional RECON data set used by DBRC when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS

```
1  2 3 4          13          22       31       40          49  50       80
                  M
R  1  E dbdname   prime-      outl-    out2-    secondary
X  2  R           ddname      ddname   ddname   index       [c] comments
                                                work data       CHNG= DBD
                                                set ddname            CARD
```

Position	Description
1	This must be either 'R' or 'X'. An 'R' defines this as a HISAM Reorganization Unload utility control statement. An 'X' defines this as a secondary index reorganization control statement. There is no default, and if this field is left blank an error message is generated.
2	This must be a 1 or a 2, depending on the number of output copies required. There is no default; if this field is left blank, an error message is generated.
3	This must be 'M', 'E', 'R', or blank for 'X' type control statements. The default is 'M' if this position is blank. M=MERGE is the index work data set created during either data base initial load or reorganization (through use of the Prefix Resolution utility) into an existing secondary index, or creates a secondary index if the data set does not exist. E=EXTRACT is a secondary index (as defined by the constant in position 49 of this statement) from either a shared index data base or from the index work data set from the Prefix Resolution utility.

R=REPLACE those segments in the secondary index that match the constant in position 49 of this statement with matching segments found in the work data set. If the secondary index segments being replaced need to be saved, an extract function should be performed prior to the replace.

4 This must be the name of the DBD that includes the ddnames of the ISAM/OSAM or VSAM data set group to be reorganized.

13 This must be the ddname of the ISAM data set of the ISAM/OSAM data set group or the KSDS ddname for the VSAM data set to be reorganized. The ddname is the primary data set name for the HISAM data base and the secondary index ddname for the secondary index data base. It must appear in the referenced DBD statement, and a corresponding DD statement must have been provided.

22 This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.

31 This must be the ddname of the second copy of the reorganized output data set. If it contains the ddname, a corresponding DD statement must be provided. This field must be blank if position 2 contains a 1.

40 This must be the ddname of the secondary index work data set if this control statement is type "X."

49 This must be the 1-byte constant specified in the DBD generation of the shared secondary index if the replace or extract function is specified in position 3 of this statement.

50 Positions 50 through 80 can contain comments or specify control interval (CI) or record size changes. To specify the source of CI or record size changes, code CHNG=DBD or CHNG=CARD, where DBD tells unload to use the DBD values for KSDS and ESDS CI and record sizes, and CARD tells unload to use the values specified on the control card that starts with CHANGE in column 1. (See below.)

```
1                                                              80
┌──────────────────────────────────────────────────────────────┐
│                                                              │
│          [KSCISZ=nnnnn]                                      │
│ CHANGE=  [,ESCISZ=nnnnn]                                     │
│          [,KSREC=nnnnn]                                      │
│          [,ESREC=nnnnn]                                      │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

This is an optional control statement. If used, at least one keyword must be specified, and even a single keyword must be enclosed within parentheses.

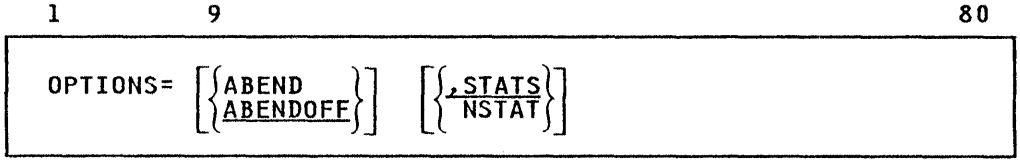
nnnnn
Is a 5-digit decimal value, including leading zeros if necessary. The maximum value is 32767.

KSCISZ
Specifies a new KSDS CI size in bytes. The size should be specified in multiples of 512 bytes.

ESCISZ
Specifies a new ESDS CI size in bytes. The size should be specified in multiples of 512 bytes.

KSREC
Specifies a new KSDS record size.

ESREC
Specifies a new ESDS record size.



This is an optional control statement.

ABEND
Terminates with user 359 ABEND if any condition arises causing termination of the run.

ABENDOFF
Turns off the ABEND function.

Note: ABENDOFF is the initial default; if ABEND has been used previously, however, it remains in effect until ABENDOFF is coded.

STATS
Provides statistics to the SYSPRINT data set and to the HISAM Reorganization Reload utility (DFSURRL0). STATS is the default if this parameter is omitted.

NSTAT
No statistics output. This causes the HISAM Reload utility to also ignore statistics output.

Note: These parameters are keywords; they are not position dependent.

RETURN CODES

This program returns codes preceded (in the case of errors) by numbered messages to the SYSPRINT data set that more fully explain the results of program execution. The return codes are as follows:

Code	Meaning
0	All requested operations have successfully completed.
4	One or more operations have not successfully completed.
8	Severe errors causing job termination have occurred.
12	A combination of error codes 4 and 8 has occurred.
16	Unable to open ddname SYSIN.

EXAMPLES

Example 1

In this example, one ISAM/OSAM data set group is to be reorganized and unloaded and one copy created of the data set group. The ISAM input ddname is DBHI2A, and the OSAM is DBHO2A. These names appear in the DBD named DI32DB01. The output ddname is DBOUT1. (The numbers above the control statements are for reference only; they are not to be included in the input stream.)

```
//DBREORG  JOB  1,1,MSGLEVEL=1
//STEP1    EXEC  PGM=DFSRRC00,PARM='ULU,DFSURUL0',REGION=250K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS        DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1330
//DBHI2A     DD  DSN=IMSVS.DBHI2A,DISP=SHR,DCB=BUFNO=10
//DBHO2A     DD  DSN=IMSVS.DBHO2A,DISP=SHR
//DBOUT1     DD  DSN=IMSVS.DBOUT1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=UNLD2A,DISP=(NEW,KEEP)
//DFSVSAMP   DD DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//SYSIN      DD  *
12 4         13       22       31      50
R1 DI32DB01 DBHI2A   DBOUT1           REORG DATA SET 2A
/*
```

Note: In this example, 10 buffers are requested by the DCB parameter of the DD statement for the ISAM input data set. This is not a requirement; performance can usually be enhanced, however, by providing additional buffers.

The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//RECON3   DD  DSN=RECON3,DISP=SHR
```

Example 2

In this example, a data base consisting of three ISAM/OSAM data set groups is to be reorganized and unloaded. Two copies of the first data set group and single copies of the second and third data set groups are to be created. (The numbers above the control statements are for reference only; they are not to be included in the input stream.)

```
//DBREORG  JOB  1,1,MSGLEVEL=1
//STEP1   EXEC  PGM=DFSRRC00,PARM='ULU,DFSURUL0',REGION=300K
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A,DCB=BLKSIZE=1330
//DBHI1A    DD  DSN=IMSVS.DBHI1A,DISP=SHR,DCB=BUFNO=10
//DBHO1A    DD  DSN=IMSVS.DBHO1A,DISP=SHR
//DBOUT1    DD  DSN=IMSVS.DBOUT1A,UNIT=TAPE,LABEL=(,SL),
//    VOL=SER=DBOUT1,DISP=(NEW,KEEP)
//DBOUT2    DD  DSN=IMSVS.DBOUT1B,UNIT=TAPE,LABEL=(,SL),
//  VOL=SER=DBOUT2,DISP=(NEW,KEEP)
//DBHI2A    DD  DSN=IMSVS.DBHI2A,DISP=SHR,DCB=BUFNO=10
//DBHO2A    DD  DSN=IMSVS.DBHO2A,DISP=SHR
//DBOUT3    DD  DSN=IMSVS.DBOUT2A,UNIT=TAPE,LABEL=(,SL),
//  VOL=SER=DBOUT3,DISP=(NEW,KEEP)
//DBHI3A    DD  DSN=IMSVS.DBHI3A,DISP=SHR,DCB=BUFNO=10
//DBHO3A    DD  DSN=IMSVS.DBHO3A,DISP=SHR
//DBOUT4    DD  DSN=IMSVS.DBOUT3A,UNIT=TAPE,LABEL=(,SL),
//  VOL=SER=DBOUT4,DISP=(NEW,KEEP)
//DFSVSAMP  DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//SYSIN     DD *
12 4        13     22     31     50                          80
R2 DI32DB01 DBHI1A DBOUT1 DBOUT2 REORG DATA SET 1A-2 COPIES
R1 DI32DB01 DBHI2A DBOUT3        REORG DATA SET 2A-1 COPY
R1 DI32DB01 DBHI3A DBOUT4        REORG DATA SET 3A-1 COPY
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 3

In this example, an index work data set passed from the Prefix Resolution utility is to be used to create: (a) unloaded versions of two secondary indexes in a shared secondary index data base, and (b) an unloaded version of a third secondary index.

The output of the example is used as input to the HISAM Reload utility to create the actual secondary indexes. The OS/VS Access Method Services utility must have been run to create the secondary index data bases. The OPTIONS statement specifies that statistics are not desired and that any serious message will cause a U359 abend. (The numbers above the control statements are for reference only; they are not to be included in the input stream.)

Note: In this example, DBRC and IRLM are turned off in the PARM EXEC card.

```
//INDREOR  JOB  1,1,MSGLEVEL=1
//STEP1    EXEC  PGM=DFSRRC00,
//         PARM='ULU,DFSURUL0,,,1,,,,,,,,,N,N'
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS        DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1330
//DBIDX1     DD  DSN=IMSVS.INDX1,DISP=SHR
//DXOUT1     DD  DSN=IMSVS.DBXOUT1,DISP=(MOD,KEEP),UNIT=TAPE,
//   LABEL=(,SL),VOL=SER=DXOUT1
//DBIDX2     DD  DSN=IMSVS.INDX2,DISP=SHR
//DXOUT2     DD  DSN=IMSVS.DBXOUT2,DISP=(,KEEP),UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=DXOUT2
//NDXDS      DD  DSN=IMSVS.NDXWDS,DISP=(OLD,DELETE)
//DFSVSAMP   DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//SYSIN      DD  *
OPTIONS=(NSTAT,ABEND)
1234         13       22    31   40    49 50
X1MDIX1DB01 DBIDX1   DXOUT1     NDXDS    CREATE NEW SECONDARY INDEX
X1MDIX1DB02 DBIDX1   DXOUT1     NDXDS    ADD SECONDARY INDEX RECS TO
                                         EXISTING ONE ABOVE
X1MDIX2DB01 DBIDX2   DXOUT2     NDXDS    RECORDS FROM SAME WORK DS PUT
                                         INTO DIFFERENT DATA BASE
/*
```

Example 4

In this example, a group of index records that had a constant of 'B' defined in the DBDGEN for the shared index is to be extracted, replaced, and merged. The options are to be changed between operations to have statistics on first, none on the second and statistics again on the third. The ABEND option is also changed. (The numbers above the control statements are for reference only; they are not to be included in the input stream.)

```
//IDEREORG JOB  1,1,MSGLEVEL=1
//STEP1   EXEC  PGM=DFSRRC00,PARM='ULU,DFSURUL0',REGION=250K
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A,DCB=BLKSIZE=1330
//DBIDX2    DD  DSN=IMSVS.INDX2,DISP=SHR
//DBIDX1    DD  DSN=IMSVS.INDX1,DISP=SHR
//DBXOUT1   DD  DSN=IMSVS.DBXOUT1,DISP=(MOD,KEEP),UNIT=TAPE,
//  LABEL(,SL),VOL=SER=DXOUT1
//DFSEXTDS  DD  DSN=IMSVS.EXTDS1,DISP=(MOD,KEEP),UNIT=TAPE,
//  LABEL=(,SL),VOL=SER=DEXTDS
//NDXWDS1   DD  DSN=IMSVS.XWDS1,DISP=(OLD,PASS)
//NDXWDS2   DD  DSN=IMSVS.XWDS2,DISP=(OLD,PASS)
//NDXWDS3   DD  DSN=IMSVS.XWDS3,DISP=(OLD,PASS)
//SYSIN     DD  *
OPTIONS=(STATS,ABEND)
1234        13      22    31  40      49 50
X1EDIX3DB01 DBIDX1  DBXOUT1   NDXWDS1 B  UNLOAD INDEX EXTRACT THOSE
                                          MARKED WITH CONSTANT B
                                          INCLUDING THOSE ON WORK
                                          DATA SET
OPTIONS=(ABENDOFF,NSTAT)
X1RDIX4DB01 DBIDX2  DBXOUT1   NDXWDS2 B  UNLOAD INDEX REPLACING THOSE
                                          HAVING CONSTANT B WITH
                                          THOSE FROM INDEX WORK
                                          DATA SET
OPTIONS=(STATS,ABEND)
X1MDIX4DB02 DBIDX2  DBXOUT1   NDXWDS3    MERGE ALL RECS OF WORK DATA
                                          SET AND CREATE A SHARED
                                          INDEX
/*
//DFSVSAMP  DD  *
1024,10
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 5

In this example, JCL is provided to: (a) create two output data sets to be used, in turn, to create two separate secondary indexes; (b) merge two secondary indexes by merging an index work data set created by the Prefix Resolution utility with an existing secondary index (which was created previously as a shared secondary index having only one constant); (c) replace the one constant already in the shared secondary index with the constants in the index work data sets, and (d) extract a constant from a shared index and put it in a form that can be used by the HISAM Reload utility to create another separate secondary index.

Each operation is separated by a different OPTIONS statement. (The numbers above the control statements are for reference only; they are not to be included in the input stream.) Although the JCL is shown only once here, each operation is considered a separate run. To perform all operations in one run, DISP=MOD must be specified for DBOUT1 and DBOUT2.

```
//INDXREOR JOB  1,1,MSGLEVEL=1
//STEP1   EXEC  PGM=DFSRRC00,PARM='ULU,DFSURUL0',REGION=250K
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A,DCB=BLKSIZE=1330
//DBIX1A    DD  DSN=IMSVS.DBIX1A,DISP=OLD
//DBIX2B    DD  DSN=IMSVS.DBIX2B,DISP=OLD
//DBOUT1    DD  DSN=IMSVS.DBOUT1A,DISP=(,KEEP),SPACE=(CYL,(2,1)),
//  UNIT=SYSDA
//DBOUT2    DD  DSN=IMSVS.DBOUT2A,DISP=(,KEEP),LABEL=(,SL),
//  VOL=SER=DBOUT2
//INDXWDS1  DD  DSN=IMSVS.INDXWDS1,DISP=(OLD,DELETE)
//INDXWDS2  DD  DSN=IMSVS.INDXWDS2,DISP=(OLD,DELETE)
//DFSEXTDS  DD  DSN=IMSVS.EXTSD1,DISP=(,KEEP),LABEL=(,SL),
//  VOL=SER=EXTDS1
//SYSIN      DD  *
1234        13      22      31      40      49 50
X1MDI32XDB1 DBIX1A  DBOUT1          INDXWDS1    DBIX1A TO BE CREATED
X1MDI33XDB2 DBIX2B  DBOUT2          INDXWDS2    DBIX2B TO BE CREATED
OPTIONS=(STATS,ABEND)
X1MDI32XDB1 DBIX1A  DBOUT1          INDXWDS2    MERGE ONE EXISTING
                                                WITH NEW
OPTIONS=(NSTAT,ABENDOFF)
X2RDI32XDB1 DBIX1A  DBOUT1  DBOUT2  INDXWDS1 A  REPLACE ANY EXISTING
                                                A'S WITH ONES FROM
                                                WORK DATA SET
OPTIONS=STATS
X1EDI32XDB1 DBIX1A  DBOUT1          INDXWSD2 B  TAKE B CONSTANTS FROM
                                                SHARED INDX AND/OR
                                                WORK DATA SET AND PUT
                                                OUT TO DFSEXTDS DD
                                                CARD
/*
//DFSVSAMP    DD  *
1024,10
/*
```

Notes:

1. An "X" or an "R" control statement must be supplied for all aliases contained in the shared index DBD. If either an "X" or "R" statement is omitted for any alias, the actual data that the alias represents may be subject to deletion if the shared secondary index data base is deleted or redefined.

2. The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD   DSN=RECON1,DISP=SHR
//RECON2   DD   DSN=RECON2,DISP=SHR
//RECON3   DD   DSN=RECON3,DISP=SHR
```

OUTPUT MESSAGES AND STATISTICS

The HISAM Reorganization Unload utility provides messages and statistics on data base content for each data set group. In addition, it provides an audit trail capability. Figure 48 is an example of the output messages and statistics obtained from this utility.

```
                H I E R A R C H I C A L   I N D E X E D   S E Q U E N T I A L
                D A T A   B A S E   R E O R G A N I Z A T I O N   U N L O A D

                     D A T A   S E T   G R O U P   S T A T I S T I C S

   DATA BASE - DI41SK01

      PRIMARY  DD- DDI3I1
      OVERFLOW DD- DDI3O1

                        PRIMARY ROOTS                    OVERFLOW ROOTS          OVERFLOW DEPENDENTS
                     IN          OUT       DELETED       IN       DELETED        IN          OUT
                      9            9            0         8            8          78           36

        TOTAL NUMBER OF RECORDS OUT =          50

   COPY 1 ON VOLUME(S)-  000000

   COPY 2 ON VOLUME(S)-  T11111

         ROOT OVERFLOW CHAINS (#)          DEPENDENT OVERFLOW CHAINS (#)       ROOTS WITHOUT OVERFLOW CHAINS (#)
      NO.  LONGEST  SHORTEST  AVERAGE    NO.  LONGEST  SHORTEST  AVERAGE     NO.  LONGEST  SHORTEST  AVERAGE
        4        3         1        2     5       40         5    15.60       2       76        42       59

                                   S E G M E N T   L E V E L   S T A T I S T I C S

MAXIMUM TWINS  AVERAGE TWINS  MAXIMUM CHILDREN  AVERAGE CHILDREN  SEGMENT NAME  SEGMENT LEVEL  TOTAL SEGMENTS BY SEGMENT TYPE  AVERAGE COUNT PER DATA BASE RECORD
            1          1.00                72             31.75      A1111111              1                               8                                1.00
            3          1.25                12              4.00      AA222222              2                              10                                1.25
            3          1.00                 9              3.00      AAA33333              3                              10                                1.25
            3          1.00                 0              0.00      AAAA4444              4                              10                                1.25
            3          1.00                 3              1.00      AAAB4444              4                              10                                1.25
            3          1.00                 0              0.00      AAABA555              5                              10                                1.25
            4          1.75                 0              0.00      AB222222              2                              14                                1.75
            3          1.37                30              9.09      AC222222              2                              11                                1.37
            3          0.90                 0              0.00      ACA33333              3                              10                                1.25
            3          0.90                24              8.00      ACB33333              3                              10                                1.25
            3          1.00                18              6.00      ACBA4444              4                              10                                1.25
            3          1.00                 0              0.00      ACBAA555              5                              10                                1.25
            3          1.00                12              4.00      ACBAB555              5                              10                                1.25
            3          1.00                 9              3.00      ACBABA66              6                              10                                1.25
            3          1.00                 6              2.00      ACBABAA7              7                              10                                1.25
            3          1.00                 0              0.00      ACBABAAA              8                              10                                1.25
            3          1.00                 0              0.00      ACBABAAB              8                              10                                1.25
            3          1.00                 0              0.00      ACBB4444              4                              10                                1.25
            0          0.00                 0              0.00      ACC33333              3                               0                                0.00
            0          0.00                 0              0.00      ACCA4444              4                               0                                0.00
            9          2.37                18              3.15      AD222222              2                              19                                2.37
            3          0.52                15              5.00      ADA33333              3                              10                                1.25
            3          1.00                12              4.00      ADAA4444              4                              10                                1.25
            3          1.00                 9              3.00      ADAAA555              5                              10                                1.25
            3          1.00                 6              2.00      ADAAAA66              6                              10                                1.25
            3          1.00                 3              1.00      ADAAAAA7              7                              10                                1.25
            3          1.00                 0              0.00      ADAAAAAA              8                              10                                1.25

        TOTAL SEGMENTS IN DATA SET GROUP =      262        AVERAGE DATA SET GROUP RECORD LENGTH=   2,149 BYTES
```

Figure 48. Example of Output Messages and Statistics—HISAM Reorganization Unload Utility

If any options were selected for this execution, various messages would be generated and appear immediately following the page heading. An explanation of all numbered messages can be found in the IMS/VS Messages and Codes Reference Manual.

The message "COPY 1 ON VOLUME(S) - volser1" indicates that the primary output data set has successfully completed. The list of volume serial numbers indicates which volumes were used and the order of their use. If a second copy was requested and the second copy successfully completed, another message, "COPY 2 ON VOLUME(S) - volser2," would appear.

Following the message for each data set group, the heading "DATA SET GROUP STATISTICS" appears. The various fields are described below.

- Data Base: Data base name
- Primary DD: VSAM KSDS or ISAM ddname of data set group
- Overflow DD: VSAM ESDS or OSAM ddname of data set group
- Primary ROOTS (#)

  Statistics dealing with root records in VSAM KSDS or ISAM

  - IN—Number of old VSAM KSDS or ISAM roots read
  - OUT—Number of new VSAM KSDS or ISAM roots written
  - DELETED—Number of old VSAM KSDS or ISAM roots deleted

(If the number of roots in and out contains one more than the user has inserted, this is the high-key record that terminates the ISAM data set.)

- Overflow ROOTS (#)

  Statistics dealing with old roots in VSAM ESDS or OSAM

  - IN—Number of old VSAM ESDS or OSAM roots read
  - DELETED—Number of old VSAM ESDS or OSAM roots deleted

- Overflow DEPENDENTS (#)

  Statistics dealing with dependent records in VSAM ESDS or OSAM

  - IN—Number of old dependent records in VSAM ESDS or OSAM read
  - OUT—Number of new dependent records in VSAM ESDS or OSAM written

The number of dependents out includes dummy OSAM dependents used to fill the last OSAM block.

- TOTAL NUMBER OF RECORDS OUT (#)

  Number of records, both VSAM KSDS and ISAM roots and VSAM ESDS and OSAM dependents, written out. This total includes at least one header record. It may also include two or more statistics records—one or more at the beginning of the data set used as a table initialization record for the Reload program, and one or more at the end of the data set containing totals unloaded by segment type so that the HISAM Reload utility can compare the numbers reloaded with those unloaded.

Note: A statistics table record consists of 20 bytes for each segment type in the DBD plus a 28-byte header for each LRECL needed to contain the entire statistics record. The approximate number of LRECLs required to contain the statistics record can be determined by applying the following formula:

(Number of segment types x 20)/(LRECL)

Multiplying the results by 2 gives the approximate number of LRECLs added to the total number of records out.

- ROOT OVERFLOW CHAINS (#)

  Statistics dealing with root records in VSAM KSDS or ISAM which had overflow chains to root records in VSAM ESDS or OSAM

  - NO.—Number of ISAM roots with chains into VSA, ESDS or OSAM
  - LONGEST—Largest number of VSAM ESDS or OSAM roots chained off one VSAM KSDS or ISAM root
  - SHORTEST—Smallest nonzero number of VSAM ESDS or OSAM roots chained off one VSAM KSDS or ISAM root
  - AVERAGE—Average number of VSAM ESDS or OSAM roots chained off one VSAM KSDS or ISAM root (of those with chains)

- DEPENDENT OVERFLOW CHAINS (#)

  Statistics dealing with roots (both VSAM KSDS or ISAM and OSAM) which had dependents in VSAM ESDS or OSAM

  - NO.—Number of roots with VSAM ESDS or OSAM dependent records
  - LONGEST—Largest number of VSAM ESDS or OSAM dependent records chained off one root
  - SHORTEST—Smallest nonzero number of VSAM ESDS or OSAM dependent records chained off one root
  - AVERAGE—Average number of VSAM ESDS or OSAM dependent records chained off one root (of those roots which had dependents)

- ROOTS WITHOUT OVERFLOW CHAINS (BYTES)

  Statistics dealing with roots (both VSAM KSDS or ISAM and OSAM) which had no dependents

  - NO.—Number of roots without dependent chains
  - LONGEST—Largest data base record (in bytes) with no VSAM ESDS or OSAM dependent records
  - SHORTEST—Shortest data base record (in bytes) with no VSAM ESDS or OSAM dependent records
  - AVERAGE—Average data base record length (in bytes) of those roots with no VSAM ESDS or OSAM dependent records

The heading "SEGMENT LEVEL STATISTICS" appears next. The various fields are described below.

- Maximum twins (#)

  This field contains the maximum number of segments of this type encountered under an immediate parent segment. At the root level, this value will always be 1.

- Average twins (#)

  This field contains the average number of segments of this type encountered under an immediate parent segment. This value is carried out to 2 decimal places.

- Maximum children (#)

  This field contains the maximum number of child segments (at all subordinate levels) under a given parent.

- Average children (#)

  This field contains the average number of child segments (at all subordinate levels) under a given parent. This value is carried out to 2 decimal places. It should be noted that the lowest level segment in any hierarchic path will have a value of zero in this field type.

- Segment name

  The segment name to which this line of statistics applies.

- Segment level

  The hierarchic level of this segment in the data base.

- Total segments by segment type (#)

  This field contains the count of the occurrences of this segment type in the entire data base. The count field in the level 1 segment type reflects the total number of data base records (root segments) in the data base.

When a data base is part of a shared secondary index data base, the segment occurrence count always appears under the first segment name. All other segment name counts are zero. Although IMS/VS is unable to distinguish which segment name matches the statistics, the count is correct.

- Average count per data base record (#)

  This field contains a count of the average number of occurrences of this segment type within a given data base record. The value is carried to 2 decimal places.

Following the individual segment type statistics for this data set group are the following two fields:

- Total segments in data set group (#)
- Average data set group record length (bytes)

  (The average length in bytes of the portion of the data base record stored in this data set group.)

HISAM REORGANIZATION RELOAD UTILITY (DFSURRL0)

The HISAM Reorganization Reload utility can be used to: (1) reload an HISAM data base unloaded by the HISAM Unload utility, (2) create or merge secondary indexes from a reorganized output data set provided by the HISAM Unload utility, and (3) reload a primary index of a HIDAM data base unloaded by the HISAM Unload utility. Sequence checking is performed on the root segment keys of data base records.

The functions of this utility can be performed by the Utility Control Facility, if desired. Refer to "Utility Control Facility" for a description of its operation.

RESTRICTIONS

The following restrictions apply with respect to using the HISAM Reorganization Reload utility:

- If an ISAM/OSAM data base is unloaded and an OPTIONS=(VSAM) control statement is included, the output is in VSAM format. The OS/VS Access Method Services utility must have been run to create the required data sets, and all of the necessary DBD changes must have been made before the HISAM Reload utility can be run. can only be used when making a conversion from ISAM/OSAM to VSAM.
- If a VSAM data base is unloaded, the reloaded data base is VSAM regardless of what is specified in an OPTIONS utility control statement. The original data set must be scratched and reallocated with the OS/VS Access Method Services utility, or a new DSNAME created by the Access Method Services utility before the HISAM Reload utility can be run.
- For information on scratching and allocating ISAM/OSAM data sets, see "Allocation for OSAM Data Sets" in "Designing the IMS/VS Online System," in IMS/VS System Administration Guide.

Figure 49 is a diagram of the HISAM Reorganization Reload utility.

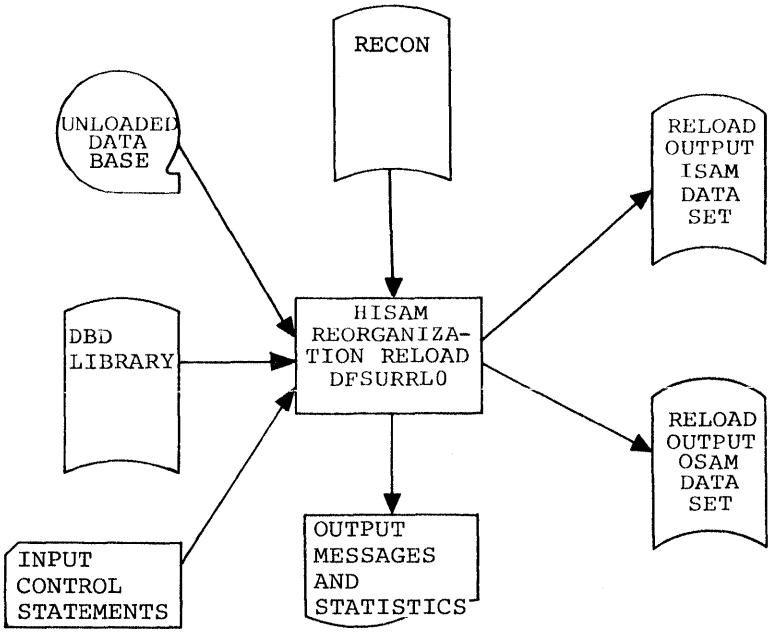


Figure 49. HISAM Reorganization Reload Utility

JCL REQUIREMENTS

The HISAM Reorganization Reload utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURRL0,DBNAME'
```

The normal IMS/VS positional parameters such as SPIE, BUF, and DBRC can follow the program name in the PARM field. See the DBBBATCH or DLIBATCH procedures in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBD that describes the data base being reorganized. This data set must reside on a direct access device.

SYSPRINT DD
Defines the output message and statistics data set. The data set can reside on a tape, direct access device, or printer, or be routed through the output stream.

DFSUINxx DD
Defines the unloaded input data set. The first input data set group to be reloaded would be defined by the ddname DFSUIN01, and each succeeding input data set would increment the last two numeric digits by 1.

isamout1 DD
Defines the VSAM KSDS or ISAM output data set to be reloaded. The ddname must be the same as the name in the DBD that was referenced when this data set was unloaded. The size of the data base space allocation can be increased by specifying a larger space parameter on this DD statement.

osamout1 DD
Defines the VSAM ESDS or OSAM output data set to be reloaded. The name must be the same as the ddname in the DBD that was referenced when this data set was unloaded. The size of the data base space allocation can be increased by specifying a larger space parameter on this DD statement. (VSAM data bases require a DEFINE control statement to alter space.)

SYSIN DD
Defines the input control information data set. The data set can reside on a card reader, tape, or direct access device, or be routed through the input stream. This DD statement is not necessary if no utility control cards are provided as input to the utility.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.)

The data set can reside on a tape, direct access device, or card reader, or be routed through the input stream.

SYSABEND DD or SYSDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence will be used for the dump.

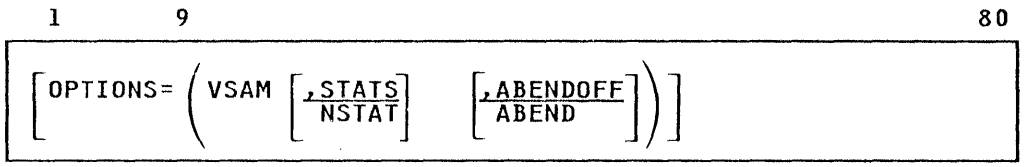
RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional data set used by DBRC when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENT



This optional utility control statement specifies the new form of the data base being reorganized, regardless of its previous form.

VSAM
This keyword is required if the output is VSAM and the input is the unloaded format of an ISAM/OSAM data base. Use of this option causes the ISAM/OSAM format to be converted to the VSAM physical format. This involves changing the length of pointers and the meaning of some pointers, as well as pointers being changed from relative block numbers to relative byte addresses; for this reason, the input data set cannot be used as input to the Data Base Recovery utility. An image copy of the data base should be made, or the HISAM Unload utility should be rerun as a reorganization of the VSAM data base to create a backup data set as input to the Data Base Recovery utility.

Before using this option, the DBD must be changed to specify VSAM as the access method, and the OS/VS Access Method Services utility must be run to do the VSAM data base definition.

If the keyword VSAM is omitted, the default is that the data base will be reloaded in the same format (VSAM or ISAM/OSAM) it had prior to being unloaded.

Unpredictable errors can occur if this keyword is specified when the data base to be reorganized is a VSAM data base. Do not specify VSAM if the input is in VSAM format or if the output desired is ISAM/OSAM.

STATS
If statistics were provided by the HISAM Unload utility, the HISAM Reload utility audits the number loaded and compares that number against the number provided by HISAM Unload.

NSTAT
Causes the statistics provided by the HISAM Unload utility to be ignored.

ABENDOFF
Turns off the abend function. An abnormal condition will cause program termination, but no abend code or dump is provided. This is the default. If ABEND is coded, it remains in effect within a jobstep until ABENDOFF is coded.

ABEND
Terminate with user abend 359, if any condition arises causing abnormal termination of the run. A dump will be printed if a SYSABEND or SYSUDUMP DD statement is supplied.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	All operations have successfully completed.
4	One or more warning messages issued.
8	One or more operations have not completed successfully.
16	Severe errors causing program termination have occurred.

EXAMPLE

In this example, a HISAM data base consisting of three ISAM/OSAM data set groups is to be reloaded.

```
//DBRELOAD  JOB  1,1,MSGLEVEL=1
//STEP1     EXEC  PGM=DFSRRC00,PARM='ULU,DFSURRL0,DI32DB03',
//              REGION=300K
//STEPLIB     DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS         DD  DSN=IMSVS.DBDLIB,DISP=OLD
//SYSPRINT    DD  SYSOUT=A
//DFSUIN01    DD  DSN=IMSVS.DBOUT1A,UNIT=TAPE,LABEL=(,SL),
//  VOL=SER=DBOUT1,DISP=(OLD,KEEP)
//DBHI1A      DD  DSN=IMSVS.DBHI1A,UNIT=SYSDA,VOL=SER=DBVOL1,
//    SPACE=(CYL,(2,,1)),DISP=(NEW,KEEP),DCB=(DSORG=IS,
//       BUFNO=10)
//DBHO1A      DD  DSN=IMSVS.DBHO1A,UNIT=SYSDA,VOL=SER=DBVOL2,
//    SPACE=(CYL,(2,1)),DISP=(NEW,KEEP)
//DFSUIN02    DD  DSN=IMSVS.DBOUT2A,UNIT=TAPE,LABEL=(,SL),
//    VOL=SER=DBOUT3,DISP=(OLD,KEEP)
//DBHI2A      DD  DSN=IMSVS.DBHI2A,UNIT=SYSDA,VOL=SER=DBVOL1,
//    SPACE=(CYL,(2,,1)),DISP=(NEW,KEEP),DCB=(DSORG=IS,
//      BUFNO=10)
//DBHO2A      DD  DSN=IMSVS.DBHO2A,UNIT=SYSDA,VOL=SER=DBVOL2,
//  SPACE=(CYL,(2,1)),DISP=(NEW,KEEP)
//DFSUIN03    DD  DSN=IMSVS.DBOUT3A,UNIT=TAPE,LABEL=(,SL),
//  VOL=SER=DBOUT4,DISP=(OLD,KEEP)
//DBHI3A      DD  DSN=IMSVS.DBHI3A,UNIT=SYSDA,VOL=SER=DBVOL1,
//  SPACE=(CYL,(2,,1)),DISP=(NEW,KEEP),DCB=(DSORG=IS,
//     BUFNO=10)
//DBHO3A      DD  DSN=IMSVS.DBHO3A,UNIT=SYSDA,VOL=SER=DBVOL2,
//  SPACE=(CYL,(2,1)),DISP=(NEW,KEEP)
//DFSVSAMP    DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//SYSIN       DD  *
OPTIONS=STATS
/*
```

Notes:

1. In this example, 10 buffers were requested by JCL. This is not a requirement; performance can usually be enhanced, however, by providing additional buffers for the ISAM data sets.

   For VSAM or secondary index data bases, the JCL is the same, except that VSAM data sets must be preallocated using the OS/VS Access Method Services utility (DISP=OLD).

2. The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

OUTPUT MESSAGES AND STATISTICS

The HISAM Reorganization Reload utility provides messages and statistics and an audit trail for each data set group reloaded. An example of the messages and statistics obtained from this utility, accompanied by explanatory information, is shown in Figure 50.

```
H I E R A R C H I C A L   I N D E X E D   S E Q U E N T I A L
D A T A   B A S E   R E O R G A N I Z A T I O N   R E L O A D


                    D A T A   S E T   G R O U P   S T A T I S T I C S

DATA BASE - DI41SK01
   PRIMARY  DD - DDI3I1
   OVERFLOW DD - DDI301

                                                DEPENDENT OVERFLOW CHAINS(#)     ROOT WITHOUT OVERFLOW CHAINS(BYTES)
     PRIMARY ROOTS  OVERFLOW DEPENDENTS  NO. LONGEST  SHORTEST  AVERAGE      NO.  LONGEST  SHORTEST  AVERAGE
                 9                   36         5        9         3      6.8         3        76        42       53


                    S E G M E N T   L E V E L   S T A T I S T I C S

     SEGMENT      SEGMENT        TOTAL SEGMENTS BY SEGMENT TYPE
      NAME        LEVEL          RELOADED      DIFFERENCE
     A1111111       1                 8
     AA222222       2                10
     AAA33333       3                10
     AAAA4444       4                10
     AAAB4444       4                10
     AAABA555       5                10
     AB222222       2                14
     AC222222       2                11
     ACA33333       3                10
     ACB33333       3                10
     ACBA4444       4                10
     ACBAA555       5                10
     ACBAB555       5                10
     ACBABA66       6                10
     ACBABAA7       7                10
     ACBABAAA       8                10
     ACBABAAB       8                10
     ACBB4444       4                10
     ACC33333       3                 0
     ACCA4444       4                 0
     AD222222       2                19
     ADA33333       3                10
     ADAA4444       4                10
     ADAAA555       5                10
     ADAAAA66       6                10
     ADAAAAA7       7                10
     ADAAAAAA       8                10

                    TOTAL SEGMENTS IN DATA SET GROUP

                    UNLOADED          RELOADED          DIFFERENCE
                        262               262

DFS340I    DATABASE DI41SK01 HAS BEEN SUCCESSFULLY RELOADED BY FUNCTION SR
DFS339I    FUNCTION SR HAS COMPLETED NORMALLY RC=00
```

Figure 50. Example of Output Messages and Statistics—HISAM Reorganization Reload Utility

If any options were selected for this execution, various messages would be generated and appear immediately following the page heading. (An explanation of all numbered messages can be found in IMS/VS Messages and Codes Reference Manual.)

Statistics are normally provided on every execution of the HISAM Reload utility. Because this increases reload time slightly, installations that are billed by processor time might want to have statistics recording suppressed from time to time. This can be done by use of the OPTIONS control statement (described below).

By specifying OPTIONS=STATS or by not using an OPTIONS card, statistics will be provided for each reload.

By specifying OPTIONS=NSTAT, no statistics will be provided for this job step.

Following the messages for each data set group, the heading "DATA SET GROUP STATISTICS" appears. The various fields are described below.

- Data Base: Data base name
- Primary DD: VSAM KSDS or ISAM ddname of data set group
- Overflow DD: VSAM ESDS or OSAM ddname of data set group
- Primary Roots

  Number of roots loaded

  (If the number of roots in and out contains more than the user has inserted, this is the high-key record that terminates the ISAM data set.)
- Overflow Dependents

  Number of dependent records loaded
- Dependent Overflow Chains (#)

  Statistics dealing with roots with dependents chained into VSAM ESDS or OSAM

  - NO.—Number of VSAM KSDS or ISAM roots with OSAM/VSAM ESDS dependent records
  - LONGEST—Largest number of VSAM ESDS or OSAM dependent records chained off one VSAM KSDS or ISAM root
  - SHORTEST—Smallest nonzero member of VSAM ESDS or OSAM dependent records chained off one VSAM KSDS or ISAM root
  - AVERAGE—Average number of VSAM ESDS or OSAM dependent records chained off VSAM KSDS or ISAM roots (of those with chains)
- Roots without Overflow Chains (BYTES)

  Statistics dealing with VSAM KSDS or ISAM roots which have no VSAM ESDS or OSAM dependent records

  - NO.—Number of roots with no VSAM ESDS or OSAM dependent records
  - LONGEST—Largest root record (in bytes) with no VSAM ESDS or OSAM dependent records
  - SHORTEST—Smallest root record (in bytes) with no VSAM ESDS or OSAM dependent records
  - AVERAGE—Average length of root records (in bytes) with no VSAM ESDS or OSAM dependent records

The heading "SEGMENT LEVEL STATISTICS" appears next. The fields, from left to right, are described below.

- Segment Name

  The segment name to which this line of statistics applies

- Segment Level

  The hierarchic level of this segment in the data base

- Total Segments by Segment Type

  - Reloaded

    The total number of segments of this type unloaded

  - Difference

    This field is blank if the counts by reload and unload are equal. If they are not equal, the difference is printed.

    When a data base is part of a shared secondary index data base, the segment occurrence count always appears under the first segment name. All other segment name counts are zero. Although IMS/VS is unable to distinguish which segment name matches the statistics, the count is correct.

Following the individual segment type statistics for this data set group are the total segments in data set statistics.

- UNLOADED—The total number of segments unloaded by the HISAM Unload utility (DFSURUL0)
- RELOADED—The total number of segments reloaded by the HISAM Reload utility (DFSURRL0)
- DIFFERENCE—The difference, if any, between the previous two totals

HD REORGANIZATION UNLOAD UTILITY (DFSURGU0)

The HD Reorganization Unload utility can be used to unload an HDAM, HIDAM, or HISAM data base to a QSAM formatted data set. If logical relationships exist, this utility generates a data set containing prefix information that is used by the HD Reorganization Reload utility to supply information to the work file generator to build the necessary records for use by the Prefix Resolution utility in resolving the relationships. No segment sequence checking is performed. There are no utility control statements for this utility.

Note: The functions of this utility can be performed by the Utility Control Facility. Refer to "Utility Control Facility" for a description of its operation.

If structural changes are to be made to a HISAM or HD data base, the HD Reorganization Unload/Reload utilities must be used. The rules and restrictions that apply to making structural changes are discussed below.

RULES AND RESTRICTIONS

The DBD of the data base being reorganized is part of the input for the HD Reorganization Unload utility. By replacing this DBD with a new version, certain structural changes can be made to a data base during the process of reorganization. The following rules and restrictions apply:

1. The HD Unload utility must have been executed against the DBD describing the current structure of the data base and updates must not have been made since the unload.

2. Logical record length and block size can be changed, and/or changes can be made from ISAM/OSAM format to VSAM format.

3. When unloading an HDAM data base, the randomizing module must be included in the JOBLIB.

4. An existing segment type can be deleted from the DBD provided all segments of this type were deleted from the data base prior to execution of the HD Unload utility.

5. New segment types can be added to the new DBD provided they do not change either the hierarchic relation among existing segment types or the concatenated keys of logically related segments.

6. Names of existing segment types must not be changed.

7. Any field statement except the one for the sequence field of a segment can be changed, added or deleted, but no attempt is made by IMS/VS to alter the data content of a segment.

8. Existing segment lengths can be changed on fixed length segments. IMS/VS cannot alter the data content, however, except to truncate data if the segment is made smaller.

   If the segment is made larger, binary zeros will be used as fill characters for the added portion of the segment. It is the user's responsibility to replace the extended portion of the segment through use of an application program running in update mode under IMS/VS.

9. The DL/I access method can be changed. ISAM/OSAM format can be changed to VSAM format or VSAM format to ISAM/OSAM. Any access method can be changed to any other with the exception of HDAM to either indexed method. HISAM/HIDAM can be changed to HDAM.

10. Segment pointer options for either HDAM or HIDAM can be changed. If, however, the data base contains logical relationships and if counter, LT, or LP pointers are changed, the Data Base Prereorganization utility must be rerun. If changing from physical to virtual pairing, all occurrences of the segment which will become virtual must be deleted.

11. There are three cases in which the HD Unload utility should not be used:

    - When changing from bidirectional virtual pairing to bidirectional physical pairing, if any logical child segments have been deleted from either the physical or logical path but not from both paths

    - When changing a real logical child from one logically related data base to another

    - When reorganizing a primary or secondary index (the HISAM reorganization utilities should be used for an index database).

See IMS/VS Data Base Administration Guide chapter on "Modifying Your Data Base" for further information on use of the HD Unload utility.

1. Use the following utility execution sequence to recover invalid logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents in a data base(s).

- Run the HD Reorganization Unload utility (DFSURGU0) against all data bases involved in a logical relationship.
- Run the Prereorganization utility (DFSURPR0) using the DBIL control statement:

  DBIL=database name1,database name2,...

  The data base name is replaced by all data base names that are involved in a logical relationship. If database name2 is involved in a logical relationship with database name3, database name3 must also be unloaded and reloaded.
- Run the HD Reorganization Reload utility (DFSURGL0) to reload the data bases using the control data set created by the Prereorganization utility as input.

  During the reload, the DBIL statement in the Prereorganization utility run causes the work data set generator program (DFSDEH0) to use the concatenated keys of the logical parents instead of the old addresses.
- Run the Prefix Resolution utility (DFSURG10) to resolve the logical relationships defined for the data base(s).
- Run the Prefix Update utility (DFSURGP0) to complete the logical relationship updates.

2. If adding a logical pointer that does not exist in a previous DBD or changing pointer options from symbolic to direct, use this same utility execution sequence, but run the new DBD after the HD Reorganization Unload utility.

   The following must be accomplished prior to executing the HD Reorganization Reload utility:

   - Assemble and link-edit the new DBD into the IMS/VS DBD library;
   - If adding or deleting new segments and logical relationships are contained within the data base, rerun the Prereorganization utility against the new DBD;
   - If the DBD name is changed and the DBD contains logical relationships, rerun the Prereorganization utility against the new DBD.

   For further information on using the HD Unload utility, see "Modifying Your Data Base" in IMS/VS Data Base Administration Guide.

3. For information on scratching and allocating ISAM/OSAM data sets, see "Allocation for OSAM Data Sets" under "Designing the IMS/VS Online System" in IMS/VS System Administration Guide.

Figure 51 is a flow diagram of the HD Reorganization Unload utility.

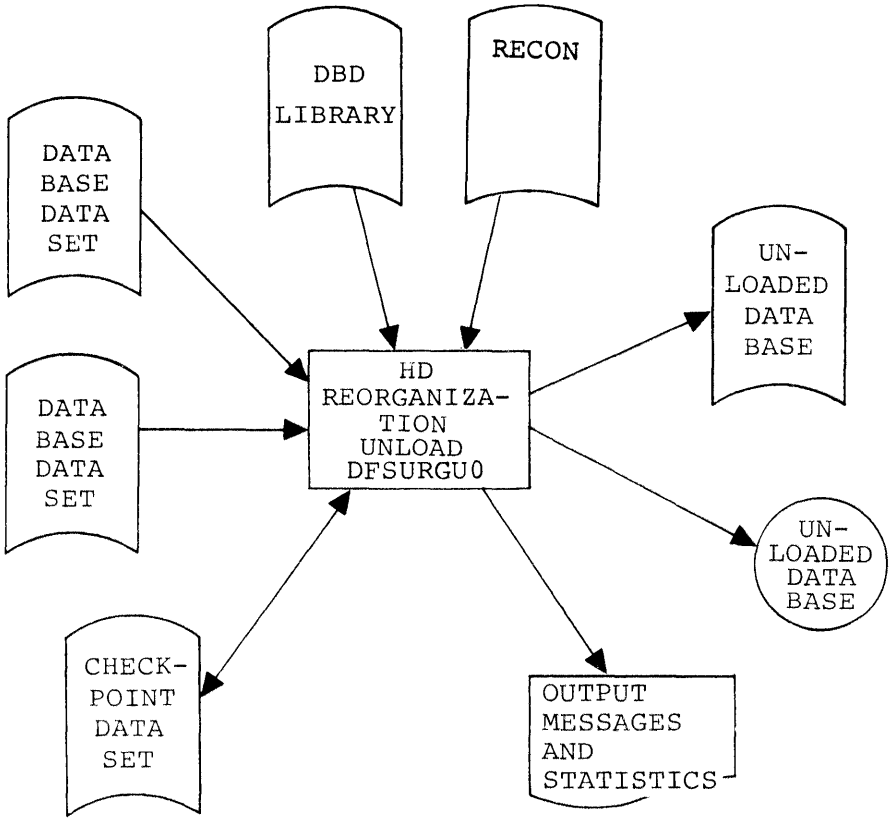


Figure 51. HD Reorganization Unload Utility

JCL REQUIREMENTS

The HD Reorganization Unload utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

The output from the HD Reorganization Unload utility is an operating system variable blocked sequential data set. Because output is blocked to the maximum size the output device can handle, standard labels must be used on output volumes.

EXEC

This statement must be in the following form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGU0,dbdname'
```

where the parameters ULU and DFSURGU0 describe the utility region and dbdname is the name of the DBD which describes the data base to be reorganized. The normal IMS/VS positional parameters such as SPIE, BUF, and DBRC can follow dbdname. See the DBBBATCH or DLIBATCH procedures in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD

Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBD that describes the data base to be reorganized (that is, DSN=IMSVS.DBDLIB,DISP=SHR). This data set must reside on a direct access device.

SYSPRINT DD
Defines the message and statistics output data set. The data set can reside on a tape, direct access device, or printer, or be routed through the output stream. DCB parameters specified for this data set are RECFM=FBA and LRECL=121. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 121.

DFSUCKPT DD
Defines the data set to be used to take checkpoints. If checkpoints are not required, this statement should not be supplied. This data set normally resides on a direct access device; however, a tape volume can be used.

DFSURSRT DD
Defines the checkpoint data set if a restart is to be attempted. This statement should be omitted if a restart is not wanted. If a restart is to be attempted, the statement should reference the same data set that the DFSUCKPT DD statement referenced when the last checkpoint was taken. This data set normally resides on a direct access device; however, a tape volume can be used.

DFSURGU1 DD
Defines the primary output data set. This DD statement must be supplied. The data set can reside on either tape or a direct access device.

DFSURGU2 DD
Defines the secondary output data set. This DD statement should only be supplied if two copies of the output are desired. The data set can reside on either tape or a direct access device.

database DD
Defines the data base data set to be reorganized. One statement must be present for each data set that is named in the DBD that describes the data base being reorganized. The ddname must match the ddname in the DBD.

If this is a HIDAM data base, DD statements must also exist for the data sets which represent the index. The DD statements used to relate to the index must contain ddnames specified in the DBD for the index data base. No DD statements are required for whatever secondary indexes may be associated with this data base.

This data set must reside on a direct access device.

DFSVSAMP DD
Describes the data set that contains the buffer pool information required by the DL/I Buffer Handler. This DD statement is required. For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.

The data set can reside on a tape, direct access device, or card reader, or be routed through the input stream.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If either statement is supplied, any return code greater than 4 will cause a U347 abend. If both statements are present, the last occurrence will be used for the dump.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC (Data Base Recovery Control) RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

Notes:

1. The checkpoint facility (see the DFSURSRT DD statement) writes a special checkpoint record to the checkpoint data set and to the output data set(s). If a restart is required, the checkpoint record is obtained from the checkpoint data set, the output volumes are positioned, and the proper position is established within the data base. The statistics table records are read, and the main storage tables are properly initialized. The program then continues with normal processing.

2. Note that multiple copies of the data base can be produced (see the DFSURGU2 DD statement). The advantage in specifying two copies is that if an I/O error occurs during execution, the utility continues to completion on the other copy. Performance would be somewhat diminished in this instance, but a total rerun would not be necessary.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	Data base unload successful.
4	One or more warning messages issued.
8	Serious error has occurred and/or copy 1 has an I/O error.
12	Possible mixed warning and serious messages issued.
16	Data base unload not successful.

EXAMPLES

Example 1

In this example, a HIDAM data base is to be reorganized using the checkpoint facility and two output copies. A restart is not requested. Three data base DD statements are provided: one is for the HIDAM OSAM data set, and two are for the ISAM and OSAM data sets of the Index data base used with HIDAM. If this were the unload of a single data set group HDAM data base, only one DD statement would be necessary for access to the data base.

```
//HDREORG    JOB 1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGU0,DI32DB02',
//          REGION=512K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1210
//DFSUCKPT   DD  DSN=IMSVS.CHKPT,UNIT=SYSDA,
//   SPACE=(TRK,(50)),VOL=SER=222222,DISP=(NEW,KEEP)
//DFSURGU1   DD  DSN=IMSVS.UNLOAD1,UNIT=TAPE,
//   VOL=SER=TAPE11,LABEL=(,SL),DISP=(NEW,KEEP)
//DFSURGU2   DD  DSN=IMSVS.UNLOAD2,UNIT=TAPE,
//   VOL=SER=TAPE21,LABEL=(,SL),DISP=(NEW,KEEP)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=SYSDA,
//   VOL=SER=DB0001,DISP=OLD
//HDINDEXI   DD  DSN=DATABASE.INDEXI,UNIT=SYSDA,DCB=DSORG=IS,
//   VOL=SER=DB0002,DISP=OLD
//HDINDEXO   DD  DSNAME=DATABASE.INDEXO,UNIT=SYSDA,
//          VOLUME=SER=DB0003,DISP=OLD
//DFSVSAMP   DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

Note: The HDPAYROL DD statement is for the OSAM data set of the HIDAM data base. The HDINDEXI DD statement is for the ISAM data set of the index data base. The HDINDEXO DD statement is for the OSAM data set of the index data base.

The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

In this example, execution of Example 1 is to be restarted after an abnormal termination. The checkpoint data set is employed in the restart.

```
//HDREORG    JOB 1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGU0,DI32DB01',
//         REGION=250K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A DCB=BLKSIZE=1210
//DFSUCKPT   DD  DSN=IMSVS.CHKPT,UNIT=SYSDA,      [Note 1]
//   VOL=SER=222222,DISP=(OLD,KEEP)
//DFSURSRT   DD  DSN=IMSVS.CHKPT,UNIT=SYSDA,      [Note 1]
//   VOL=SER=222222,DISP=(OLD,KEEP)
//DFSURGU1   DD  DSN=IMSVS.UNLOAD1,UNIT=TAPE,     [Note 2]
//   LABEL=(,SL),VOL=(,,,2,SER=(TAPE11,TAPE12)),
//   DISP=(MOD,KEEP)
//DFSURGU2   DD  DSN=IMSVS.UNLOAD2,UNIT=TAPE,     [Note 2]
//   LABEL=(,SL),VOL=(,,,2,SER=(TAPE21,TAPE22)),
//   DISP=(MOD,KEEP)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=SYSDA,
//   VOL=SER=DB0001,DISP=OLD
//HDINDEXI   DD  DSN=DATABASE.INDEX1,UNIT=SYSDA,DCB=DSORG=IS,
//   VOL=SER=DB0002,DISP=OLD
//HDINDEXO   DD  DSNAME=DATABASE.INDEX0,UNIT=SYSDA,
//         VOLUME=SER=DB0003,DISP=OLD
//DFSVSAMP   DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

Notes:

1. The DFSURSRT DD statement and the DFSUCKPT DD statement can reference the same data set. If restart is successful, the old checkpoint record is overwritten by the next checkpoint taken.

2. The primary and secondary output DD statements were changed to supply two volumes; only one volume was supplied by the previous execution of the utility. This assumes the previous abnormal termination was caused by an output I/O error that might, for example, have occurred at the end of the volume because there were no additional volumes available.

   Because the program does not differentiate between various causes of termination, it positions the volume in use at the time the checkpoint was taken to the applicable checkpoint record. It then issues a FEOV to cause volume-switching and continues with the new output volume.

   To avoid considerable tape handling on restart of a multivolume output execution, the volumes that have completed normally should be removed from the DD statements before submitting the job. The program opens the output and checks the volume currently mounted to ensure that it was the volume mounted when the checkpoint was taken. If it is not that volume, it issues a FEOV to get the next volume mounted. This could obviously result in a large amount of tape handling.

OUTPUT MESSAGES AND STATISTICS

The HD Reorganization Unload utility provides output messages and statistics. An example of the messages and statistics obtained from this utility, accomplished by explanatory information, is shown in Figure 52 on page 192.

HIERARCHICAL DIRECT DATA BASE REORGANIZATION UNLOAD

DFS353I WARNING - NO CHK PNT DATA SET SUPPLIED. NO CHECK POINTS TAKEN
DFS350I NO CHECK POINT INPUT. NORMAL UNLOAD REQUESTED
DFS344I DDNAME - DFSURGU2 - NOT SUPPLIED. 1 COPY CREATED
DATA BASE - DH41DB02 HAS BEEN UNLOADED
COPY 1 ON VOLUME(S) - STORAGE
DFS352I NO ERRORS DETECTED - DATA BASE UNLOAD SUCCESSFUL

DATA BASE STATISTICS

SEGMENT LEVEL STATISTICS						RECORD LEVEL STATISTICS	
MAXIMUM TWINS	AVERAGE TWINS	MAXIMUM CHILDREN	AVERAGE CHILDREN	SEGMENT NAME	SEGMENT LEVEL	TOTAL SEGMENTS BY SEGMENT TYPE	AVERAGE COUNT PER DATA BASE RECORD
1	1.00	15	6.00	A11NXXXX	1	10	1.00
1	1.00	1	1.00	AB2PG1XX	2	10	1.00
1	1.00	0	0.00	ABCTJMXX	3	10	1.00
4	1.30	5	1.15	AD2TGIJK	2	13	1.30
5	1.15	0	0.00	ADEPAFXX	3	15	1.50
3	1.20	0	0.00	AF2TADFX	2	12	1.20

TOTAL SEGMENTS IN DATA BASE = 70 AVERAGE DATA BASE RECORD LENGTH = 816 BYTES

Figure 52. Example of Output Messages and Statistics—HD Reorganization Unload Utility

Following the page heading are the various messages generated as a result of the options selected for this execution. An explanation of all numbered messages can be found in IMS/VS Messages and Codes Reference Manual.

Although not shown in the example, the message "DATA BASE - databasename HAS BEEN UNLOADED" appears on every execution of this program.

The message "COPY 1 ON VOLUME(S) - volser1" would appear if the primary output data set successfully completed. The list of volume serial numbers indicates which volumes were used and the order of their use. If a second copy was requested and successfully completed, the message "COPY 2 ON VOLUME(S) - volser2" would appear. The heading "DATA BASE STATISTICS" follows the messages.

The subtitles "Segment Level Statistics" and "Record Level Statistics" denote the type of statistics under their respective headings.

Under Segment Level Statistics, the fields, from left to right, are as described below.

- Maximum twins

  This field contains the maximum number of segments of this type encountered under an immediate parent segment. At the root level, this value is always 1.

- Average twins

  This field contains the average number of segments of this type encountered under an immediate parent segment. This value is carried out to 2 decimal places.

- Maximum children

  This field contains the maximum number of child segments (at all subordinate levels) under a given parent.

- Average children

  This field contains the average number of child segments (at all subordinate levels) under a given parent. This value is carried out to 2 decimal places. Note that the lowest level segment in any hierarchic path will have a value of zero in this field type.

The next two fields are the segment name to which this line of statistics applies, and the hierarchic level of this segment in the data base. Note that the segment descriptions are mapped from top to bottom, in the same order in which they were described in the DBD.

Under Record Level Statistics, the fields, from left to right, are as described below.

- Total segments by segment type

  This field contains the count of the number of occurrences of this segment type in the entire data base. Note that the count field in the level 1 segment type reflects the total number of data base records (root segments) in the data base.

- Average count per data base record

  This field contains a count of the average number of occurrences of this segment type within a given data base record. The value is carried to 2 decimal places.

Following the individual segment type statistics, a count is written of the total number of all segments in the data base and the average data base record length in bytes. The average data base record length includes both the data length and the prefix size. Note that the product of the number of segments at level 1 times the average data base record length gives the total number of bytes in the data base. Because all physically stored records may not use all available data positions, this figure can only be used as an approximation of the data set space required.

Note: When variable-length segments are involved, the calculation of the average data base record length is based on the maximum possible segment length.

HD REORGANIZATION RELOAD UTILITY (DFSURGL0)

The HD Reorganization Reload utility can be used to: (a) reload an HDAM, HIDAM or HISAM data base from a data set created by the HD Unload utility, and (b) create work data sets (if the data base that is reloaded includes logical relationships or secondary indexes) that are used as input to the logical relationship resolution utilities.

Sequence checking is performed on all applicable segments of the data base records.

If logical relationships or secondary indexes exist, a work data set is created that must be used later as input to the Data Base Prefix Resolution utility to resolve any logical or secondary index relationships that exist. Refer to "Rules and Restrictions" under "HD Reorganization Unload Utility (DFSURGU0)" on page 184.

The following must be accomplished before executing the HD Reorganization Reload utility:

- Assemble and link-edit the new DBD into the IMS/VS DBD library;
- If adding or deleting new segments and logical relationships are contained within the data base, rerun the Prereorganization utility against the new DBD;
- If the DBD name is changed and the DBD contains logical relationships, rerun the Prereorganization utility against the new DBD.
- For information on scratching and allocating ISAM/OSAM data sets, see "Allocation for OSAM Data Sets" under "Designing the IMS/VS Online System" in IMS/VS System Administration Guide.
- There are two cases in which the HD Reload utility should not be used:
  - When changing from bidirectional virtual pairing to bidirectional physical pairing, if any logical child segments have been deleted from either the physical or logical path but not from both paths, and
  - When changing a real logical child from one logically related data base to another.

For further information on using the HD Unload utility, see "Modifying Your Data Base" in IMS/VS Data Base Administration Guide.

The functions of this utility can be performed by the Utility Control Facility. Refer to "Utility Control Facility" for a description of its operation.

A flow diagram of the HD Reorganization Reload utility is shown in Figure 53 on page 195.

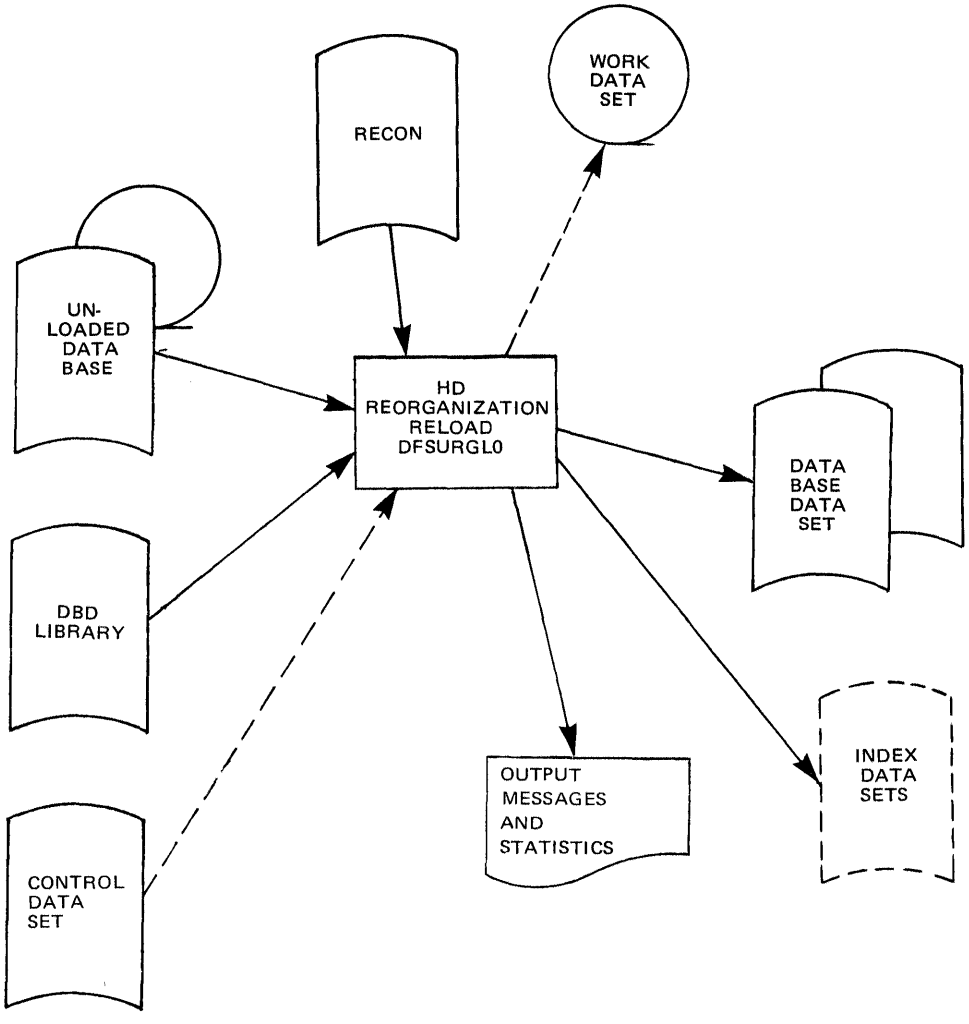


Figure 53. HD Reorganization Reload Utility

JCL REQUIREMENTS

The HD Reorganization Reload utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGL0,dbdname'
```

where the parameters ULU and DFSURGL0 describe the utility region and dbdname is the name of the DBD which includes the data base to be reloaded. The normal IMS/VS positional parameters such as SPIE, BUF, and DBRC can follow dbdname. See the DBBBATCH or DLIBATCH procedures in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Describes the library containing the DBD referenced in the EXEC statement PARM field. (Normally, this is IMSVS.DBDLIB.) This data set must reside on a direct access device.

SYSPRINT DD
Defines the message output data set. The data set can reside on a tape, or direct access device, or be routed through the output stream.

DFSUINPT DD
Describes the input data set containing the data to be reloaded. This is the data set created by the HD Reorganization Unload utility. The data set must reside on either tape or a direct access device.

DFSURWF1 DD
Describes the work data set to be created during reload that will be used as input by the Prefix Resolution utility (DFSURG10) to resolve logical or secondary index relationships. It can be specified as DUMMY if the data base being reloaded is not involved in a logical relationship or with a secondary index.

The DCB parameters for the DD statement must include RECFM=VB and BLKSIZE specified to be the same as that specified for the work data set of the user's initial load program or for the Data Base Scan utility (DFSURGS0). A value of LRECL=900 is recommended, but a smaller value (as small as 300) can be used if no secondary indexes are present.

The data set must reside on either tape or a direct access device.

database DD
Defines the data base data set to be reorganized. One statement of this type must be present for each data set that appears in the DBD which describes this data base. The ddname must match the ddname in the DBD.

If this is a HIDAM data base, DD statements must also exist for the data sets which represent the index. The DD statements which relate to the index must contain ddnames specified in the DBD for the index data base. No DD statements are required for whatever secondary indexes may be associated with this data base.

This data set must reside on a direct access device.

DFSURCDS DD
Defines the control data set for this program. The data set must be the output generated by the Prereorganization utility (DFSURPR0). This DD statement must be included if logical relationships exist.

This data set must reside on either tape or a direct access device.

DFSVSAMP DD
Describes the data set that contains the buffer pool information required by the DL/I buffer handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pool" in IMS/VS Installation Guide.)

Note: The buffer pool size will affect performance when reloading a data base. For best results, the buffer pool should contain enough blocks to hold a data base record (a root segment and all its dependents). Blocks already created will be referenced when a segment is inserted, that is, pointed to be a parent or twin in a prior block. If many segments were inserted between these two segments, the buffer containing the parent or twin may have been written to the data set and will have to be read in again.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If either statement is supplied, any return code greater than 0 will cause a U355 abend. If both statements are present, the last occurrence will be used for the dump.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

RETURN CODES

The following return codes are provided at program termination.

Code	Meaning
0	Data base reload successful
4	There were no segments to reload
8	Reload count differs from unload count
16	Data base reload unsuccessful.

EXAMPLES

Example 1

This example shows the JCL for an HDAM reorganization reload.

```
//HDRELOAD   JOB 1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGL0,DH32DB01',
//           REGION=250K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=OLD
//SYSPRINT   DD  SYSOUT=A
//DFSUINPT   DD  DSN=IMSVS.UNLOAD1,UNIT=TAPE,
//   VOL=SER=TAPE11,LABEL=(,SL),DISP=OLD
//DFSURWF1   DD  DSN=IMSVS.WRKTAPE1,UNIT=TAPE,
//   VOL=SER=WKTAPE,LABEL=(,SL),DISP=(NEW,KEEP),
//   DCB=(BLKSIZE=1008,LRECL=900,RECFM=VB)
//HDSKILLS   DD  DSN=DATABASE.SKILLS,UNIT=SYSDA,
//   VOL=SER=DB0002,DISP=(NEW,KEEP),SPACE=(CYL,(10,10))
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=SYSDA,DISP=(OLD,KEEP),
//   VOL=SER=IMSMSC
//DFSVSAMP   DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

This example shows the JCL for a HIDAM reorganization reload. Note that the primary index data base data sets are also described.

```
//HIRELOAD   JOB 1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGL0,HD32DB02',
//           REGION=250K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=OLD
//SYSPRINT   DD  SYSOUT=A
//DFSUINPT   DD  DSN=IMSVS.UNLOAD1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=TAPE11,DISP=OLD
//DFSURWF1   DD  DSN=IMSVS.WRKTAPE1,UNIT=TAPE,
//   VOL=SER=WKTAPE,LABEL=(,SL),DISP=(NEW,KEEP),
//   DCB=(BLKSIZE=1008,LRECL=900,RECFM=VB)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=SYSDA,
//   VOL=SER=DB0001,DISP=OLD
//HDINDEXI   DD  DSN=DATABASE.INDEXI,UNIT=SYSDA,
//   VOL=SER=DB0003,DCB=DSORG=IS,DISP=(NEW,KEEP),
//   SPACE=(CYL,(5))
//HDINDEXO   DD  DSN=DATABASE.INDEXO,UNIT=SYSDA,
//   VOL=SER=DB0003,DISP=(NEW,KEEP),SPACE=(CYL,(10,10))
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=SYSDA,DISP=(OLD,KEEP),
//   VOL=SER=IMSMSC
//DFSVSAMP   DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//RECON3   DD  DSN=RECON3,DISP=SHR
```

Example 3

This example shows the JCL for a HIDAM VSAM data base unload and reload.

```
//HDREORG    JOB 1,1,MSGLEVEL=1
//UNLOAD EXEC   PGM=DFSRRC00,PARM='ULU,DFSURGU0,DHVBTZ01'
//STEPCAT    DD DISP=SHR,DSN=IMSCAT1
//STEPLIB    DD DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD DSN=IMSVS.RESLIB,DISP=SHR
//IMS        DD DSN=IMSCT.V1,DBDLIB,DISP=SHR
//             DD DSN=IMSCT.V1,PSBLIB,DISP=SHR
//DFSURGU1   DD DSN=UNLOAD,UNIT=SYSDA,SPACE=(TRK,(10,5)),              X
//             DISP=(NEW,PASS)
//PRINTDD    DD SYSOUT=A
//SYSPRINT   DD SYSOUT=A
//SYSUDUMP   DD SYSOUT=A
//DXSK0302   DD DSN=VVDX0302,DISP=OLD
//DXSK0301   DD DSN=VVDX0301,DISP=OLD
//DHSK0301   DD DSN=VVDH0301,DISP=OLD
//DFSVSAMP   DD *
2048,10
/*
//STP98  EXEC   PGM=IDCAMS
//STEPCAT    DD DISP=SHR,DSN=IMSCAT1
//SYSPRINT   DD SYSOUT=A
//SYSABEND   DD SYSOUT=A
//VSA        DD UNIT=SYSDA,DISP=OLD,VOL=SER=VSIMSA
//SYSIN      DD *
 DELETE VVDH0301
 DELETE VVDX0301
 DELETE VVDX0302
 DEF CL (NAME(VVDX0301) CYL(1 1) RECSZ(16 16) VOL(VSIMSA) IXD-
      CISZ(2048) FSPC(25) KEYS(10 5))
 DEF CL (NAME(VVDH0301)   TRK(10 5) RECSZ(2041 2041) VOL(VSIMSA) NIXD-
  CISZ(2048)  )
 DEF CL (NAME(VVDX0302)   TRK(10 5) RECSZ(2041 2041) VOL(VSIMSA) NIXD-
  CISZ(2048)  )
//RELOAD EXEC   PGM=DFSRRC00,PARM='ULU,DFSURGL0,DHVBTZ01'
//STEPCAT    DD DISP=SHR,DSN=IMSCAT1
//STEPLIB    DD DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD DSN=IMSVS.RESLIB,DISP=SHR
//IMS        DD DSN=IMSCT.V1,DBDLIB,DISP=SHR
//             DD DSN=IMSCT.V1,PSBLIB,DISP=SHR
//DFSUINPT   DD DSN=UNLOAD,UNIT=SYSDA,DISP=(OLD,PASS)
//PRINTDD    DD SYSOUT=A
//SYSPRINT   DD SYSOUT=A
//SYSUDUMP   DD SYSOUT=A
//DHSK0301   DD DSN=VVDH0301,DISP=OLD
//DXSK0302   DD DSN=VVDX0302,DISP=OLD
//DXSK0301   DD DSN=VVDX0301,DISP=OLD
//DFSVSAMP   DD *
2048,10
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//RECON3   DD  DSN=RECON3,DISP=SHR
```

OUTPUT MESSAGES AND STATISTICS

The HD Reorganization Reload utility provides output messages and statistics. Figure 54 is an example of the messages and statistics obtained from this utility, accompanied by explanatory information.

HIERARCHICAL DIRECT DATA BASE REORGANIZATION RELOAD

SEGMENT LEVEL STATISTICS

SEGMENT NAME	SEGMENT LEVEL	TOTAL SEGMENTS BY SEGMENT TYPE RELOADED	DIFFERENCE
A11NXXXX	1	10	
AB2PG1XX	2	10	
ABCTJMXX	3	10	
AD2TG1JK	2	13	
ADEPAFXX	3	15	
AF2TADEX	2	12	

TOTAL SEGMENTS IN DATA BASE

UNLOADED	RELOADED	DIFFERENCE
70	70	

DFS354I NO ERRORS DETECTED. DATA BASE RELOAD SUCCESSFUL

Figure 54. Example of Output Messages and Statistics—HD Reorganization Reload Utility

Following the messages, the heading "SEGMENT LEVEL STATISTICS" appears. The fields, from left to right, are as described below.

- Segment name

  The segment name to which this line of statistics applies.

- Segment level

  The hierarchic level of this segment in the data base. It should be noted that the segments are mapped from top to bottom, in the same order they were described in the DBD, and in the same order they appeared in the HD Unload statistics.

The next two fields appear under the heading "TOTAL SEGMENTS BY SEGMENT TYPE."

- Reloaded

  The number of occurrences of this segment type in the reload of the entire data base.

- Difference

  This field is blank if the reload count equals the unload count for this segment. If it is not blank, a '+' will be to the right if there were more counted in reload than in unload; a '-' will be there if there were more counted in unload than in reload.

Following the individual segment type statistics, the heading "TOTAL SEGMENTS IN DATA BASE" appears. The fields, from left to right, are as follows:

- Unloaded

  The total number of all segments in the data base counted by the HD Unload utility.

- Reloaded

  The total number of all segments in the data base counted by the HD Reload utility.

- Difference

  This field is blank if the counts by reload and unload are equal. If they are not equal, the difference is printed.

DATA BASE SURVEYOR UTILITY (DFSPRSUR)

The Data Base Surveyor utility (DFSPRSUR) scans all or part of an HDAM or HIDAM data base and produces a report describing the physical organization of the data base, which aids the user in determining the need for reorganization. In addition, the Surveyor utility identifies the size and location of free space areas, which can receive reorganized records during a partial data base reorganization. See the description of the Partial Data Base Reorganization utility later in this chapter for more information on how to use the Surveyor utility output.

The Surveyor utility can run as a batch message processing program (BMP) against an online data base, or as a batch program.

INPUT

The data base to be analyzed can be shared when Surveyor is executing as a BMP. Input utility control statements direct the utility to the desired sections (key ranges or block number ranges) of a particular data base. See "JCL Requirements" and "Utility Control Statements" for a detailed explanation of all required input. See "Examples" for examples of coding the JCL and utility control statements.

A PSB containing a PCB for the data base being surveyed must be defined for the use of the Surveyor utility. (More than one data base PCB may be contained in this PSB.) The data base PCB must contain SENSEG statements for all segments defined in the corresponding DBD. The PSB should specify PROCOPT=G. When the Surveyor is to be executed as a BMP, OLIC=YES must be specified on the PSBGEN statement, and the PSB must be defined to the IMS/VS online control region.

OUTPUT

The Surveyor utility provides the following as output:

- Data base Surveyor report
- Return codes
- Error messages

For each partition of a range specified, the Surveyor report contains such statistics as the distribution of the number of blocks accessed to read a data base record, the average number of blocks accessed per record, the average size of a record, the total size of all records accessed, and the actual number of records read. Also, the report may indicate the total amount of free space, the distribution of contiguous free space areas by size, or the location of the largest areas of contiguous free space. See "Examples" for an annotated sample Surveyor report. See "Return Codes" for a list of Surveyor utility return codes and their meanings. See IMS/VS Messages and Codes Reference Manual for explanations of all messages produced by the Surveyor utility.

For each range specified, the utility divides the number of blocks in the secondary data base into 10 parts and lists the portion of segments in each of the 10 parts that belong to the specified range. For example, if there are 56 blocks, the utility divides the number of blocks into 10 parts (ranging from 1 through 6, 7 through 12...15 through 56) and lists the portion of segments in each part that belong to the specified range.

JCL REQUIREMENTS

The Data Base Surveyor utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define input and output data sets are required.

EXEC

To run the Surveyor utility as a batch program, specify:

```
PGM=DFSRRC00,PARM='DBB,DFSPRSUR,...' (if prebuilt
blocks are being used; otherwise,
PARM='DLI,DFSPRSUR,...')
```

To run the Surveyor utility as a batch message processing program, specify:

```
PGM=DFSRRC00,PARM='BMP,DFSPRSUR,...'
```

The normal IMS/VS positional parameters can follow the program name in the PARM field.

See "Member Name DBBBATCH," "Member Name DLIBATCH," and "Member Name IMSBATCH" in IMS/VS System Programming Reference Manual for additional information on executing a batch or batch message processing program.

STEPLIB DD

Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD

Points to an authorized library that contains the IMS/VS SVC modules.

SYSIN DD
Defines the input control data set for this program. The data set can reside on card reader, tape, or direct access device, or be routed through the input stream. LRECL and BLKSIZE must both be 80.

IMS DD
Defines the libraries containing the DBD and PSB that describe the data base to be analyzed. These data sets must reside on a direct access device. This statement is required and must always define the DBD library: the PSB library is only required when PARM=DLI is specified.

IMSACB DD
Defines the library containing the ACB that describes the data base to be analyzed. This data set must reside on a direct access device. This statement is required only when PARM=DBB is specified.

SYSPRINT DD
Defines the message and report output data set. The data set can reside on tape, direct access device, or printer, or be routed through the output stream. This statement is always required.

DCB parameters specified for this data set are RECFM=FBM and LRECL=121. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 121.

IEFRDER DD
Defines the IMS/VS log data set. This statement is required when Surveyor executes as a batch program, but may be specified as DUMMY.

database DD
Defines the data base to be analyzed. The ddname must match the ddname in the DBD. This statement is only required when Surveyor executes as a batch program. (When Surveyor executes as a BMP, the data base data sets must be defined in the control region JCL.)

DFSVSAMP DD
Describes the data set that contains the buffer pool information required by the DL/I Buffer Handler. This DD statement is required when Surveyor executes as a batch program. (For additional information on control statement format and buffer pool structure see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.)

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence will be used for the dump.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS

Input statements are used to describe processing options for the Data Base Surveyor utility. The first character of a statement must start before column 17. Multiple keywords cannot be specified on a single statement. If there are any blanks embedded within an operand, any characters following the blank(s) are assumed to be comments. Continuation statements are allowed when a keyword's operands extend beyond one input card. There must be a nonblank character in column 72 to continue a statement. Comment-only statements are specified by placing an asterisk in column 1. The format of the Surveyor utility control statement is:

```
DBNAME=dbdname
{ KEYRANGE= ( {ALL          } { EOD          } ,partitionsize)
            ( {lowkeyvalue, } { highkeyvalue } }
  FROMAREA= ( {ALL            } { EOD            } ,partitionsize)
            ( {lowblocknumber, } { highblocknumber } }
  TOAREA=   (ddname,{ALL            } { EOD            } ,partitionsize) }
                    {lowblocknumber, } { highblocknumber } }
[ SAMPLE={1} ]
         {n}
[ MODE= {BATCH  } ]
        {ONLINE }
```

where:

DBNAME=
Specifies the data base to be surveyed to find data base distortions and/or free space. This operand must be the name of a DBD with HD organization. DBNAME is required and must appear only once.

KEYRANGE=
Specifies the range of keys to be analyzed for data base distortions. Only one KEYRANGE definition is allowed. If KEYRANGE is specified, FROMAREA or TOAREA may not be specified on other input statements in the same job stream. KEYRANGE is invalid if the data base is HDAM. The operands are the root segment keys or generic keys, with a maximum of 255 bytes each. Keys may be expressed in hexadecimal by preceding the value with an X and enclosing them in quotation marks; for example, KEYRANGE=(X'C8C5E7',X'D2C5E8',1000). The high key value may be specified as "EOD" if the upper limit is the end of the data base. The low key value may be specified as "ALL" and the high key value omitted, in which case, the range will be the entire data base.

The partition size is specified as the number of data base records to be included in each partition of the range. The number specified must be from 1 to 9999. The Surveyor utility will produce statistics for each partition of the range and also for the entire range.

FROMAREA=
Specifies one range of blocks to be analyzed for data base distortions. Only one FROMAREA definition is allowed. If FROMAREA is specified, KEYRANGE or TOAREA may not be specified on other input statements in the same job stream. FROMAREA is invalid if the data base is HIDAM. The operands are block numbers in the root addressable area. The high block number may be specified as "EOD" if the upper limit is the last block in the root addressable area. The low block number may be specified as "ALL" and the high

block number omitted, in which case, the range will be the entire root addressable area.

The partition size is specified as the number of blocks of root addressable area to be included in each partition of the range. The number specified must be from 1 to 9999. Surveyor will produce statistics for each partition of the range and also for the entire range.

TOAREA=
Specifies one area of the data base to be analyzed for free space. TOAREA keywords are defined to be a range within a data set group. Up to ten TOAREA definitions may be specified for a single data base. If TOAREA is specified, KEYRANGE or FROMAREA may not be specified on other input statements in the same job stream. DDname must be the name of a DD statement defining a data set in the data base being surveyed.

The block numbers can be coded as any block number within the limits of the data set, "ALL," or "EOD." The low block number has a minimum value of two. If the low block number is specified as less than two, two will be assumed. If the low block number is specified as "ALL," with the high block number omitted, the range will be the entire data set group. If the high block number is specified as "EOD," the upper limit of the range is the last block of the data set group.

The partition size is specified as the number of blocks to be included in each partition of the range. The number specified must be from 1 to 9999. Surveyor will produce statistics for each partition of the range and also for the entire range.

SAMPLE=
Specifies that the utility is to sample only a part of each range. The operand for this keyword is specified as a number between 1 and 1000. The operand specifies number of records for KEYRANGE's and FROMAREA's and number of blocks for TOAREA's. Every nth record (or block) will be accessed and intervening records (or blocks) will be ignored for analysis. For example, if n=10, then the first record/block, the 11th record/block, and the 21st record/block, etc., will be accessed and used for reporting data base storage information. If the SAMPLE keyword is omitted, the default will be to access every record (or block) in the range.

MODE=
Specifies whether Surveyor will be run as a BMP (MODE=ONLINE) or batch program (MODE=BATCH). BATCH is the default.

RETURN CODES

The Surveyor utility produces the following return codes at program termination:

Code	Meaning
0	No errors detected
4	Warning message issued
8	Abnormal program termination

EXAMPLES

Example 1

This example shows the JCL and utility control statements required to execute DFSPRSUR as a BMP to analyze portions of a data base using the KEYRANGE keyword. Note that the DD statements for the data base being analyzed must be included in the IMS/VS control region JCL.

```
//SUREX1   JOB  1,1,MSGLEVEL=(1,1)
//         EXEC PGM=DFSRRC00,PARM=(BMP,DFSPRSUR,PRPSB01Y)
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
        MODE=ONLINE
        DBNAME=PR01RW00
        KEYRANGE=(000300,000500,10)
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

This example shows the JCL and utility control statements required to execute DFSPRSUR as a batch program to analyze portions of a data base, using the FROMAREA keyword.

```
//SUREX2   JOB 1,1,MSGLEVEL=(1,1)
//         EXEC PGM=DFSRRC00,PARM=(DLI,DFSPRSUR,PRPSB11Y)
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//         DD  DSN=IMSVS.PSBLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SYSUDUMP DD  SYSOUT=A
//PR11DD   DD  DSN=PR11RW00,DISP=(OLD,KEEP)
//SYSIN    DD  *
    MODE=BATCH
    DBNAME=PR11RW00
    FROMAREA=(1,EOD,5)
    SAMPLE=2
/*
//DFSVSAMP DD  *
1024,4
4096,8
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Figure 55 is a sample partition report produced by Example 2 execution. One of these reports is produced for each partition.

```
SURVEYOR FROMAREA PARTITION REPORT FOR DBD PR11RW00

PARTITION BLOCK NUMBERS      1 TO      5

TOTAL NUMBER OF ROOTS =            7
TOTAL LENGTH OF SEGMENTS =      4412
AVERAGE LENGTH PER DBR =         630

SEGMENTS OF A DATA BASE RECORD (DBR) MAY SPREAD ACROSS SEVERAL BLOCKS FOR MANY REASONS.  THIS TABLE SUMS
THE NUMBER OF DATA BASE RECORDS BY THE NUMBER OF BLOCKS THEY OCCUPY.
# OF BLOCKS/SUM OF # DBR WHICH OCCUPY THIS # OF BLOCKS

NO. BLK    1       2       3       4       5       6-8     9-11    12-14   15-17    > 17
NO. DBR    5       1       1       0       0       0       0       0       0        0

THIS TABLE SHOWS THE NUMBER AND LENGTH OF  SEGMENTS IN EACH TENTH OF EACH DSG FOR THIS  DATA BASE.

  DSG                          BLOCK          TOTAL LENGTH       NUMBER          AVERAGE LENGTH      PERCENT OF
 DDNAME      BLKSIZE       LOW    HIGH       OF SEGMENTS      OF SEGMENTS       OF SEGMENTS       AREA OCCUPIED
PR11DD         1690          1-     6            1084             58                18                10.7
                             7-    12               0              0                 0                 .0
                            13-    18               0              0                 0                 .0
                            19-    24               0              0                 0                 .0
                            25-    30               0              0                 0                 .0
                            31-    36               0              0                 0                 .0
                            37-    42               0              0                 0                 .0
                            43-    48               0              0                 0                 .0
                            49-    54            2396            134                17                23.6
                            55-    56             932             52                17                27.6
```

Figure 55. Surveyor FROMAREA Partition Report

Figure 56 is a sample range report produced by Example 2 execution. One of these reports is produced to summarize the entire range.

```
SURVEYOR FROMAREA RANGE REPORT FOR DBD PR11RW00

RANGE BLOCK NUMBERS      1 to     50

TOTAL NUMBER OF ROOTS =           17
TOTAL LENGTH OF SEGMENTS =     12412
AVERAGE LENGTH PER DBR =         730

SEGMENTS OF A DATA BASE RECORD (DBR) MAY SPREAD ACROSS SEVERAL BLOCKS FOR MANY REASONS.  THIS TABLE SUMS
THE NUMBER OF DATA BASE RECORDS BY THE NUMBER OF BLOCKS THEY OCCUPY.
# OF BLOCKS/SUM OF # DBR WHICH OCCUPY THIS # OF BLOCKS

NO. BLK    1       2       3       4       5       6-8     9-11    12-14   15-17    > 17
NO. DBR   13       3       1       0       0       0       0       0       0        0
```

Figure 56. Surveyor FROMAREA Range Report

The various parts of the Surveyor reports are:

PARTITION BLOCK NUMBERS
Low and high block numbers in the partition.

KEYRANGE
Low and high key values in the partition.

TOTAL NUMBER OF ROOTS
Total number of roots in the partition.

AVERAGE LENGTH PER DBR
Average length of a data base record.

A table, which shows the relationship between the number of records and the number of blocks in which they are spread across, appears next. The various fields are described below.

NO. BLK
Heading line of number of blocks.

NO. DBR
Number of DBRs spread across the number of blocks in heading above.

A table that shows the characteristics of each tenth of each DSG in the data base being surveyed, appears next. The various fields are described below.

DSG DDNAME
DDNAME of DSG.

BLKSIZE
Block size of the DSG.

BLOCK LOW and HIGH
Low and high block numbers of this portion of the DSG.

TOTAL LENGTH OF SEGMENTS
Total length of all segments (found in partition being reported) which physically reside in this portion of this DSG.

NUMBER OF SEGMENTS
Number of segments (found in partition being reported) which physically reside in this portion of this DSG.

AVERAGE LENGTH OF SEGMENTS
Average length of segments (found in partition being reported) which physically reside in this portion of this DSG.

PERCENT OF AREA OCCUPIED
Percentage of area occupied by segments (found in partition being reported) that physically reside in this portion of this DSG. Note that segments belonging to data base records outside the range being reported may physically reside within the DSG area being reported but will not be reflected in this report.

Example 3

This example shows the JCL and utility control statements required to execute DFSPRSUR as a batch program to analyze and entire data base, using the TOAREA keyword.

```
//SUREX3   JOB 1,1,MSGLEVEL=(1,1)
//         EXEC PGM=DFSRRC00,PARM=(DLI,DFSPRSUR,PRPSB11Y)
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//         DD  DSN=IMSVS.PSBLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SYSUDUMP DD  SYSOUT=A
//PR01DD   DD  DSN=PR01RW00,DISP=(OLD,KEEP)
//SYSIN    DD  *
    MODE=BATCH
    DBNAME=PR01RW00
    TOAREA=(PR01DD,2,8,10)
    SAMPLE=2
/*
//DFSASAMP DD  *
1024,4
4096,8
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//RECON3   DD  DSN=RECON3,DISP=SHR
```

Figure 57 is a sample partition report produced by the Example 3 execution. One of these reports is produced for each partition.

```
     SURVEYOR TOAREA PARTITION REPORT FOR DBD PR01RW00 DSG PR01DD

PARTITION BLOCK NUMBERS      2 TO       6
TOTAL NUMBER OF FREE SPACE ELEMENTS    =        2
TOTAL AMOUNT OF FREE SPACE             =     6350
TOTAL NUMBER OF BLOCKS ACCESSED        =        3
AVERAGE AMOUNT OF FREE SPACE PER FSE   =     3175
PERCENT OF FREE SPACE TO TOTAL AREA    =     51.7


                    TEN LARGEST AREAS OF FREE SPACE
 SIZE     NUMBER    LOCATION (FIRST TEN)
  4084          1          2
  2266          1          6
```

Figure 57. Partial Reorganization TOAREA Partitions Report

Figure 58 is a sample range report produced by the Example 3 execution. One of these reports is produced to summarize the entire range.

```
     SURVEYOR TOAREA RANGE REPORT FOR DBD PR01RW00

RANGE BLOCK NUMBERS      2 TO      6
TOTAL NUMBER OF FREE SPACE ELEMENTS    =         2
TOTAL AMOUNT OF FREE SPACE             =      6350
TOTAL NUMBER OF BLOCKS ACCESSED        =         3
AVERAGE AMOUNT OF FREE SPACE PER FSE   =      3175
PERCENT OF FREE SPACE TO TOTAL AREA    =      51.7
```

Figure 58. Partial Reorganization TOAREA Range Report

The various parts of the reports are described here:

PARTITION BLOCK NUMBERS
Low and high block numbers in the partition.

TOTAL NUMBER OF FREE SPACE ELEMENTS
Total number of free space elements found in partition.

TOTAL AMOUNT OF FREE SPACE
Total amount of free space found in partition.

TOTAL NUMBER OF BLOCKS ACCESSED
Total number of blocks accessed in partition.

AVERAGE AMOUNT OF FREE SPACE PER FSE
Average amount of free space per free space element which was found in partition.

PERCENT OF FREE SPACE TO TOTAL AREA
Percentage of free space found in the partition blocks that were surveyed.

A table, which shows the 10 largest areas of free space, appears next. The various fields are described below.

SIZE
Size of free space element.

NUMBER
Number of free space elements of the size indicated under SIZE which were found in this partition.

LOCATION (FIRST 10)
Block number of the first 10 free space elements of the size indicated under SIZE which were found in this partition.

PARTIAL DATA BASE REORGANIZATION UTILITY (DFSPRCT1)

The Partial Data Base Reorganization utility can be used to reorganize user-specified ranges of an HDAM or HIDAM data base. (A "range" is either a group of HIDAM records with continuous key values or a group of HDAM records with continuous relative block numbers. A range is specified using a low and high pair of key or block number values.) The Data Base Surveyor utility can be used to aid in the selection of the ranges to be reorganized. For more information and samples of output, see the complete description of the Data Base Surveyor utility earlier in this chapter.

The partial Data Base Reorganization utility, in one execution, performs operations similar to several other reorganization utilities, such as:

- Unloading, in hierarchic sequence, all root segments within a specified range and all segments dependent on those roots
- Reloading those root and dependent segments (in hierarchic order) into specified contiguous free space
- Resolving all pointers relating to the reloaded segments, including:
  - Logically related segments in the same data base
  - Logically related segments in other data bases
  - Physical twin pointers of roots at boundaries of selected ranges

To accomplish the reorganization, two separate steps are executed: "Step 1" performs preorganization functions to check for user errors without requiring use of the data base; "Step 2" performs the unload/reload/pointer resolution functions with the data base offline. A detailed description of the input, output, and function of both steps follows.

RESTRICTIONS

The following limitations apply to this utility:

- Structural changes to the data base are not allowed.
- HISAM logically related data bases cannot have a direct pointer in a logical child or logical parent segment.
- Up to 49 related data bases may be processed.
- As many as 500 segment types may participate in a reorganization.
- Up to 500 scan and reload actions are allowed for pointer resolution.
- Up to 10 KEYRANGEs or FROMAREAs are allowed.
- Up to 10 TOAREAs are allowed after each FROMAREA or KEYRANGE.

Step 1 (Prereorganization)

The first step of partial data base reorganization performs initialization functions consisting of:

- Reading control cards that specify the range of records
- Creating control tables for use by Step 2
- Determining logically related data bases that may require pointer resolution
- Preparing a report

Step 1 requires, as input, the DBD of the data base to be reorganized and utility control statements defining the record ranges and sort options.

The primary output is the set of control tables to be used by Step 2 to perform the partial reorganization. Optionally, PSB source statements can be produced for creating a PSB for use in Step 2. (This PSB must contain PCBs for four required GSAM work data sets and will have to be assembled and link-edited before

Step 2; see the example at the end of this section.) After the PSB generation is done for the data bases to be reorganized, that process does not have to be repeated for subsequent reorganizations of the same data bases. Step 1 also produces two scan reports. The "required segment scan report" lists any logically related segments of logically related data bases that will be automatically scanned during Step 2 processing. The "optional segment scan report" lists any logically related segments of logically related data bases for which a scan is optional. (To select an optional segment for scanning, provide a SCANSEG control statement as Step 2 input.)

The scan examines every occurrence of all segment types being scanned and builds a work record for each occurrence. Step 2 uses these work records to speed its location of the segment occurrences that need pointer updating. If a scan is not performed for the optional scan segments, the prefix update phase of Step 2 must follow pointer chains to locate segments to be updated.

Scanning the optional segments has advantages when:

- Other segments in the same data base require the scan, or
- A large proportion of the optional scan segments contains pointers to required scan segments being moved in a logically related data base.

Additional Step 1 outputs include reports, possibly error messages, and a return code.

Note: Sometime before beginning Step 2, DBDs for the four GSAM work data sets must be created; follow the examples shown at the end of this section.

Step 2 (Unload/Reload/Pointer Resolution)

The second step reads the control tables produced by Step 1 and the user-supplied control statements. According to the specified record ranges, all segments (roots and their dependents) are unloaded in hierarchic order to an intermediate data set. The space within the data base that these records occupied is freed. Again according to the user's specification, the records are reloaded into ranges of free space within the data base and the new record locations saved in work records. Then, all logically related data bases are scanned for pointers to the records that have been moved: work records are created to designate where pointer resolution is required. All work records are then sorted (by OS/VS SORT) according to data base name and segment name. Finally, all pointers to the records having new locations are changed to point to the new locations.

Output from Step 2 includes the partially reorganized data base, as well as a report of what was done and a return code. In the case of an unsuccessful execution, an error message is issued.

Checkpoint/Restart

The Partial Data Base Reorganization utility has restart capabilities. Checkpoints are taken before the unload/reload phase, at the end of each sort phase, and at the beginning of the prefix update phases. A restart may be performed from any checkpoint. Before restarting, the Data Base Backout utility must be used to undo any changes made after the checkpoint was taken. (For more information, see the Data Base Backout utility in Chapter 7, "Data Base Recovery Utilities.")

JCL REQUIREMENTS

The Partial Data Base Reorganization utility is executed as two standard OS/VS jobs in an IMS/VS batch processing region. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define input and output data sets are required for each job.

Step 1

EXEC
Describes the program to be run and should be in this format:

```
PGM=DFSPRCT1
```

STEPLIB DD
Points to the IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

IMS DD
Defines the library containing the DBDs that describe the data base to be reorganized and all logically related data bases.

SYSPRINT DD
Defines the message and report output data set. The data set can reside on tape, a direct access device, or a printer, or be routed through the output stream. This statement is required.

DCB parameters for this data set are RECFM=FBA and LRECL=121. BLKSIZE must be provided on the DD statement and must be a multiple of 121.

SYSIN DD
Defines the input control data set for this job. The data set can reside on a card reader, tape, or direct access device, or be routed through the output stream. The LRECL must be specified as 80.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence will be used for the dump.

SYSPUNCH DD
Defines the data set that will contain the PSB source statements if the user elected to have them produced.

DCB parameters for this data set are RECFM=FBS and LRECL=80. BLKSIZE must be provided on this DD statement and must be a multiple of 80.

DFSPRCOM DD
Defines the data set that will contain the control tables that are to be used by Step 2.

Step 2

EXEC
Describes the program to be run and should be in this format:

```
PGM=DFSRRC00,PARM=(DLI,DFSPRCT2,psbname)
```

Where DLI describes the region, DFSPRCT2 names the Step 2 program, and psbname is the name of the PSB which includes the data base to be reorganized. The normal IMS/VS positional parameters can follow the psbname in the PARM field. See "Member Name DBBBATCH" or "Member Name DLIBATCH" in IMS/VS System Programming Reference Manual for

additional information on executing a batch processing program.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library which contains the IMS/VS SVC modules.

IMS DD
Defines the libraries containing the DBDs that describe the data base to be reorganized and all logically related data bases. This library must also contain the PSB named in the EXEC statement and the required GSAM DBDs.

DFSPRWF1 DD
Defines an intermediate work data set. DISP=NEW should be specified for this data set.

DFSPRWF2 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=18 and RECFM=FB. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001, in which case DISP=OLD should be specified. A GSAM DBD must be generated for this data set.

DFSPRWF3 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=18 and RECFM=FB. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001, in which case DISP=OLD should be specified. A GSAM DBD must be generated for this data set.

DFSPRWF4 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=1000 and RECFM=VB. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001, in which case DISP=OLD should be specified. A GSAM DBD must be generated for this data set.

DFSPRWF5 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=1000 and RECFM=VB. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001, in which case DISP=OLD should be specified. A GSAM DBD must be generated for this data set.

DFSPRWF6 DD
Defines an intermediate work data set. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001, in which case DISP=OLD should be specified.

DFSPRWF7 DD
Defines an intermediate work data set. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR002, in which case DISP=OLD should be specified.

DFSPRWF8 DD
Defines an intermediate work data set. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR003, in which case DISP=OLD should be specified.

DFSPRWF9 DD
Defines an intermediate work data set. DISP=NEW should be specified for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR004, in which case DISP=OLD should be specified.

DFSPRWFA DD
Defines an intermediate work data set. DISP=NEW should be specified for this data set.

SORTLIB DD
Defines the data set containing the load modules for the operating system SORT/MERGE program. This DD statement is required.

SORTWKnn DD
Defines intermediate storage data sets for the operating system Sort/Merge program. Refer to the appropriate operating system Sort/Merge manual regarding specification and size of intermediate storage data sets. These DD statements are required.

IEFRDER DD
Defines the IMS/VS log data set, which must reside on a direct access device. This statement is required.

Note: If this data set is specified as DD DUMMY, no checkpoint/restart capability is available.

SYSPRINT DD
Defines the message and report output data set. The data set can reside on tape, a direct access device, or a printer, or be routed through the output stream. This statement is required.

DCB parameters for this data set are RECFM=FBA and LRECL=121. BLKSIZE must be provided on this DD statement and must be a multiple of 121.

DFSPRCOM DD
Defines the data set created by Step 1 containing the control tables.

SYSIN DD
Defines the input control data set for this job. The data set can reside on a card reader, tape, or a direct access device, or be routed through the output stream.

SYSOUT DD
Defines the message output data set for SORT/MERGE. The DD name is the one specified during generation of invoked sort (normally, the DD name is SYSOUT). This data set can reside on a printer, tape, or a direct access device, or be routed through the output stream. This DD statement is required.

database DD
Defines the data base data sets to be reorganized, all logically related data base data sets, and all secondary index data sets. The ddnames must match those specified in the DBD.

If this is a HIDAM data base, DD statements for the index data sets must also be supplied. The ddnames must match those specified for the index data sets in the DBD.

These data sets must reside on a direct access device. DISP=OLD is recommended.

DFSVSAMP DD
Describes the data set that contains the buffer pool information required by the DL/I buffer handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pool" in IMS/VS Installation Guide.)

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence will be used for the dump.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS

Input statements are used to describe processing options for Partial Data Base Reorganization. The first character on a statement must start before column 17. Multiple keywords cannot be specified on a single statement. If there are any blanks on the input statement, with the exception of blanks embedded within an operand, any characters following the blank(s) are assumed to be comments. Continuation statements are allowed when a keyword's operands extend beyond one input card. Continuation is allowed only between operands of a keyword and not in the middle of an operand, with the exception of KEYRANGE key operands. A nonblank character must be placed in column 72 to continue a statement. The first character of a continuation statement must start in column 16. Comment-only statements are specified by placing an asterisk in column 1.

The following are the valid input keywords for Step 1.

```
1          17
  DBNAME=dbdname
{ KEYRANGE=(lowkeyvalue,{EOD|highkeyvalue})          }
{ FROMAREA=(lowblocknumber,{EOD|highblocknumber})    }
  TOAREA=(ddname,{EOD|lowblocknumber},{EOD|highblocknumber})
  [PSB=psbname]
  [SORTOPT=sortoptions]
```

where:

DBNAME=
Specifies the data base to be partially reorganized. This operand must be the name of a DBD with HD organization.

DBNAME is required as the first keyword and must appear only once.

KEYRANGE=
Specifies one range of keys to be reorganized. At least one KEYRANGE must be provided if the data base is HIDAM. Up to 10 KEYRANGEs are allowed. The operands are the root segment or generic keys, with a maximum or 255 bytes each. Keys are specified as either character or hexadecimal values. Character keys are the default. Keys may be expressed in hexadecimal by preceding the value with an X and enclosing it in single quotation marks; for example, KEYRANGE=(X'C8C5E7',X'D2C5E8'). The high key value may be specified as EOD if the upper limit is the end of the data base.

The KEYRANGE keyword must be immediately followed by a TOAREA keyword. KEYRANGE is invalid if the data base is HDAM.

FROMAREA=
Specifies one range of blocks to be reorganized. At least one FROMAREA must be provided if the data base is HDAM. Up to 10 of these keywords are allowed. The operands are block numbers in the root addressable area. The high block number may be specified as EOD if the upper limit is the last block in the root addressable area.

FROMAREA is invalid if the data base is HIDAM.

The FROMAREA keyword must be immediately followed by a TOAREA keyword.

TOAREA=
Specifies one area of the data base into which reorganized segments must be placed. TOAREA keywords are grouped in sets, with one occurrence per set for each data set group in the data base being reorganized. One set must be provided for each KEYRANGE or FROMAREA specified. ddnames must be the name of a DD defining a data set in the data base being organized.

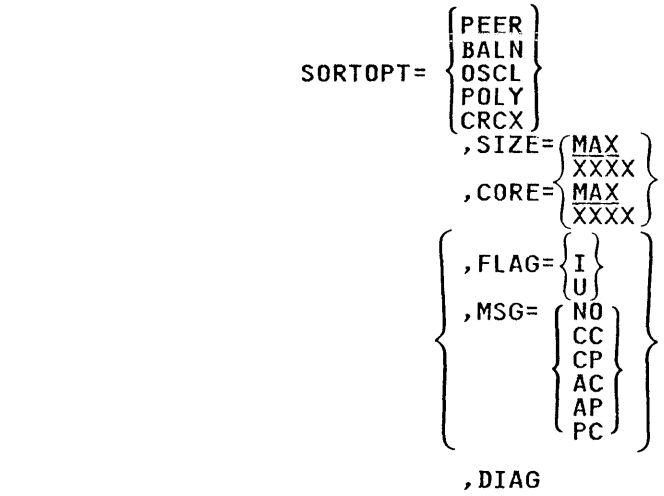
If both low block number and high block number are specified, the rules are the same as for FROMAREA.

EOD may be the value for either the low block number or the high block number. When EOD is specified as the low block number, the segments will be placed beginning at the end of the overflow area. In this case, no high block number is specified. When EOD is specified as the high block number, the TOAREA extends from the low block number to the end of the data set group.

PSB=
the name of a PSB to be generated in Step 1, to be used in Step 2. This keyword may be omitted, if a PSB has already been generated by a previous execution of Partial Data Base Reorganization, and a new PSB is not desired. This keyword may also be omitted if the user provides a PSB that exactly follows the PSB example at the end of this section.

SORTOPT=
options for all sorts. SORTOPT is optional, and it may be specified only once. The character string located in quotes may not exceed 69 characters. Only those sort options which the user wishes to use are entered in the operand string. No extra commas are needed for omitted SORTOPT operands. Default options for sorts in Partial Data Base Reorganization are the normal OS/VS sort option defaults.

SORT options that might be used with the Partial Data Base Reorganization utility are:



For a description of all the options, refer to the appropriate Sort/Merge manual listed in the "How to Use This Publication" section in this manual.

The following are the valid input keywords for Step 2:

```
1              17
 DBNAME=dbdname
[RESTART=checkpointidnumber]
[SCANSEG=(dbdname,segmentname),...]
```

where:

DBDNAME=
Specifies the data base to be partially reorganized. The DBDname must be the same name as that which was specified for Step 1.

RESTART=
Specifies that partial reorganization is to be restarted from the checkpoint named. If RESTART is specified it must appear only once. All other keywords except DBNAME are ignored.

SCANSEG=
Specifies segment types to be included in the scan process. One or more SCANSEG statements may be included in a single execution of Step 2. Only DBD names and segment names listed in the optional scan section of the Step 1 option report can be used as operands. (Segments listed as requiring the scan need not be specified with this statement; they will automatically be scanned.)

If other segments in the same data base require the scan, including the optional segments for the scan may result in improved performance.

RETURN CODES

This program provides return codes preceded, in the case of errors, by numbered messages to the SYSPRINT data set that more fully explain the results of program execution. For a detailed explanation of the numbered messages, see the IMS/VS Messages and Codes Reference Manual. The return codes are as follows:

Code	Meaning
0	All requested operations have completed successfully.
4	Warning message issued.
8	Severe errors causing job termination have occurred.

EXAMPLES

Step 1 Examples

The following example shows the JCL and utility control statements to run Step 1 of the Partial Data Base Reorganization utility.

```
//PDBREX1  JOB  1,1,MSGLEVEL=(1,1)
//         EXEC PGM=DFSPRCT1
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD   DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A,DCB=(BLKSIZE=1210)
//SYSUDUMP DD   SYSOUT=A
//SYSPUNCH DD   DSN=&&DPRPSB,SPACE=(TRK,(2,1)),UNIT=SYSDA,
//          DISP=(,PASS),DCB=(BLKSIZE=800)
//DFSPRCOM DD   DSN=&&DPR,SPACE=(TRK,(2,1)),UNIT=SYSDA,DISP=(,PASS)
//SYSIN    DD   *
  DBNAME=PR07RW12
  KEYRANGE=(000100,000200)
  TOAREA=(PR07DD,005,007)
  PSB=PTSTN07
```

The following are sample output reports from Step 1. Figure 59 contains the input statements. Figure 60 contains the range values. Figure 61 and Figure 62 contain the names of segments for required and optional scanning.

```
        IMS/VS PARTIAL REORGANIZATION - STEP 1 INPUT STATEMENTS                 164/78     PAGE   1

0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
     DBNAME=PR07RW12
     KEYRANGE=(000100,000200)
     TOAREA=(PR07DD,005,007)
     PSB=PTSTN07
```

Figure 59. Partial Reorganization Step 1 Input Statements

```
        IMS/VS PARTIAL REORGANIZATION - RANGE VALUES          FOR DBD - PR07RW12     164/78     PAGE    2
KEYRANGE = '000100'
        TO '000200',
TOAREA   =  00000005 TO 00000007  DDNAME = PR07DD
```

Figure 60. Partial Reorganization Range Values

```
IMS/VS PARTIAL REORGANIZATION - REQUIRED SEGMENT SCAN          FOR DBD - PR07RW12     164/78     PAGE    3
         NO SEGMENT FOUND FOR REQUIRED SCAN
```

Figure 61. Partial Reorganization Required Segment Scan

```
IMS/VS PARTIAL REORGANIZATION - OPTIONAL SEGMENT SCAN          FOR DBD - PR07RW12     164/78     PAGE    4
         NO SEGMENT FOUND FOR OPTIONAL SCAN
```

Figure 62. Partial Reorganization Optional Segment Scan

PSB EXAMPLE

The following example shows the PSB source statements generated by Step 1 if the PSB keyword is included in the Step 1 input control statements.

```
      PCB    TYPE=DB,NAME=PR07RW12,KEYLEN=12,PROCOPT=GIR
         SENSEG    NAME=D,PARENT=0
         SENSEG    NAME=F,PARENT=D
      PCB     TYPE=DB,NAME=PR071,KEYLEN=6,PROCOPT=G
         SENSEG    NAME=INDEX,PARENT=0
      PCB     TYPE=DB,NAME=PR07R,KEYLEN=12,PROCOPT=GIR
         SENSEG    NAME=A,PARENT=0
         SENSEG    NAME=C,PARENT=A
PCB     TYPE=GSAM,DBDNAME=DFSPRWF2,PROCOPT=L
PCB     TYPE=GSAM,DBDNAME=DFSPRWF3,PROCOPT=L
PCB     TYPE=GSAM,DBDNAME=DFSPRWF4,PROCOPT=L
PCB     TYPE=GSAM,DBDNAME=DFSPRWF5,PROCOPT=L
PSBGEN LANG=COBOL,CMPAT=YES,PSBNAME=PTSTN07
```

The JCL for assembling and link-editing the PSB, which must be executed before Step 2, is shown below:

```
//C        EXEC  PGM=IFOX00,REGION=250K,PARM='LOAD,NODECK'
//SYSLIB   DD  DSN=IMSVS.MACLIB,DISP=SHR
//SYSGO    DD  UNIT=SYSDA,DISP=(,PASS),SPACE=(80,(200,50),RLSE)
//SYSPRINT DD  SYSOUT=A
//SYSPUNCH DD  SYSOUT=B
//SYSUT1   DD   UNIT=SYSDA,DISP=(,DELETE),SPACE=(1700,(100,50))
//SYSUT2   DD   UNIT=SYSDA,DISP=(,DELETE),SPACE=(1700,(100,50))
//SYSUT3   DD   UNIT=(SYSDA,SEP=(SYSLIB,SYSUT1,SYSUT2),
//          SPACE=(1700,(100,50))
//SYSIN    DD  DSN=*.PTSTN07.SYSPUNCH,DISP=(OLD,DELETE)
//L        EXEC  PGM=DFSILNK0,REGION=256K
//SYSLIN   DD  DSN=*.C.SYSGO,DISP=(OLD,DELETE)
//SYSPRINT DD  SYSOUT=A
//SYSLMOD  DD  DSN=IMSVS.PSBLIB(PTSTN07),DISP=SHR
//SYSUT1 DD UNIT=SYSDA,DISP=(,DELETE),SPACE=(1024,(100,10),RLSE)
```

DBD EXAMPLES

The following DBDs must be generated before executing Step 2.

```
DBD  NAME=DFSPRWF2,ACCESS=(GSAM,BSAM)                              *
DATASET  DD1=DFSPRWF2,RECFM=FB,RECORD=18,BLOCK=10
DBDGEN
FINISH
END

DBD  NAME=DFSPRWF3,ACCESS=(GSAM,BSAM)                              *
DATASET  DD1=DFSPRWF3,RECFM=FB,RECORD=18,BLOCK=10
DBDGEN
FINISH
END

DBD  NAME=DFSPRWF4,ACCESS=(GSMA,BSAM)
DATASET  DD1=DFSPRWF4,RECFM=VB,RECORD=1000,BLOCK=1
DBDGEN
FINISH
END

DBD  NAME=DFSPRWF5,ACCESS=(GSAM,BSAM)                              *
DATASET  DD1=DFSPRWF5,RECFM=VB,RECORD=1000,BLOCK=1
DBDGEN
FINISH
END
```

STEP 2 EXAMPLES

The following example shows the JCL and utility control statements to run Step 2.

```
//STEP2     EXEC PGM=DFSRRC00,REGION=512K,
//            PARM=(DLI,DFSPRCT2,PTSTN07)
//STEPLIB   DD   DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//DFSPRWF1  DD   UNIT=SYSDA,SPACE=(CYL,(10,5)),DISP=(,PASS)
//DFSPRWF2  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS),
//             DCB=(RECFM=FB,LRECL=18,BLKSIZE=180)
//DFSPRWF3  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS),
//             DCB=(RECFM=FB,LRECL=18,BLKSIZE=180)
//DFSPRWF4  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS),
//             DCB=(RECFM=VB,LRECL=1000,BLKSIZE=1004)
//DFSPRWF5  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS),
//             DCB=(RECFM=VB,LRECL=1000,BLKSIZE=1004)
//DFSPRWF6  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS)
//DFSPRWF7  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS)
//DFSPRWF8  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS)
//DFSPRWF9  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS)
//DFSPRWFA  DD   UNIT=SYSDA,SPACE=(CYL,(01,1)),DISP=(,PASS)
//SORTLIB   DD   DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD   UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK02  DD   UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK03  DD   UNIT=SYSDA,SPACE=(CYC,1)
//IMS       DD   DSN=IMSVS.PSBLIB,DISP=(SHR)
//            DD   DSN=IMSVS.DBDLIB,VOL=SER=IMSQAW,UNIT=SYSDA,DISP=SHR
//IEFRDER   DD   DSN=IMSVS.LOG1,UNIT=TAPE,DISP=(NEW,KEEP),VOL=SER=222222
//SYSPRINT  DD   SYSOUT=A,DCB=(BLKSIZE=1210)
//DFSPRCOM  DD   DSN=*.PTSTN07.DFSPRCOM,DISP=(OLD,KEEP)
//SYSUDUMP  DD   SYSOUT=A
//SYSOUT    DD   SYSOUT=A
//PR07DD    DD   DSN=PR07RW12,DISP=OLD
//PR07RDD   DD   DSN=PR07R,DISP=OLD
//PR07RIDD  DD   DSN=PR07RI,DISP=OLD
//PR07IDD   DD   DSN=PR07I,DISP=OLD
//SYSIN     DD   *
   DBNAME=PR07RW12
//DFSVSAMP  DD   *
1024,4
4096,8
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD   DSN=RECON1,DISP=SHR
//RECON2   DD   DSN=RECON2,DISP=SHR
//RECON3   DD   DSN=RECON3,DISP=SHR
```

The following are sample output reports from Step 2. Figure 63 contains the input statements. Figure 64 , Figure 65, and Figure 66 contain unload statistics. Figure 67 on page 224 and Figure 68 on page 224 contain reload statistics. Figure 69 on page 224 contains scan statistics.

```
         IMS/VS PARTIAL REORGANIZATION - STEP 2 INPUT STATEMENTS                 164/78     PAGE    1

0....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
   DBNAME=PR07RW12
```

Figure 63. Partial Reorganization Step 2 Input Statements

```
         IMS/VS PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE  1      FOR DBD - PR07RW12     164/78     PAGE    2

                                  *              SEGMENT STATISTICS                *        RECORD STATISTICS         *
SEGMENT      SEG    DSG    BLOCK    % OF SEG IN SAME BLK AS:     AVERAGE    AVERAGE    AVERAGE    TOTAL      AVE SEG PER
NAME         LVL    NUM    SIZE      PHY-TWIN.     PHY-PAR      TWINS     CHILDREN    LENGTH    SEGMENTS   DB RECORD

D              1      1    4096         79.3          N/A         N/A         2.2       44.0         29          1.0
F              2      1    4096         90.6         92.2         2.2         0.0       36.0         64          2.2

TOTAL SEGMENTS UNLOADED =          93

AVERAGE DATA BASE RECORD LENGTH =      123.2

NUMBER OF ROOT ANCHOR POINTS PER BLOCK =    1

KEYRANGE = '000100'
        TO '000200'
```

Figure 64. Partial Reorganization Unload Statistics

```
         IMS/VS PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE  1      FOR DBD - PR07RW12     164/78     PAGE    3

      RANGE OF UNLOADED SEGMENTS

  DATA SET      LOW BLOCK    HIGH BLOCK
;ROUP NUMBER     NUMBER       NUMBER

      1             2            5
```

Figure 65. Range of Unloaded Segments

```
MS/VS PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE  1      FOR DBD - PR07RW12     164/78     PAGE    4

                DISTRIBUTION OF DATA BASE RECORDS

     NUMBER OF    OBSERVED     PERCENT     CUMULATIVE    CUMULATIVE
      BLOCKS      FREQUENCY    TOTAL       PERCENT       REMAINDER
         1            27        93.10        93.10          6.90
         2             2         6.90       100.00          0.00

     MAXIMUM BLOCKS FOR A DATA BASE RECORD =    2

     MEAN OBSERVED FREQUENCY                =  1.07
```

Figure 66. Distribution of Data Base Records

```
          IMS/VS PARTIAL REORGANIZATION - RELOAD STATISTICS FOR RANGE  1      FOR DBD - PR07RW12     164/78     PAGE    5

SEGMENT    SEG    DSG    BLOCK    % OF SEG IN SAME BLK AS:      RELOAD    DIFFERENCE
NAME       LVL    NUM    SIZE      PHY-TWIN      PHY-PAR       COUNT     RELOAD-UNLOAD

D            1      1    4096          96.6          N/A          29          0
F            2      1    4096          98.4         98.4          64          0

TOTAL SEGMENTS RELOADED =        93
```

Figure 67. Partial Reorganization—Reload Statistics

```
          IMS/VS PARTIAL REORGANIZATION - RELOAD STATISTICS FOR RANGE  1      FOR DBD - PR07RW12     164/78     PAGE    6

                  RANGE OF RELOADED SEGMENTS

  DATA SET      LOW BLOCK     HIGH BLOCK     BYTE COUNT INSERTED
GROUP NUMBER     NUMBER        NUMBER          TO OVERFLOW

      1              5             6              N/A

DFS3000I SUCCESSFUL COMPLETION OF PARTIAL DATA BASE REORGANIZATION
```

Figure 68. Range of Reloaded Segments

```
          IMS/VS PARTIAL REORGANIZATION - SCAN STATISTICS                FOR DBD - PR05RW08     164/78     PAGE    7

          DATA BASE NAME    SEGMENT NAME    SCAN COUNT

             PR05R              D                100

          TOTAL SEGMENTS SCANNED =        100

DFS3000I SUCCESSFUL COMPLETION OF PARTIAL DATA BASE REORGANIZATION
```

Figure 69. Partial Reorganization Scan Statistics

The various parts of the reports (Figures 64 through 69) are explained here:

SEGMENT NAME
The name of the segment type being unloaded/reloaded.

SEG LVL
The hierarchic level number of the segment type being unloaded/reloaded.

DSG NUM
The data set group number of the segment type being unloaded/reloaded.

BLOCK SIZE
Block size of the data set group.

% OF SEG IN SAME BLK AS PHY-TWIN
The percentage of segments of this type which occupy the same data base block as the previously unloaded or reloaded physical twin segment.

% OF SEG IN SAME BLK AS PHY-PAR
The percentage of segments of this type which occupy the same data base block as the previously unloaded or reloaded physical parent segment.

AVERAGE TWINS
The average number of physical twins within the twin families.

AVERAGE CHILDREN
The average number of children of this segment type.

AVERAGE LENGTH
The average length of segments of this type.

TOTAL SEGMENTS
The total number of segments being unloaded/reloaded.

AVE SEG PER DB RECORD
The average number of segments per data base record.

RANGE OF UNLOADED SEGMENTS
The actual range of the segments being unloaded. (For an HDAM data base, this number is probably different from the range specified.)

DATA SET GROUP NUMBER
Same as "DSG Number."

LOW BLOCK NUMBER
The lowest block number of the segments unloaded within the data set group.

HIGH BLOCK NUMBER
The highest block number of the segments unloaded within the data set group.

DISTRIBUTION OF DATA BASE RECORDS
The distribution of the data base records unloaded over the number of physical blocks. This report only tabulates 1 through 40 blocks: distributions over 40 blocks are accumulated in one entry.

NUMBER OF BLOCKS
The number of physical blocks occupied by a data base record.

OBSERVED FREQUENCY
The number of data base records observed for the distribution.

PERCENT TOTAL
The percentage of the data base records observed over the total data base records unloaded.

CUMULATIVE PERCENT
Total percentage up to this point.

CUMULATIVE REMAINDER
Total percentage remaining up to this point.

MAXIMUM BLOCKS FOR A DATA BASE RECORD
The maximum number of blocks occupied by a data base record.

MEAN OBSERVED FREQUENCY
The average number of blocks occupied by unloaded data base records.

RELOAD COUNT
The number of segments reloaded.

DIFFERENCE RELOAD-UNLOAD
The difference between the number of segments unloaded and the number reloaded.

TOTAL SEGMENTS RELOADED
The total number of segments reloaded for the specific range.

BYTE COUNTINSERTED TO OVERFLOW
The number of bytes inserted into the overflow area (HDAM only)

SCAN COUNT
The number of segments scanned for this segment type.

DATA BASE PREREORGANIZATION UTILITY (DFSURPR0)

The Data Base Preorganization utility creates a control data set that is used by the other logical relationship resolution utilities. It also indicates which data bases and segments (if any) must be scanned by the Data Base Scan utility. If secondary indexes exist when initially loading or reorganizing an indexed data base, the Prereorganization utility must be executed against the indexed data base to create a control data set used in the creation of a secondary index. This utility must also be executed when data bases being loaded and/or reorganized contain logical relationships.

The input to this utility is a data set that consists of the utility control statements that name the data bases(s) being loaded and/or reorganized. The DBDs that are used for the data bases named on these statements must define each data base as it is to exist after logical relationships are resolved. These DBDs must not be modified until the Prefix Update utility has been successfully executed.

The output is a control data set that is used by the Data Base Scan utility and the Data Base Prefix Resolution utility.

A flow diagram of the Data Base Prereorganization utility is shown in Figure 70 on page 227.

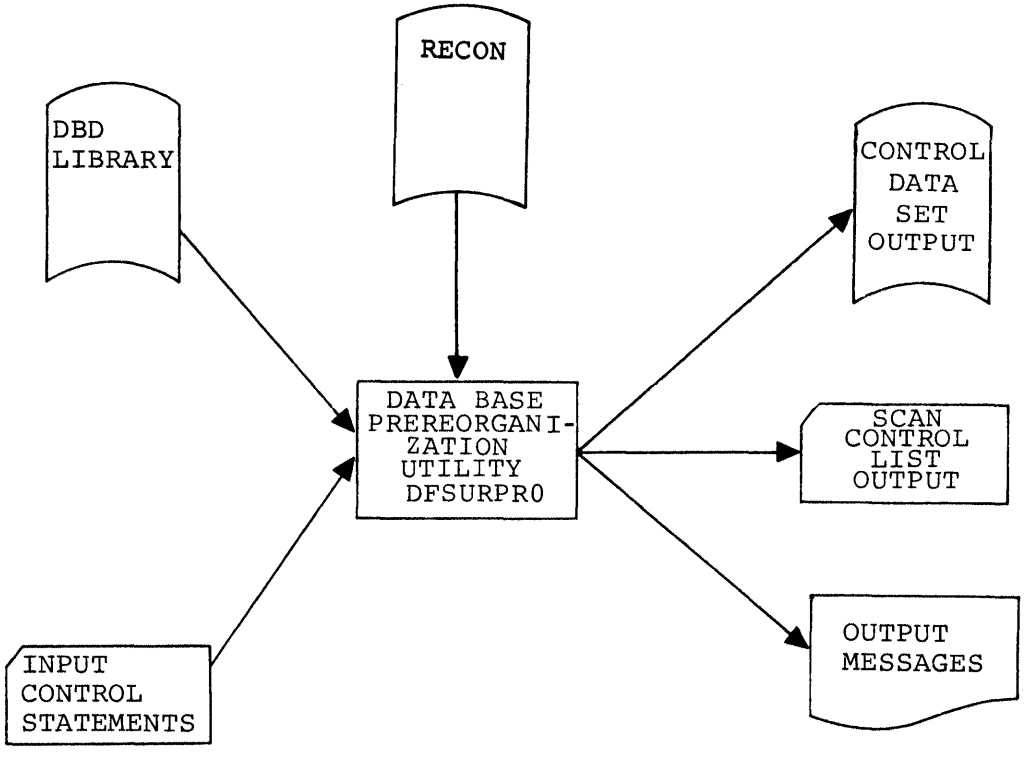


Figure 70. Data Base Prereorganization Utility

JCL REQUIREMENTS

The Data Base Prereorganization utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC
This statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURPR0'
```

The normal IMS/VS positional parameters such as SPIE and BUF can follow program name in the PARM field. See the DBBBATCH or DLIBATCH procedure in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD
Points to IMSVS.RESLIB, that contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBDs which describe the data bases named on the input control statements. This DD statement must always be included. The data set must reside on a direct access device.

SYSIN DD
This data set will contain input control statements. The data set can reside on a card reader, tape, or direct access device, or be routed through the input stream. This DD statement must always be included.

DCB parameters specified within this program are RECFM=FB and LRECL=80. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSPRINT DD
Defines the message output data set. The data set can reside on a printer, tape, direct access device, or be routed through the output stream. This DD statement must always be included.

DCB parameters specified within this program are RECFM=FB and LRECL=120. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSPUNCH DD
Defines the punch-type output data set. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream. This DD statement must be included if an "OPTIONS=(PUNCH)" control statement is provided.

DCB parameters specified within this program are RECFM=FB and LRECL=80. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSURCDS DD
Defines the output data set for this program. This data set is the control data set used at data base load time, by the Prefix Resolution utility and the Data Base Scan utility. This DD statement must always be included.

DCB parameters specified within this program are RECFM=FB and LRECL=1600. BLKSIZE must be provided on this DD statement.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence will be used for the dump.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

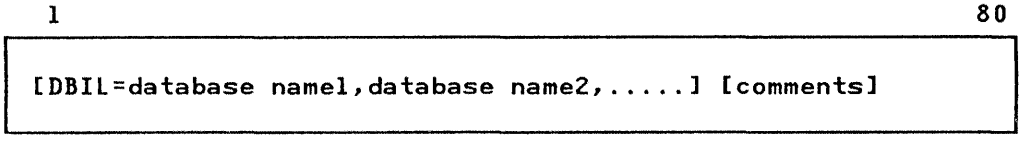
RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS

DBIL Statement

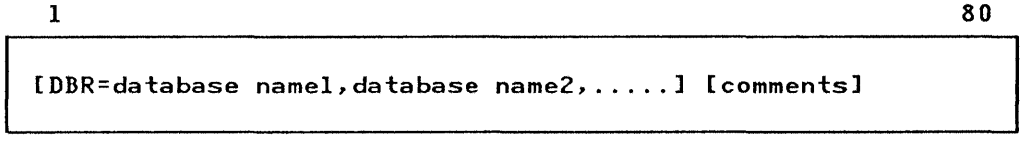


This utility control statement names data bases that are to be initially loaded or reorganized. This statement is necessary for reorganization when there has been a DBD change affecting logical pointers or counters. One or more of these statements can be provided. Each DBD name must be left-justified to provide a total length of 8 characters. If the DBD name is less than 8 characters, sufficient trailing blank characters must be provided to make a complement of 8 characters. A blank must follow the last entry on each statement. If a HIDAM data base is to be initially loaded, only its DBD name should be listed on a DBIL control statement. (Neither the HIDAM primary index nor any secondary index DBD names should be listed.)

When structural changes affecting logical relationships are made to a data base during a data base reorganization process, the following must be performed prior to the HD reload step.

- Assemble and link-edit the new DBD into the IMS/VS DBD library.
- Rerun the Prereorganization utility against the new DBD, specifying DBIL=database name.

DBR Statement



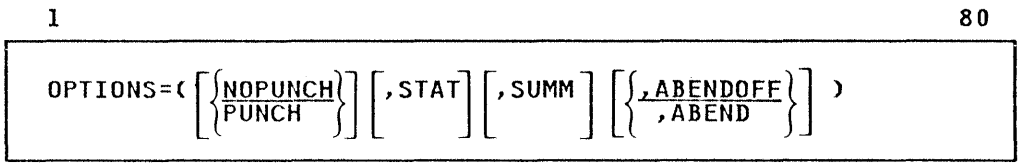
This utility control statement names data bases that are either being converted from DOS/VS DL/I or being reorganized. One or more of these statements can be provided. Each DBD name must be left-justified to provide a total length of 8 characters. If the DBD name is less than 8 characters, sufficient trailing blank characters must be provided to make a complement of 8 characters. A blank must follow the last entry on each statement.

Note: Use the DBIL statement when there has been a DBD change affecting logical pointers or counters.

It should be noted that, if a HISAM data base is to be reorganized using the HISAM Reorganization Unload/Reload utilities, the HISAM DBD name must not be listed on a DBR control statement. If a HISAM data base is to be reorganized using the HD Reorganization Unload/Reload utilities, however, the HISAM DBD name must be listed on a DBR control card.

If a HIDAM data base is to be reorganized, only its DBD name should be listed on a DBR control statement. (Neither the HIDAM primary index nor secondary index DBD names should be listed.)

OPTIONS Statement



This utility control statement indicates whether any optional information is to be provided during the reorganization or initial load of the data base. These parameters are not positional and can be specified in any order. If more than one parameter is specified, a comma must be included between the parameters. Information specified on this statement affects output in the execution of the Prereorganization utility (DFSURPR0) and the Prefix Resolution utility (DFSURG10).

PUNCH
Causes the data base scan list to be written to both the SYSPUNCH data set and SYSPRINT data set. This output is used as input to the Data Base Scan utility via the SYSIN data set. (See the description of the SYSIN DD statement for the Data Base Scan utility.)

NOPUNCH
Prevents the scan list from being written to the data set defined by the SYSPUNCH DD statement. This is the default if this parameter or the PUNCH parameter is omitted.

STAT
Causes the Data Base Prefix Resolution utility (DFSURG10) to accumulate statistics on segments that are updated.

SUMM
Causes the Data Base Prefix Resolution utility to accumulate and then print merely the number of times the message DFS878I was issued.

ABENDOFF
Turns off the abend function. This is the default. If ABEND was coded, it remains in effect within a job step until ABENDOFF is coded.

ABEND
Terminates with user abend 955, if any condition arises causing abnormal termination of the run. A dump will be printed if a SYSABEND or SYSUDUMP DD statement is present.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	No errors detected.
8	One or more error messages have been issued.

EXAMPLE

This example shows the job control statements and utility control statement required to execute DFSURPR0 for two data bases that are to be initially loaded. The DBD names for the two data bases are PAYR and SKILLINV.

```
//RLUTIL JOB     1,1,MSGLEVEL=1,REGION=250K
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURPR0'
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=SYSDA,DISP=(NEW,KEEP),
//   VOL=SER=IMSMSC,DCB=(BLKSIZE=1600),SPACE=(CYL,1)
//SYSIN  DD  *,DCB=BLKSIZE=80
DBIL=PAYRbbbb,SKILLINV
/*
```

OUTPUT MESSAGES

The output messages issued by this utility indicate the data base operations that must be performed prior to execution of the Prefix Resolution and the Prefix Update utilities. For instance:

- Data bases listed after the characters DBIL= in message DFS861I must be initially loaded.
- Data bases listed after the characters DBR= in message DFS861I must be reorganized using the HD Reorganization Unload/Reload utilities.
- Data bases listed after the characters DBS= in message DFS862I must be scanned using the Data Base Scan utility.

Other forms of output messages created by this utility are: (1) a listing of the control statements that were provided as input, (2) an optional deck of scan control statements for use with the Data Base Scan utility, (3) error messages, and (4) a termination message.

DATA BASE SCAN UTILITY (DFSURGS0)

The Data Base Scan utility scans data bases not being loaded or reorganized for segments that contain logical relationships that are affected by loading and/or reorganizing other data bases. For each segment of this type, the utility generates one or more output records, depending upon the relationships in which that segment is involved. The records are written to the DFSURWF1 output work data set allocated to this utility.

This utility generates a work data set that serves as one of the inputs to the Prefix Resolution utility.

If execution of the Data Base Scan utility is abnormally terminated for any reason other than a data base I/O error, program execution can be resumed by a restart operation.

If execution of this utility is abnormally terminated because of a data base I/O error, the following steps should be followed:

- Determine the cause of the error, and take the necessary action to correct it;
- Restore the data base as it existed before execution of this utility;
- Request a restart operation.

Note: The functions of this utility can be performed by the Utility Control Facility. Refer to "Utility Control Facility" for a description of its operation.

Figure 71 on page 232 shows a flow diagram of the Data Base Scan utility.

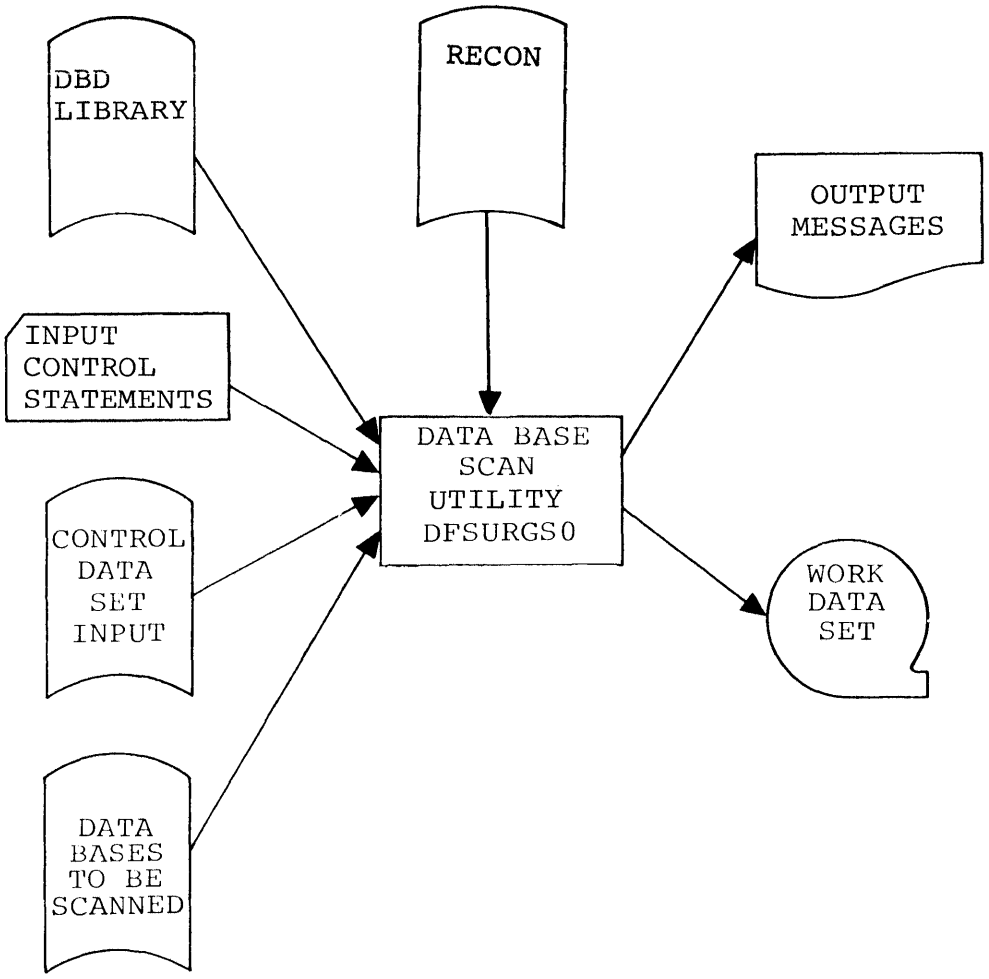


Figure 71. Data Base Scan Utility

JCL REQUIREMENTS

The Data Base Scan utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGS0'
```

The normal IMS/VS positional parameters such as SPIE and BUF can follow the program name in the PARM field. See the DBBBATCH or DLIBATCH procedure in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

The Data Base Scan utility can be passed a buffer size parameter on this statement. The buffer size would be meaningful, however, only if the block size of the data base was such that more than 7K bytes was required to have two blocks in storage. This would allow sequential GETs in one block while the second was being read in from a storage device.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBDs that describe the data base(s) to be scanned, plus any logically related data bases. This DD statement must always be included. The data set must reside on a direct access device.

SYSIN DD
Defines the input data set for this program. The data set can reside on a card reader, tape, or direct access device, or be routed through the input stream. This DD statement need not be included unless utility control statements are provided as input to this program.

DCB parameters specified within the program are RECFM=FB and LRECL=80. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSPRINT DD
Defines the message output data set. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream. This DD statement must always be included.

DCB parameters specified within this program are RECFM=FB, LRECL=120. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSUDUMP DD
Defines the dump data set for this program. This DD statement is required only if the ABEND utility control statement is included. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB, LRECL=120. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSURCDS DD
Defines the control data set for this program. It must be the output control data set generated by the Prereorganization utility. This DD statement must always be included. The data set must reside on either tape or a direct access device.

DFSURSRT DD
Defines the data set to be used for restart purposes. This data set is not required if restart is not desired.

For restart processing, the DFSURSRT DD concatenation should be the same as the DFSURWF1 DD concatenation from the previous scan. However, you can begin the DFSURSRT concatenation with the data set that contains the record identified in the RSTRT command. Examine the checkpoint information in SYSPRINT from the previous scan execution to determine which data set this is.

If the original DFSURWF1 was a single data set, the data set should be specified in the DFSURSRT DD statement.

database DD
References the data base(s) that is to be scanned as indicated by the Prereorganization utility. DD statements must be present for each data base. The ddnames must match the ddname indicated in the DBD. The data set must reside on a direct access device.

DFSURWF1 DD
Defines the input/output work data sets for this program. This data set will be supplied as one of the inputs to the Prefix Resolution utility, and as the output for the Initial Load and Data Base Scan utilities. The data set can reside on tape or a direct access device.

DCB parameters specified within this program are RECFM=VB and LRECL=300. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I Buffer Handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.)

The data set can reside on a tape, direct access device, or card reader, or it can be routed through the input stream.

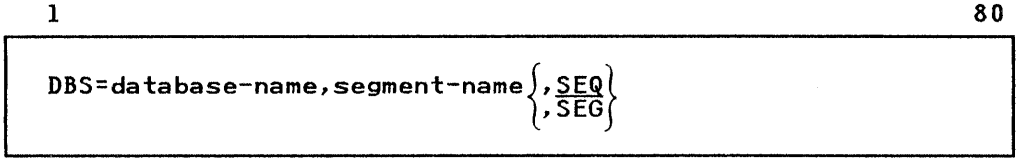
RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

DBS Statement



This utility control statement names a data base segment that is to be scanned by the Data Base Scan utility. One or more of these statements can be provided. The data base name and segment name must be padded with sufficient blanks to provide a total length of 8 characters each. User comments can be included following the parameter specifications.

If DBS control statements are not provided, the scan information provided by the control data set from the Prereorganization utility is used and only those data base names and segment names appearing in the control data set are accepted. (It should be noted that all data bases provided on the scan list must be scanned prior to execution of the Prefix Resolution utility.)

If DBS control statements are not provided, the data bases to be scanned will be:

- Scanned sequentially, using unqualified GN calls for HISAM data bases
- Scanned by segment, using GN calls qualified by segment names for HDAM and HIDAM data bases

If DBS control statements are provided to the Scan utility, the scan list provided in the control data set is ignored entirely and work data set records are generated only for those segments named on the DBS statements. These segment names must, of course, exist in the control data set or the DBS control statements will also be ignored.

It should be noted that, even if a scan list is provided through DBS control statements, the control data set must still be provided to this utility because it contains other information that is required by the Scan utility.

The scan list contained in the SYSPUNCH output data set of the Prereorganization utility can be used as input to the Scan utility. The value of using the SYSPUNCH output as input to this utility is that, if multiple data bases need to be scanned, they can be scanned in parallel with multiple executions of the Scan utility. This can be done by separating the SYSPUNCH output (DBS= statements) by data base name and submitting scan control statements for each data base to different executions of the utility.

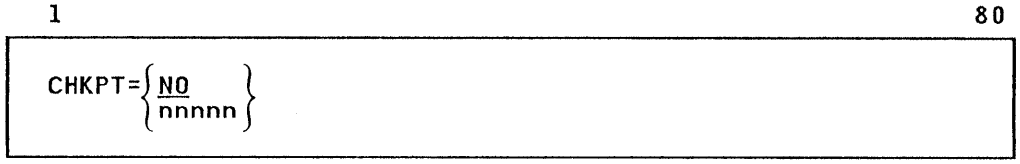
The SEQ and SEG options specify the method to be used to scan a data base. The SEQ option indicates that a data base is to be scanned sequentially by using unqualified GN calls. The SEG option indicates that a data base is to be scanned by using GN (Get Next) calls qualified by segment name. The scan method option specified on a DBS control statement applies to all segments to be scanned in the data base named on the control statement, not just to the specific segment named on the control statement.

If the scan method option is specified on multiple DBS control statements for a particular data base, the method specified on the last DBS control statement encountered for that data base is used for the entire data base. If neither option is specified,

the SEQ option will default for HISAM data bases and the SEG option will default for HDAM and HIDAM data bases.

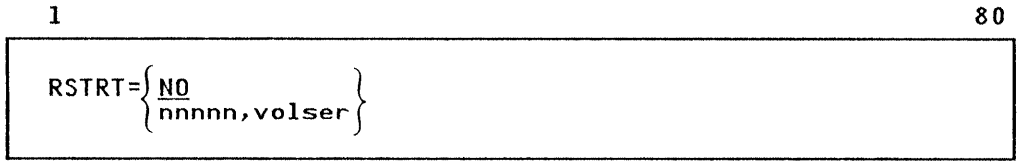
The efficiency of scanning an HD data base can be improved by specifying the SEQ option if many segment types are to be scanned. Conversely, the efficiency of scanning a HISAM data base can be improved by specifying the SEG option if few segments are to be scanned. The relative efficiency obtained by either scan method depends upon the particular structure of the data base to be scanned. In some cases, the best method may have to be determined by usage.

CHKPT Statement



This utility control statement indicates that checkpoint operations are to be performed during operation of this program. A checkpoint record is written to the data set specified by the DFSURWF1 DD statement every time the number of records specified by "nnnnn" is generated. As each checkpoint record is generated, a checkpoint message (DFS867) is issued to the OS/VS system console. The message indicates the name of this program, the checkpoint number of the checkpoint record written, and the output volume serial of the volume being written. The checkpoint messages should be saved in case a restart operation is required.

RSTRT Statement



This utility control statement indicates that a restart operation is to be performed by this program. "nnnnn" is a 5-digit decimal number and "volser" is the volume serial identifier of the input volume containing the checkpoint record numbered "nnnnn." The parameters "nnnnn" and "volser" can be obtained from the checkpoint messages that were issued to the OS/VS system console.

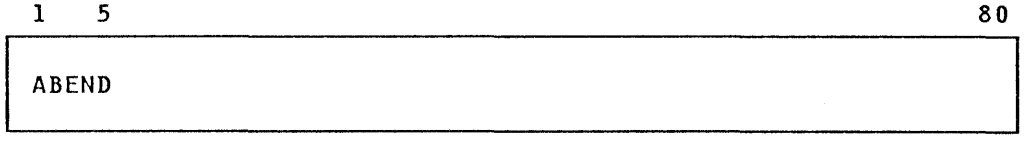
If the volume specified on this statement is not mounted, FEOVs are issued until the correct volume is made available. The volume specified on this statement must be identified by the DFSURSRT DD statement. The restart module reads records present on the volume identified by this DD statement until the checkpoint record numbered "nnnnn" is encountered.

After each record is read, it is written to the data set identified by the DFSURWF1 DD statement. When the correct checkpoint record is located, a message (DFS378I) is issued to the OS/VS system console indicating this program name, the checkpoint number, and the volume serial. Because the checkpoint record specified on the "RSTRT" control statement was written to the data set identified by the DFSURWF1 statement, the current volume available to that data set will appear in the restart-completed message.

Processing continues from the restart point. The volume containing the specified checkpoint record, and any succeeding volumes that were written during a previous execution of this

program, must not be included in the data set supplied to the Prefix Resolution utility (DFSURG10).

ABEND Statement



This optional utility control statement should be included when a storage dump is required for diagnostic purposes. If included in the input stream, and if any abnormal condition arises during execution of this utility, any return code greater than zero causes user abend 955 to be issued. The abend is issued at end of program execution, and a storage dump is provided at that time. If this control statement is included, a SYSUDUMP DD statement is also required.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	No errors detected.
8	One or more error messages issued.

EXAMPLE

This example shows the JCL required to scan the data base defined by the HDRELTD DD statement. This data base is logically related to one or more other data bases which the user indicated on either DBIL or DBR control statements supplied to the Prereorganization utility. The information in the control data set from the Prereorganization utility is used in preference to control statements.

```
//RLUTIL JOB     1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGS0',REGION=250K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSURWF1   DD  DSN=IMSVS.URWF1,UNIT=TAPE,VOL=SER=TAPE11,
// LABEL=(,SL),DISP=(NEW,KEEP),DCB=(LRECL=900,BLKSIZE=1008,
//   RECFM=VB)
//HDRELTD    DD  DSN=DATABASE.DBRELATD,UNIT=SYSDA,
//   VOL=SER=DB0003,DISP=OLD
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=SYSDA,
//   VOL=SER=IMSMSC,DISP=OLD
//DFSVSAMP   DD DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

Note: DD statements must be included for all logically related data bases.

The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

OUTPUT MESSAGES

The output messages issued by this utility include DFS339I and DFS340I. The DFS339I message indicates when the scan processing started for the data base named on the DBS utility control statement. Message DFS340I indicates that scan processing is completed.

Other output messages issued by this utility denote error conditions are fully explained in IMS/VS Messages and Codes Reference Manual.

DATA BASE PREFIX RESOLUTION UTILITY (DFSURG10)

The Data Base Prefix Resolution utility accumulates the information generated on work data sets during the load and/or reorganization of one or more data bases. It produces an output data set that contains the prefix information needed to complete the logical relationships defined for the data base(s) and, optionally, an output data set containing information needed to create and/or update secondary index data bases. There are no utility control statements for this utility.

Note: The functions of this utility can be performed by the Utility Control Facility. Refer to "Chapter 8. Utility Control Facility (DFSUCF00)" on page 308 for a description of its operation.

RESTRICTIONS

The Data Base Prefix Resolution utility uses the OS/VS Sort/Merge programs. Because the maximum sort field permitted by Sort/Merge is 256 characters, certain limits must be observed. The following restrictions apply:

1. For any given logical parent/logical child pair, the sum of items a and b below must not exceed 200 characters (the balance of 56 characters is used by IMS/VS for control purposes):

   a. The length of the logical parent's concatenated key

   b. The length of the sequence field of the logical child as seen by its logical parent

2. The sum must be computed once for the logical parent and once for the logical child. These summations are treated separately.

3. One or more of the above quantities may be omitted from the summations as described below.

   a. The logical parent's concatenated key length must be included in both limit checks if the logical parent is being initially loaded, or if the logical child does not point to the logical parent with a logical parent pointer.

   b. The logical child's sequence field length as seen by its logical parent must be included in the logical child's limit check if the logical child is being initially loaded and if it has a logical twin chain. Otherwise, it may be omitted.

      If the above limit check is not satisfied for either a logical parent or a logical child, the user can omit loading of the logical parent or logical child at initial data base load time. The logical parent or logical child can then be inserted into the data base at a later time by an application program operating in an update mode. Once a data base is loaded, one or more of the components of the limit check may be omitted.

The Data Base Prereorganization utility performs the above limit check for logical parent/logical child combinations affected by an intended data base initial load or reload. It should be noted that the limit check is a worst-case check. If the limit check fails for a logical parent/logical child combination, message DFS885 will be issued. Refer to IMS/VS Messages and Codes Reference Manual for an explanation of the message.

Note: IMS/VS makes no assumption of sequence for unkeyed or nonunique keyed segments during initial data base load, and the operating system Sort/Merge does not guarantee first-in/first-out sequence of records with equal key values. For these reasons, the sequence of logical child segments of these types may be inconsistent in successive runs of this program or in successive reorganization runs.

Figure 72 is a flow diagram of the Data Base Prefix Resolution utility.

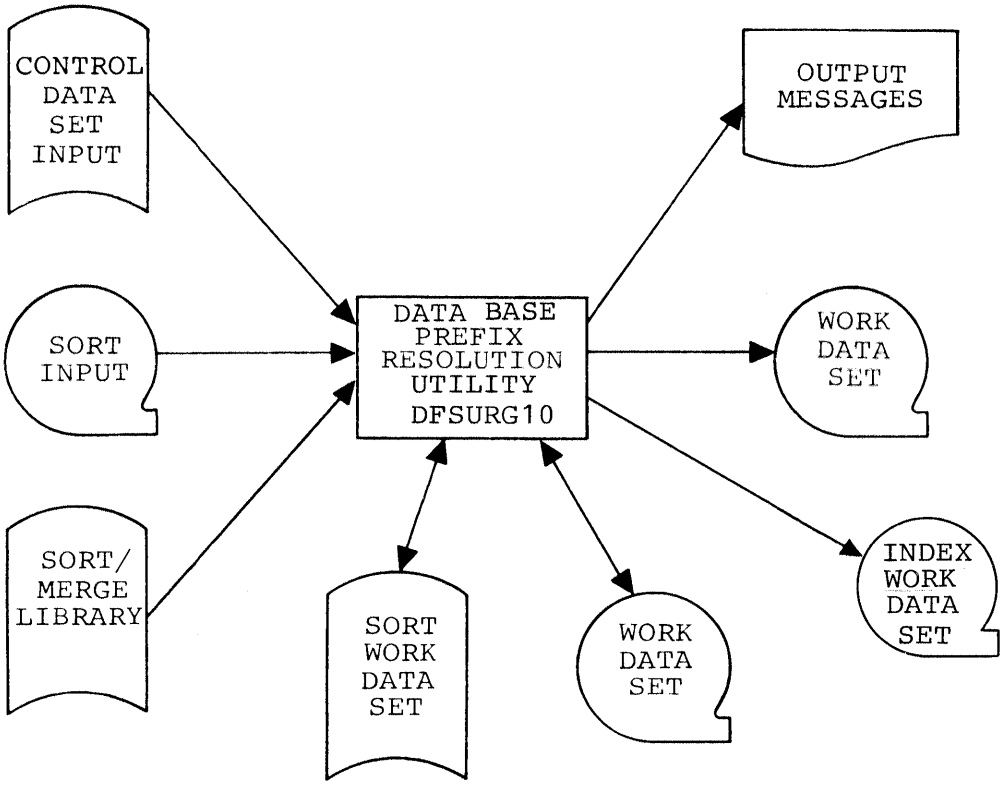


Figure 72. Data Base Prefix Resolution Utility

JCL REQUIREMENTS

The Data Base Prefix Resolution utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

This utility attaches the Sort/Merge program. The IMS/VS-supported PARM field options available to the Sort/Merge program can be used with this utility by specifying the desired options in the PARM field of the EXEC statement for this utility. (For a description of the options, refer to the appropriate Sort/Merge manual listed in the "How to Use This Publication" section in this manual.)

EXEC

This statement must be of the form:

```
PGM=DFSURG10,REGION=nnnk,PARM='options'
```

where nnn is the region size. Because this program invokes the operating system Sort, program efficiency can be improved by increasing region size.

The PARM field specifications are not positional.

The PARM field options and defaults are:

```
PARM=' {BALN}          [,CKPT] {FILSZ=n }          ,CORE={nnnnnn}
       {OSCL}                  {,SIZE=n }                {061440}
       {POLY}                  {FILSZ=En}                {MAX   }
                               {SIZE=En }

      ,MSG={NO}
           {CC}
           {CP} [,ABEND] ,{DUMPWF1}={n  }
           {AC}           {DUMPWF2}={ALL}
           {AP}           {DUMPWF }
                                                                  '
```

If OSCL is specified, a SIZE parameter must be specified.

If the defaults indicated above for CORE and MSG are not specified, the Sort/Merge program determines the sort method to be used.

Only the keywords shown above will be passed to the attached Sort/Merge program. For the MSG parameter, any valid Sort/Merge option will be passed to the Sort/Merge program.

If the CKPT parameter is specified, the Prefix Resolution utility links to the Sort/Merge program instead of attaching as previously stated.

The value of CORE must be a 6-digit figure and should be calculated based on the type of SORT and SORT devices used.

ABEND should be specified only when a storage dump is required for diagnostic purposes. When specified, a SYSUDUMP DD statement is also required.

FILSZ=n

n is the exact number of records in the data set to be sorted. It must take into account records to be inserted or deleted at exit E15, if any.

SIZE=n

n is the exact number of records in the input data set, excluding any changes to be made at exit E15. SM1 will accept with FILSZ or SIZE, but FILSZ is always to be preferred when its use is possible, because it allows better optimization.

If the number of records in the input data set is not the same as the value n specified, the program terminates with the value n placed in the IN field of the message IGH047A or IGH054I. This applies to both FILSZ and SIZE.

FILSZ=En or SIZE=En

n is the estimated number of records to be sorted; the value specified for n should be large enough to include both the input data set and any records added or deleted at exit E15.

For example, if total data set size is estimated to be 5000 records, specify FILSZ=E5000. The maximum allowable size is E+8(Ennnnnnnn).

If the balanced disk technique is being used, the Sort/Merge program prints message IGH070I, which states either (1) that the size of the file has not been specified, or (2) the decimal number of records has not been specified.

If this operand is omitted, the Sort/Merge program assumes that:

- If intermediate storage is tape, the input data set can be contained on one volume at the blocking factor used by the sort.
- If intermediate storage is direct access, the input data set will fit into the space you have allocated.

If possible, an estimated file size should be given, because this greatly improves SM1's optimization and hence performance.

The DUMPWF1, DUMPWF2, and DUMPWF parameters are diagnostic tools to be used only when it is required to see the DFSURWF1 and DFSURWF2 workfile records exactly as output from the first or second executions of the SORT.

If DUMPWF1 is included, the specified number of records as produced by the first sort (the sorted DFSURWF1) is printed in SYSPRINT.

Specification of DUMPWF2 requests the same service for sorted DFSURWF2.

Both DUMPWF1 and DUMPWF2 requests can be included, but, if the same value of n is desired for both, DUMPWF is an equivalent specification.

The value specified for n can contain up to 9 digits.

STEPLIB DD
Points to the IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

SYSPRINT DD
Defines the message output data set for this program. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream. This DD statement must always be included.

DCB parameters specified within this program are RECFM=FB and LRECL=121. BLKSIZE must be provided on the SYSPRINT DD statement, and must be a multiple of LRECL.

SYSUDUMP DD
Defines the dump data set for this program. This DD statement is required only if the ABEND utility control statement is included. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB, LRECL=120. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSOUT DD
Defines the message output data set for Sort/Merge. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream. This DD statement must always be included.

SORTLIB DD
Defines a data set containing load modules for the operating system Sort/Merge program. This DD statement must always be included.

SORTWKnn DD
Defines intermediate storage data sets for the operating system Sort/Merge program. Refer to the appropriate operating system Sort/Merge manual regarding specification of number and size of intermediate storage data sets. These DD statement(s) must be included.

SORTIN DD
Defines the input data set for this program. This DD statement must always be included. It is referenced by the operating system Sort/Merge program and must conform to its JCL requirements. The data set(s) referenced by this DD statement must be the output work data set(s) produced during a data base initial load, reload, or scan operation; those work data sets must be concatenated to form the SORTIN data set. If there are multiple data sets with unlike DCB attributes, the data set with the largest LRECL should be first in the concatenation.

DCB parameters specified within this program are RECFM=VB, and LRECL=900. The BLKSIZE must be the same as that specified for the work data sets written during initial data base load, data base reload, or data base scan. BLKSIZE should be the same as that specified on the DFSURWF1 DD card for those programs. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSURWF2 DD
Defines an intermediate sort work data set. This DD statement must always be included. The data set can reside on a tape or direct access device. The size of the data set will be approximately the same as that of the input data set defined by the SORTIN DD statement.

DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSURWF3 DD
Defines the output work data set that contains all output data from this program. This statement must always be included. The output data set defined by this statement will be supplied as input to the Prefix Update utility. The data set can reside on tape or direct access device. Its size will be approximately the same as that of the input data set defined by the SORTIN DD statement.

DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on the DFSURWF3 DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSURCDS DD
Defines the control data set for this program. It must be the output control data sets generated by the Prereorganization utility. This DD statement must always be included.

DFSURIDX DD
Defines an output work data set which will be used if secondary indexes are present in the DBDs being reorganized/loaded. This data set must be used as input to the HISAM Unload program (DFSURUL0) for

creating, replacing, merging, or extracting secondary indexes (shared or unshared). This DD statement is required only if secondary indexes are present. DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on the DFSURIDX DD statement.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	No errors detected.
4	Returned when any of the following messages have been issued during program execution or the following messages have been preempted by the SUMM parameter during DB Prereorganization. DFS878, DFS885, DFS961
8	Returned when one or more of the following messages has been issued during program execution: DFS852, DFS855, DFS857, DFS876, DFS877, DFS879, DFS880, DFS881 or when data required by prefix update is not written to the WF3 data set.
12	Returned when either one or both of the messages listed under return code 4 and any one or more of the messages listed under return code 8 have been issued.
16	Returned by the OS/VS Sort/Merge program. This return code takes precedence over the above return codes.

Note: For return codes larger than 16, the meaning stated above for return code 16 applies.

If either an 8, 12, or 16 return code is provided by the Prefix Resolution utility (DFSURG10), the Prefix Update utility (DFSURGP0) should not be executed because the input work data set required by DFSURGP0 may not have been completely generated by DFSURG10. The errors indicated by the diagnostic messages should be corrected, and the data base operations should be redone before the Prefix Resolution utility is again attempted.

If return code 4 is provided, a legitimate error may or may not be present. Refer to IMS/VS Messages and Codes Reference Manual for an explanation of the DFS878 and DFS885 cautionary messages.

EXAMPLE

The following example shows use of the Prefix Resolution utility to resolve logical relationships and secondary indexes. Only three work data sets are supplied to Sort/Merge, and Sort/Merge is allowed to choose the sort method. SORTIN is the work data set created at either reload time or initial load time as the DFSURWF1 ddname.

DFSURWF2 is an intermediate work file and is deleted at the end of the step.

The DFSURWF3 data set is created here. It will be used as input to the Prefix Update utility.

DFSURIDX is an output data set on which will be written those segments required to build a secondary index using the HISAM Unload/Reload utilities.

```
//RLUTIL JOB 1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSURG10
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSOUT DD      SYSOUT=A
//SYSPRINT DD    SYSOUT=A,DCB=BLKSIZE=1210
//SORTLIB    DD  DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTWK01   DD  UNIT=SYSDA,SPACE=(CYL,(02),,CONTIG)
//SORTWK02   DD  UNIT=SYSDA,SPACE=(CYL,(02),,CONTIG)
//SORTWK03   DD  UNIT=SYSDA,SPACE=(CYL,(02),,CONTIG)
//SORTIN     DD  DSN=IMSVS.URWF1,UNIT=TAPE,VOL=SER=TAPE11,
//  LABEL=(,SL),DISP=(OLD,KEEP),DCB=(LRECL=900,BLKSIZE=1008,
//   RECFM=VB)
//DFSURWF2   DD  DSN=IMSVS.URWF2,UNIT=TAPE,VOL=SER=TAPE21,
//  LABEL=(,SL),DISP=(NEW,DELETE),DCB=(LRECL=900,
//    BLKSIZE=1008,RECFM=VB)
//DFSURWF3   DD  DSN=IMSVS.URWF3,UNIT=TAPE,VOL=SER=TAPE31,
//  LABEL=(,SL),DISP=(NEW,KEEP),DCB=(LRECL=900,BLKSIZE=1008,
//   RECFM=VB)
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=SYSDA,DISP=OLD,
//          VOL=SER=IMSMSC
//DFSURIDX   DD  DSN=IMSVS.URIDX,DISP=(,KEEP),
//          UNIT=TAPE,VOL=SER=TPEIDX,
//  LABEL=(,SL),DCB=(LRECL=900,BLKSIZE=1008,RECFM=VB)
```

OUTPUT MESSAGES AND STATISTICS

If no errors are detected by this program, the only output message issued will be a normal program termination message, unless "STAT" or "SUMM" was specified in the Prereorganization utility control statement; if either was specified, statistics will be printed.

DATA BASE PREFIX UPDATE UTILITY (DFSURGP0)

The Data Base Prefix Update utility uses the output generated by the Prefix Resolution utility to update the prefix of each segment whose prefix information was affected by a data base load and/or reorganization.

The output of the Prefix Resolution utility consists of one or more update records to be applied to each segment that contains logical relationship prefix information. The update records have been sorted into data base and segment physical location order by the Prefix Resolution utility. The prefix fields updated by this program include the logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents.

Log output data sets are optionally created if log (IEFRDER, IEFRDER2) DD statements are included in the JCL. This log output can be used if a later data base recovery is required. It is especially useful for the update of scanned data bases. However, batch backout may not be performed using the log output. If DBRC is active when prefix update executes and no log DD statements are supplied, a DBRC NOTIFY.REORG is automatically recorded in RECON for each data set updated.

If execution of this program is abnormally terminated for any reason other than a data base I/O error, program execution can be resumed by requesting a restart action.

If execution of this program is abnormally terminated because of a data base I/O error, the cause of the I/O error should be determined and rectified. The data base should then be restored as it existed before execution of this program; the program should then be reexecuted, using the original input work data set.

Note: The functions of this utility can be performed by the Utility Control Facility, if required. Refer to "Utility Control Facility" for a description of its operation.

Figure 73 is a flow diagram of the Data Base Prefix Update utility.

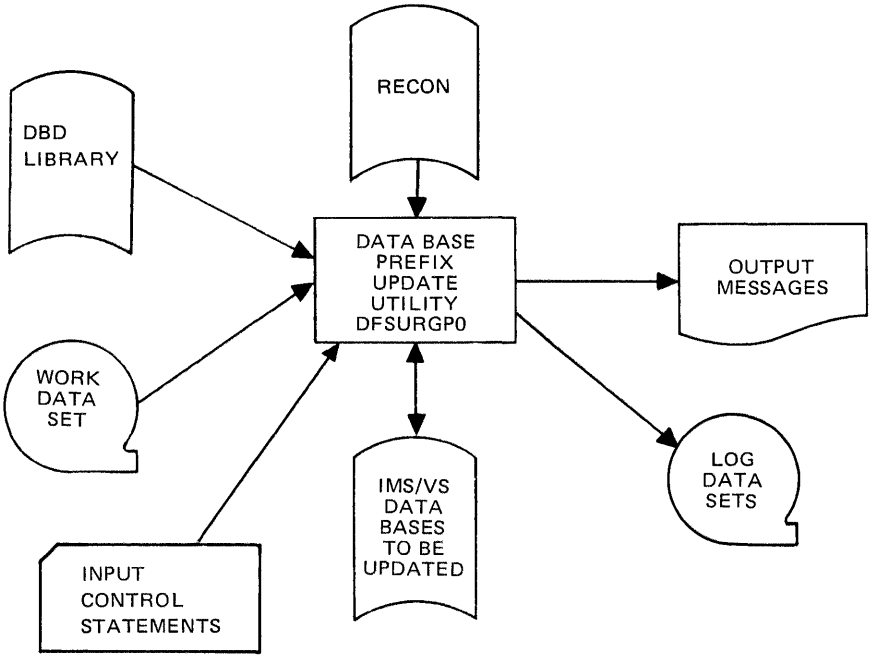


Figure 73. Data Base Prefix Update Utility

JCL REQUIREMENTS

The Data Base Prefix Update utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC
This statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGP0'
```

The normal IMS/VS positional parameters such as SPIE and BUF can follow the program name in the PARM field. See the DBBBATCH or DLIBATCH procedures in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBDs that describe the data base(s) that was loaded and/or reorganized. This DD statement must always be included. The data set must reside on a direct access device.

SYSIN DD
Defines the data set that is to contain input control statements. The data set can reside on a card reader, tape, or direct access device, or be routed through the input stream. This DD statement need not be included unless control statements are supplied as input to this program.

DCB parameters specified within this program are RECFM=FB and LRECL=80. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSPRINT DD
Defines the message data set. The data set can reside on a printer, tape, or direct access device, or be routed through the output stream. This DD statement must always be included.

DCB parameters supplied by the program are RECFM=FB and LRECL=120. BLKSIZE must be specified on this DD statement and must be a multiple of LRECL. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SNAPDD
Defines the snap output data set for this program. This statement is required only if the SNAP control statement is specified in the SYSIN data stream. The data set can reside on a printer, tape, direct access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB, LRECL=120. BLKSIZE must be provided on this DD statement and must be a multiple of LRECL. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

SYSUDUMP DD
Defines the dump data set for this program. This DD statement is required only if the ABEND utility control statement is included. The data set can reside on a printer, tape, direct access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB, LRECL=120. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no default, and the results are unpredictable.

DFSURWF3 DD
Defines the input work data set for this program. It must be the output data set generated by the Prefix Resolution utility on the //DFSURWF3 DD statement. The data set can reside on a tape or direct access device. This DD statement must always be included.

database DD
References the data base(s) that were initially loaded and/or reorganized and/or scanned. One or more DD statements must be present for each data set group of a data base that has logical relationships. The ddname must match the ddname indicated in the DBD. If a HIDAM data base is operated upon with this utility, DD statements must be supplied for its primary index data base.

This data set must reside on a direct access device.

IEFRDER DD
Describes the system log data set created during prefix update. The data set usually resides on tape; however, a direct address volume may be used. This statement is optional and should be included only when log output is desired.

IEFRDER2 DD
Describes the secondary system log data set created during prefix update. The data set usually resides on tape; however, a direct address volume may be used. This statement is optional and should be included only when dual log output is desired.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I Buffer Handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.)

The data set can reside on a tape, direct access device, or card reader, or be routed through the input stream.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS

```
1                                                              80
┌──────────────────────────────────────────────────────────────┐
│ CHKPT={NO }                                                  │
│       {YES}                                                  │
└──────────────────────────────────────────────────────────────┘
```

This utility control statement indicates that checkpoint operations are to be performed during operation of this program. The Prefix Resolution utility automatically writes records on the DFSURWF3 data set as a service for the Prefix Update utility, and these records are used as checkpoints. As each checkpoint record is encountered, a checkpoint message (DFS867) is issued to the OS/VS system console. The message indicates the name of this program, the checkpoint number of the checkpoint record, and the volume serial identifier of the volume being processed. The checkpoint messages should be saved in case a restart operation is required.

```
1                                                              80
┌──────────────────────────────────────────────────────────────┐
│ RSTRT={NO                 }                                  │
│       {nnnnn, volser      }                                  │
└──────────────────────────────────────────────────────────────┘
```

This control statement indicates that a restart operation is to be performed by this program. "nnnnn" is a 5-digit decimal number and "volser" is the volume serial identifier of the input volume containing the checkpoint record numbered "nnnnn." The two parameters, "nnnnn" and "volser," may be obtained from the checkpoint messages that were issued to the OS/VS system console.

If the volume specified on this control statement is not mounted, FEOVs are issued until the correct volume is made available. When restart is completed, a message (DFS378) is issued indicating the checkpoint number, volume serial, and program name. Processing continues from the restart point.

This optional utility control statement indicates that IMS/VS control blocks are to be printed to the SNAPDD data set for diagnostic information.

The STATUS parameter causes snaps to be taken whenever an abnormal status code is returned by DL/I or an abnormal return code is returned from the buffer handler.

The nnnn parameter specifies that a snap is taken after nnnn number of calls are made to the buffer handler. As many as 25 SNAP statements with nnnn specified are accepted. The nnnn parameter must be in the range from 1 to 9999999 with no blanks or commas embedded. The significant digits are required, but blanks cannot appear between the equal sign and the first digit of the relative call number.

There is a one-to-one correlation between the buffer handler calls and the records from the DFSURWF3 data set.

This optional utility control statement should be included when a storage dump is required for diagnostic purposes. If included in the input stream, and if any abnormal condition arises during execution of this utility, any return code greater than zero causes user abend 955 to be issued. The abend is issued at end of program execution, and a storage dump is provided at that time. If this control statement is included, a SYSUDUMP DD statement is also required.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	No errors detected.
8	One or more error messages issued. The messages contain details on the error(s) and are printed as part of the system output.

EXAMPLE

This example shows the JCL required to execute DFSURGP0 to update the two logically related data bases defined by the HDPAYROL and HDSKILLS DD statements.

```
//RLUTIL JOB 1,1,MSGLEVEL=1
//STEP1  EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGP0',REGION=250K
//STEPLIB    DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS    DD      DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSURWF3   DD  DSN=IMSVS.URWF3,UNIT=TAPE,VOL=SER=TAPE31,
// LABEL=(,SL),DISP=(OLD,KEEP),DCB=(LRECL=300,BLKSIZE=1008,
//   RECFM=VB)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=SYSDA,
//   VOL=SER=DB0001,DISP=OLD
//HDSKILLS   DD  DSN=DATABASE.SKILLS,UNIT=SYSDA,
//   VOL=SER=DB0002,DISP=OLD
//DFSVSAMP   DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

OUTPUT MESSAGES

If no errors are detected by this program, the only output message issued will be a normal program termination message that indicates the number of records processed.

The following is an example of reorganizing a HIDAM OSAM data base (OLCDB112), its index ISAM data base (OLCDI112), and its secondary index VSAM data base (OLCDX112).

```
//DBREORG  JOB  A
//JOBLIB   DD DSN=IMSVS.RESLIB,DISP=SHR
//JOBCAT   DD DSN=VCATQAV,DISP=SHR
//*****************************************************************
//*     -DBDNAME-      -DDNAME-      -DSNAME-      -ACCESS-
//*      OLCDB112      OLCD112       OLC.D112      HIDAM OSAM
//*      OLCDI112      OLCDI112      OLC.DI112     INDEX ISAM
//*      OLCDI112      OLCDO112      OLC.DO112     INDEX OSAM (OVFLOW)
//*      OLCDX112      OLCDX112      OLC.DX112     INDEX VSAM (SEC)
//*****************************************************************
//*
//*****************************************************************
//*      PREREORGANIZATION UTILITY
//*****************************************************************
//*
//PREREORG EXEC PGM=DFSRRC00,PARM='ULU,DFSURPR0'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(,PASS),
//            SPACE=(TRK,1),DCB=BLKSIZE=1600
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//SYSIN    DD *
DBR=OLCDB112
/*
//*
```

```
//*******************************************************************
//*      UNLOAD THE HIDAM DATA BASE - OLCDB112
//*******************************************************************
//*
//UNLOAD   EXEC PGM=DFSRRC00,REGION=180K,
//         PARM='ULU,DFSURGU0,OLCDB112'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURGU1 DD DSN=&&ULD1,DISP=(NEW,PASS,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(1,1))
//OLCD112  DD DSN=OLC.D112,DISP=SHR                                ,V
//OLCDI112 DD DSN=OLC.DI112,DISP=SHR                                ,
//OLCDO112 DD DSN=OLC.DO112,DISP=SHR                                ,
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//DFSVSAMP DD *
2048,5
4096,5
IOBF=(512,7)
IOBF=(1024,4)
IOBF=(2048,4)
IOBF=(8192,4)
/*
//*
//*******************************************************************
//*      SCRATCH THE HIDAM OSAM, INDEX ISAM AND OSAM DATA SETS
//*              (NOTE:  THIS STEP IS UNNECCESSARY FOR OSAM
//*               DATA BASES UNLESS ADDITIONAL SPACE IS DESIRED.)
//*******************************************************************
//*
//SCRATCH  EXEC PGM=IEFBR14,REGION=6K
//DD1      DD DSN=OLC.D112,DISP=(OLD,DELETE)
//DD2      DD DSN=OLC.DI112,DISP=(OLD,DELETE)
//DD3      DD DSN=OLC.DO112,DISP=(OLD,DELETE)
/*
//*
//*******************************************************************
//*      RELOAD THE HIDAM DATA BASE - OLCDB112
//*******************************************************************
//*
//RELOAD   EXEC PGM=DFSRRC00,REGION=180K,
//         PARM='ULU,DFSURGL0,OLCDB112'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSUINPT DD DSN=&&ULD1,DISP=(OLD,DELETE,DELETE),UNIT=SYSDA
//DFSURWF1 DD DSN=&&WF1,UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(1,1)),
//            DCB=(RECFM=VB,LRECL=300,BLKSIZE=1008)
//OLCDI112 DD DSN=OLC.DI112,UNIT=SYSDA,DISP=(,CATLG),
//            VOL=SER=IMSQAC,SPACE=(CYL,1),DCB=DSORG=IS
//OLCDO112 DD DSN=OLC.DO112,UNIT=SYSDA,DISP=(,CATLG),
//            VOL=SER=IMSQAC,SPACE=(TRK,(1,1))
//OLCD112 DD DSN=OLC.D112,UNIT=SYSDA,DISP=(,CATLG),
//            VOL=SER=IMSQAC,SPACE=(TRK,(1,1))
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//DFSVSAMP DD *
2048,4
4096,4
IOBF=(512,7)
IOBF=(1024,4)
IOBF=(2048,4)
IOBF=(8192,4)
/*
//*
```

```
//*******************************************************************
//*       PREFIX RESOLUTION
//*******************************************************************
//*
//PREFRES  EXEC PGM=DFSURG10
//SYSOUT   DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//DFSURIDX DD DSN=&&IDX,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(1,1)),
//            DCB=(RECFM=VB,LRECL=300,BLKSIZE=1008)
//DFSURWF2 DD DSN=&&WF2,DISP=(,DELETE),SPACE=(CYL,(1),,CONTIG),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=300,BLKSIZE=1008)
//DFSURWF3 DD DSN=&&WF3,DISP=(,PASS),SPACE=(CYL,(1),,CONTIG),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=300,BLKSIZE=1008)
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSURCDS DD DSN=&&URCDS,DISP=(OLD,DELETE)
//SORTIN   DD DSN=&&WF1,DISP=(OLD,DELETE)
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//*
//*******************************************************************
//*       UNLOAD SECONDARY INDEX - DBDNAME OLCDX112
//*******************************************************************
//*
//URUL     EXEC PGM=DFSRRC00,PARM='ULU,DFSURUL0'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//OLCDX112 DD DSN=OLC.DX112,DISP=SHR
//DDIXOUT1 DD DSN=&&XOUT1,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(1,1))
//DFSURIDX DD DSN=&&IDX,DISP=(OLD,DELETE)
//DFSVSAMP DD *
2048,4
4096,4
IOBF=(512,7)
IOBF=(1024,4)
IOBF=(2048,4)
IOBF=(8192,4)
//SYSIN    DD *
X1MOLCDX112 OLCDX112 DDIXOUT1          DFSURIDX
/*
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//*
//*******************************************************************
//*     SCRATCH AND REALLOCATE SECONDARY INDEX
//*******************************************************************
//*
//VSAM     EXEC PGM=IDCAMS,REGION=200K
//SYSPRINT DD  SYSOUT=A
//VSDATA   DD  UNIT=SYSDA,VOL=SER=IMSQAV,DISP=SHR
//SYSIN DD *
 DELETE  OLC.DX112
 DEFINE  CLUSTER (NAME (OLC.DX112) -
            TRK(1,1) -
            SPEED -
            VOL (IMSQAV) -
            FREESPACE (20,10) -
            SHAREOPTIONS (3,3) ) -
          DATA (NAME(OLC.DX112D) -
            RECSZ(12,12) KEYS (6,5) CISZ (512) )-
          INDEX (NAME(OLC.DX112S) CISZ (2048) )
//*
```

```
//*****************************************************************
//*     RELOAD SECONDARY INDEX - OLCDX112
//*****************************************************************
//*
//URRL     EXEC PGM=DFSRRC00,PARM='ULU,DFSURRL0'
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//DFSUIN01 DD DSN=&&XOUT1,DISP=(OLD,DELETE)
//OLCDX112 DD DSN=OLC.DX112,DISP=SHR                                  ,
//DFSVSAMP DD *
2048,4
4096,4
IOBF=(512,7)
IOBF=(1024,4)
IOBF=(2048,4)
IOBF=(8192,4)
/*
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//*
//*****************************************************************
//*       PREFIX UPDATE
//*****************************************************************
//*
//UPDATE   EXEC PGM=DFSRRC00,
//         PARM='ULU,DFSURGP0'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURWF3 DD DSN=&&WF3,DISP=(OLD,DELETE),UNIT=SYSDA
//OLCDI12  DD DSN=OLC.DI12,DISP=SHR
//OLCDI112 DD DSN=OLC.DI112,DISP=SHR                                  ,
//OLCDO112 DD DSN=OLC.DO112,DISP=SHR                                  ,
//OLCDX112 DD DSN=OLC.DX112,DISP=SHR                                  ,
//DFSVSAMP DD *
2048,4
4096,4
IOBF=(512,7)
IOBF=(1024,4)
IOBF=(2048,4)
IOBF=(8192,4)
/*
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//
```

EXAMPLE USING DFSLRLOD PROCEDURE

```
//*
//*        EXECUTION OF THE FOLLOWING PROCEDURE CAUSES THREE LOGICALLY
//*        RELATED DATABASES TO BE INITIALLY LOADED AND THEIR LOGICAL
//*        RELATIONSHIPS TO BE RESOLVED.
//*
//*
//LOAD     EXEC    PROC=DFSLRLOD,PSB=HZBLHP40,PGML=DFSDDLT0
//*
//*        THE FOLLOWING JCL DEFINES THE INPUT DATA FOR THE
//*        PREREORGANIZATION UTILITY (DFSURPR0).
//*
//P.SYSIN  DD      *
DBIL=DI41HP02,DH41HP03,DH41HP02
//*
//*        THE FOLLOWING JCL DEFINES THE INPUT DATA TO BE LOADED
//*        BY THE INITIAL LOAD PROGRAM.
//*
//L.SYSIN  DD      DSN=ICS.CSOR(HPHD02),DISP=SHR
//         DD      DSN=ICS.CSOR(HPHD03),DISP=SHR
//         DD      DSN=ICS.CSOR(HPHI02),DISP=SHR
//*
//*        THE FOLLOWING JCL DEFINES THE THREE LOGICALLY RELATED
//*        DATABASES TO BE LOADED BY THE INITIAL LOAD PROGRAM.
//*        THE FIRST TWO DATABASES ARE HIDAM.  THE THIRD DATABASE
//*        IS HISAM.
//*
//L.DHSK0201 DD    DSN=IMS.DDB.DHSK0201,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//L.X      DD      DSN=ICS.DDB.INDEXI,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=(DSORG=IS,
//         OPTCD=WM)
//L.X0     DD      DSN=ICS.DDB.INDEX0,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1,1))
//L.DHSK0301 DD    DSN=IMS.DDB.DHSK0301,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//L.X2     DD      DSN=ICS.DDB.INDEXI2,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=(DSORG=IS,
//         OPTCD=WM)
//L.X02    DD      DSN=ICS.DDB.INDEX02,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(,KEEP),SPACE=(CYL,(1,1))
//L.DISK0201 DD    DSN=IMS.DDB.DISK0201,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=DSORG=IS
//L.DISK0202 DD    DSN=IMS.DDB.DISK0202,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1,1))
//*
//*
//*        THE FOLLOWING JCL DEFINES THE THREE LOGICALLY RELATED
//*        DATABASES TO BE UPDATED BY PREFIX UPDATE UTILITY (DFSURGP0).
//*
//U.DHSK0201 DD    DSN=*.LOAD.L.DHSK0201,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=SYSDA
//U.X      DD      DSN=*.LOAD.L.X,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.X0     DD      DSN=*.LOAD.L.X0,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.DHSK0301 DD    DSN=*.LOAD.L.DHSK0301,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=SYSDA
//U.X2     DD      DSN=*.LOAD.L.X2,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.X02    DD      DSN=*.LOAD.L.X02,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.DISK0201 DD    DSN=*.LOAD.L.DISK0201,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=SYSDA
//U.DISK0202 DD    DSN=*.LOAD.L.DISK0202,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
```

The following is an example of reorganizing two logically related data bases. The two data bases are: DIVNTZ02 (a HISAM VSAM data base) and DHVNTZ02 (a HIDAM VSAM data base with an index VSAM data base (DXVNTZ02).

```
//DBREORG2  JOB A
//JOBLIB    DD DSN=IMSVS.RESLIB,DISP=SHR
//JOBCAT    DD DSN=VCATREC,DISP=SHR
//*
//*****************************************************************
//*  -DBDNAME-      -DDNAME-       -DSNAME-      -ACCESS-
//*   DIVNTZ02      DBHVSAM1       JDSG1RC       HISAM VSAM (PRIME)
//*      "          DBHVSAM2       JDSG1RC0      HISAM VSAM (OVERFLOW)
//*   DHVNTZ02      HIDAM          KDSG1RC       HIDAM VSAM (GROUP1)
//*      "          HIDAM2         KDSG2RC       HIDAM VSAM (GROUP2)
//*   DXVNTZ02      XDLBT04I       KINDXRC       INDEX VSAM
//*****************************************************************
//*
//*****************************************************************
//*   PREREORGANIZATION
//*****************************************************************
//*
//PREREORG EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURPR0,,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,
//         UNIT=SYSDA,
//         DISP=(,PASS,DELETE),
//         SPACE=(TRK,(10,10),RLSE),
//         DCB=(BLKSIZE=1600)
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//SYSIN    DD *
OPTIONS=(NOPUNCH,STAT,SUMM)
DBR=DIVNTZ02
DBR=DHVNTZ02
/*
//*****************************************************************
//*      UNLOAD THE HIDAM DATA BASE - DHVNTZ02
//*****************************************************************
//*
//UNLOAD1  EXEC PGM=DFSRRC00,REGION=1024K,COND=(1,LT),
//         PARM='ULU,DFSURGU0,DHVNTZ02,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,ASN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,
//         UNIT=SYSDA,DISP=(OLD,PASS)
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//DFSURGU1 DD DSN=&&ULD1A,DISP=(NEW,PASS,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(5,1))
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
```

```
//*******************************************************************
//*      UNLOAD THE HISAM DATA BASE - DIVNTZ02
//*******************************************************************
//*
//UNLOAD2  EXEC PGM=DFSRRC00,REGION=1024K,COND=(1,LT),
//         PARM='ULU,DFSURGU0,DIVNTZ02,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,
//         UNIT=SYSDA,DISP=(OLD,PASS)
//DBHVSAM1 DD DSN=JDSG1RC,DISP=SHR
//DBHVSAM2 DD DSN=JDSG1RC0,DISP=SHR
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//DFSURGU1 DD DSN=&&ULD1B,DISP=(NEW,PASS,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(5,1))
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
/*
//*******************************************************************
//*      SCRATCH AND REALLOCATE THE VSAM DATA BASES
//*******************************************************************
//*
//SCRATCH  EXEC  PGM=IDCAMS,COND=(1,LT)
//SYSPRINT DD SYSOUT=*
//VSAMDD   DD UNIT=SYSDA,DISP=SHR,VOL=SER=RECRES
//SYSIN    DD *
 DELETE  JDSG1RC   PURGE FILE(VSAMDD)
 DELETE  JDSG1RC0  PURGE FILE(VSAMDD)
 DELETE  KDSG1RC   PURGE FILE(VSAMDD)
 DELETE  KDSG2RC   PURGE FILE(VSAMDD)
 DELETE  KINDXRC   PURGE FILE(VSAMDD)
 DEFINE CLUSTER (NAME (JDSG1RC) -
          CYLINDERS (2,1) -
          VOL (RECRES) -
          FREESPACE (30,20) -
          SHAREOPTIONS (3,3) -
          RECSZ (200,200) -
          KEYS (5,6) -
          UNIQUE SPEED) -
        DATA (NAME(JDSG1RC1) -
          CISZ (1024)) -
        INDEX (NAME(JDSG1RC2) -
          CISZ (1024)) -
        CATALOG (VCATREC)
```

```
 DEFINE CLUSTER (NAME (JDSG1RC0) -
          CYLINDERS (1,1) -
          VOL (RECRES) -
          SHAREOPTIONS (3,3) -
          RECSZ (200,200) -
          NIXD -
          UNIQUE) -
        DATA (NAME(JDSG1RC3) -
          CISZ (512)) -
        CATALOG (VCATREC)
 DEFINE CLUSTER (NAME (KDSG1RC) -
          CYLINDERS (2,1) -
          VOL (RECRES) -
          SHAREOPTIONS (3,3) -
          RECSZ (2041,2041) -
          NIXD -
          UNIQUE) -
        DATA (NAME(KDSG1RC1) -
          CISZ (2048)) -
        CATALOG (VCATREC)
 DEFINE CLUSTER (NAME (KDSG2RC) -
          CYLINDERS (1,1) -
          VOL (RECRES) -
          SHAREOPTIONS (3,3) -
          RECSZ (505,505) -
          NIXD -
          UNIQUE) -
        DATA (NAME(KDSG2RC1) -
          CISZ (512)) -
        CATALOG (VCATREC)
 DEFINE CLUSTER (NAME (KINDXRC) -
          CYLINDERS (1,1) -
          VOL (RECRES) -
          FREESPACE (30,20) -
          SHAREOPTIONS (3,3) -
          KEYS (5,5) -
          RECSZ (12,12) -
          SPEED -
          UNIQUE) -
        DATA (NAME(KINDXRC1) -
          CISZ (512)) -
        INDEX (NAME(KINDXRC2) -
          CISZ (1024)) -
        CATALOG (VCATREC)
//*
//*****************************************************************************
//*      RELOAD THE HIDAM DATA BASE - DHVNTZ02
//*****************************************************************************
//*
//RELOAD1  EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURGL0,DHVNTZ02,,1,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSUINPT DD DSN=&&ULD1A,DISP=(OLD,DELETE),UNIT=SYSDA
//DFSURWF1 DD DSN=&&WF1A,UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(2,1)),
//            DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
```

```
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
//*
//*****************************************************************
//*      RELOAD THE HISAM DATA BASE - DIVNTZ02
//*****************************************************************
//*
//RELOAD2  EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURGL0,DIVNTZ02,,1,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSUINPT DD DSN=&&ULD1B,DISP=(OLD,DELETE),UNIT=SYSDA
//DFSURWF1 DD DSN=&&WF1B,UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(2,1)),
//            DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//DBHVSAM1 DD DSN=JDSG1RC,DISP=SHR
//DBHVSAM2 DD DSN=JDSG1RC0,DISP=SHR
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
//*
//*****************************************************************
//*     PREFIX RESOLUTION
//*****************************************************************
//*
//PREFRES  EXEC PGM=DFSURG10,REGION=1024K
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSOUT   DD SYSOUT=A
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//SORTIN   DD DSN=&&WF1A,UNIT=SYSDA,DISP=(OLD,DELETE)
//         DD DSN=&&WF1B,UNIT=SYSDA,DISP=(OLD,DELETE)
//DFSURWF2 DD DSN=&&WF2,
//         UNIT=SYSDA,
//         DISP=(,DELETE),
//         SPACE=(CYL,(2,2)),
//         DCB=BLKSIZE=1008
```

```
//DFSURWF3 DD DSN=&&WF3,
//          DISP=(,PASS),
//          UNIT=SYSDA,
//          SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=VB,LRECL=300,BLKSIZE=13030,BUFNO=8)
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSURIDX DD DUMMY,DCB=BLKSIZE=1008
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//*
//*****************************************************************
//*     PREFIX UPDATE
//*****************************************************************
//*
//UPDATE   EXEC PGM=DFSRRC00,REGION=1024K,
//          PARM='ULU,DFSURGP0,,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMSVS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURWF3 DD DSN=&&WF3,
//          DISP=(OLD,DELETE),
//          UNIT=SYSDA
//DBHVSAM1 DD DSN=JDSG1RC,DISP=SHR
//DBHVSAM2 DD DSN=JDSG1RC0,DISP=SHR
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
//
```

UTILITY PROCEDURES

To reduce the input JCL required to perform the various data base utility functions described in this chapter, it is recommended that each using installation develop a set of utility procedures. This chapter provides several sample procedures and JCL examples that will enable the user to perform commonly used data base operations with the minimum JCL.

SAMPLE PROCEDURES FOR DATA BASE REORGANIZATION/LOAD UTILITIES

The procedure names and symbolic parameters used in the procedures are for illustrative purposes only; they can be changed to meet the needs of each installation.

These procedures are not included on the IMS/VS system distribution tapes; they are not supported as part of the IMS/VS program product. In most cases, the suggested procedures will have to be modified to conform to the requirements of each installation. The particular areas that will probably require modification are highlighted by notes in the description of each procedure.

The following procedures are included in this section:

Procedure	Description
DFSLRLOD	A four-step procedure for initially loading one or more data bases containing logical relationships
DFSPRL	A one-step procedure for executing the Prereorganization utility (DFSURPR0)
DFSILOAD	A one-step procedure for executing an initial data base load program
DFSHDUR	A two-step procedure for reorganizing a data base using the HD Unload/Reload utilities (DFSURGU0, DFSURGL0)
DFSSCAN	A one-step procedure for scanning one or more data bases using the Data Base Scan utility (DFSURGS0)
DFSLRRES	A two-step procedure for resolving logical relationships and updating one or more data bases using the Prefix Resolution utility (DFSURG10) and the Prefix Update utility (DFSURGP0)

DESCRIPTION OF DFSLRLOD PROCEDURE

This procedure can be used to initially load one or more data bases and to resolve any logical relationships which may be present in the data bases. All the data bases must be loaded during the same execution of the initial load program. If it is not possible to load all data bases during the same execution of the initial load program, or if the user wishes to intermix one or more initial data base loads with one or more data base reorganizations, it is necessary to use the other procedures.

The DFSLRLOD procedure consists of four steps:

1. Step P—Execute Prereorganization utility (DFSURPR0).
2. Step L—Execute initial data base load program.
3. Step R—Execute Prefix Resolution utility (DFSURG10).
4. Step U—Execute Prefix Update utility (DFSURGP0).

The symbolic parameters of this procedure have the following meanings:

PGML
Is the name of the initial data base load program. This program is supplied by the using installation. It must execute as a DL/I batch program.

PSB
Is the name of the PSB to be used by the initial load program. This PSB must contain one PCB for each data base to be initially loaded. PROCOPT for each PCB should be L or LS, as appropriate.

SOUT
Provides definition of the SYSOUT CLASS for the print output data set.

BUF
Specifies the data base buffer pool size (in 1K increments).

SPIE
0=none; 1=issue of SPIE is effective at entry.

TEST

Specifies that parameter list address validity-checking is to be performed. Validity check values: 0=none; 1=all.

Notes:

1. Control statements for DFSURPR0 (step P) must be supplied through the data set defined by a //P.SYSIN DD statement.

2. DD statements must be provided for the data sets representing the DL/I data bases being loaded. DD statements must be provided for both steps L and U.

3. Input data to be loaded into the data base(s) must be defined by DD statements provided for step L.

4. The definition of the data sets defined by the following DD statements should be modified according to the size of the data bases being initially loaded: DFSURWF1, SORTIN, SORTWK01, SORTWK02, SORTWK03,..., DFSURWF2, DFSURWF3. The parameters that most likely require modification are: SPACE=, VOL=, UNIT=, and DSN=.

DESCRIPTION OF DFSPRL, DFSILOAD, DFSHDUR, DFSSCAN, AND DFSLRRES PROCEDURES

These procedures can be used to intermix the initial load of one or more data bases with the reorganization and/or scan of one or more data bases. The purpose of each procedure is described below:

1. DFSPRL

   Step P—Execute Prereorganization utility (DFSURPR0).

2. DFSILOAD

   Step L—Execute initial data base load program.

3. DFSHDUR

   Step UL—Execute HD Unload utility (DFSURGU0).

   Step RL—Execute HD Reload utility (DFSURGL0).

4. DFSSCAN

   Step S—Execute Scan utility (DFSURGS0).

5. DFSLRRES

   Step R—Execute Prefix Resolution utility (DFSURG10).

   Step U—Execute Prefix Update utility (DFSURGP0).

A typical sequence of executing the above procedures is as follows:

1. Execute procedure DFSPRL. If no errors are detected, output messages will indicate which of the other procedures must be executed: The data bases listed after the characters DBIL= must be initially loaded (the DFSILOAD procedure should be used); the data bases listed after the characters DBR= must be reorganized by using the HD Unload/Reload utilities (the DFSHDUR procedure should be used); the data bases listed after the characters DBS= must be scanned using the Data Base Scan utility (the DFSSCAN procedure should be followed).

2. Execute procedures DFSILOAD, DFSHDUR, and DFSSCAN as indicated in item 1 above. The procedure DFSILOAD can be executed one or more times, depending upon the operation of

the user-provided initial load program. The procedure DFSHDUR must be executed once for each data base to be reorganized. The procedure DFSSCAN is executed once (the procedures are defined such that only one execution of DFSSCAN is required for all data bases which must be scanned). It should be noted that DFSILOAD, DFSHDUR, and DFSSCAN can be executed concurrently. The data sets produced for the DD statement named DFSURWF1 during execution of each procedure must be concatenated to form the SORTIN data set for execution of the DFSLRRES procedure.

3. Execute procedure DFSLRRES. This procedure resolves any logical relationships which may be present in the data base(s) that were initially loaded, reorganized, or scanned.

The symbolic parameters of these procedures have the following meanings:

PGML, PSB, SOUT, BUF, SPIE, TEST
Same meanings as those for the DFSLRLOD procedure.

DBDUL
Is the name of the DBD that describes the data base to be unloaded.

DBDRL
Is the name of the DBD that describes the data base to be reloaded.

Notes:

1. Control statements for DFSURPR0 (step P of procedure DFSPRL) must be supplied through the data set defined by a //P.SYSIN DD statement.

2. DD statements must be supplied for the data sets representing the DL/I data bases being loaded, reorganized, or scanned. The data base DD statement requirements by procedure and step are:

   DFSILOAD
   Step L—DD statements for each data base to be initially loaded during each execution of the load program.

   DFSHDUR
   Step UL—DD statements for the data base to be unloaded.

   Step RL—DD statements for the data base to be reloaded.

   DFSSCAN
   Step S—DD statements for each data base to be scanned.

   DFSLRRES
   Step U—DD statements for each data base to be updated. DD statements should be present for each data base that was initially loaded, reorganized, or scanned.

3. Input data to be loaded into the data base(s) must be defined by DD statements provided for step L each time the DFSILOAD procedure is executed.

4. The definition of the data sets defined by the following DD statements should be modified according to the size of the data bases being initially loaded, reorganized, or scanned: DFSURWF1, SORTIN, SORTWK01, SORTWK02, SORTWK03,...., DFSURWF2, DFSURWF3, DFSURGU1, DFSUINPT. The parameters that probably require modification are SPACE=, VOL=, UNIT=, and DSN=.

5. These procedures pass temporary data sets to one another. If the procedures are not executed within the same operating system job, the appropriate DD statements must be modified to retain these data sets between jobs.

6. These procedures can be used for ISAM data bases if the appropriate DD statements are changed and /DFSVSAMP DD* statement and proper buffer control statements are added to those procedures requiring VSAM data sets.

DFSPRL PROCEDURE

```
//          PROC    SOUT=3,SOUTP=B
//P         EXEC    PGM=DFSRRC00,PARM='ULU,DFSURPR0',REGION=250K
//*
//*         THIS STEP EXECUTES THE PREREORGANIZATION UTILITY
//*         (DFSURPR0).
//*
//STEPLIB   DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD      DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD      DSN=IMSVS.DBDLIB,DISP=SHR
//DFSURCDS  DD      UNIT=SYSDA,SPACE=(TRK,1),DSN=&&CDS,DISP=(,PASS),
//          DCB=BLKSIZE=1600
//SYSPRINT  DD      SYSOUT=&SOUT,DCB=BLKSIZE=1200
//SYSPUNCH  DD      SYSOUT=&SOUTP,DCB=BLKSIZE=400
```

DFSILOAD PROCEDURE

```
//          PROC    PSB=,PGML=,BUF=8,SPIE=0,TEST=0,SOUT=3
//L         EXEC    PGM=DFSRRC00,PARM='DLI,&PGML,&PSB,&BUF,&SPIE&TEST',
//          REGION=150K
//*
//*         THIS STEP EXECUTES THE INITIAL DATABASE LOAD PROGRAM.
//*
//STEPLIB   DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD      DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD      DSN=IMSVS.PSBLIB,DISP=SHR
//          DD      DSN=IMSVS.DBDLIB,DISP=SHR
//DFSURWF1  DD      DSN=&&WF1,UNIT=SYSDA,DISP=(MOD,PASS),
//          SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=900,
//          BLKSIZE=1008)
//DFSURCDS  DD      DSN=&&CDS,DISP=(OLD,PASS),UNIT=SYSDA
//DFSVSAMP  DD      *
XXXX,YY
```

DFSHDUR PROCEDURE

```
//          PROC    DBDUL=,DBDRL=,SOUT=3,BUF=8
//UL        EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGU0,&DBDUL,&BUF',
//          REGION=150K
//*
//*         THIS STEP EXECUTES THE HD UNLOAD UTILITY (DFSURGU0).
//*
//STEPLIB   DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD      DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD      DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD      SYSOUT=&SOUT
//DFSURGU1  DD      DSN=&&UL,UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(1,1))
//RL        EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGL0,&DBDRL,&BUF',
//          REGION=150K,COND=(1,LT,UL)
//*
//*         THIS STEP EXECUTES THE HD RELOAD UTILITY (DFSURGL0).
//*
//IMS       DD      DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD      SYSOUT=&SOUT
//DFSUINPT  DD      DSN=&&UL,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSURWF1  DD      DSN=&&WF1,UNIT=SYSDA,DISP=(MOD,PASS),
//          SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=900,
//          BLKSIZE=1008)
//DFSURCDS  DD      DSN=&&CDS,DISP=(OLD,PASS),UNIT=SYSDA
```

DFSSCAN PROCEDURE

```
//          PROC    SOUT=3,BUF=8
//S         EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGS0,,&BUF',REGION=150K
//*
//*         THIS STEP EXECUTES THE SCAN UTILITY (DFSURGS0).
//*
//STEPLIB   DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD      DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD      DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD      SYSOUT=&SOUT,DCB=BLKSIZE=1200
//DFSURWF1  DD      DSN=&&WF1,UNIT=SYSDA,DISP=(MOD,PASS),
//          SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=900,
//          BLKSIZE=1008)
//DFSURCDS  DD      DSN=&&CDS,DISP=(OLD,PASS),UNIT=SYSDA
```

DFSLRRES PROCEDURE

```
//          PROC    SOUT=3,BUF=8
//R         EXEC    PGM=DFSURG10,REGION=150K
//*
//*         THIS STEP EXECUTES THE PREFIX RESOLUTION UTILITY (DFSURG10).
//*
//STEPLIB   DD      DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT  DD      SYSOUT=&SOUT,DCB=BLKSIZE=1200
//SYSOUT    DD      SYSOUT=&SOUT
//SORTLIB   DD      DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD      UNIT=SYSDA,SPACE=(CYL,(02),,CONTIG)
//SORTWK02  DD      UNIT=SYSDA,SPACE=(CYL,(02),,CONTIG)
//SORTWK03  DD      UNIT=SYSDA,SPACE=(CYL,(02),,CONTIG)
//SORTIN    DD      DSN=&&WF1,UNIT=SYSDA,DISP=(OLD,DELETE),
//          DCB=(LRECL=900,BLKSIZE=1008,RECFM=VB)
//DFSURWF2  DD      UNIT=SYSDA,SPACE=(CYL,(1),,CONTIG),DSN=&&WF2,
//          DISP=(,PASS),DCB=(LRECL=900,BLKSIZE=1008,RECFM=VB)
//DFSURWF3  DD      UNIT=SYSDA,SPACE=(CYL,(1),,CONTIG),DSN=&&WF3,
//          DISP=(,PASS),DCB=(LRECL=900,BLKSIZE=1008,RECFM=VB)
//DFSURCDS  DD      DSN=&&CDS,DISP=(OLD,PASS),UNIT=SYSDA
//U         EXEC    PGM=DFSRRC00,PARM='ULU,DFSURGP0,,&BUF',REGION=150K,
//          COND=(7,LT,R)
//*
//*         THIS STEP EXECUTES THE PREFIX UPDATE UTILITY (DFSURGP0).
//*
//IMS       DD      DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD      SYSOUT=&SOUT,DCB=BLKSIZE=1200
//DFSURWF3  DD      DSN=&&WF3,UNIT=SYSDA,DISP=(OLD,PASS)
```

EXAMPLES USING DFSPRL, DFSILOAD, DFSHDUR, AND DFSLRRES PROCEDURES

```
//*
//*        EXECUTION OF THE FOLLOWING PROCEDURES CAUSES ONE DATA BASE TO
//*        BE INITIALLY LOADED AND TWO OTHER LOGICALLY RELATED DATA BASES
//*        TO BE REORGANIZED.  THE TWO DATA BASES TO BE REORGANIZED ARE
//*        HIDAM.  THE DATA BASE TO BE INITIALLY LOADED IS HISAM.
//*
//*
//*
//PREREORG EXEC    PROC=DFSPRL
//*
//*        THE FOLLOWING JCL DEFINES THE INPUT DATA FOR THE
//*        PREREORGANIZATION UTILITY (DFSURPR0).
//*
//P.SYSIN  DD      *
DBIL=DI41HP02
DBR=DH41HP02,DH41HP03
/*
//LOAD     EXEC    PROC=DFSILOAD,PSB=HZBLHP40,PGML=DFSDDLT0
//*
//*        THE FOLLOWING JCL DEFINES THE INPUT DATA TO BE LOADED.
//*
//STEPLIB  DD      DSN=IMSVS.RESLIB,DISP=SHR
//L.SYSIN  DD      DSN=ICS.CSOR(HPHI02),DISP=SHR
//*
//*        THE FOLLOWING JCL DEFINES THE DATA BASE TO BE LOADED.
//*
//L.DISK0201 DD    DSN=IMS.DDB.DISK0201,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=DSORG=IS
//L.DISK0202 DD    DSN=IMS.DDB.DISK0202,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1,1))
//REORG1   EXEC    PROC=DFSHDUR,DBDUL=DH41HP02,DBDRL=DH41HP02
//*
/*        THE FOLLOWING JCL DEFINES THE FIRST OF THE TWO DATA BASES
//*        TO BE REORGANIZED.  NOTE THAT TWO SETS OF DD CARDS ARE
//*        REQUIRED, ONE SET FOR THE UNLOAD STEP AND ONE SET FOR THE
//*        RELOAD STEP.
//*
//UL.DHSK0201 DD   DSN=IMS.DDB.DHSK0201,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//UL.X     DD      DSN=ICS.DDB.INDEXI,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),DISP=(OLD,DELETE)
//UL.XO    DD      DSN=ICS.DDB.INDEXO,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//RL.DHSK0201 DD   DSN=IMS.DDB.DHSK0201,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//RL.X     DD      DSN=ICS.DDB.INDEXI,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),SPACE=(CYL,(2,,1)),
//         DISP=(NEW,KEEP)
//RL.XO    DD      DSN=ICS.DDB.INDEXO,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1,1))
//REORG2   EXEC    PROC=DFSHDUR,DBDUL=DH41HP03,DBDRL=DH41HP03
```

```
//*
//*        THE FOLLOWING JCL DEFINES THE SECOND OF THE TWO DATA BASES
//*        TO BE REORGANIZED.  NOTE THAT TWO SETS OF DD CARDS ARE
//*        REQUIRED, AS DESCRIBED ABOVE.
//*
//UL.DHSK0301 DD   DSN=IMS.DDB.DHSK0301,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//UL.X2    DD      DSN=ICS.DDB.INDEXI2,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),DISP=(OLD,DELETE)
//UL.XO2   DD      DSN=ICS.DDB.INDEXO2,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//RL.DHSK0301 DD   DSN=IMS.DDB.DHSK0301,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//RL.X2    DD      DSN=ICS.DDB.INDEXI2,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),SPACE=(CYL,(2,,1)),
//                    DISP=(NEW,KEEP)
//RL.XO2   DD      DSN=ICS.DDB.INDEXO2,UNIT=SYSDA,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1.1))
//RESOLVE  EXEC    PROC=DFSLRRES
//*
//*        THE FOLLOWING JCL DEFINES THE THREE LOGICALLY RELATED
//*        DATA BASES TO BE UPDATED BY PREFIX UPDATE UTILITY (DFSURGP0)
//*
//U.DHSK0201 DD    DSN=*.LOAD.L.DHSK0201,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.X      DD      DSN=*.LOAD.L.X,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.XO     DD      DSN=*.LOAD.L.XO,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.DHSK0301 DD    DSN=*.LOAD.L.DHSK0301,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=SYSDA
//U.X2     DD      DSN=*.LOAD.L.X2,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.XO2    DD      DSN=*.LOAD.L.XO2,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=SYSDA
//U.DISK0201 DD    DSN=*.LOAD.L.DISK0201,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=SYSDA
```

CHAPTER 7. DATA BASE RECOVERY UTILITIES

IMS/VS provides a combination of system facilities, utility programs, and procedures to help recover data bases that have been physically damaged or improperly updated. This chapter first outlines the steps used in recovery and then describes the IMS/VS Data Base Recovery utilities used in this task.

IMS/VS uses a simple system for physical data base recovery. It creates an image copy of a data base at user-specified times and keeps a log of all changes made to the data base after those times. Then, if the data base is physically damaged, IMS/VS fully restores it by: (1) writing the image copy to DASD, and (2) applying the logged changes to this copy.

To undo changes made by an application program, an IMS/VS utility program can be used that will back out the data base to a specified point.

The system facilities and utilities used to perform recovery are listed by function below (IMS/VS data base recovery utilities are marked with an asterisk):

1. Create a copy of a data base

   *Data Base Image Copy utility, Online Data Base Image Copy utility, or HISAM Reorganization Unload utility

2. Log changes to the data base

   Data Base Change Logger

3. Accumulate changes from IMS/VS log data sets

   *Data Base Change Accumulation utility

4. Recover a data base that has been physically damaged

   *Data Base Recovery utility

5. Undo changes made to a data base by selected application programs

   *Batch Backout utility

Notes:

1. All IMS/VS data base recovery utilities operate in the IMS/VS batch processing region, except the Online Data Base Image Copy utility, which runs online as a batch message processing program.

2. If a VSAM data set has been defined in a user catalog, it is also necessary to identify the user catalog by means of either a JOBCAT or a STEPCAT DD statement. See OS/VS Virtual Storage Access Method Programmer's Guide for more details.

RECOVERY CONSIDERATIONS

USE OF THE SYSTEM LOG FOR DATA BASE RECOVERY

IMS/VS logs all information associated with the modification of data within an existing data base. The information placed on an IMS/VS system log whenever a data base modification is made, includes the following:

- Identification of the data base;
- Identification of the data set within a data base (an offset to the DDNAME within the DBD of the data set);
- Identification of the modified record within the data set (accomplished either by ISAM/KSDS key or OSAM/ESDS relative direct access block number);
- Content of the data base record before it is updated ("before" image);
- Content of the data base record after it is updated ("after" image).

Log records are not created when a data base is initially loaded (that is, when the processing option 'L' or 'LS' is selected). For this reason, and because HSAM and GSAM do not support updates to data bases, the data base recovery utilities do not support the HSAM or GSAM data base organization.

For an HSAM or GSAM data base, it is the user's responsibility to create a backup copy of the data base for recovery from any I/O error that might occur. In addition, when creating a new master of an HSAM or GSAM data base, the user should maintain a copy of the old master along with a copy of all transactions used to create the new master. Then, in the event of an I/O error on the new master, the user can apply this composite backup to create the new master again.

All log records created by online processing are written to DASD data sets referred to as online log data sets (OLDS). When the OLDS is filled and is available to be archived, the Log Archive utility (DFSUARC0) reads the OLDS, creates a system log data set (SLDS), and, optionally, creates a recovery log data set (RLDS). The SLDS or RLDS is input to data base recovery, accounting, and statistics and can be on either DASD or tape. The Log Archive utility is described in the "Log Maintenance Utilities" chapter of this manual.

CHANGE ACCUMULATION

While recovery can be done by using the log data set and the image copy, the time and trouble taken to run the job can be reduced if some processing is done to the changes beforehand. The Change Accumulation utility is provided for this purpose. This utility selects the changed data base log records from the log data set and sorts them in order by data set within data base. The Change Accumulation data set so obtained can then be used as input to the Recovery utility.

BACKOUT

When the status of a data base is not known because the program that was updating the data base terminated abnormally, the Batch Backout utility can be used to back out the effects of the program. (This should only be necessary in the IMS/VS batch processing environment or when program isolation is used for online processing). This utility reads the log created by the processing of the failing program. Using the data base log records thus read, the utility restores the data base to its status at the time the program that failed was originally scheduled or to the program's last sync point record. It also creates a log that must be used as input to any future data base recovery operation or as input to the Data Base Change Accumulation utility.

DATA BASE IMAGE COPY UTILITY (DFSUDMP0)

The Data Base Image Copy utility is a batch utility that creates an output copy of the data sets within a data base. This output is used as input to the Data Base Recovery utility. For information on creating an online copy, see "Online Data Base Image Copy Utility (DFSUICP0)" on page 276.

Multiple data sets or areas can be copied on mixed DASD devices with one execution of the Image Copy utility. For the convenience of operations, all data sets of a data base should be copied at the same time.

For DEDBs, you can specify multiple area data sets of an area as input to the Image Copy utility if the area is registered in the DBRC (Data Base Recovery Control) RECON data set.

Note: HSAM, GSAM, and MSDB data bases cannot be copied with this utility.

The user has the option of creating one or two output image copies. The advantage in specifying two copies is that if an I/O error occurs during copy execution, the utility continues to completion on the other copy. Performance would be somewhat diminished in this instance, but a total rerun would not be necessary.

Figure 74 on page 270 is a flow diagram of the Data Base Image Copy utility.

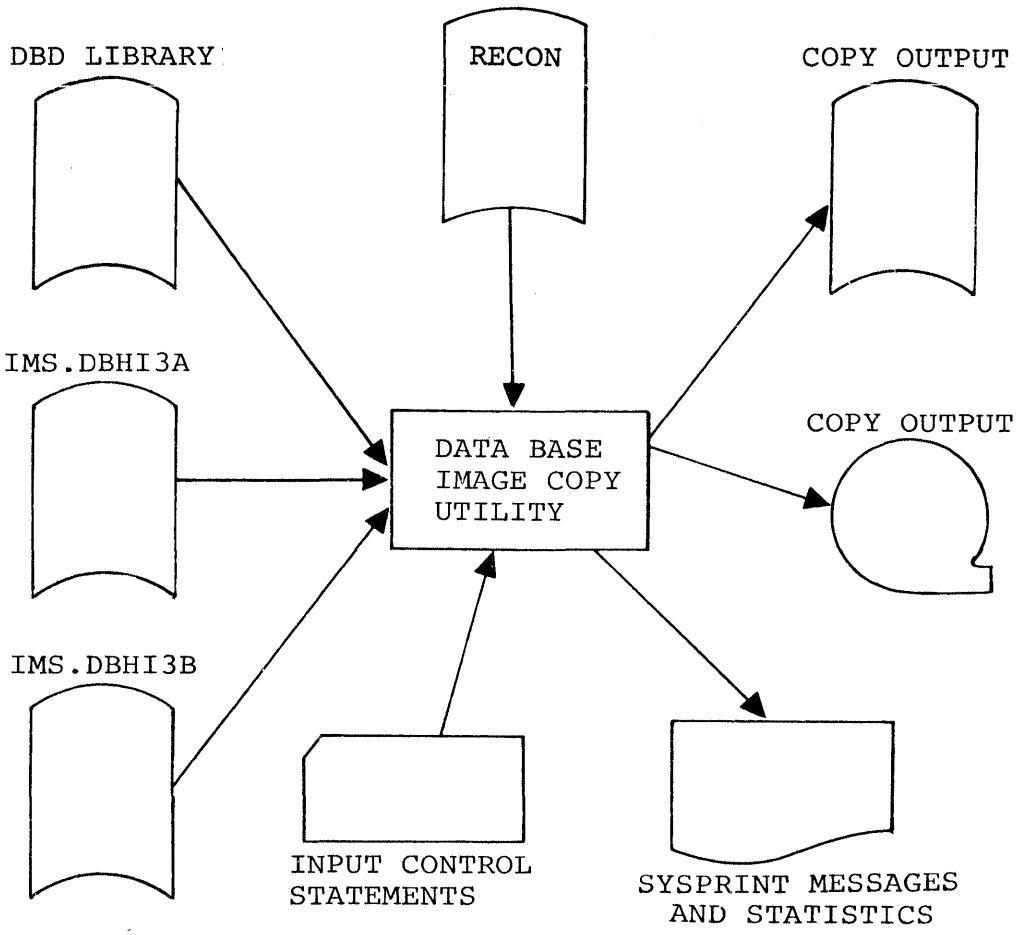


Figure 74. Data Base Image Copy Utility

Performance can be enhanced by providing additional buffers through the appropriate job control statements. Each installation will have to determine the optimum number of buffers to allocate based on their particular requirements.

Because logical record lengths and blocking factors are calculated at execution time, standard labels must be used on all output copies created by the Image Copy utility.

The first record on the output copy is a dump header record which includes information such as data set identification, creation date and time. The creation date and time are required for subsequent use by the Data Base Recovery utility for verification of input.

The frequency of creating image copies must be determined by the user's requirements for timely recovery. The minimum requirement is that a copy be created immediately after a data base is reorganized, reloaded, or initially loaded. Because data base recovery is done on a physical replacement basis, a reloaded data set is not physically the same as it was before unload.

When a data base is stopped and DBDGEN is run to insert and/or delete one or more areas, the Image Copy utility must be run for all areas that follow the inserted and/or deleted area(s) before the data base is started again.

The output from the HISAM Reorganization Unload utility can also be used as input to the Data Base Recovery utility. The Data

Base Reorganization/Load Processing chapter of this manual describes how this is accomplished.

When doing an Image Copy of a shared secondary index, you must only specify the first DBD. This will copy the entire data set.

The functions of this utility can be performed under control of the Utility Control Facility, if required. Refer to "Utility Control Facility" for a description of its operation.

USER CONSIDERATIONS

If Data Base Image Copy is run concurrently with execution of the online system or another batch system, you must make sure the online or batch system does not update the data base being copied. This is normally accomplished for the online system by issuing the /DBDUMP or /DBRECOVERY command for the data base to be copied. As indicated by message DFS175, "DBDUMP COMMAND COMPLETED," these commands must complete successfully before starting data base image copy.

If updates are made to the data base while Data Base Image Copy is running, the image copy cannot be relied upon to successfully recover the data base.

To use this utility with multiple DEDB area data sets, the area name specified in the control statement must be registered in the DBRC RECON.

The CIC parameter can be used to select concurrent processing for copying DEDBs. It cannot be used with the ULU or UDR parameters.

JCL REQUIREMENTS

The Data Base Image Copy utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement may either invoke a cataloged procedure containing the required statements or be in any of the following forms:

```
PGM=DFSUDMP0,PARM='DBRC=x,CIC'

or

PGM=DFSRRC00,PARM='ULU,DFSUDMP0'

or

PGM=DFSRRC00,PARM='UDR,DFSUDMP0,dbdname'
```

In the first form, the Image Copy utility is executed independently of the IMS/VS region controller. This is the normal execution mode. If the first form is used, PARM='DBRC=' can be specified as Y or N to override the specification of DBRC= on the IMSCTRL macro statement made during IMS/VS system definition. DBRC will be used during the execution of this utility if DBRC=Y was specified during IMS/VS system definition, unless overridden by the DBRC=N on this utility's EXEC statement. Specification of DBRC=N means that DBRC will not be used for this execution of the utility and DBRC should not be used to generate the JCL.

If DBRC=NO was specified during IMS/VS system definition, DBRC will not be used during the execution of this utility unless overridden by DBRC=Y on this utility's EXEC

parameter. Specification of DBRC=Y means that DBRC will be used for this execution of this utility.

If DBRC=FORCE was specified on the IMSCTRL macro statement during IMS/VS system definition, it may not be overridden by the DBRC= parameter on the EXEC statement of this utility. DBRC will always be used during the execution of this utility. If you attempt to override DBRC=FORCE, message DFS044I will be issued and a nonzero return code will be returned. The second and third forms are maintained for upward compatibility and, if either of the last two forms is used, the normal IMS/VS positional parameters can follow. (See "Member Name DLIBATCH|DBBATCH" in IMS/VS System Programming Reference Manual for additional parameters that can be specified in executing a batch processing region.) When PARM=UDR is specified, a valid dbdname is required, but is ignored by the Image Copy utility.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBD that describes the data base to be dumped. This is usually DSNAME=IMSVS.DBDLIB. The data set must reside on a direct-access volume.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct-access volume, or printer, or be routed through the output stream.

SYSIN DD
Defines the input control statement data set. The data set can reside on a tape, direct-access volume, or card reader, or be routed through the input stream.

datain DD
Defines the input data set to be dumped. The ddname on this statement must be the same as the name in the DBD that describes this data set; the ddname must also appear on the utility control statement. One DD statement of this type must be present for each data set to be dumped. The data set must reside on a direct-access volume. If this data set is VSAM, performance will improve when the AMP parameter is used to specify additional VSAM buffers. For ISAM/OSAM, DCB=BUFNO=n can be specified, where n is the number of blocks/CI per track.

datain (Fast Path)
For multiple DEDB area data sets, up to seven datain DD statements can be specified. If the area is registered in the RECON data set, the ddname specified in each datain DD statement must not be the area name but must match the names registered in the ADS list of the target area. If the area is not registered, the ddname specified in the datain DD statement must be the area name (ddname operand in the DBD area macro).

dataout1 DD
Defines the first copy of the dumped output data set. One DD statement is required for each data set to be dumped. The ddname may be any 1- to 8-character string, but the ddname must appear in the associated utility control statement. The output device must be either direct access or tape. Standard labels must be used. The BLKSIZE used

is the largest multiple of the logical record length that does not exceed the maximum BLKSIZE. If a BLKSIZE is specified in the JCL, that BLKSIZE is considered the maximum. For devices other than 3380, the default maximum BLKSIZE is the BLKSIZE that was specified as the maximum for that device in the MVS I/O gen. For 3380s, the maximum BLKSIZE is 23K, unless the record length exceeds 23K. If the logical record length exceeds 23K, the maximum BLKSIZE is 32K.

dataout2 DD
Required only if the associated utility control statement requests two copies of the dump. The name must appear in the control statement. The name must be that of either a tape or direct access device. Standard labels must be used. The default for BLKSIZE is the maximum capacity of the output device.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set that the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENT

```
   DBNAME   DATAIN   DATAOUT1 DATAOUT2
1234        13       22       31       40                  80
┌──────────────────────────────────────────────────────────────┐
│ D1IDI32DB01 DFHI3A   DBDMP1   [DBDMP2] [COMMENTS]            │
└──────────────────────────────────────────────────────────────┘
```

Position	Description
1	This must be the character "D." The "D" identifies the statement as a Data Base Image Copy data set utility control statement.
2	This must be a 1 or a 2, depending on the number of copies required.
3	This must be blank or the character "I." If coded "I," an image copy of an index of a KSDS is requested and position 13 must reference the KSDS ddname. The "I" option is not valid for ISAM/OSAM data sets. If position 3 is blank, and if position 13 specifies the ddname for the KSDS, an image copy of the KSDS is obtained. Position 13 is to be used only if the image produced is to be used for Track Recovery. Note that image copy and recovery of an embedded index of a KSDS are not possible and should not be attempted. However, a normal full recovery of the KSDS rebuilds an embedded index and the KSDS data area.
4	This must be the name of the physical DBD that includes the name of the data set to be dumped.
13	This must be the ddname of the input data set or area name to be dumped. It must appear in the

referenced DBD, and a corresponding DD statement must have been provided.

22 This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.

31 This must be the ddname of the second copy of the dumped data set. This field must be blank if position 2 contains a 1. If present, a corresponding DD statement must be provided.

40 Comments may be placed in positions 40 through 80.

RETURN CODES

The Data Base Image Copy utility provides the following return codes:

Code	Meaning
0	Successful completion of all operations.
4	Warning messages were issued.
8	One or more operations not successful.
16	Severe errors have caused the job to terminate without completing all operations.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

EXAMPLES

Example 1

In this example, the data set with the ddname DBHI3A is to be copied from the data base named DI32DB01. The output data set ddname is DBAOUT1. (The numbers above the control statements are for reference only and are not to be included in the input stream.)

```
//DBDUMP   JOB 1,1,MSGLEVEL=1
//STEP1   EXEC PGM=DFSUDMP0
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//DBHI3A   DD  DSN=IMS.DBHI3A,DISP=SHR
//DBAOUT1  DD  DSN=IMS.DBAOUT1,UNIT=TAPE,
//   VOL=SER=DBDMP1,LABEL=(,SL),DISP=(NEW,KEEP)
//SYSIN    DD  *

12 4        13       22       31       40                              80
D1 DI32DB01 DBHI3A   DBAOUT1           DUMP SINGLE DATA SET
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

In this example, two data sets with the ddnames DBHI3A and DBHI3B are to be copied from the data base named DI32DB01. Two copies of the data set DBHI3A are to be created.

```
//DBDUMP   JOB 1,1,MSGLEVEL=1
//STEP1    EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//DBHI3A   DD  DSN=IMS.DBHI3A,DISP=SHR
//DBHI3B   DD  DSN=IMS.DBHI3B,DISP=SHR
//DBAOUT1  DD  DSN=IMS.DBAOUT1,UNIT=TAPE,VOL=SER=DBDMP1,
//  LABEL=(,SL),DISP=(NEW,KEEP)
//DBAOUT2  DD  DSN=IMS.DBAOUT2,UNIT=TAPE,VOL=SER=DBDMP2,
//  LABEL=(,SL),DISP=(NEW,KEEP)
//DBBOUT1  DD  DSN=IMS.DBBOUT1,UNIT=TAPE,VOL=SER=DBDMP3,
//  LABEL=(,SL),DISP=(NEW,KEEP)
//SYSIN    DD  *

12 4        13       22       31       40                              80
D2 DI32DB01 DBHI3A   DBAOUT1  DBAOUT2  DATA SET 1-DUMP 1+2
D1 DI32DB01 DBHI3B   DBBOUT1           DATA SET 2-DUMP 1
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 3

In this example the area with the area name AREANAM1 is to be copied from the data base named DI32DB01. The ddnames and dsnames in the ADS list for the area are DDNAME1, DDNAME2, DDNAME3 and DSNAME1, DSNAME2, DSNAME3.

The output data set ddname is DBAOUT1. (The number above the control statement is for reference only and is not to be included in the input stream.)

```
//DBDUMP    JOB  1.1,MSGLEVEL=1
//STEP1     EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'
//STEPLIB   DD   DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD   DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD   SYSOUT=A
//DDNAME1   DD   DSN=DSNAME1,DISP=SHR
//DDNAME2   DD   DSN=DSNAME2,DISP=SHR
//DDNAME3   DD   DSN=DSNAME3,DISP=SHR
//DBAOUT1   DD   DSN=IMS.DBAOUT1,UNIT=TAPE,
//  VOL=SER=DBDMP1,LABEL=(,SL),DISP=(NEW,KEEP)
//RECON1    DD   DSN=RECON1,DISP=SHR
//RECON2    DD   DSN=RECON2,DISP=SHR
//RECON3    DD   DSN=RECON3,DISP=SHR
//SYSIN     DD   *
12 4        13       22       31       40                          80
D1 DI32DB01 AREANAM1 DBAOUT1          DUMP DUAL DEDB AREA
/*
```

ONLINE DATA BASE IMAGE COPY UTILITY (DFSUICP0)

The Online Data Base Image Copy utility runs as a batch message processing program (BMP) and creates an output copy of a data set within a data base while the data base is allocated to and being used by the online system. Output is provided in the same format as from the batch Data Base Image Copy utility. This output is used as input to the Data Base Recovery utility.

The output from the HISAM Reorganization/Unload utility and batch Data Base Image Copy utility can also be used as input to the Data Base Recovery utility. See the descriptions of these other utilities in the "Data Base Reorganization/Load Processing" chapter and in this chapter for more information on the recovery process.

The user has the option of creating one or two output image copies. The advantage in specifying two copies is that, if an I/O error occurs during copy execution, the utility continues to completion on the other copy. Performance would be somewhat diminished in this instance, but a total rerun would not be necessary.

Because default block sizes are calculated at execution time if the user does not specify them in the JCL, standard labels must be used on all output copies created by the Online Image Copy utility.

The utility prints the time stamp of the system log when the utility starts and the time stamp again when the utility completes.

The first record on the output copy is a dump header record which includes information such as data identification, creation date and time. The creation date and time are required for subsequent use by the Data Base Recovery utility for verification of input.

The frequency of creating image copies must be determined by the user's requirements for timely recovery and available online resources.

Note: The Online Data Base Image Copy utility does not support CICS/VS.

USER CONSIDERATIONS

The user should note the following limitations when running the Online Image Copy utility:

- GSAM and HSAM data bases cannot be copied.
- Fast Path MSDBs and DEDBs cannot be copied.
- The index portion of a VSAM KSDS cannot be copied without copying the entire KSDS.
- This utility requires a PSB that names the data base to be copied and contains the OLIC=YES operand. This PSB can contain one or more PCBs and must be defined by an APPLCTN macro in the online system definition. The PSBGEN "LANG=" keyword must not specify PL/I. Refer to "DL/I Data Base PCB Statement" in the "PSB Generation" chapter for details on constructing a PSB for use with this utility.

In addition, the following should be considered: If the online image copy is created concurrently with application program updating of the same data base, the changes need to be applied through the Data Base Recovery utility. The time stamp of the first log data set required for recovery are printed on SYSOUT.

CHECKPOINT/RESTART

Two optional DD statements provide the ability to restart an image copy job after a system or utility failure. If the statements are not included in the JCL for the Online Image Copy utility, no checkpoint/restart functions are available. The DFSUCKPT DD statement defines the data set to which the utility writes checkpoint information, consisting of volume serial number and relative record numbers, during execution. The DFSURSRT DD statement defines a checkpoint data set to be used to restart the job. DFSUCKPT and DFSURSRT can define the same data set.

By default, if the DFSUCKPT DD statement is included, the Online Image Copy utility writes a checkpoint for every 5000 records copied. The user can override this checkpoint interval by specifying a different interval on the control statement. See "Utility Control Statements."

If the restart function is to be used, the DD statements defining the image data sets must be coded with DISP=KEEP or CATLG and nonspecific serial numbers. The disposition of KEEP or CATLG will ensure that the volume rewinds properly in event of a restart. Not coding specific volume serial numbers allows the operator to mount multiple volumes in any order during a restart. See "Job Control Statements."

An error received on the checkpoint data set during utility execution causes a message to be printed, but the utility will continue. No further checkpoints are taken.

An error received on the restart data set during restart causes a message to be printed and the utility to abnormally terminate.

JCL REQUIREMENTS

The Online Data Base Image Copy utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define input and output data sets are required. Note that the DD statements for the data sets to be copied are in the control region job control language data set and not in the copy job itself.

EXEC
This statement may either invoke a cataloged procedure containing the required statements or be in the form:

```
PGM=DFSRRC00,PARM='BMP,DFSUICP0,psbname,,destname'
```

where the parameters BMP and DFSUICP0 describe the utility region; psbname is the name of a PSB that has been described by an APPLCTN macro in the online system definition, specifies the data base to be copied, and contains the OLIC=YES parameter; and destname is the output destination for critical error messages (the default destination is the OS/VS console).

The normal IMS/VS positional parameters can follow. (See "Member name IMSBATCH" in IMS/VS System Programming Reference Manual for additional parameters that can be specified in executing a batch message processing region.)

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBD that describes the data base to be dumped. This is usually DSNAME=IMSVS.DBDLIB. The data set must reside on a direct access volume.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct access volume, or printer, or be routed through the output stream.

SYSIN DD
Defines the input control statement data set. The data set can reside on a tape, direct access volume, or card reader, or be routed through the input stream.

dataout1 DD
Defines the first copy of the dumped output data set. One DD statement is required for each data set to be dumped. The ddname may be any 1- to 8-character string, but the ddname must appear in the associated utility control statement. The output device must be either direct access or tape. Standard labels must be used. A user block size can be specified in the BLKSIZE subparameter of the DCB operand; the utility will round the block size down to an even multiple of the data base data set logical record size plus 8, rounded to the next doubleword. The block size specified must be larger than the logical record length and smaller than or equal to the device maximum. The default for BLKSIZE is the maximum block size of the output device. If the restart function will be used, the disposition of the image data sets should be KEEP or CATLG.

dataout2 DD
Required only if the associated utility control statement requests two copies of the dump. Same requirements as dataout1. The block size specified for dataout2 can be different from that specified for dataout1.

DFSUCKPT DD
Defines the optional checkpoint data set, to which the utility will write checkpoint information. A single track on a direct access device is required. Data set characteristics are specified by the utility.

DFSURSRT DD
Defines the optional restart data set, indicating that a previous checkpoint is to be used for restarting the job.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENT

```
   DBNAME   DATAIN   DATAOUT1 DATAOUT2
1234        13       22       31       40                    80
┌──────────────────────────────────────────────────────────────┐
│ D1 DI32DB01 DFHI3A   DBDMP1   [DBDMP2] [CHECKPOINT INTERVAL] │
└──────────────────────────────────────────────────────────────┘
```

Position	Description
1	This must be the character "D." The "D" identifies the statement as an Online Data Base Image Copy utility control statement.
2	This must be a 1 or 2, depending on the number of copies required.
3	This must be blank. Note that image copy and recovery of an embedded index of a KSDS are not possible and should not be attempted. However, a normal full recovery of the KSDS rebuilds an embedded index as well as the KSDS data area.
4	This must be the name of the physical DBD that includes the name of the data set to be dumped.
13	This must be the ddname of the input data set to be dumped. It must appear in the referenced DBD, and a corresponding DD statement must have been provided.
22	This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.
31	This must be the ddname of the second copy of the dumped data set. This field must be blank if position 2 contains a 1. If present, a corresponding DD statement must be provided.
40	This can be a 4-digit number specifying a checkpoint interval. This field is checked only if a DFSUCKPT DD statement is included in the JCL used to execute this utility. If this field is blank, nonnumeric, or zero, the default of 5000 is used (and, in the last two cases, a warning message indicating invalid field format is issued).

RETURN CODES

The Online Data Base Image Copy utility provides the following return codes:

Code	Meaning
0	Successful completion of all operations.
4	Warning messages were issued.
8	One or more operations not successful.
12	Error during restart.
16	Severe errors have caused the job to terminate without completing all operations.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

EXAMPLES

Example 1

Example 1 shows the JCL and utility control statements used to create two copies of the ISAM index of a HIDAM data base, with no checkpoints.

```
//DLICEX1  JOB 1,1,MSGLEVEL=(1,1)
//OLIC     EXEC PGM=DFSRRC00,PARM='BMP,DFSUICP0,OLINSK41'
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSUDUMP DD  SYSOUT=A
//DMP1     DD  DSN=OLIC2.IMAGE1.ISAM,DISP=(,KEEP),
//  UNIT=TAPE,LABEL=(1,SL)
//DMP2     DD  DSN=OLIC2.IMAGE2.ISAM,DISP=(,KEEP),
//  UNIT=TAPE,LABEL=(1,SL)
//SYSIN    DD  *
D2 DX41SK01 DXSK0101 DMP1    DMP2
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

Example 2 shows the JCL and utility control statements used to copy four OSAM data set groups associated with a HIDAM data base, with checkpoints taken.

```
//DLICEX2  JOB  1,1,MSGLEVEL=(1,1)
//OLIC     EXEC PGM=DFSRRC00,PARM='BMP,DFSUICP0,HHTASK41'
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSUDUMP DD  SYSOUT=A
//DFSUCKPT DD  DSN=OLIC2.CKPT2,DISP=(,KEEP),VOL=SER=IMSQAW,
//  UNIT=SYSDA,SPACE=(TRK,(1,1))
//DMP1     DD  DSN=OLIC2.IMAG1.OSAM,DISP=(,KEEP),
//  UNIT=TAPE,LABEL=(1,SL)
//DMP2     DD  DSN=OLIC2.IMAG2.OSAM,DISP=(,KEEP),
//  UNIT=TAPE,LABEL=(1,SL)
//DMP3     DD  DSN=OLIC2.IMAG3.OSAM,DISP=(,KEEP),
//  UNIT=TAPE,LABEL=(1,SL)
//DMP4     DD  DSN=OLIC2.IMAG4.OSAM,DISP=(,KEEP),
//  UNIT=TAPE,LABEL=(1,SL)
//SYSIN    DD  *
D1 DH41SK01 DHSK0101 DMP1            1000
D1 DH41SK01 DHSK0102 DMP2            1000
D1 DH41SK01 DHSK0103 DMP3            2000
D1 DH41SK01 DHSK0104 DMP4            1000
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

DATA BASE CHANGE ACCUMULATION UTILITY (DFSUCUM0)

The purpose of the Data Base Change Accumulation utility is to provide the Data Base Recovery utility with a sequential data set that contains only that information from the log data sets which is necessary for recovery. This is done by eliminating

all nondata base records; by specifying a purge date (or dates) to eliminate all data base records before that date; by sorting the acceptable data base records, and by combining all data base records that update the same segment. The resulting records are sequenced by data set within the data base. This utility invokes the Sort/Merge program that is used in the installation.

This utility may be executed several times over a period of time to incorporate additional data base changes and to delete changes which are no longer useful. The input to this utility may be one or more IMS/VS system log data set(s) (SLDSs) or recovery log data set(s) (RLDSs), and a previous data base accumulation change tape, if one exists.

The Change Accumulation utility can be run independently of IMS/VS and is an excellent means of combining and reducing all of the log data sets that may be required for data base recovery. The new log output data set created by the Change Accumulation utility is used by the Data Base Recovery utility.

The input to the Data Base Change Accumulation utility consists of:

1. All SLDS created since either the last image copy utility execution or the last run of this utility. This can include the new log output data sets resulting from previous executions of this utility.

2. The previous data base Change Accumulation utility data set. This would be the output from the last execution of this utility.

3. DBD library that is normally called IMSVS.DBDLIB.

4. Control statements (ID, DB0 and DB1) that specify any purge dates and how the data base log records are to be processed. (See Figure 75 on page 282 and "Utility Control Statements" later in this chapter.)

Output from the Data Base Change Accumulation utility consists of:

1. A new Change Accumulation utility data set which is a sequential data set containing the combined data base records for the data base/data sets identified on a DB0 control statement.

2. A new log output data set that contains data base records identified by the data base/data sets on a DB1 control statement.

The new log output data set can be used to select records for critical data bases and then as input to individual change accumulation runs to obtain an accumulated change data set with changes for only a particular data base. This would minimize the time otherwise required to recover a data base by eliminating the need to pass unwanted records from other data bases.

It should be noted that the new log output data set cannot be used as input to the Data Base Backout utility.

Restriction: When using DBRC, the creation of an optional log output data set using this utility is not recorded in the RECON data set, as is the creation of other output data sets.

The functions of this utility can be performed by the Utility Control Facility, if required. Refer to "Utility Control Facility" for a discussion of its operation.

Figure 75 on page 282 depicts the sources of input to the Data Base Change Accumulation utility and the output created by this utility.

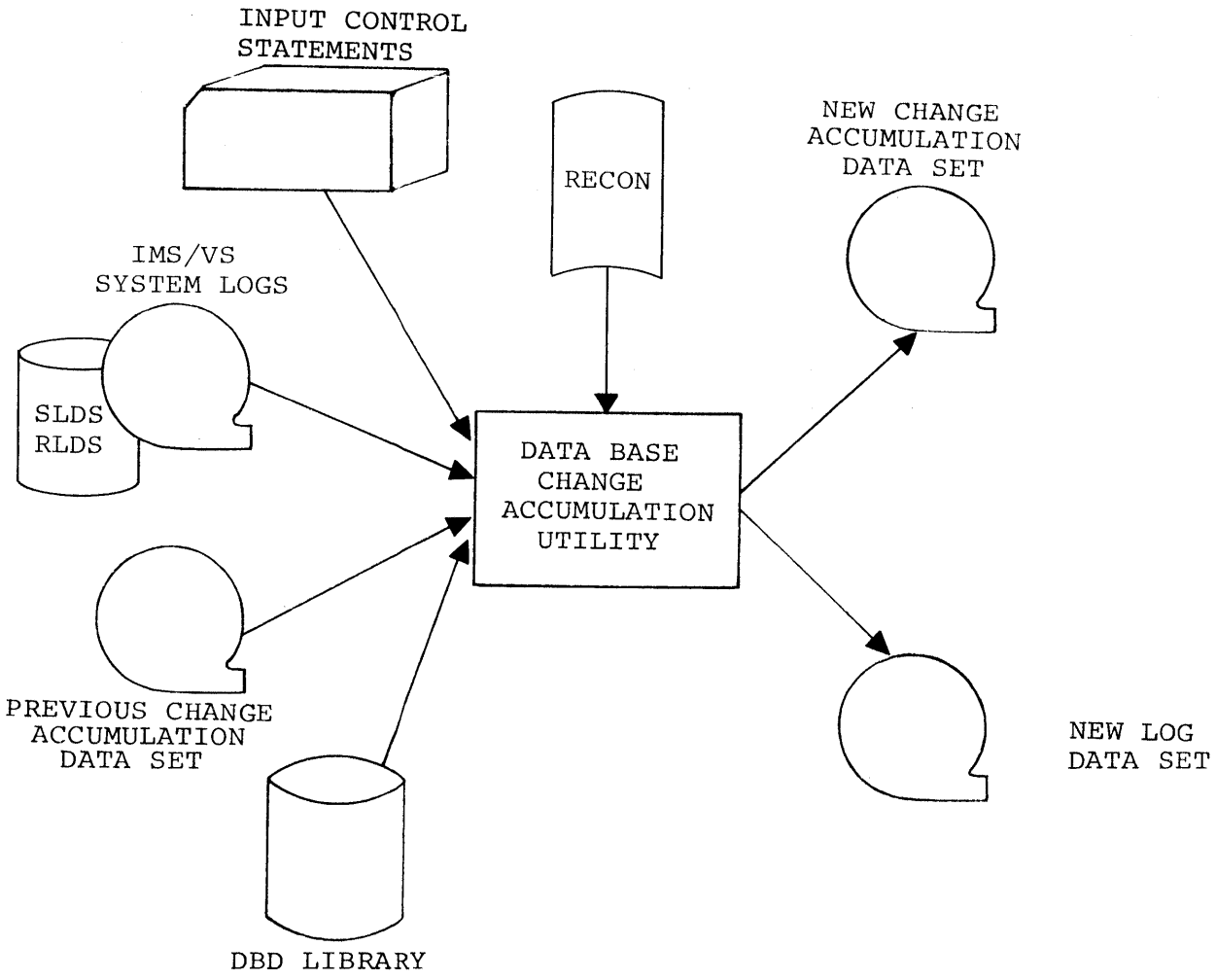


Figure 75. Data Base Change Accumulation Utility

USE OF PURGE DATE AND TIME

If change accumulation records or an input log span an Image Copy time or a reorganization time, the input log contains change records that were made before and after the image copy or reorganization time. A purge date and time corresponding to the image copy time or reorganization time must be specified. This ensures elimination of data base change records not valid in a recovery.

The user may specify one purge date and time for all data base log records being processed. The purge date and time can be specified for either all data base log records that update a particular data base or for all data base log records that update a data set within a data base. The user may also specify multiple purge dates and times, and these can be different, for different data sets within a data base.

It should be noted that, if a purge date is specified without the associated time, the time defaults to zeros.

For input log tapes, the change accumulation utility compares the purge date specified to the date and time in each data base update record. If the purge date is later than the date and time in the update record, then the input log record is dropped.

For previous change accumulation tapes, the utility compares the purge date specified to the date and time in the DFSUCUM0 records. If the purge date is later than the date and time in the DFSUCUM0 record, then the input log record is dropped.

Note: The date and time in these records represents the latest update contained in the data base block. For this reason, it is possible that there is an update in that record which is earlier than the purge date, and that update will not be dropped, because the latest update in that same record occurred after the purge date specified. This note is particularly important if a change accumulation spans a reorganization. You must always specify a purge date the first time a data base is accumulated following a reorganization, so that old records are correctly discarded, and DFSUCUM0 blocks are correctly discarded.

In the following example, processing steps are listed to show the use of a purge date and time. There is no old accumulated change data set to be updated. (All the processing steps represent one sequence for one example. The intervening comments are for clarification only.)

```
Load Data Base 1 (DB1)
Process - Log 1
Process - Log 2
Accumulate (Log 1, Log 2) into Accumulation Log-A
Process - Log 3
Process - Log 4
Reorganize IDB1
Process - Log 5
Process - Log 6
Accumulate (Log 5, Log 6) into Accumulation Log-B
```

- For this accumulation run, are purge date and time needed? No, not unless the Accumulation Log-A is input.
- Accumulation Log-B contains only the change log records since DB1 was reorganized.

```
Load Data Base 2 (DB2)
Process DB1 and DB2 - Log 7
Process DB1 and DB2 - Log 8
Reorganize IDB2
Accumulate (Log 7, Log 8) into Accumulation Log-C
```

- Are purge date and time needed now? Yes, to eliminate previous log records for DB2 because DB2 has been reorganized.

A purge date and time can be specified on any DB0 or DB1 utility control statement (described later in this chapter). Log records, which are identified by the control statement and which were created prior to the purge date, are eliminated. In the case of updating an old data base change accumulation data set through execution of this utility, old accumulation change records matching a DB0 identifier and having a date that is before the purge date are eliminated and are not written to the new accumulation change tape.

JCL REQUIREMENTS

The Data Base Change Accumulation utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement may either invoke a cataloged procedure containing the required statements, or be in the form:

```
PGM=DFSUCUM0,PARM='CORE=ssssss,DBRC=x',REGION=xxx
```

where ssssss is the amount of storage in bytes that Sort/Merge can use for this application. The program defaults to 100000 bytes if no value is specified. The region size specified should be the ssssss value plus 100K bytes, or 200K bytes if no ssssss value is specified. The region size required is dependent on many variables, including input and output buffer requirements, number of data base/data sets specified on control statements, and the size of DBDs.

PARM='DBRC' can be specified as Y or N to override the specification of DBRC= on the IMSCTRL macro statement made during IMS/VS system definition.

The following table summarizies the outcome of specifying DBRC=:

System Definition	Utility EXEC Statement	
	DBRC=N	DBRC=Y
DBRC=NO	not used	used
DBRC=YES	not used	used
DBRC=FORCE	invalid	used

If DBRC=FORCE was specified on the IMSCTRL macro statement during IMS/VS system definition, it may not be overridden by the DBRC= parameter on the EXEC statement of this utility. DBRC will always be used during the execution of this utility. If you attempt to override DBRC=FORCE, message DFS044I will be issued and a nonzero return code will be returned.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct access volume, or printer, or be routed through the SYSOUT stream.

IMS DD
Defines the library containing the DBDs that describe all data bases to be accumulated. This is usually DSNAME=IMSVS.DBDLIB. The data set must reside on a direct access volume.

SYSOUT DD
Defines the output message data set for the Sort/Merge program. The data set can reside on a tape, direct access volume, or printer, or be routed through the SYSOUT stream. The Sort/Merge program specifies AP (all messages to printer).

SORTLIB DD
Defines a data set containing load modules for the execution of Sort/Merge. This is usually DSNAME=SYS1.SORTLIB. The data set must reside on a direct access volume.

SORTWKnn DD
These DD statements define the intermediate storage data sets for the Sort/Merge program. The data sets normally reside on a direct access volume; however, tape can be used. (For specification of number and size of intermediate storage data sets, refer to the applicable sort/merge manual.)

DFSUCUMN DD
Defines the new accumulated change output data set. The data set can reside on a tape or a direct access volume. The output is blocked to maximum device capacity but will not exceed 16383.

DFSUCUMO DD
Defines the old accumulated change input data set that is to be merged with the log input data in order to create the new accumulated change data set. If no old accumulated changes are to be merged, the following DD statement must be used:

```
//DFSUCUMO DD DUMMY,DCB=BLKSIZE=100
```

This data set can reside on tape or a direct access volume.

DFSUDD1 DD
Defines the new log output data set. The output data set can reside on a tape or a direct access volume. The output is blocked to maximum device capacity. Standard labels must be used.

DFSULOG DD
Defines the log input data set containing the change records to be accumulated. This data set can reside on tape or a direct-access volume.

If the data set is a CICS journal, DCB=(RECFM=VB) must be specified.

Multiple log data sets can be used as input by concatenating the data sets. When log input comes from both an IMS/VS Release 1.2 or later system and an earlier release of the system, the log input must be in the sequence in which it was created.

SYSIN DD
Defines the control statement data set. The data set can reside on a tape, direct access volume, or card reader, or be routed through the input stream.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

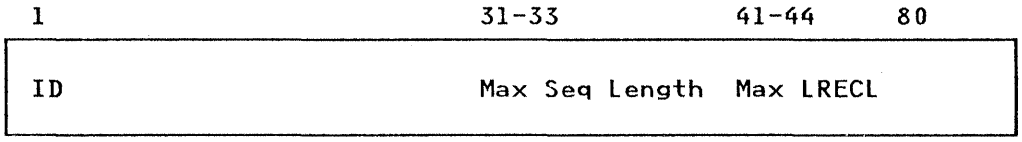
If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS

Three types of utility control statements are used by the Data Base Change Accumulation utility: ID, DB0, and DB1 statements. All or any combination of these statements can be used, subject to the limitations described in the following section.

- ID statement

This optional statement is used to describe both the table requirements and the sort requirements needed for this change accumulation execution. If it is included, it must be the first statement supplied. If it is not included, default values are assigned as described below.

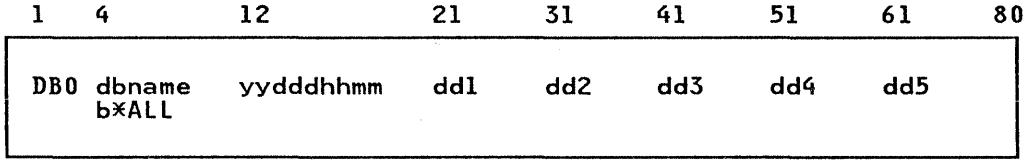


Position	Description
1	Positions 1 and 2 must contain the characters ID.
31	Positions 31 through 33 contain the maximum length root sequence field contained within the log records to be processed as a result of a DBO control statement. This value is used to pad the sequence field with binary zeros for sorting purposes. If there are no ISAM or VSAM KSDS records to be processed, this value should be specified as 4, the length of the relative block number field. This value must be in the range 1 through 236 and must be left-justified or supplied with leading zeros. The default value for this entry is 10.
41	Positions 41 through 44 contain the maximum length of a data base change type log record plus the maximum sequence length specified in positions 31 through 33. The default value is 0767. Data base change type log records created by IMS/VS are limited to a maximum of 512 bytes and the maximum sequence length is 255 bytes. This parameter should be specified when the user expects Fast Path data base change type records to be written to the log. For Fast Path, this parameter must be equal to the maximum control interval size for any area plus 255. The value must be left-justified or supplied with leading zeros. Note that the MAX LRECL size may change with different releases of IMS/VS.

- DBO statement

  This optional statement is used to describe which records are to be accumulated for output to the new change accumulation data set. One or more of these statements may be included. The DBO statements are generally used to indicate what is to be accumulated from the input log data set(s) into the output accumulation data set. Input from the old accumulation data set for any data base(s) or data set(s) that are specified in a DBO statement will be included in the output accumulation data set, unless it should be purged due to a purge date in the DBO statement. All other input from the old accumulation data set for any other data base(s) or data set(s) is carried forward to be included in the output accumulation data set. The options that are available are described in the following:

  Note: Each combination of data base name/ddname may appear on one control statement only.



Position	Description
1	Positions 1 through 3 must contain the characters DBO.
4	Positions 4 through 11 can contain a data base name or a b*ALL where position 4 is blank and positions 5 through 8 contain the characters *ALL. If *ALL is specified, all records will be accumulated, the purge date and time in positions 12 through 20 will be applied to all records, and no DB1 statements can be specified. If DBRC is being used, the *ALL option of DBO control statements is not valid. If a data base name and ddnames are supplied, the purge date is applied only to those records whose data base name/ddname combinations match. If no ddnames are supplied, the purge date applies to all records that match the data base name.
12	Positions 12 through 20 can contain a purge date and time in the form yydddhhmm, where yy=year, ddd=day of year, hh=hour, mm=minute. If a purge date and time are specified, all records that match a data base name/ddname description and dated before the purge date will be eliminated. If an old accumulated change input is supplied, all records that match the data base name/ddname description and dated before the purge date will not be merged into the new change accumulation date set. If this field is blank, no purge date is used. If any fields are coded, all the fields for time and date must be coded.
21	Positions 21 through 28, 31 through 38, 41 through 48, 51 through 58, and 61 through 68 can contain 1 to 5 ddnames which, combined with the data base name supplied, make up the record identification. All records matching this identification will be sorted and accumulated and have the purge date and time applied to them. As many combinations of data base name, purge date, and ddname specifications as are required can be specified by submitting additional DBO control statements. If no ddnames are supplied, the purge date and time are applied to all records that match the data base name. All ddnames must be left-justified; unused positions must be blank. If no ddnames are specified but a dbname is specified in positions 4 through 11, all changes to the data base are written to the accumulation change data set.

- DB1 statement

  This statement is used to describe which records are to be written out to the new log output data set. These records are not sorted; they are written in the same order they are read. Any log records that are not data base change records are not written to the output data set. Any number of DB1 statements describing data base name/ddname combinations may be included.

  Note: Each combination of data base name/ddname may appear on one control statement only.

1	4	12	21	31	41	51	61	80
DBl	dbname b*ALL b*OTHER	yydddhhmm	ddl	dd2	dd3	dd4	dd5	

Position	Description
1	Positions 1 through 3 must contain the characters DBl.
4	Positions 4 through 11 can contain a data base name, b*ALL or b*OTHER. If *ALL is specified, all records are written to the new log data set, and no DBO control statements may be included. If *OTHER is specified, all records not described by a DBO control statement are written to the new log data set. If a purge date and time are specified in position 12, all records identified by this control statement and dated before the purge date are not written to the new log data set. If *ALL was specified on a DBO statement, it cannot also be specified on a DBl statement. Only one DBl statement specifying *OTHER can be supplied.
12	Positions 12 through 20 can optionally contain a purge date and time in the form yydddhhmm, where: yy=year, ddd=day of year, hh=hour, mm=minutes. All records matching a record identification combination and dated before the purge date will be eliminated.
21	Positions 21 through 28, 31 through 38, 41 through 48, 51 through 58, and 61 through 68 may contain 1 to 5 ddnames. These names, combined with the data base name, make up the record identification. All records matching this identification and dated after the purge date will be written to the new log data set. All ddnames supplied must be left-justified; unused positions must be blank. If no ddnames are specified but a dbname is specified in positions 4 through 11, all changes to the data base are written to the new data base log data set.

The Data Base Change Accumulation utility makes the following assumptions in the absence of utility control statement information:

- If no utility control statements are present, all data base log records are to be sorted and combined to produce a new change accumulation data set. No purge date and a maximum prime key length of 10 bytes are assumed.
- If no ID control statement is present, a maximum prime key length of 10 is assumed.
- If no DBO or DBl control statements are present, all data base log records are to be sorted and combined to produce a new change accumulation tape. No purge date is assumed. If a DBO statement is used, the user may not also use a DBl *ALL statement. If a DBO *ALL statement is used, the user may not also use a DBl statement.

RETURN CODES

The Data Base Change Accumulation utility provides the following return codes:

Code	Meaning
0	Successful completion.
4	Warning messages were issued.
8	Unsuccessful completion.
16	Returned by SORT/MERGE program.

There may be other return codes returned by Sort/Merge if the ERRET=ABEND option was specified when Sort/Merge was installed.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

Note: Message 1EC161I is a normal message during an IMS/VS data base recovery of a VSAM data set.

EXAMPLES

Example 1

In this example, two data base logs are to be accumulated. There is no old accumulated change data set to be updated. The first data base (DI32DB01) is to be accumulated in its entirety, and all records before day 175 of year 73 and before 1200 hours are to be eliminated. The second data base (DI32DB02) is to be accumulated selectively.

The data base (DI32DB02) data set with the ddname DDI3IA is to be accumulated, and all records before day 173 of year 73 and before 1500 hours are to be eliminated.

The data base (DI32DB02) data set with the ddname DDI3OA is to have its records written out to the new log data set, and all records before day 173 of year 73 and before 1500 hours are to be eliminated.

All other records are to be written out to the new log data set.

```
//ACCUM1  JOB  1,1,MSGLEVEL=1
//STEP1  EXEC  PGM=DFSUCUM0,PARM='CORE=50000',REGION=150K
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS  DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//SYSOUT  DD  SYSOUT=A
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUM0  DD  DUMMY,DCB=BLKSIZE=100
//DFSUCUMN  DD  DSN=IMSVS.CUM1,UNIT=TAPE,VOL=SER=CUMTAP,DISP=(,KEEP)
//DFSUDD1  DD  DSN=IMSVS.NEWLOG,UNIT=TAPE,VOL=SER=LOGTAP,DISP=(,KEEP)
//DFSULOG  DD  DSN=IMSVS.LOG1,UNIT=TAPE,VOL=SER=LTAPE1,DISP=OLD
//         DD  DSN=IMSVS.LOG2,UNIT=TAPE,VOL=SER=LTAPE2,DISP=OLD
//SYSIN  DD  *
DB0DI32DB01731751200
DB0DI32DB02731731500DDI3IA
DB1DI32DB02731731500DDI30A
DB1 *OTHER
/*
```

Note: The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

In this example, all data base change records are to be accumulated. The maximum root segment sequence field length is specified as 4 bytes, because all log records reflect HD-type organizations. There are no ISAM-type or VSAM KSDS-type change records. The DB0 control statement specifies that all records are to be accumulated, and that those records before day 200 of year 73 are to be eliminated. An old change accumulation data set is to be merged with the new change accumulation data set. The purge date will be applied to the old accumulation data set. DFSUDD1 (new log output data set) is defined as DUMMY because a DB1 control statement is not specified.

```
//ACCUM2  JOB  1,1,MSGLEVEL=1
//STEP1  EXEC  PGM=DFSUCUM0,PARM='CORE=50000',REGION=150K
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS   DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//SYSOUT  DD  SYSOUT=A
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD  UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUM0  DD  DSN=IMSVS.CUM1,UNIT=TAPE,VOL=SER=CUMTAP,DISP=OLD
//DFSUCUMN  DD  DSN=IMSVS.CUM2,UNIT=TAPE,VOL=SER=CUMTP2,DISP=(,KEEP)
//DFSUDD1  DD  DUMMY
//DFSULOG  DD  DSN=IMSVS.LOG1,UNIT=TAPE,VOL=SER=LTAPE3,DISP=OLD
//         DD  DSN=IMSVS.LOG2,UNIT=TAPE,VOL=SER=LTAPE4,DISP=OLD
//SYSIN  DD  *
ID                              004
DB0 *ALL   732000000
/*
```

DATA BASE RECOVERY UTILITY (DFSURDB0)

The Data Base Recovery utility program is designed to restore a physically damaged data set within an IMS/VS data base. This utility does not provide a means of recovery from application logic errors; it is the user's responsibility to ensure the integrity of the data in the data base.

This utility performs recovery by updating a copy of the data base with the changes that have been logged after the copy was made.

The Data Base Recovery utility program is executed in a special IMS/VS batch region. This allows data base recovery to be run independently of the IMS/VS system. The Data Base Recovery utility can be used with multiple DEDB area data sets, but the utility will recover one area data set at a time.

This utility will recover only one copy of a DEDB area during each execution. If multiple-copy DEDB area data set support is required, the DEDB Area Data Set Create utility must be used to create the additional copies.

If an area is registered in the DBRC RECON data set, the following restrictions must be satisfied:

- The ddname and dsname in the dataset1 DD statement must match the names registered in the ADS list of the target area.
- If full recovery is requested, the target area must be in the recovery-needed status in the DBRC RECON data set. If a track recovery is requested, the target area must be in the recovery-not-needed status and the area data set must be in an available status in the DBRC RECON data set.

If the target area is not registered in the DBRC RECON data set and the DBRC RECON data set has a NOFORCE attribute, the ddname in the dataset1 DD statement must match the target area name.

If the target area is not registered in the DBRC RECON data set and the DBRC RECON data set has a FORCER attribute, this utility terminates with an error message.

When recovering a shared secondary index, you must specify only the first DBD. The utility will recover the entire data set.

For full recovery of an ISAM/OSAM data set, the data set must be scratched and reallocated before executing the Data Base Recovery utility. Because the recovery utility does not use OSAM to build the output data set, preallocating the primary data space across multiple volumes is not supported. For information on scratching and allocating ISAM/OSAM data sets, see "Allocation for OSAM Data Sets" under "Designing the IMS/VS Online System" in IMS/VS System Administration Guide.

The functions of this utility can be performed by the Utility Control Facility. Refer to "Utility Control Facility" for a description of its operation.

The copy of the data base to be supplied to the Data Base Recovery utility can be:

- An image copy created by the DB Image Copy utility (DFSUDMP0) or the online image copy utility (DFSUICP0). (Note that neither of these utilities may be used for HSAM or GSAM data bases. Because of this, the Data Base Recovery utility cannot be used with HSAM or GSAM data bases.)
- A HISAM unload data set created by the HISAM Reorganization Unload utility (DFSURUL0).
- If the user has already restored the data base with a copy created by an OS/VS or utility, the image copy input data set is not required.

The changes to the data base to be supplied as input must be:

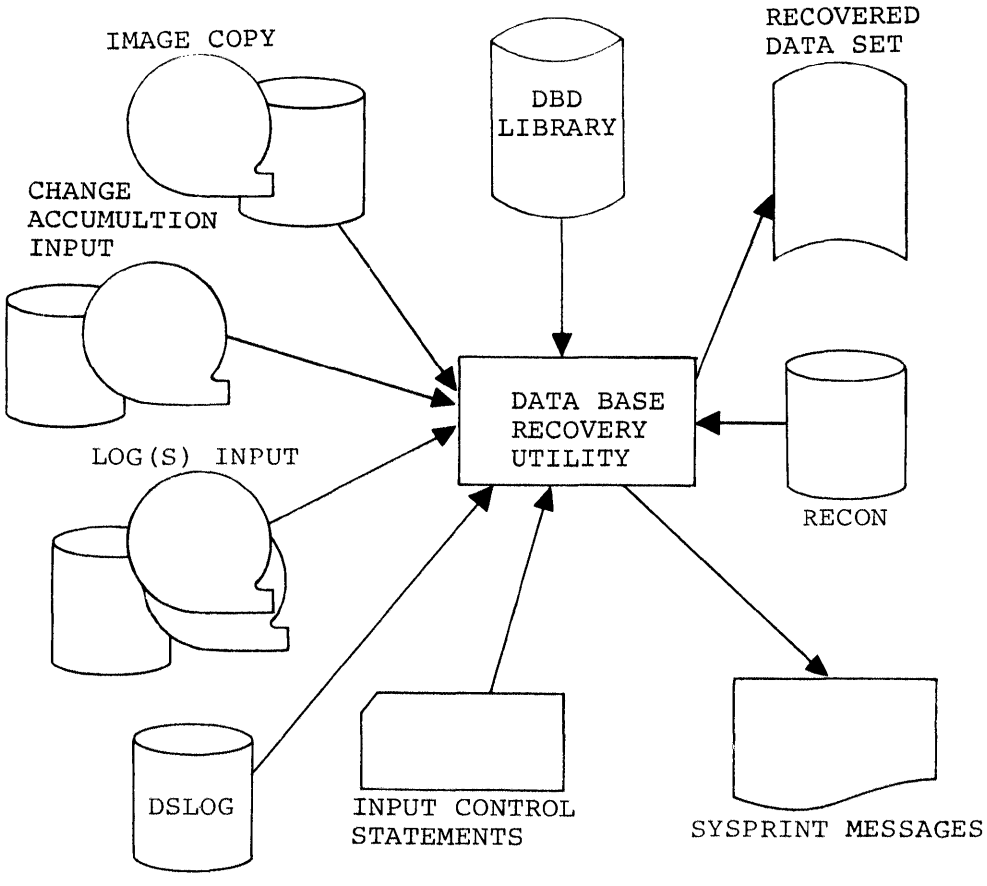
- The SLDS created by IMS/VS during normal execution;
- An accumulation of the changes on the log created by the DB Change Accumulation utility (DFSUCUM0); or
- A combination of both the SLDS and the log created by DB Change Accumulation.

If the Online Data Base Image Copy utility is used to back up a data base data set, changes made concurrent with and subsequent to the image copy may be required. The Online Image Copy utility provides on SYSOUT the volume serial number of the first log tape that may contain applicable changes.

Optional track recovery for VSAM data sets can be requested to recover a portion of a data base data set for which errors have been logged. In this case, an accumulated change data set created by the DB Change Accumulation utility and an image copy data set created by the DB Image Copy utility, the Online Data Base Image Copy utility, or the HISAM Reorganization Unload utility are used to update an otherwise current DASD copy of the data base data set.

If the target data set fails to open because of a failure during the current or a previous run, the data set should be scratched and reallocated before rerunning this utility.

Figure 76 on page 293 is a flow diagram of the Data Base Recovery utility.



Notes:

1. The image copy can be a dump created by the batch Data Base Image Copy utility, or the Online Data Base Image Copy utility, or, in the case of HISAM, a reorganization dump created by the HISAM Reorganization Unload utility.
2. Multiple logs can be provided by concatenating the logs in date and time sequence. The DB Recovery utility ignores log records created before the dump input.
3. Only one data set recovery is allowed for each execution.
4. SYSPRINT can be blocked but must be a multiple of 121.
5. It is normal to receive VSAM information message IEC161I 072-053 when recovering a VSAM data set.
6. HD reload should not be used as input to recovery. There is no guarantee that physical location of segments after a second reload will match log records created from updates to a data base after the first reload.
7. Output from the Log Tape Merge utility should not be used as input to recovery.

Figure 76. Data Base Recovery Utility

If the HISAM Unload data set is to be used as the DFSUDUMP data set input to the Data Base Recovery utility, the data base must be reloaded immediately following the unload. This procedure ensures that the Recovery utility will ignore any records on the IMS/VS system log that were created before the unload/reload operation. This is necessary because the HISAM unload tape, when processed by the Recovery utility, reorganizes the data base. Reloading after unload creates an equivalent copy of the

data base. A new Change Accumulation tape should be started after unload, or it should be purged of log entries prior to the unload tape.

For full recovery of a VSAM data set, the VSAM data set must be deleted and redefined before executing the Data Base Recovery utility.

TRACK RECOVERY OPTION

Data sets or area data sets that are stored using VSAM can be recovered with the track recovery option (TRCV). This option is designed to recover defective tracks within a data set that have a data check or missing address marker condition indicated. The track recovery option is not supported for the index portion of a VSAM KSDS on the 3850 Mass Storage System. Track recovery on the IBM 3850 cannot recover from stage errors or I/O errors that have caused the track format to be damaged.

If track recovery is attempted after a staging error, and an I/O error is returned, the entire data base must be recovered.

One execution of track recovery recovers the defective tracks for the data set that are in the data space defined by the DFSTRCV DD statement. If the defective tracks exist in more than one VSAM data space, Track recovery must be executed for each VSAM data space associated with the data set that contains the tracks whose contents are to be recovered. Whereas full recovery creates a new data set, the track recovery option only modifies one or more tracks in the existing data set. Recovery with this option is usually faster than full recovery.

Processing with the track recovery option consists of (1) locating all tracks in error in the data set by means of error log records read from the accumulated change input file; (2) assigning alternate tracks where permanent DASD errors exist; (3) taking image copy input data and application data from the accumulated change input for the records in error, and replacing them in the data set. The Data Base Change Accumulation utility must be executed before recovery to provide a current input file because logs are not used with the track recovery option. Note that image copy and recovery of an embedded index of a KSDS are not possible and should not be attempted. However, a normal full recovery of the KSDS rebuilds an embedded index as well as the KSDS data area. The track recovery option is specified on control statements.

JCL REQUIREMENTS

The Data Base Recovery utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC

This statement may either invoke a cataloged procedure containing the required statements, or be in the form:

```
PGM=DFSRRC00,PARM='UDR,DFSURDB0,dbdname'
```

where dbdname is the name of the DBD that includes the data set to be recovered.

The normal IMS/VS positional parameters such as BUF and SPIE can follow dbdname. (See "Member Name DLIBATCH" in IMS/VS System Programming Reference Manual for additional parameters that can be specified in executing a batch processing region.)

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBD that describes the data base data set to be recovered. This is usually DSNAME=IMSVS.DBDLIB. This data set must reside on a direct access volume.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct access volume, or printer, or be routed through the output stream. SYSPRINT may be blocked but must be a multiple of 121.

SYSIN DD
Defines the input control data set. It can reside on a tape, direct access volume, or card reader, or be routed through the input stream.

DFSUDUMP DD
Defines the image copy input data set. It can be a data set created by either the batch Data Base Image Copy utility, the Online Data Base Image Copy utility, or the HISAM Reorganization Unload utility. If the input data set is created with the HISAM Reorganization Unload utility, the TRCV option cannot be used to recover the first control interval (RBA 0) of an ESDS. If no image copy or HISAM unload copy input is supplied, this statement must be coded as DD DUMMY. The ddname on this statement can be other than DFSUDUMP. If this is the case, the ddname must also be included in position 22 of the utility control statement. The data set can reside on tape or a direct access volume.

DFSUCUM DD
Defines the accumulated change input data set. If no accumulated change input is supplied, this statement must be coded DD DUMMY. This data set can reside on tape or a direct access volume.

DFSULOG DD
Defines the log change input. If no log changes are to be applied, this statement must be coded as DD DUMMY. This data set can reside on tape or a direct access volume. This data set is not used with the track recovery option.

Multiple logs can be used as input by concatenating the data sets. This requires that the DD statements be in date and time sequence. (See Example 2 for details.)

dataset1 DD
Defines the data set to be recovered. The ddname must be the same as the one in the DBD that describes this data set. It must also be in the utility control statement.

For DEDBs, this DD statement defines the area data set of the area to be recovered and the ddname must be the same as the one in the DBD that describes this area.

If an area is registered in the DBRC RECON data set, the ddname and dsname must match the names registered in the ADS list of the target area. If an area is not registered in the DBRC RECON data set and the DBRC RECON data set has the NOFORCER attribute, the ddname must be the same as the area name and must be in the utility control statement.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required:

- If only change accumulation input is used
- If log input is used
- For recovering a VSAM ESDS with data from HISAM unload as input
- For recovery when a null image copy data set is used as input

For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide. The data set can reside on a tape, direct access device, or card reader, or be routed through the input stream.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence will be used for the dump.

DFSTRCV DD
Defines the volume and extent information used by ATLAS to assign alternate tracks on the volume. The DSNAME operand must contain the name of the VSAM data space associated with the data set that contains the defective tracks. The data space name is the name in the FORMAT1 DSCB that describes the extents for the defective tracks.

If the data or index component of the VSAM cluster is defined with the UNIQUE attribute, then the data space name is the same as the component name. If the components are not defined with the UNIQUE attribute, the system generates data space name. To determine the generated data space name, list the VSAM catalog and the VTOC for the volumes containing the error tracks.

For a detailed description of VSAM data spaces, see OS/VS1 Access Method Services or OS/VS2 Access Method Services.

The DFSTRCV statement can be coded as DD DUMMY when performing track recovery on an IBM 3850 Mass Storage System, because the ATLAS SVC is not used.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENT

```
1   5                                                              80
|                                                                   |
| ABEND                                                             |
|                                                                   |
```

This utility control statement requests that the Data Base Recovery utility terminate with a user abend 302 if any abnormal condition is encountered. A storage dump is provided if a SYSUDUMP DD statement is supplied. If this statement is omitted, the Data Base Recovery utility issues error messages for any abnormal condition encountered and continues processing.

This control statement must come before the Data Base Recovery utility control statement.

```
1       7                                                          80
|                                                                  |
| NOSEQCK                                                          |
|                                                                  |
```

This utility control statement requests that the Data Base Recovery utility not perform sequence checking on the input log(s). This statement should be used with extreme caution, because recovery may not be possible if logs have sequencing problems.

This control statement must come before the Data Base Recovery utility control statement.

```
1  2  3  4       12 13       22         45                    80
|                                                              |
| S  T  U  D132DB01 I DBH13A   [DFSUDUMP] [COMMENTS]           |
|                                                              |
```

Position	Description
1	This must be the character "S." The "S" defines this statement as a Data Base Recovery utility control statement.
2	This must be blank or the character "T." If it is blank, a full data set recovery will take place. If "T" is specified, track recovery is requested for the data set.
3	This must be blank or the character "U." If it is a blank, the track is updated only if ATLAS detects an I/O error on the track. If "U" is specified, the track is updated even though ATLAS does not detect an error on the track. When performing track recovery on a 3850 Mass Storage System, this position is ignored, because the ATLAS SVC is not used.
4	This must be the name of the DBD that describes the data base containing the data set to be recovered. This name must also appear in the PARM field of the EXEC statement.
12	This must be blank or the character "I." If blank, this data set is not an index of a KSDS. The "I" specification requests track recovery of an index of a KSDS. If "I" is specified, position 2 must contain the character "T."
13	This must be the ddname of the data set or areaname to be recovered. It must be the same as the ddname in the DBD and dataset1 DD statement.
22	This is the ddname of the data set to be used for the image copy input. If this field is left blank, the ddname 'DFSUDUMP' is the default.

31-44	Reserved for DBRC use. Refer to the "Time-Stamp Recovery of a Data Base Data Set" section in IMS/VS Data Base Recovery Control: Guide and Reference for a description of the contents of columns 31 through 44.
45-80	Positions 45 through 80 can optionally contain comments.

RETURN CODES

The Data Base Recovery utility provides the following return codes:

Code	Meaning
0	Requested recovery successful.
4	Warning error occurred.
8	Serious error occurred.
16	Catastrophic error occurred; requested recovery unsuccessful.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

EXAMPLES

Example 1

This example shows the JCL for recovering an ISAM data set with a ddname of DBHI3A in a data base named DI32DB01. Input is provided from the image copy and change accumulation data sets.

```
//RECOVERY  JOB  1,1,MSGLEVEL=1
//STEP1  EXEC  PGM=DFSRRC00,PARM='UDR,DFSURDB0,DI32DB01'
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP DD  DSN=IMS.DBAOUT1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=DBDMP1,DISP=(OLD,KEEP)
//DFSUCUM DD  DSN=IMS.CUM1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=DBCUM1,DISP=(OLD,KEEP)
//DFSULOG   DD  DUMMY
//DBHI3A    DD  DSN=IMS.DBHI3A,UNIT=SYSDA,
//   VOL=SER=DBASE1,DISP=(NEW,KEEP),
//   SPACE=(CYL,(20,,2)),DCB=(DSORG=IS)
//DFSVSAMP DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//SYSIN  DD    *
S  DI32DB01 DBHI3A        RECOVERY FOR ISAM DATA SET
/*
```

Note: The SYSPRINT data set can be blocked but must be a multiple of 121. Any ISAM data sets to be recovered must be specified DISP=NEW.

The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 2

This example shows the JCL for recovery of an HDAM OSAM data set with a ddname of DBHD3B in a data base named DD32DB01. Input is provided from an image copy data set and multiple system log data sets. Note that the ddname on the image data set is not defaulted to DFSUDUMP in the control statement.

```
//RECOVERY  JOB  1,1,MSGLEVEL=1
//STEP1   EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,DD32DB01'
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT    DD  SYSOUT=A
//DUMPDS DD  DSN=IMS.DBBOUT1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=DBDMP3,DISP=(OLD,KEEP)
//DFSUCUM DD  DUMMY
//DFSULOG  DD  DSN=IMSLOG.MONDAY,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=LOG1,DISP=(OLD,KEEP)
//         DD  DSN=IMSLOG.TUESDAY,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=LOG2,DISP=(OLD,KEEP)
//DBHD3B   DD  DSN=IMS.DBHD3B,UNIT=SYSDA,VOL=SER=DBASE2,
//   SPACE=(CYL,(20,10)),DISP=(NEW,KEEP)
//DFSVSAMP DD  DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//SYSIN  DD  *
S  DD32DB01 DBHD3B  DUMPDS   RECOVERY FOR HDAM OSAM DATA SET
/*
```

Note: The log input can be concatenated but must be in date and time sequence.

The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 3

Example 3 shows the JCL for recovering a VSAM data set with the track recovery option after an I/O error is detected. The data set recovered is the data component of a VSAM KSDS (key sequenced data set) that is the primary data set of a HISAM data base. The ddname of the VSAM KSDS is DBVI3A in a data base named DI32DB01. The DFSTRCV dd statement defines the data space name in the format 1 DSCB and the volume serial number for the component of the VSAM KSDS that had the I/O error. Input is provided from the image copy and change accumulation data sets.

```
//RECOVERY  JOB  1,1,MSGLEVEL=1
//STEP1  EXEC  PGM=DFSRRC00,PARM='UDR,DFSURDB0,DI32DB01'
//STEPLIB   DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT    DD  SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP DD  DSN=IMS.DBCOUT1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=DBDMP1,DISP=(OLD,KEEP)
//DFSUCUM DD  DSN=IMS.CUM1,UNIT=TAPE,LABEL=(,SL),
//   VOL=SER=DBCUM1,DISP=(OLD,KEEP)
//DFSULOG  DD  DUMMY
//DBVI3A   DD  DSN=IMS.DBVI3A,DISP=OLD
//DFSTRCV DD DSN=VSAM0001.HISAM.PRIME.DATA,
//   DISP=OLD,VOL=SER=KSDS01
//DFSVSAMP   DD  DSN=IMS.VSAM PARM(OPTIONS),DISP=SHR
//SYSIN DD *
ABEND
ST DI32DB01  DBVI3A
/*
```

Note: The DFSVSAMP data set contains user-defined buffer information.

The following DD statements are also required if DBRC (Data Base Recovery Control) without dynamic allocation is being used:

```
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//RECON3   DD  DSN=RECON3,DISP=SHR
```

Example 4

This example shows the JCL for recovery of a DEDB area with an area name of AREANAM1 in a data base named DI32DB01. Input is provided from an image copy data set and change accumulation data sets. DDNAME1 and DSNAME1 are the ddname and dsname in the ADS list of the area to be recovered. (The numbers above the control statement are for reference only and are not to be included in the input stream.)

```
//RECOVERY  JOB  1,1,MSGLEVEL=1
//STEP1     EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,DI32DB01'
//STEPLIB   DD   DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//IMS       DD   DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD   SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP  DD   DSN=IMS.DBAOUT1,UNIT=TAPE,
//  VOL=SER=DBDMP1,LABEL=(,SL),DISP=(OLD,KEEP)
//DFSUCUM   DD   DSN=IMS.CUM1,UNIT=TAPE
//  VOL=SER=DBCUM1,LABEL=(,SL),DISP=(OLD,KEEP)
//DFSULOG   DD   DUMMY
//DDNAME1   DD   DSN=DSNAME1,DISP=OLD
//DFSVSAMP  DD   DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//RECON1    DD   DSN=RECON1,DISP=SHR
//RECON2    DD   DSN=RECON2,DISP=SHR
//RECON3    DD   DSN=RECON3,DISP=SHR
//SYSIN     DD   *
1234         12 13             31                          80
S  DI32DB01     AREANAM1       RECOVERY FOR DEDB
/*
```

BATCH BACKOUT UTILITY (DFSBB000)

The Batch Backout utility program restores data bases from a point when updates were applied to a point before the program was initiated, or to a checkpoint. The input log data sets can be OLDS and/or SLDS. The logs used must not have been created before a reorganization. The program operates as a normal IMS/VS batch job. It uses the PSB used by the program whose effects are to be backed out. All data sets of PCBs within a PSB that require backout must be provided as input to the Batch Backout utility. Thus all data base data sets for the PSB do not have to be provided as input. Also, even though a secondary data set group may not have been updated during the execution of the program, the data set will have to be supplied as input to backout if any other data set group within the data base was updated.

The usefulness of the utility is dependent on the type of error(s) encountered, the configuration of the IMS/VS system, and whether or not the proper sequence of recovery steps are taken by the user. The circumstances under which this utility should be run are described in IMS/VS Operations and Recovery Guide.

A log is created during any backout process. This log must be included in any subsequent data base recovery attempt made against any of the data bases involved in the backout. The log contains backout records that are matched during the recovery run against log records created during the update run to remove

the update effect on the data base. This log must not be used as input to any subsequent backout attempt.

All logs with any changes for a data set must be mounted if backout is to a checkpoint. The extent of the backout is determined by the region type being backed out, the IMS/VS system configuration, and control statement input in the case of batch checkpoint.

When using a multiple volume log data set with the Batch Backout utility, the JCL must specify the log data sets in the order of oldest to newest. The utility will process them in reverse order, newest to oldest.

When the Batch Backout utility is used to back out changes made by an online region, multiple log data sets may be required. Up to 11 data sets are allowed and these can be a mixture of tape and DASD. The log data sets must be processed in reverse order to their creation; that is, the most recently created log is input to the first execution of the utility. This method of processing is a result of an OS/VS restriction that prevents this utility from reading multiple concatenated data sets backward in a single job step.

One of the following values may be specified for the DBRC parameter in the EXEC JCL statement for this utility (see member name DBBBatch or DLIBatch in the IMS/VS System Programming Reference Manual for the correct position of the DBRC parameter).

- Null

  DBRC will be included based on the specification of the IMSCTRL macro statement keyword DBRC= during IMS/VS system definition.

- Y

  DBRC will be used during this execution of IMS/VS.

- N

  DBRC will not be used during this execution of Batch Backout. DBRC=FORCE system definition specification may be overridden by the specification of DBRC=N on the EXEC procedure if the previous execution of IMS/VS used DBRC but not IRLM.

- Specification of DBRC=C will permit you to back out batch jobs that terminated normally during an execution that included DBRC but did not include IRLM. During batch backout of jobs that terminated normally, IMS/VS performs authorization for data bases and then backs out data base changes, just as is done for jobs that terminate abnormally. However, if the previous execution of IMS/VS included both DBRC and IRLM, batch backout will fail if the previous execution completed normally.

If the IMS/VS resource lock manager (IRLM) is being used during this execution of IMS/VS and DBRC=N is specified, IMS/VS abnormally terminates with message DFS044I and a nonzero return code will be issued. If the IRLM is not being used, using DBRC provides additional data base security. If a common RECON data set is shared among subsystems, DBRC allows proper authorization to registered data bases so the subsystems can share data at the data base level.

Figure 77 on page 302 presents a summary of conditions that terminate the processing of the Batch Backout utility.

Summary of Conditions Ending the Backout of an Individual Region		
Region Being Backed Out	CHKP-ID specified	CHKP-ID not specified
MPP/BMP	1. First CHKP record encountered. 2. Scheduling record for this PSB. 3. Completed ROLB call.	1. First CHKP record encountered. * 2. Scheduling record for this PSB. 3. Completed ROLB call. 4. Message GU record.
DLI/DBB	1. CHKP record requested.	1. First CHKP record encountered. * 2. Completed ROLB call. 3. ' Batch Started ' Accounting record.
*Note: The order of occurrence is referenced as from the end of the log toward the start of the log . (Read-backward processing.)		

Figure 77. Conditions That Terminate the Batch Backout Utility

Figure 78 on page 303 depicts the data set requirements for the Batch Backout utility.

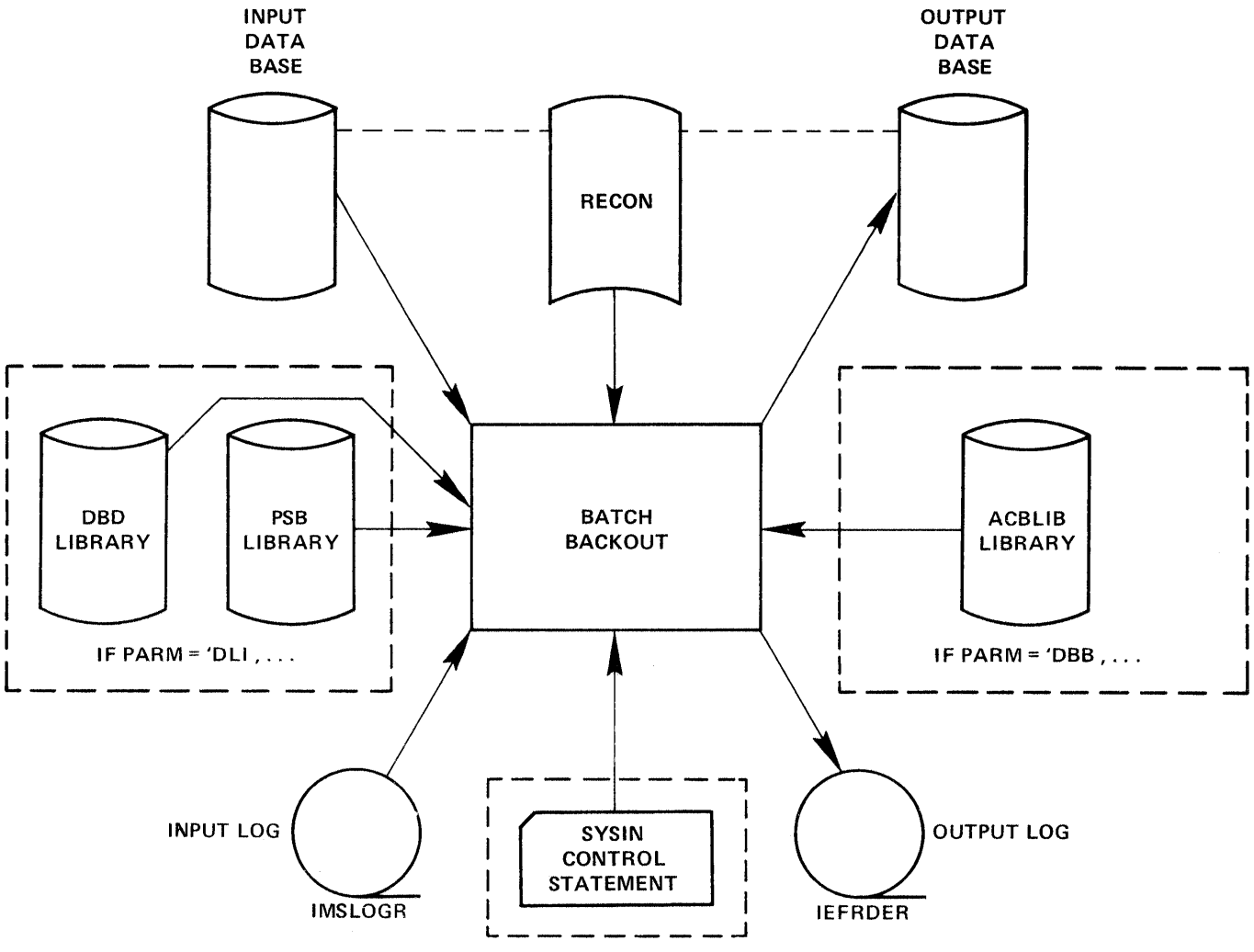


Figure 78. Data Set Requirements for the Batch Backout Utility

JCL REQUIREMENTS

The Batch Backout utility is executed as a standard OS/VS job. It requires a JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs.

EXEC

This statement may be in the form:

```
PGM=DFSRRC00,PARM='DBB,DFSBB000,psbname,nnn' (if
prebuilt blocks are being used; otherwise,
PARM='DLI,DFSBB000,...')
```

where psbname is the name of the PSB used by the program to be backed out and nnn is the number of 1K-byte blocks required for the data base buffer pool.

See "Member Name DBBBATCH or DLIBATCH" in IMS/VS System Programming Reference Manual for additional information on executing a batch processing region.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Describes the IMSVS.PSBLIB and IMSVS.DBDLIB data sets, (PARM='DLI,...'). These data sets must reside on a direct access volume.

OR:

IMSACB DD
Describes the IMSVS.ACBLIB data set, (PARM='DBB,...'). This data set must reside on a direct access volume.

IMSLOGR DD
Describes the input log file. It may reside on tape or DASD.

IMSLOGRn
Describes additional input logs. The data sets may be OLDS and/or SLDS. The suffix n can be 0 through 9.

Note: IMSLOGRx DD statements specify the input log data sets. The data sets may be OLDSs and/or SLDSs. The suffix x can be blank, or 0 through 9. If there is only one input log data set, it is specified by the IMSLOGR DD statement. If there are multiple input log data sets, then IMSLOGR is the ddname for the least current, IMSLOGR0 the DD name for the next, up to IMSLOGRn for the most current. Because IMSLOGRn is the most current (contains the most recent log data), it will be used first by Batch Backout utility and IMSLOGR, being the least current will be used last.

IEFRDER DD
Describes the system log created during backout. The data set resides on tape or DASD.

IEFRDER2 DD
Describes the secondary system log created during backout. The data set resides on tape or DASD. Include this statement only when dual log output is desired.

data base DD
These are the data base DD statements required by the PSB referenced in the EXEC statement. This data set must reside on a direct access volume. If this data set is VSAM, performance will improve when the AMP parameter is used to specify additional VSAM buffers.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.) The data set can reside on a tape, direct access device, or card reader or be routed through the input stream.

SYSIN DD
Required only if a CHKPT control statement is supplied. This data set can reside on a tape, direct access volume, or card reader, or be routed through the input stream.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

UTILITY CONTROL STATEMENTS (OPTIONAL)

Backout of a batch region normally terminates at the first checkpoint encountered on the input log data set (processed backward). If it is necessary to back out to an earlier checkpoint, the CHKPT control statement identifies this earlier checkpoint.

```
 1     7          16                                              80
┌──────────────────────────────────────────────────────────────────┐
│ CHKPT AAAAAAAA [COMMENTS]                                        │
└──────────────────────────────────────────────────────────────────┘
```

Position	Description
1	Positions 1 through 5 must be the characters 'CHKPT'. These characters define the control statement.
7	This is the 8-character checkpoint ID supplied to IMS/VS with the 'CHKPT' call. The ID is displayed as part of message DFS681I at the time the 'CHKPT' call is made.
16	Positions 16 through 80 can optionally contain comments.

Backout normally will not go beyond a system checkpoint, when a valid termination has been found for all active PSTs. This may make the data base difficult to recover in the case of a user error, such as when a PSB is inadvertently started again after some type of failure in which the PSB and the data base(s) were stopped, but before the data base had been recovered. Backout will allow for a manual system checkpoint override. This is done with the IGNORE utility control statement, which may be submitted in addition to the CHKPT utility control statement, if appropriate. This will force batch backout to search as far as necessary, possibly all the way back to the beginning of the log, in order to determine if a successful termination has occurred for all PSTs. This will cause a lot of unnecessary processing and should only be used under exceptional conditions, such as operator errors. The format of the control statement is as follows:

```
 1       8                                                       80
┌──────────────────────────────────────────────────────────────────┐
│ IGNORE   [COMMENTS]                                              │
└──────────────────────────────────────────────────────────────────┘
```

Position	Description
1	Positions 1 through 6 must be the characters 'IGNORE'. These characters define the control statement.
8	Positions 8 through 80 may optionally contain comments.

RETURN CODES

The Batch Backout utility provides the following return codes:

Code	Meaning
0	Backout successful. (DFS395I)
4	PSB incorrect. (DFS396I)
8	Unable to open data base. (DFS397I)
12	Data base I/O error. (DFS398I)
16	Buffer pool too small. (DFS399I)
20	Unable to open input log. (DFS400I)
24	Call to DBRC failed. (DFS401I)
28	No data base record in log. (DFS888I)
32	No block size for IMSLOGR DD statement. (DFS890I)
36	Invalid record on input log. (DFS894I)
44	No CHKPT ID on control statement. (DFS957I)
48	Specified CHKPT not found. (DFS958I)
52	Specified CHKPT not within last schedule. (DFS959I)
60	Input log was specified as DD DUMMY or multiple log DD statements were specified, but correct ddname and/or order was not specified. (DFS2296A)
64	Backout processing incomplete for PSB. (DFS2298A)
68	DBRC is required for this execution of backout. (DFS044I)
72	IRLM is required for this execution of backout. (DFS045I)
76	Unable to locate X'42' log record (DFS091I)
80	A normally completed job may not be backed out if IRLM was active (DFS173I)

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step. Each return code, however, causes a message to be printed. (The message that corresponds to each return code is shown above.)

EXAMPLE

The following example shows the JCL for backout of data base changes made by a program that uses PSB PLVAPZ12.

```
//BTCHBO   JOB A,USER,CLASS=H,TIME=1440,REGION=1500K
//JOBLIB   DD DSN=IMSVS.RESLIB,DISP=SHR
//JOBCAT   DD DSN=VCATSHR,DISP=SHR
//EXAMPLE  EXEC PGM=DFSRRC00,REGION=156K,
//            PARM='DBB,DFSBB000,PLVAPZ12,008,1,,,,,,,IMSS,,Y,Y,IRLM'
//SYSABEND DD SYSOUT=A
//IMSACB   DD DSN=IMSVS.ACBLIB,DISP=SHR
//IEFRDER  DD DSN=LGBKOUTF,DISP=(,KEEP),UNIT=SYSDA,VOL=SER=000000,
//            SPACE=(CYL,(1,1)),
//            DCB=(RECFM=VB,BLKSIZE=8192,LRECL=8188,BUFNO=12)
//IMSLOGR  DD DSN=DSHR.OLDSP0,DISP=OLD,UNIT=SYSDA,VOL=SER=IMSQAD
//IMSLOGR0 DD DSN=DSHR.OLDSP1,DISP=OLD,UNIT=SYSDA,VOL=SER=IMSQAD
//IMSLOGR1 DD DSN=DSHR.OLDSP2,DISP=OLD,UNIT=SYSDA,VOL=SER=IMSQAD
//DFSVSAMP DD *
512,6
1024,6
2048,6
//DBHVSAM1 DD  DSN=DSHR.FJVHSG1K,DISP=SHR
//DBHVSAM2 DD  DSN=DSHR.FJVHSG1E,DISP=SHR
//HIDAM    DD  DSN=DSHR.FKVHIG1E,DISP=SHR
//HIDAM2   DD  DSN=DSHR.FKVHIG2E,DISP=SHR
//XDLBT04I DD  DSN=DSHR.FKVHIIXK,DISP=SHR
//RECON1   DD  DSN=RCONDFI1,DISP=SHR
//RECON2   DD  DSN=RCONDFI2,DISP=SHR
```

CHAPTER 8. UTILITY CONTROL FACILITY (DFSUCF00)

GENERAL DESCRIPTION

The concepts and procedures discussed in this chapter apply specifically to the Utility Control Facility (UCF) and to the manner in which it implements the functions of those utilities described in the "Data Base Recovery" and "Data Base Reorganization" chapters of this manual. The correct execution of the UCF is contingent on the reader's knowledge of the requirements for those utilities.

The UCF is a program that acts as a controller for the execution of the utilities. The UCF is driven by control statements coded in a freeform manner. The user can supply multiple control statements that cause the UCF to execute multiple utilities (such as execution of the Data Base Recovery, Reorganization, and batch Image Copy utilities) on the same or multiple data bases in the same job step. Other advantages in using the UCF are as follows:

- Restart processing is provided and can be initiated by a single EXEC parameter or control statement.
- Most operations within the control stream can be stopped and restarted at a later time at the user's convenience.
- The outstanding Write to Operator with Reply (WTOR) can be used to stop the job as well as to enter certain options.
- User exits are provided to access data records being processed.
- The UCF constructs a control data set, based on control statement specifications, that organizes and executes all functions in a manner that protects the user from certain operational problems. (See "Normal Processing" later in this chapter.)

RESTRICTIONS

1. For restart of the user's Initial Load program and/or the HD Reorganization Reload utility under the UCF, the following restrictions apply:
   - The restart applies only to a VSAM data set.
   - Multiple load PCBs may be included in the same PSB. However, during the initial load, or the restart of the initial load, the initial load program must use only one load PCB per execution of a FUNCTION=IL statement.
   - Root segments must be keyed such that the user's HDAM randomizing module can locate the root.
   - Restart must begin with an ISRT for the next root segment following the root with the key equal to the key feedback area.
   - The user-written initial load program is responsible for repositioning the input file.

2. The following precautions must be observed for restart of the user's Initial Load program or if restart is being done for an HDAM data base that was being reloaded by the HD Reorganization Reload utility:

   - The checkpoint value for this restart must be the same value as specified in the failing UCF function.
   - If the root sequence field is nonunique, any root segments that were inserted after the checkpoint at which the restart was made will remain in the data base. The only valid condition for restart would therefore be a controlled termination (application program returns with a nonzero return code).
   - If the root sequence field is nonunique and a root segment insert is done for a segment that already exists in the data base because of segments inserted after the checkpoint, the data will be compared. If the segment data is the same, the old segment will be overlaid with its replacement and the dependent segments will be dropped because they will be reinserted by subsequent user/reload insert. This will occur only until a non-duplicate root is found. After a segment with a new key or with different data is encountered, LB status codes will be returned for any subsequent duplicates.

   Note: On a restart, if a path call is used to insert a root and its dependent when the root already exists in the data base, any insert of the dependent segment in that root in the path call will fail.

3. The user must explicitly close any data sets opened in any user-written routine; otherwise, the UCF abends.

4. When reorganizing a VSAM data base using one execution of the UCF, the user must specify EXEC=STOP on the utility control statement requesting the unload function. When the unload operation has been completed, the UCF stops execution. The data base can then be deleted and reallocated using the Access Method Services utility, and the UCF can then be RESTARTed.

   If multiple data bases are unloaded, the user can specify 'STOP' in each unload function or in only the last unload function that causes UCF to stop after all data bases are unloaded. UCF functions are performed on data bases in the collating sequence order of the data base names (DBX before DBY, etc.). The 'STOP' should be placed in the statement referring to the last data base name.

5. If using the track recovery option under the UCF, the Change Accumulation utility must have been run in this step or prior to this step.

6. The UCF cannot be used for data base backout/restart; the Data Base Backout utility does not require any of the functions provided by the UCF.

7. During reorganization of a data base whose DBD has both changed and contains logical relationships, UCF execution must be stopped after the unload function if one of the following conditions exist:

   - Either counter, LT, or LP pointers are changed
   - Adding or deleting segments involved in logical relationships change
   - The DBD name is changed and the DBD contains logical relationships

To stop UCF execution, use the EXEC=STOP parameter on the last unload function. The new DBD must then be generated followed by a new UCF execution (not a restart) to do the reload.

8. The UCF cannot be used to initially load a GSAM data base.

9. The UCF cannot be used to execute the Online Data Base Image Copy utility, the Data Base Surveyor utility, or the Partial Data Base Reorganization utility.

10. The FUNCTION=OP card must have the same checkpoint value and request parameters as the FUNCTION=OP card in the failing UCF step.

11. In case of an abrupt system termination, such as a loss of power, the user must terminate the unclosed output data set before restart is begun.

12. If your data base has logical relationships, utilities should be run either with or without control of UCF. Mixing UCF and non-UCF utilities can cause unpredictable results.

13. If, during restart of an initial load or reload program, UCF determines there are no valid checkpoints from which to restart, UCF will restart from the beginning of the load or reload. If any segments had already been written to the data base, message DFS730I, reason code I,30 is issued. When this occurs, you must scratch and reallocate data base data sets before restarting again.

14. DBRC is not supported for UCF restart processing when DSLOG input is used. If DBRC is active during UCF restart processing, the results will be unpredictable.

15. This utility does not support CICS/VS.

NORMAL PROCESSING

Normal processing requires control statements for direction of all utility functions except Data Base Scan (DFSURGS0), Prefix Resolution (DFSURG10), and Prefix Update (DFSURGP0). The UCF automatically generates all required statements for these three functions.

Note: If an unload or reload only is requested, Data Base Scan, Prefix Resolution, and Prefix Update processing is not automatic; the user must provide control statements to request execution of these utilities. If only an initial load is requested, processing of these utilities is automatic unless keywords on their control statements indicate no processing is to take place. One control card is necessary to request the execution of data base scan. Subsequent scan requests will be ignored.

The control statements are read at one time and a control data set is built. All entries in the control data set are cross-referenced to verify that no conflicting requests are made and that all logical relationship functions are either requested by the user or generated by the UCF.

The UCF executes the utilities in a particular order, regardless of the order in which the user submits the control statements. The order of execution is as follows:

1. Change Accumulation

2. Recovery

3. HISAM Reorganization Unload

4. HISAM Reorganization Reload

5. Data Base Scan

6. HD Reorganization Unload

7. HD Reorganization Reload

8. Initial Load

9. Prefix Resolution

10. Prefix Update

11. Reorganization Unload for Secondary Indexes

12. Reorganization Reload for Secondary Indexes

13. Image Copy (batch)

14. Data Base Zap

15. Module Zap

When the same function is requested on several control statements, the UCF executes the functions in ascending order of data base name. If two or more control statements for the same function also specify the same data base name, then these statements are executed in the order they were read in by the UCF.

In summary, the control statements are executed in the order of functions shown above. Within a function they are executed in data base name order; within the same function and data base name, they are executed in the order submitted.

Processing proceeds by determining the function to be started next, recording the event in the journal data set, and attaching the appropriate utility.

All functions are thus organized and executed in a manner that safeguards the user against certain operational difficulties that might otherwise occur. Consider, for example, the case of reorganizing a data base containing a logical parent when the logical child has a direct pointer to it. If the logical parent data base is unloaded and reloaded before data base scan is executed, the logical relationship is destroyed. The UCF, however, scans the logical child data base first, unloads, and then reloads the logical parent data base.

The standalone utilities statistics and summary reports are part of the UCF output. Where options to suppress statistics exist within the utilities, the REQUEST keyword values STATS and NOSTATS determine whether or not statistics are printed.

INITIAL LOAD APPLICATION PROGRAM CONSIDERATIONS

Initial load programs can be run under control of the UCF and thereby take advantage of the UCF restart capability. In most instances, only minor changes are required to these programs to allow for the proper interface. The programs must be modified to recognize when restart processing is occurring, when WTOR stop-processing requests have been entered, and when checkpoints have been taken. The interface is:

```
Register 0 ----> 4-word parameter list

                 1st word ----> PCB list
                 2nd word ----> DFSPRINT data set
                 3rd word ----> PST
                 4th word ----> During restart of the user's
                                initial load program, the
                                address of area containing the
                                last segment loaded prior to
                                checkpoint.
```

Register 1 ----> PCB list

When restart processing of a user's initial load program is in progress, a status code of UR (this is a restart) is returned in the load PCB upon entry to the program. Another parameter, the fully concatenated key contained in the PCB key feedback area, is also passed. The information in this key feedback area determines the nature of the restart, and two conditions apply:

- If the information is other than zeros, restart processing begins from the point of termination.
- If the information is zeros, restart processing is from the beginning of the program.

Because the user's program should respond with the next root or dependent segment to be loaded, the user I/O files may have to be adjusted by the application program.

Because further inspection might be necessary to determine the restart point on the input files, the actual segment data is also provided. (This would be important, for example, if there are nonunique or unsequenced fields.)

Additionally, the user's program should be modified to recognize when a DL/I status code indicates that the user's I/O files are to be checkpointed and that processing is to be terminated as a result of an operator's reply. This status code is returned only on a call that would have had a status of blanks.

The DL/I status codes and their meanings are:

UC Checkpoint has been taken. The data base is checkpointed at the logical record preceding the root that was just loaded.

UR This is a restart.

US The operator has requested initial load program to stop processing.

UX A combination of UC and US.

Note: For both the US and UX conditions, processing can be stopped at the next checkpoint.

The user should be aware of certain restrictions that apply to Initial Load application programs. See "Restrictions" (items 1 and 2) earlier in this chapter.

Initial Load Exit Routine

The UCF checkpoint module (DFSUCP90) is given control directly from the Load Insert module (DFSDDLE0) to allow the UCF to checkpoint the user's Initial Load program and to process replies (if any) to the outstanding WTOR. The DFSUCP90 module in turn provides interface to a user's exit routine to allow the user to change checkpoint intervals and/or to checkpoint work data sets. The interface to the exit routine is:

Upon entry,

Register 1 ----> 3-word parameter list

1st word ----> common area defined by DSECT UCFCMVEC

2nd word ----> DFSPRINT data set

3rd word ----> PST

The common area includes fields U7CURCKP and U7CKPTNK. The U7CURCKP field contains the checkpoint intervals currently in

effect. The U7CKPTNK field serves as a 4-byte counter and contains the number of records to be processed before a checkpoint is taken. The user can synchronize (in whatever manner desired) checkpointing of individual data sets with checkpoints currently in effect for the Initial Load program by:

- Monitoring the count and checkpointing data sets when the counter (U7CKPTNK) reaches zero, or
- Forcing a checkpoint by clearing the counter to hexadecimal zeros and checkpointing data sets.

Upon return to DFSUCP90, if the U7CKPTNK field is zero, a checkpoint record will be written to the journal data set, and the field will be reinitialized to the value contained in the U7CURCKP field.

TERMINATION/ERROR PROCESSING

Errorchecking occurs both during execution of the utility and after completion. If there are no errors, the completion of the event is recorded and a check is made for a request to stop processing. In the event that a request to stop processing was made, the appropriate restart messages are generated and the job ends with a return code of 4. If no stop processing request was made, the next function is determined and processing continues as described for normal processing.

If errors are discovered during the error-checking phase of execution, the completion of the event is recorded as an error and processing ends to allow for restart of the function. When the entire job is completed, appropriate return codes are passed. With a normal completion, restart processing is not necessary and is prevented by a special record written on the journal data set. Any termination other than normal can be restarted. Statistics are printed upon termination of each function and at job termination.

A return code of zero in register 15 indicates normal completion of the user's Initial Load program and that a restart is not to be done at a later time. If restart is to be done later, however, register 15 contains a return code greater than 4. This causes a checkpoint to be taken, and the restart proceeds with the next root segment.

CHECKPOINT/RESTART

The UCF contains an internal checkpoint/restart feature for abnormal termination and user-termination requests. The checkpoint function requires a journal data set, and the user must allocate this for the UCF; the IMS/VS system log normally used for batch processing is not used in this instance. Checkpoints are taken when a functional utility is started or ended, and when a control function has been started or ended. Checkpoints are also taken at specific points within the processing of a functional utility as a result of user-supplied and default record counts. At each of these checkpoints, records are written to the journal data set to define the type of checkpoint and the appropriate restart control information.

These checkpoints are used by the restart processor and are internal checkpoints rather than OS/VS checkpoints. The frequency of checkpoints can be specified as a user option or can be defaulted to every 2000 data base related records.

Examples of the JCL for executing restart of the UCF are provided at the end of this chapter.

RESTART PROCESSING

When restart processing is required, the control data set that was in use when the program last ended is read and the journal log that was last used is also read. The last function started and/or completed is determined by matching the journal records to the functions in the control data set, and processing continues as in normal processing.

The restart function occurs in one of the following ways, depending on the type of checkpoint records that were last written to the journal data set.

- If the records indicate the start of a functional utility (that is, one of the data base utilities or the user's initial load program), restart processing begins at the function that was started. If the records indicate the end of a functional utility, restart processing begins at the next function to be performed.
- If the records indicate the start or end of a control function (such as building the control data set), restart processing begins either at the control function that was started or at the next function to be performed.
- If the records indicate a checkpoint, restart processing begins at that point, and the IMS/VS data sets are positioned as necessary.

Restart processing requires the availability of all data sets that were in use at the time the checkpoint was taken.

During restart, when repositioning any multiple volume output data set, the DD statement must be changed to remove those volume serial numbers not used in the original execution. Failure to do this may result in incorrect repositioning of the output data set.

Restart of either the Change Accumulation utility or the Prefix Resolution utility must be from the beginning of execution. For these utilities, all data sets created up to the point of termination must be scratched and the utilities must be restarted as if for the first time.

USER EXIT PROCESSING

UCF allows user exits in all utilities to examine records or to compile statistics. While no IMS/VS control data may be altered during the user exit processing, the user data can be altered as long as the segment record length requirements are observed. The user exit is required to maintain all IMS/VS data unchanged during the exit processing. The HD Reorganization Unload utility (DFSURGU0) and the HD Reorganization Reload utility (DFSURGL0) user exits can delete or insert segments and view blocks after they have been loaded.

The user exit routines are specified on the utility control statements associated with a given function by coding the EXITRTN keyword and specifying the name of the exit routine. The user exit routine must reside in the LINKLIB or be defined in a STEPLIB or JOBLIB data set.

On entry to the user exit routine, register 1 points to a parameter list that contains three entries:

- Address of the data
- Address of the DFSPRINT DCB
- Address of the partition specification table (PST) at successful (nonabend) completion

The user exit is entered a final time with the address of the data equal to 0.

The HD Reorganization Reload utility user exit is entered at one of two locations:

- At offset 0, when the reload record has been read from the unload tape
- At offset 4, when the record has been loaded into the data base

The user exit routine may write on the DFSPRINT data set by issuing a PUT macro statement for a "MOVE MODE" operation. The DFSPRINT data set is opened before the exit routine is entered. Any nonutility data set used by the exit routine must be opened and closed by the exit.

The user exit routines used with the HD Reorganization utilities can pass return codes in register 15 that inform the utility of segment disposition. The return codes recognized by the HD Reorganization utilities are:

Code	Meaning
0	Process normally.
4	Delete the segment passed to the exit routine.
8	Insert the segment pointed to by register 1 before the current segment. Return to the exit routine with the same segment that was originally passed to it.

The segment name and the segment level can be found in the PCB pointed to by the PST. Any logical children of the segment being inserted must be inserted separately into their own data bases.

SERVICE AIDS

There are two types of service aids available with the UCF—error-point abends, and data base zaps and module zaps. The zap aids should be used only by an IBM Field Engineering Program Support Representative or by someone authorized and under proper direction.

Error-Point Abends

The UCF allows selective abend requests. If a diagnostic message is generated during processing, a control statement can be used to select points at which to invoke an abend to the program. Specifying REQUEST=MSGALL indicates that all "A" or "W" type diagnostic messages are to be set as abend requests. The MSGNUM keyword indicates the exact message at which to abend if that message is issued during processing. (See the FUNCTION=OP control statement for additional information.)

Data Base Zap/Module Zap

The UCF can be used to zap data base blocks, UCF modules, or UCF utilities to force abend conditions or to correct logic errors. The control statement rules for this zap facility follow the control statement rules for the SPZAP program. Only the VERIFY and REP control statements are supported, however, and certain restrictions apply to their use. Exactly 8 bytes of data must be specified, and in 2-byte hexadecimal form. The data must not be separated by commas or spaces. (For further information on SPZAP, refer to OS/VS2 System Programming Library: Service Aids.)

Zaps on modules are performed in storage, while data base zaps are performed on disk. If a zap statement contains a RELATE keyword, the module zap is performed before execution of the RELATEd module. Data base zaps are performed before unload utility functions and after load utility functions.

Module zaps are performed prior to each separate execution of the specified module name unless restricted by the RELATE and SEQ keywords. If RELATE is specified for a module zap, only those functional control statements with a matching module name are zapped. If SEQ is specified, this further restricts the zap to the module with a matching sequence number. Data base zaps are performed before unload utility functions, after load utility functions, and/or after all requested functional utilities have been executed.

Data Base Zaps

Before Unload Utility Functions	After Load Utility Functions
IM	SR
SU	DR
RU	RR
DU	RV

All other data base zaps are executed after the last requested functional utility has executed.

WRITE-TO-OPERATOR-WITH-REPLY (WTOR) FUNCTION

This UCF function issues messages to and processes replies from the UCF user. The outstanding WTOR provided by the UCF allows the user to interrogate the UCF to determine the status of its execution, to change checkpoint values, to stop the UCF and then to restart it at a later time. The restart cannot, however, be initiated by a WTOR. The WTOR is processed between executions of the utilities and at checkpoint time.

In the example that follows, the user receives message DFS367I at the console, requests the status of the UCF's execution, and learns that it is executing the HD Reorganization Reload utility with the data base "DBNAME." (The DBNAME is not included in the message for functions CA and PR.)

```
@09   *DFS367I UTILITY CONTROL FACILITY RUNNING,
        ENTER REQUESTS AS NEEDED
r 9, status
IEE600I  REPLY TO 09 IS 'STATUS'
DFS369I  FUNCTION IS DR FOR DATABASE DBDNAME
@10   *DFS367I UTILITY CONTROL FACILITY RUNNING,
        ENTER REQUESTS AS NEEDED
```

In response to the last DFS367I message, the user might, for example, request that a checkpoint value be changed immediately as follows:

```
r 10,ckpnt=100
```

Or the user might, for example, request the UCF to stop processing with:

```
r 10,END
```

The UCF stops processing upon recognition of the END request by the executing utility.

Additionally, the user can enter certain control replies and option replies in response to message DFS367I.

Control Replies	Option Replies
END FUNCTION xx END STATUS	Identifiers FUNCTION= SEQ= EXEC= Values REQUEST= CKPNT=

Control Replies

Only one of the control replies can be entered on one response.

END
A reply of END stops processing at the next checkpoint in, or after, the current function (whichever checkpoint occurs first).

FUNCTION xx END
This reply stops processing after the first execution of the specified function. The UCF can be restarted after FUNCTION xx END by adding a control statement punched FUNCTION=OP, COND=RESTART to the DFSYSIN data set and resubmitting the job. Example 3 at the end of this chapter shows the JCL for executing restart of the UCF.

STATUS
This reply causes a WTO message that explains which function, and, if possible, which data base or data set are being processed. Status requests are processed between executions of the utilities or at a checkpoint, whichever occurs first.

Option Replies

Either of the FUNCTION= or SEQ= identifiers can be entered alone or with the EXEC= identifier. The EXEC= identifier, however, can only be entered if either the FUNCTION= or SEQ= identifier is entered.

If both FUNCTION= and SEQ= are entered on the same reply, they must agree with the control data set entry of the same sequence number. For example, if the user's entries are FUNCTION=CA,SEQ=004, the control data set with SEQ=004 must be associated with FUNCTION=CA.

If the FUNCTION= identifier is not entered on an option reply, FUNCTION=OP is assumed. FUNCTION=DX and FUNCTION=SX cannot be entered on a WTOR reply.

The values REQUEST= or CKPNT= can be entered alone or with the identifiers.

Multiple use of one identifier in a reply causes only the last entered value for that identifier to be accepted.

JCL REQUIREMENTS

The UCF is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required. For additional information on JCL requirements, see the UCF examples at the end of this chapter.

EXEC
This statement can be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSUCF00'.
```

It can also be in the form of a procedure that contains the required job control and utility control statements. A region size of 600K bytes is the minimum for execution.

For restart processing, PARM='ULU,DFSUCF00,,,0001' should be specified. (For further information on specifying PARM options, see "Member Name DLIBATCH" in Chapter 1 of IMS/VS System Programming Reference Manual.)

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSVS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

IMS DD
Defines the library containing the DBDs and PSBs that describe the data bases and their processing blocks (that is, DSN=IMSVS.DBDLIB and IMSVS.PSBLIB).

DFSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct access device, or printer, or be routed through the output stream. This file is specified in the program with the DCB operand LRECL=121 and RECFM=FBA. BLKSIZE must be specified on this DD statement. If BLKSIZE is not specified there is no default, and the results are unpredictable.

DFSYSIN DD
Defines the input control statement data set. This data set can reside on a tape, direct access device, or system reader. This file is specified in the program with DCB operands LRECL=80 and RECFM=FB. BLKSIZE can be specified on this DD statement.

DFSNJRNL DD
Defines the new UCF journal data set. This data set can reside on a tape or direct access device. It contains control records and checkpoint records for use in restart processing. This data set must always be specified and is specified in the program with DCB parameters RECFM=VB, BLKSIZE=4008, and LRECL=4000. DISP=(,KEEP) must be specified by the user.

To use a multivolume data set, you must preformat all volumes except the first one. A volume sequence number of 1 must also be specified in the VOLUME parameter.

DFSOJRNL DD
Defines the old UCF journal data set created in a prior run that requires restart processing. This data set can reside on a tape or a direct access device. It must always be specified. When restarting, it is defined in the program with the following DCB values: RECFM=VB, LRECL=4000, and BLKSIZE=4008. When not restarting, it must be specified as DD DUMMY.

DFSNCDS DD
Defines the new control data set for this program. This data set can reside on a tape or a direct access device. DISP=(,KEEP) is required because this data set must be used as input to restart processing. This data set must always be specified and is specified in the program with DCB parameters of LRECL=1600 and RECFM=FB. BLKSIZE must also

be specified; if it is not, there is no default, and the results are unpredictable.

DFSOCDS DD
Defines the old control data set that was created in a prior run that requires restart processing. This data set can reside on a tape or a direct access device. It must always be specified when restarting. The DCB is defined in the program with LRECL=1600 and RECFM=FB. When not restarting, the data set can be specified as DD DUMMY. BLKSIZE must be specified on the DD statement only if the data set resides on unlabeled tape.

DFSRDER DD
This DD statement can be omitted or specified as DD DUMMY. It defines the IMS/VS system log. The system log is not currently used by the initial load or reload utilities.

DFSCNTRL DD
Defines the control data set that is created just prior to attaching the functional utility. This data set must always be specified and can reside on a tape or direct access device. The program defines the DCB with LRECL=80, RECFM=FB, and BLKSIZE=80. DD DUMMY cannot be used to specify this data set.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required if any of the data bases used are VSAM, ISAM, or OSAM data sets. (For additional information on control statement format and buffer pool structure, see "Defining the IMS/VS Buffer Pools" in IMS/VS Installation Guide.) The data set can reside on a tape, direct access device, statement reader, or be routed through the input stream.

The following DD statements are required if the job execution involves a Change Accumulation or a Prefix Resolution execution.

SYSOUT DD
Defines the output data set used by the Sort/Merge program. This data set can reside on a tape, or direct access device, or printer, or be routed through the SYSOUT stream. The DCB parameters are established by the Sort program. (For a description of these parameters, refer to the applicable sort/merge manual listed in the "How to Use This Publication" section of this manual.)

SORTLIB DD
Defines the data set containing load modules for the sort/merge execution (that is, DSNAME=SYS1.SORTLIB).

SORTWKnn DD
These statements define the intermediate storage data sets for the Sort/Merge program. The "nn" designation is a 2-digit number.

The following DD statements are required if the job execution involves reorganization of a data base with logical relationships, initial loads, the Prefix Resolution utility, or the Prefix Update utility executions.

DFSURWF1 DD
Defines the data set used to resolve logical or secondary index relationships. The data set is used as input to the Prefix Resolution utility. This data set can reside on a tape or direct access device. DISP=KEEP must be specified since this data set may be involved in restart processing. The user must specify RECFM=VB, LRECL=900, and BLKSIZE on this DD statement. For further information on work file space requirements, see "Work Data Set Allocation" in the chapter "Data Base Reorganization/Load Processing" in IMS/VS Primer. All references to a particular DFSURWF1 data set must occur within a complete execution of the UCF

(interruption and restart are considered parts of a complete execution).

DFSURWF2 DD
Defines the intermediate sort work data set. This data set can reside on a tape or direct access device. Its size is approximately the same as that of the data set defined by the DFSURWF1 DD statement. The DCB parameters specified in the program are RECFM=VB and LRECL=900. BLKSIZE must be specified on this DD statement.

DFSURWF3 DD
Defines the output work data set that contains all update data for segments involved in logical relationships for that particular execution. This data set is created by the Prefix Resolution utility and is used by the Prefix Update utility. It can reside on a tape or direct access device. The DCB parameters are defined in the program are RECFM=VB and LRECL=900. BLKSIZE must be specified on this DD statement. DISP=KEEP must be specified for restart purposes in the event that the UCF terminates while running the Prefix Update utility.

DFSURIDX DD
Defines an output work data set that will be used if secondary indexes are present in the DBDs being processed. The requirements for this data set are the same as those described above for DFSURWF3. This DD statement is required only if secondary indexes are present. DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on the DFSURIDX DD statement.

The following DD statement is required if REQUEST=EXTRACT was specified for a secondary index reorganization unload.

DFSEXTDS DD
Used to create an unloaded version of those records extracted from a shared secondary index as specified in control statements. This optional DD statement is required only if "E" (REQUEST=EXTRACT) is specified on the FUNCTION=RU control statement. The DCB attributes are determined dynamically, depending on the type of output device and the VSAM LRECLs used. Standard labels must be used.

The following DD statements are required if the job execution uses the Change Accumulation utility.

DFSULOG (or ddname coded for LOGIN operand)|DD
Defines the IMS/VS system log input data set. This data set contains the change records to be accumulated. It can reside on a tape or direct access device. The DCB parameters for this data set are defined when the data set is created.

DFSUDD1 (or ddname coded for LOGOUT operand)|DD
Defines the new log data set. This data set can reside on a tape or direct access device. Standard labels must be used for tape. The DCB is defined with RECFM=VB, LRECL=300, and BLKSIZE=1008. This can be specified as DD DUMMY if no DBNAME operands include a RECID=1 parameter in the input control statements.

DFSUCUMN (or ddname coded for CUMOUT operand)|DD
Defines the new accumulated change output data set. This data set can reside on a tape or direct access device. Standard labels must be used for tape. The DCB is defined with RECFM=VB. BLKSIZE and LRECL specifications are determined by the Change Accumulation utility and are based on the type of output device selected. This can be specified as DD DUMMY if no DBNAME operands include a RECID=0 parameter in the input control statements.

DFSUCUM0 (or ddname coded for CUMIN operand)|DD
Defines the old accumulated change input data set that is to be merged with the log input data in order to create the new accumulated change data set. If old changes are not to be merged, use the following definition: DD DUMMY,DCB=BLKSIZE=100.

The following DD statements are required if the job execution uses the Data Base Recovery utility.

DFSUDUMP (or ddname coded for INDDS operand) DD
Defines the image copy input data set. It can be a data set created by either the batch Data Base Image Copy utility or the HISAM Reorganization Unload utility. If no image copy or HISAM unload copy input is supplied, this statement must be coded as DD DUMMY.

DFSUCUM (or ddname coded for CUMIN operand) DD
Defines the accumulated change input data set. If no accumulated change input is supplied, this statement must be coded DD DUMMY. If no image copy or HISAM unload copy input is supplied, no accumulation change input can be supplied.

The following DD statements are required if the job execution uses the track recovery option of the Data Base Recovery utility.

DFSTRCV DD
Defines a data set that was involved in an I/O error and that is to be used by IEHATLAS when replacing records involved in a track recovery. This DD statement must be provided if REQUEST=T is specified on a FUNCTION=RV control statement. The DD statement must have a DSNAME that is defined in a Format 1 data set control block (DSCB) and must specify the tracks that contain the error(s).

RECON1 DD
Defines the first DBRC RECON data set.

Note: DBRC is not supported for UCF restart processing with DSLOG input. If DBRC is active during UCF restart processing with DSLOG input, the results may be unpredictable.

RECON2 DD
Defines the second DBRC RECON data set.

Note: DBRC is not supported for UCF restart processing with DSLOG input. If DBRC is active during UCF restart processing with DSLOG input, the results may be unpredictable.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, these RECON data set ddnames should not be used.

DD statements are also required to define all data bases referenced by the functional utilities during this execution, and all output data files created during this execution (for example, Reorganization Unload and batch Image Copy). Input files used by the Data Base Recovery and Reorganization Reload utilities must also be defined. Figure 79 summarizes the use of the FUNCTION keyword on the various DD statements.

	FUNCTION=Keywords Used on Utility Control Statements																		
	*UCF	OP	CA	DR	DU	DX	IL	IM	PR	PU	RR	RU	RV	SN	SR	SU	SX	ZB	ZM
IMS	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DFSPRINT	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DFSYSIN	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DFSNJRNL	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DFSOJRNL	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
DFSNCDS	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DFSOCDS	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
DFSRDER	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
DFSCNTRL	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DFSVSAMP				V	V	V	V	V		V	V	V	V	V	V	V	V	V	
SYSOUT			R			R			R										
SORTLIB			R			R			R										
SORTWKnn			R			R			R										
DFSURWF1				R		R	R		O					R					
DFSURWF2						R			R										
DFSURWF3						R			R	R									
DFSURIDX												R							
DFSEXTDS												E							
DFSULOG			O										R						
DFSUDD1			C																
DFSUCUMN			O																
DFSUCUMO			C																
DFSUDUMP													O						
DFSUCUM													O						
DFSTRCV													T						
Data Set DDs				R	R	R	R	R		R	R	R	R	R	R	R	R	R	

R - Required
D - Specify as DD DUMMY when not restarting
V - Required for VSAM organized files
T - Required if Track Recovery option specified
E - Required if REQUEST=EXTRACT option specified
O - Required but can override the ddnames

* These DD statements apply to execution of the UCF.

Figure 79. JCL DD Statements Summary Table

UTILITY CONTROL STATEMENTS

The UCF control statements can be coded in a freeform manner, but at least one valid keyword must appear on each statement. The keywords must be separated by a comma. A statement can be continued by punching a nonblank character in position 72 and beginning the following statement anywhere before position 72. A complete control statement is comprised of the first statement plus any continuation statements. For example:

```
FUNCTION=OP
FUNCTION=DX,DBNAME=dbname,OUTDDS=ddname,                    x
   INDDS=ddname
```

In the above example, FUNCTION=OP is one complete statement contained on a single line, and FUNCTION=DX... is one complete statement contained on two lines.

When a parameter allows for several values to be enclosed in parentheses, the values are not positional and can appear in any order. For example:

```
 ,REQUEST=(SUMM,MSGALL)
      ,REQUEST=(MSGALL,SUMM,STATS)
```

Each utility control statement must contain the FUNCTION keyword. This keyword is defined as follows:

FUNCTION=xx

where xx is:

- OP for Option statement
- CA for Change Accumulation utility
- DR for HD Reorganization Reload utility
- DU for HD Reorganization Unload utility
- DX for combined HD Reorganization Unload and Reload utilities
- IL for User's Initial Load Program
- IM for batch Data Base Image Copy utility
- PR for Prefix Resolution utility
- PU for Prefix Update utility
- RR for Secondary Index Reload
- RU for Secondary Index Unload
- RV for Data Base Recovery utility
- SN for Data Base Scan utility
- SR for HISAM Reorganization Reload utility
- SU for HISAM Reorganization Unload utility
- SX for combined HISAM Reorganization Unload and Reload utilities
- ZB for Data Base Zaps
- ZM for Module Zaps

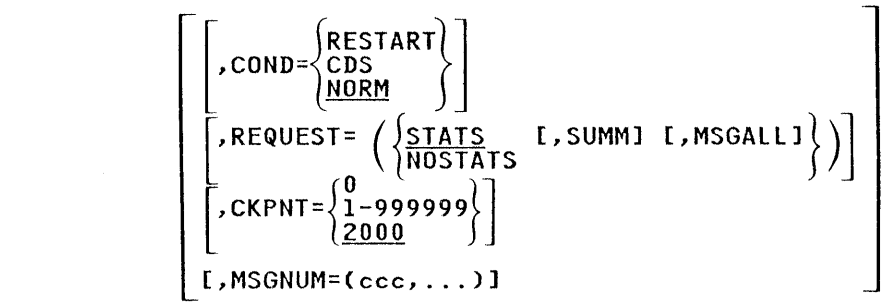
The functions are requests for invocation of the functional utilities and/or the user's initial load of a data base.

The FUNCTION=OP Statement

Because UCF takes the place of Prereorganization, the REQUEST= options from this statement will be used by Prefix Resolution to control the report writing. Therefore, the REQUEST keyword should be coded primarily for Prereorganization/Prefix Resolution, and secondarily for UCF defaults.

This control statement establishes certain UCF run specifications. The format of the statement is:

FUNCTION=OP



where:

COND=
Can only be used once for each UCF run. It indicates the type of processing for this execution of the UCF. COND=RESTART means restart processing is required. COND=CDS builds a control data set from input control statements for purposes of validating data requests. COND=NORM is the default and is used to indicate normal processing.

REQUEST=(STATS or NOSTATS)
STATS specifies that statistics are to be printed after each functional utility completes processing. STATS is the default. NOSTATS specifies that statistics are not to be printed. If REQUEST=NOSTATS is specified for the initial execution of UCF, then all subsequent restart steps must also specify REQUEST=NOSTATS. This parameter is also used by Prefix Resolution to control its output report.

REQUEST=SUMM
Specifies that a summary of the statistics is to be printed after the functional utility completes processing. This parameter is also used by Prefix Resolution to control its output report.

REQUEST=MSCALL
Specifies that all "A" and "W" type messages are to be set as abend requests. This parameter is used only as a diagnostic aid.

CKPNT=
Specifies the checkpoint interval to be used as the default specification during the UCF run. The value specified must be between 0 and 999999. All function control statements that do not have the CKPNT keyword default to this value. If CKPNT is not specified for FUNCTION=OP, 2000 is used as the default CKPNT value on all UCF control statements. CKPNT=0 means that checkpoints are not to be taken for this function. See the appropriate control statement for the default values.

MSGNUM=

Is used to set the error point abend flags. The value "ccc" is the message number extracted from the message identifier "DFSccc I" and is used to set a condition that causes an abend if this message is to be issued during the ensuing processing.

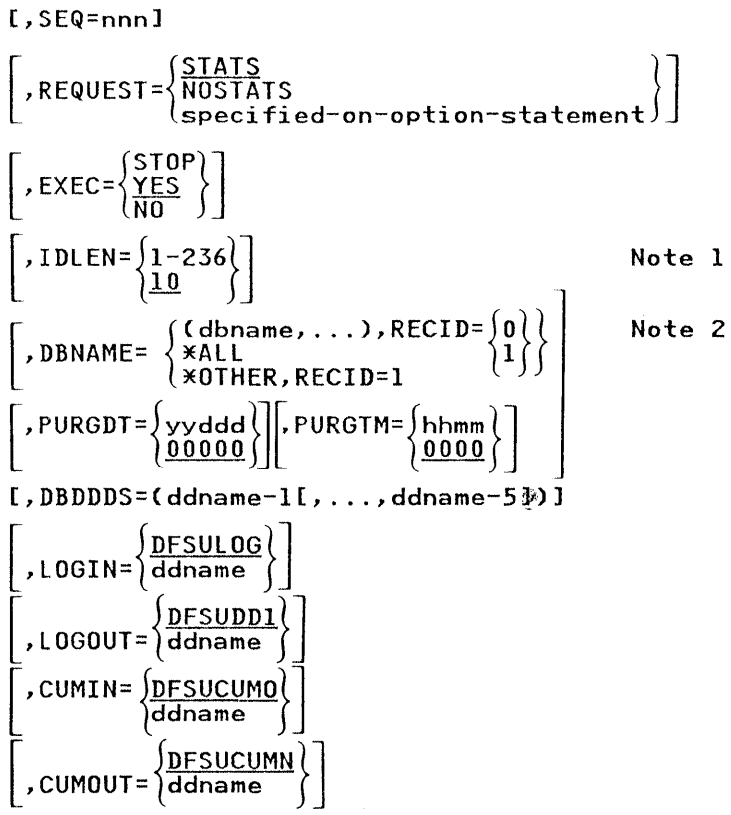
The FUNCTION=CA Statement

This control statement causes the UCF to execute the Data Base Change Accumulation utility (DFSUCUM0). The purpose of this utility is to provide the Data Base Recovery utility with a sequential data set containing only that information from logs which is necessary for recovery.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=CA control statement is:

FUNCTION=CA



Notes:

1. This keyword can be specified only once per UCF job step.
2. If multiple dbnames are specified for the DBNAME keyword, all related keywords (RECID, PURGDT, PURGTM, and DBDDDS) are applicable to all data bases specified.

where:

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

IDLEN=
Specifies the maximum length of the root-sequence field within log records to be processed. The value must be in the range 1 through 236. The default is 10. This keyword corresponds to the "max sequence length" field on the "ID" utility control statement in the Data Base Change Accumulation utility. If there are no ISAM or VSAM KSDS records to be processed, this value should be specified as 4. This keyword can be specified only once for each UCF job step.

DBNAME=
Specifies the data base name(s) or specification corresponding to the data base specification on the DB0 and DB1 utility control statements in the Data Base Change Accumulation utility.

DBNAME=dbname specifies a data base that is to be included in the output accumulation process. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with that name. Specification of up to 5 data set names within a data base is made by specifying the DBDDS keyword on the same control statement. If no DBDDDS keyword is specified, all data sets comprising the data base are used.

DBNAME=*ALL cannot be specified more than once for each UCF run. If DBNAME=*ALL was specified along with RECID=1 (same control statement), no other RECID specification can be made for the UCF run.

DBNAME=*OTHER is valid only for RECID=1 control statements. If DBNAME=*OTHER is specified, no other RECID=1 statements are permitted for the same UCF job step.

RECID
Specifies the type of accumulation to be performed during the Change Accumulation utility.

RECID=0 corresponds to the DB0 utility control statement in the Data Base Change Accumulation utility. All accumulated records of the RECID=0 type are written to the New Change Accumulation data set.

RECID=1 corresponds to the DB1 utility control statement of the Data Base Change Accumulation utility. This statement is used to describe which records are to be written to the New data base log data set. These records are not sorted and are written as read. Any log records that are not data base change records are eliminated.

If RECID is specified, DBNAME must also be specified for that same control statement. If these two keywords are specified for a FUNCTION=CA control statement, PURGDT, PURGTM, and the DBDDDS keyword specifications are permitted.

If RECID is not specified along with DBNAME for any FUNCTION=CA control statement (except the first for this job step), the last RECID specified during this job step is assumed.

If FUNCTION=CA was specified during the UCF job step and the RECID and DBNAME specifications were not, all data base log records are sorted and combined to produce a new Change Accumulation data set.

PURGDT=
Specifies a purge date in the form "yyddd" where "yy" (00 to 99) is the year and "ddd" (000 to 365) is the day of the year. All records matching the RECID and DBNAME specifications and dated before the purge date will be eliminated. The PURGTM keyword is used to specify a precise time of day along with the purge date. The default for PURGDT, if not specified, is 00000.

PURGTM=
Specifies a purge time in the form "hhmm" where "hh" (00 to 23) is the hour of the day and "mm" (00 to 59) is the minute of the hour. This specification should be used along with PURGDT if the record accumulation process requires only a portion of a day's transaction records. All records matching the RECID and DBNAME specifications and dated before the purge time will be eliminated. The default for PURGTM, if not specified, is 0000.

DBDDS=
Specifies from 1 to 5 data sets to be used to limit the accumulation process within a data base. If not specified, all data sets of the data base are used for accumulation. Selection of more than five data sets can be accomplished by using multiple control statements for the same data base.

LOGIN=
Defines the ddname of the IMS/VS system log input for Change Accumulation. If a ddname is not specified, a default of DFSULOG is assumed.

LOGOUT=
Defines the ddname of the new data base log data set to be created on a Change Accumulation run. If a ddname is not specified, a default of DFSUDD1 is assumed.

CUMIN=
Defines the ddname on the Change Accumulation data set used as input to the referenced utilities. If a ddname is not specified, a default of DFSUCUM0 is assumed.

CUMOUT=
Defines the ddname of the new Change Accumulation data set. If a ddname is not specified, a default of DFSUCUMN is assumed.

OUTDDS=
Allows the user to verify that all sort-work-file data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the sort-work-files specified, a message is issued and restart preparation begins to terminate the UCF.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=DR Statement

This control statement causes the UCF to execute the HD Reorganization Reload utility (DFSURGL0) to reload an HDAM, HIDAM, or HISAM data base from a data set created by the HD Reorganization Unload utility (DFSURGU0).

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=DR control statement is:

FUNCTION=DR

```
 ,DBNAME=dbname
[,INDDS={DFSUINPT      }]
        {reload-file-ddname}
[,SEQ=nnn]
[,EXEC={STOP}]
       {YES }
       {NO  }
[,CKPNT={0                                 }]
        {1-999999                          }
        {specified-on-option-statement     }
        {2000                              }
[,EXITRTN=exit-module-name]
[,DBDDS=(ddname,...)]
[,WFlDDS={DFSURWF1}]
         {ddname   }
```

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

INDDS=
Specifies a ddname for the input data set containing the data to be reloaded. This is the data set specified by the OUTDDS keyword of the FUNCTION=DU control statement (HD Reorganization Unload utility). The INDDS keyword must not specify a data base name and can be specified only once for this control statement. If a ddname is not specified, a default of DFSUINPT is assumed.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during the execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

DBDDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

WF1DDS=
Defines that output work data set that will be supplied as an input to the Prefix Resolution utility. The data set can reside on tape or a direct access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

The FUNCTION=DU Statement

This control statement causes the UCF to execute the HD Reorganization Unload utility to unload an HDAM, HIDAM, or HISAM data base to a QSAM-formatted data set. If the DBD has both changed and contains logical relationships, UCF execution must be stopped after the unload function if one of the following conditions exist.

- Either counter, LT, or LP pointers are changed.
- Adding or deleting segments such that the level of segments involved in logical relationships change.
- The DBD name is changed and the DBD contains logical relationships.

To stop UCF execution, use the EXEC=STOP parameter on the last unload function. The new DBD must then be generated followed by a new UCF execution (not a restart) to do the reload.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=DU control statement is:

FUNCTION=DU

```
    ,DBNAME=dbname
    ,OUTDDS= [{DFSURGU1 }] [{ DFSURGU2 }]
             [{ddname-1 }] [{,ddname-2 }]
  [,SEQ=nnn]
  [,REQUEST= {STATS                          }]
             {NOSTATS                        }
             {specified-on-option statement  }
  [,EXEC={STOP}]
         {YES }
         {NO  }
   ,CKPNT={0                                 }
          {1-999999                          }
          {specified-on-option-statement     }
          {2000                              }
  [,EXITRTN=exit-module-name]
  [,DBDDS=(ddname,...)]
```

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

OUTDDS= [(DFSURGU1 ddname-1) (DFSURGU2 , ddname-2)]
Specifies up to 2 ddnames that define the primary and secondary data sets for the HD Reorganization Unload utility. These data sets can reside on either tape or direct access devices. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.) If multiple DU functions are included in one execution of UCF, each must specify a unique output data set. If ddname-1 is not specified, a default name of DFSURGU1 is assumed. If ddname-2 is not specified, a default name of DFSURGU2 is assumed.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a maching-sequence identifier.

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

Note: EXEC=STOP must be specified when reorganizing a VSAM data base. This is further explained under "Restrictions" earlier in this chapter.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

DBDDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

The FUNCTION=DX Statement

This control statement causes the UCF to execute the HD Reorganization Unload and Reload utilities for data base reorganization as described for FUNCTION=DR and FUNCTION=DU.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=DX control statement is:

FUNCTION=DX

```
 ,DBNAME=dbname
[,OUTDDS=(([{IDFSURGU1}][{DFSURGU2 }]))]
         [{ddname-1 }][{,ddname-2}]
[,INDDS={DFSUINPT}]
        {ddname   }
[,SEQ=nnn]
[,EXEC={STOP}]
       {YES }
       {NO  }
[,CKPNT={0                                 }]
        {1-999999                          }
        {specified-on-option-statement     }
        {2000                              }
[,EXITRTN=unload-exit-module-name]
[,EXITRLD=reload-exit-module-name]
[,DBDDDS=(ddname,...)]
[,WF1DDS={DFSURWF1}]
         {ddname   }
```

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

OUTDDS=
Specifies up to 2 ddnames that define the primary and secondary data sets for the HD Reorganization Unload utility. These data sets can reside on either tape or direct access devices. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.) If multiple DX functions are included in one execution of UCF, each must specify a unique output data set. If ddname-1 is not specified, a default name of DFSURGU1 is assumed. If ddname-2 is not specified, a default name of DFSURGU2 is assumed.

INDDS=
Specifies the input data set containing the data to be reloaded. This ddname must correspond to the ddname specified for the OUTDDS keyword during the unload portion of the data base reorganization. The data set must reside on tape or a direct access device. If a ddname is not specified, a default name of DFSUINPT is assumed.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

Notes:

1. The EXEC=STOP parameter remains in effect for the entire control statement. A STOP will be performed for both the unload and reload functions, requiring a restart to continue executing subsequent functions.

2. EXEC=STOP must be specified when reorganizing a VSAM data base. This is further explained under "Restrictions" earlier in this chapter.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified for function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control at the unload phase of this reorganization to allow the user to examine records or compile statistics. The unload-exit-module-name must reside in a library known to this function.

EXITRLD=
Specifies an exit routine to be given control at the reload phase of this reorganization to allow the user to examine records or compile statistics. The reload-exit-module-name must reside in a library known to this function.

DBDDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

WF1DDS=
Defines the output work data set that will be supplied as an input to the Prefix Resolution utility. The data set can reside on tape or a direct access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

The FUNCTION=IL Statement

This control statement attaches a user-supplied Initial Load program. Restart capabilities are available for these programs under the UCF. In most cases, only minor changes are required to the programs to provide the proper interface to the UCF. For additional information, see "Initial Load Application Program Considerations" earlier in this chapter.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=IL control statement for the user's Initial Load program is:

```
FUNCTION=IL

    ,DBNAME=dbname

    ,ILPGM=loadpgm

    ,ILPSBNAM=psbname

   [,SEQ=nnn]

   [,EXEC={STOP}]
          {YES }
          {NO  }

   [,CKPNT={0                                  }]
           {1-999999                           }
           {specified-on-option-statement      }
           {2000                               }

   [,DSDDNAM=(ddname,...)]

   [,OUTDDS={DFSURWF1}]
            {ddname  }

   [,DBDDDS=(ddname,...)]

   [,EXITRTN=exit-module-name]
```

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

ILPGM=
Defines the name of the user's initial load program to be attached. This module must reside in the LINKLIB or in a JOBLIB or STEPLIB data set or must reside in a LINKPACK area because it is located by BLDL and ATTACH commands.

ILPSBNAM=
Defines the name of the PSB that is to be used by the user's initial load program. This PSB must reside in the data set defined by the IMS/VS DD statement. This keyword must be specified once on each complete IL control statement; it cannot be specified more than once for each control statement.

SEQ=
Links the ZAP functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of roots loaded in the data base and must be between 1 and 999999.

If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default for this function. CKPNT=0 means that checkpoints are not to be taken for this function.

DSDDNAM=
Allows the user's initial load function to verify that all data sets are defined before executing the program. The UCF executes a DEVTYPE macro against this ddname; if there is not a DD statement for this data set, a message is issued and restart prepartion begins to terminate the UCF.

OUTDDS=
Defines the input work data set that will be supplied as an input to the Prefix Resolution utility. The data set can reside on tape or a direct access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

DBDDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

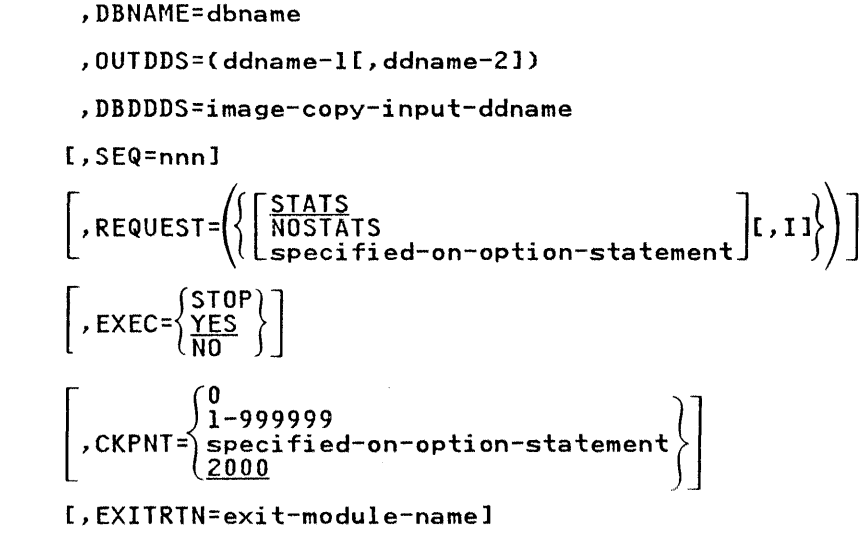
The FUNCTION=IM Statement

This control statement causes the UCF to execute the batch Data Base Image Copy utility (DFSUDMP0) to create an output copy of the data sets within a data base.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=IM control statement is:

FUNCTION=IM



where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

OUTDDS=
Specifies up to 2 copies for the Image Copy output data set. The ddname-1 defines the primary output data set; ddname-2 defines the secondary output data set. They can reside on either tape or direct access devices. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to be normal end of job.)

DBDDDS=
Specifies the Image Copy input data set that is to be dumped. This data set must reside on a direct access device and must be specified once per control statement. If REQUEST=I is specified, the data set must be a KSDS.

SEQ=
Links the zap functions (ZB and ZM) to this functional control statement by using a matching-sequence identifier.

REQUEST=

STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

REQUEST=I specifies an image copy of an index of a KSDS. If specified, the DBDDDS keyword value must refer to a KSDS. The only recovery that can be used with this Image Copy is a track recovery.

EXEC=

Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=

Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=

Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=PR Statement

This control statement causes the UCF to execute the Data Base Prefix Resolution utility (DFSURG10). This utility accumulates the information generated on work data sets during the load and/or reorganization of one or more data bases. It produces an output data set that includes one or more updated records to be applied to each segment that contains logical relationship information.

The updated records have been sorted in physical-location order by data base and segment. The prefix fields that are updated include the logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents. This program optionally produces an output data set that contains information necessary to create and/or update secondary index data bases.

The STATS and/or SUMM reports are controlled by the REQUEST= keyword on the FUNCTION=OP statement.

This statement is optional, because it is automatically generated by the UCF for normal processing.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=PR control statement is:

```
FUNCTION=PR

     [,INDDS={DFSURWF1}]
             {ddname  }
     [,SEQ=nnn]
     [,EXEC={STOP}]
            {YES }
            {NO  }
     [,DBNAME=dbname]
     [,EXITRTN=exit-module-name]
     [,OUTDDS=(sort-work-file,...)]
```

where:

INDDS=
Specifies that the DFSURWF1 or user-defined data set generated by the Data Base Scan utility is to serve as one of the inputs to the Data Base Prefix Resolution utility.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC
Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name. The function will be performed for all data bases on the work file whether the DBNAME parameter is coded or not.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

OUTDDS=
Allows the user to verify that all sort-work-file data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the sort-work-files specified, a message is issued and restart preparation begins to terminate the UCF.

The FUNCTION=PU Statement

This control statement causes the UCF to execute the Data Base Prefix Update utility (DFSURGP0). This utility uses the output generated by the Prefix Resolution utility to update the prefix of each segment whose prefix information was affected by a data base load and/or reorganization.

This statement is optional, because it is automatically generated by the UCF for normal processing.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=PU control statement is:

FUNCTION=PU

```
[,SEQ=nnn]

[,EXEC={STOP}]
       {YES }
       {NO  }

[,CKPNT={0                                  }]
        {1-999999                           }
        {specified-on-option-statement      }
        {2000                               }

[,DBNAME=(dbname1,dbname2,...)]

[,EXITRTN=exit-module-name]

[,INDDS={DFSURWF3                           }]
        {ddname-from-Prefix-Resolution-run  }

[,DBDDS=(ddname,...)]
```

where:

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of input work records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

DBNAME=
Specifies the data base required by the UCF for execution purposes and to set up parameters for the WTOR. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name. This function will be performed for all data bases on the work file whether the DBNAME parameter is coded or not.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

INDDS=
Used to specify the ddname of the input data set. The default (WF3 data set) applies to this particular UCF run. If a Prefix Resolution run was done outside of this UCF run, the data set could have some other ddname (whatever the user specified).

DBDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart processing begins to terminate the UCF.

The FUNCTION=RR Statement

This control statement causes the UCF to execute the HISAM Reorganization Reload utility (DFSURRL0) to create, merge, or replace a member in a secondary index.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=RR control statement is:

FUNCTION=RR

```
 ,DBNAME=dbname
[,KDSDD=ddname]
 ,INDDS={DFSUIN01                          }
        {(reload-file-ddname,...)          }
[,DBDDDS=ddname]
[,SEQ=nnn]
[,REQUEST={STATS                           }]
          {NOSTATS                         }
          {specified-on-option-statement   }
[,EXEC={STOP)}]
       {YES  }
       {NO   }
[,CKPNT={0                                 }]
        {1-999999                          }
        {specified-on-option-statement     }
        {2000                              }
[,EXITRTN=exit-module-name]
```

where:

DBNAME=
Specifies the data to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

KDSDD=
Specifies a keyed data set ddname that is to be operated on. This keyword is used to verify the presence of the DD statement for the keyed data set.

INDDS=
Specifies the input data set containing the data to be reloaded. This ddname must correspond to the ddname specified for the OUTDDS keyword during the unload portion of the data base reorganization. If this keyword is omitted, the default is DFSUIN01. The data set may reside on either tape or a direct access device.

IBDDS=
Defines the ddname of the overflow (ESDS) data base data set required by this function. Only one DBDDDS can be specified for each control statement.

SEQ=
Links the zap functions (ZB and ZM) to this functional control statement by using a matching-sequence identifier.

REQUEST=
STATS specifies statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CHKPNT=0 means that checkpoints are not to be taken for this function.

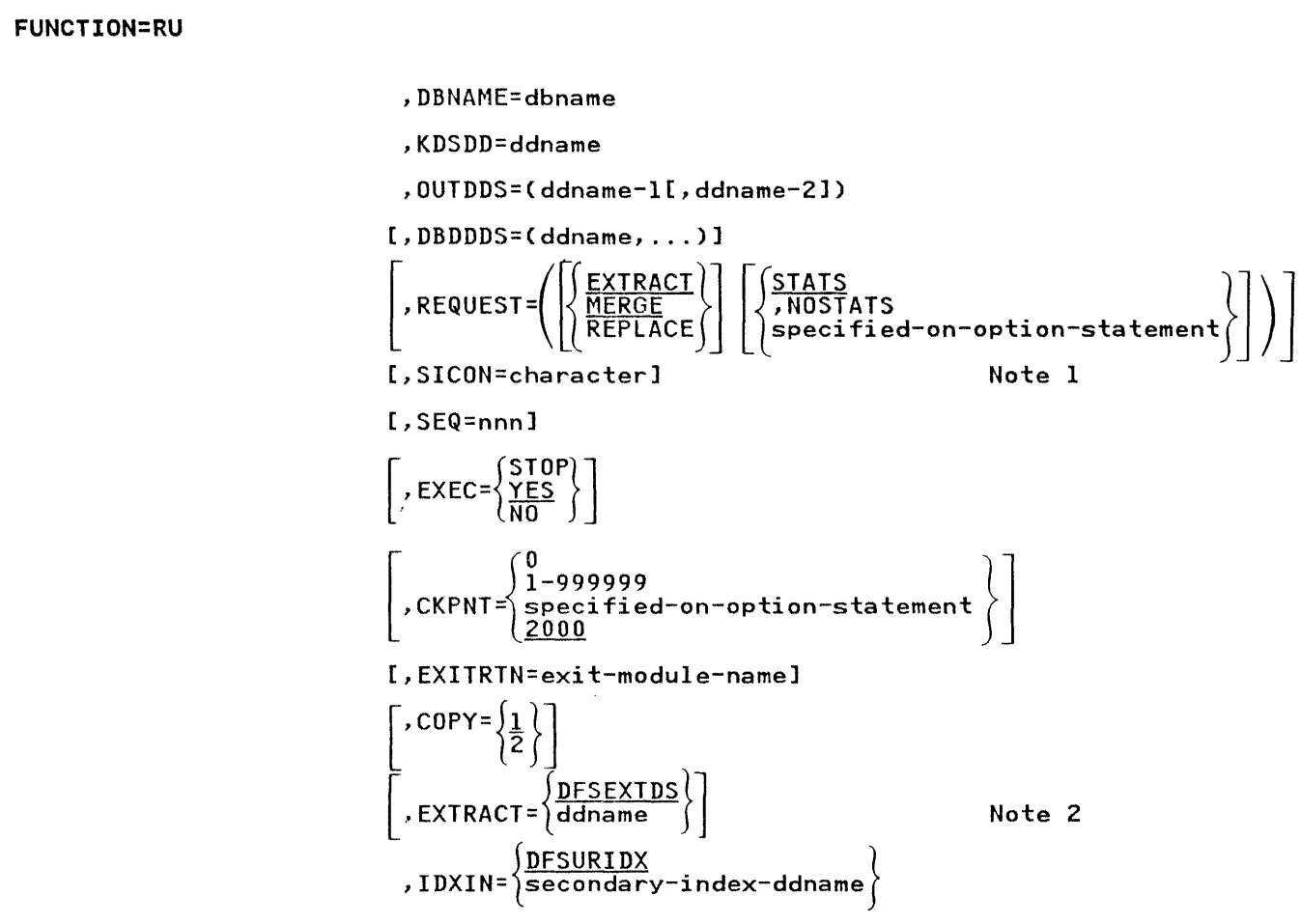
EXITRTN=
Specifies an exit routine to be given control in order to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=RU Statement

This control statement causes the UCF to execute the HISAM Reorganization Unload utility (DFSURUL0) to create an input work data set to the HISAM Reorganization Reload utility.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=RU control statement is:



Notes:

1. Required if either REQUEST=EXTRACT or REQUEST=REPLACE is specified.

2. Can only be specified with REQUEST=EXTRACT.

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

KDSDD=
Defines the VSAM KSDS or ISAM data set of the data base to be reorganized. The ddname must be the same as the name in the DBD that describes this data set. This keyword is used with the DBDDDS keyword for each reorganization of the HISAM data base.

OUTDDS=
Specifies up to 2 copies for the Secondary Index output data set. The ddname-1 defines the primary output data set. If ddname-2 is specified, the COPY keyword must have a value of 2. These copy data sets can reside on either tape or direct access devices.

DBDDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

REQUEST=STATS or MERGE or REPLACE
EXTRACT will extract a secondary index (as defined by the SICON keyword) from either a shared index data base or from the index work data set from Prefix Resolution. If REQUEST=EXTRACT is specified, the SICON and EXTRACT keywords are required.

MERGE will merge the index work data set created during either data base initial load or reorganization (through use of the Prefix Resolution utility) into an existing secondary index, or create a secondary index if the data set does not exist. MERGE is the default.

REPLACE will replace those segments in the secondary index that match the character constant specified by the SICON keyword with matching segments found in the work data set. If the secondary index segments being replaced need to be saved, an extract operation should be performed prior to the replace. The SICON keyword is required for a replace operation.

REQUEST=STATS or NOSTATS or specified-on-option-statement
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

SICON=
Specifies a 1-byte constant defined in the DBD generation of the shared secondary index. This keyword is required if REQUEST=EXTRACT or REQUEST=REPLACE is defined on this control statement.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that a checkpoint is not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

COPY=
Provides for multiple copies of output data sets. COPY can be specified only once per control statement. COPY=1 produces only 1 copy and is the default. COPY=2 produces an additional output copy and should be specified if a ddname-2 is specified for the OUTDDS keyword. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.)

EXTRACT=
Defines the ddname of the EXTRACT data base data set and must be specified if REQUEST=EXTRACT is used. DFSEXTDS is the default if no ddname is specified.

IDXIN=
Describes the output data set (DFSURIDX) from the Prefix Resolution utility that contains secondary index information. This keyword is required and can be specified only once for each control statement. This keyword corresponds to the index DD statement in the HISAM Reorganization Unload utility. DFSURIDX is the default if no ddname is specified.

The FUNCTION=RV Statement

This control statement causes the UCF to execute the Data Base Recovery utility to restore a physically damaged data set within an IMS/VS data base. (This utility does not provide recovery from application logic errors; it is the user's responsibility to ensure the integrity of the data in the data base.)

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=RV control statement is:

FUNCTION=RV

```
   ,DBNAME=dbname

   ,DBDDDS=recovered-data-set-ddname

  [         {DFSUDUMP         }]
  [ ,INDDS= {image-copy-ddname}]

  [         {DFSULOG}]
  [ ,LOGIN= {ddname }]

  [,SEQ=nnn]
  [             (  {  [ {STATS                        } ]            } ) ]
  [ ,REQUEST=   (  {  [ {NOSTATS                      } ] [{T[,I]}]  } ) ]
  [             (  {  [ {specified-on-option-statement} ]            } ) ]

  [        {STOP}]
  [ ,EXEC= {YES }]
  [        {NO  }]

  [         {0                            }]
  [         {1-999999                     }]
  [ ,CKPNT= {specified-on-option-statement}]
  [         {2000                         }]

  [ ,CUMIN= {ddname }]
  [         {DFSUCUM}]

  [,EXITRTN=exit-module-name]
```

(Defaults, underlined in the original: DFSUDUMP, DFSULOG, STATS, YES, 2000, DFSUCUM.)

Note: Only data sets that are stored using VSAM can be recovered with the track recovery option "T." This option is designed to recover only those tracks that are in error within a data set. Whereas full recovery creates a new data set, the track recovery option only modifies one or more tracks in the existing data set. Recovery with this option is usually faster than full recovery.

where:

DBNAME=
: DBNAME specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

DBDDDS=
: Defines the data set to be recovered. This keyword is required for the recovery function and can only be specified once per control statement. This data set must reside on a direct access volume.

INDDS=
Defines the image-copy-input data set. It can be created by either the Data Base Image Copy utility or the HISAM Reorganization Unload utility. The default ddname is DFSUDUMP.

LOGIN=
Defines the ddname of the IMS/VS system log input for Data Base Recovery or the data base log data set that was output from a change accumulation and input for Data Base Recovery. The default ddname is DFSULOG.

SEQ=
Links the zap functions (ZB and ZM) to this functional control statement by using a matching-sequence identifier.

REQUEST= STATS or NOSTATS or specified-on-option-statement
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed.

If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

REQUEST="(,T,I)"
"T" specifies that the track recovery option of the Data Base Recovery utility is to be used. If "T" is not specified, a full data set recovery takes place.

"I" specifies that the track recovery option applies to an index of a KSDS. This parameter is specified along with the "T" parameter; it has no meaning if specified alone.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of log records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

CUMIN=
Defines the change accumulation data set. If this input is not supplied, the statement must be coded DD DUMMY. If an image copy or HISAM unload copy is not supplied, change accumulation input cannot be supplied. The data set can reside on tape or a direct access volume; standard labels must be used. The default data set name is DFSUCUM.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=SN Statement

This control statement causes the UCF to execute the Data Base Scan utility (DFSURGS0). This utility scans nonreorganized data bases that contain logical relationships that are affected by loading and/or reorganizing other data bases. It also generates output work data sets that will be used by the Data Base Prefix Resolution utility.

The SCAN function is executed for initial loads and HD Reloads. When SCAN is not required, the DFSURWF1 DD statement must be present unless EXEC=NO is specified on the FUNCTION=SN statement.

The total number of ddnames specified for this function must not exceed the limit of 20.

This statement is optional, because it is automatically generated by the UCF for normal processing. The format of the FUNCTION=SN control statement is:

FUNCTION=SN

```
[                                                              ]
  ,DBNAME=dbname
 [,DBDDS=(data-base-data-sets,...)]
 [          {DFSURWF1}  ]
 [,OUTDDS={ddname    }  ]
 [,SEGNAME=segment-name]
 [          {SEQ}       ]
 [,SCANTYP={SEG}        ]
 [,SEQ=nnn]
 [          {STATS                                  }       ]
 [,REQUEST={NOSTATS                                 }       ]
 [          {specified-on-option-statement          }       ]
 [          {STOP}      ]
 [,EXEC=    {YES }      ]
 [          {NO  }      ]
 [          {0                                      }       ]
 [          {1-999999                               }       ]
 [,CKPNT=   {specified-on-option-statement          }       ]
 [          {2000                                   }       ]
                                                               ]
  [,EXITRTN=exit-module-name]
```

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

DBDDS=
Allows the user to verify that all data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

OUTDDS=
Defines the output data set that will be used to resolve logical relationships. This data set will be used as an input to the Prefix Resolution utility. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

SEGNAME=
Defines the segment name to be scanned for. This keyword is used in conjunction with a DBNAME= keyword in this same control statement. It can be specified only once for each complete control statement.

SCANTYPE=
Defines the order of scan to be performed on the DBNAME associated with this same control statement. This keyword can be specified only once for each complete control statement. The value "SEQ" means that the order of search is with unqualified GN calls from the beginning of the data base. The value "SEG" means that the order of search is with GN calls qualified on the segment name from the beginning of the data base.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed.

If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of calls that equals the number of root segments read and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=SR Statement

This control statement causes the UCF to execute the HISAM Reorganization Reload utility (DFSURRL0). This utility can be used to reload a HISAM or HIDAM primary index data base from a QSAM-formatted data set created by the HISAM Reorganization Unload utility.

Reorganization of HISAM data bases is significantly faster using the HISAM Unload/Reload utilities instead of the HD Unload/Reload utilities.

The HISAM Unload/Reload utilities cannot be used to make structural changes other than changes to logical record length and block size. The HD Unload/Reload utilities, however, allow structural changes to be made to a data base.

Note: The HISAM Reorganization Unload/Reload utilities cannot be used to reorganize HISAM data bases that are indexed by a secondary index or that contain segments with direct address pointers used in logical relationships. The HD Reorganization Unload/Reload utilities must be used instead.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=SR control statement is:

FUNCTION=SR

```
 ,DBNAME=dbname
          { DFSUIN01                  }
 ,INDDS=  {(reload-file-ddname,...)   }
[,DBDDDS=(ddname,...)]
[,KDSDD=ddname]
[,SEQ=nnn]
[            (  { [ {STATS                         } ]          } )]
[,REQUEST=   (  { [ {NOSTATS                       } ] [,VSAM]  } )]
[            (  { [ {specified-on-option-statement } ]          } )]
[       {STOP}]
[,EXEC= {YES }]
[       {NO  }]
[        {0                            }]
[        {1-999999                     }]
[,CKPNT= {specified-on-option-statement}]
[        {2000                         }]
[,EXITRTN=exit-module-name]
```

(Underlined defaults: DFSUIN01, STATS, YES, 2000.)

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

INDDS=
Specifies the input data set containing the data to be reloaded. This is the data set created from the FUNCTION=SU control statement (HISAM Reorganization Unload utility). If this keyword is omitted, the default is DFSUIN01.

DBDDDS=
Used to verify the existence of a data base data set that is to be reloaded.

KDSDD=
Defines the VSAM KSDS or ISAM output data set to be reloaded. The ddname must be the same as the ddname used for the KDSDD keyword during the HISAM Reorganization Unload (FUNCTION=SU).

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

VSAM
Specifies that the ISAM/OSAM input reload copy is to be reloaded to a VSAM data set.

The following restrictions apply with respect to using the REQUEST=VSAM keyword for a FUNCTION=SR control statement:

- If an ISAM/OSAM data base is unloaded and the REQUEST=VSAM keyword is specified, the output is in VSAM format. The OS/VS Access Method Services utility must have been run to create the required data sets, and all of the necessary DBD changes must have been made before the HISAM Reload utility can be run.
- REQUEST=VSAM can only be used when making a conversion from ISAM/OSAM to VSAM.
- If a VSAM data base is unloaded, the reloaded data base is VSAM regardless of what is specified for the REQUEST= keyword statement. The original data set must be scratched and reallocated with the OS/VS Access Method Services utility, or a new data set name must be created by the Access Method Services utility before the HISAM Reload utility can be run.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records and must be between 1 and 999999. If a CKPNT is not specified, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=SU Statement

This control statement causes the UCF to execute the HISAM Reorganization Unload utility (DFSURUL0). This utility unloads a HISAM data base and creates a reorganized output that can be used as input to either the Data Base Recovery utility or the HISAM Reorganization Reload utility.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=SU control statement is:

FUNCTION=SU

```
   ,DBNAME=dbname

   ,KDSDD=ddname

   ,OUTDDS=(ddname-1[,ddname-2])

  [,DBDDDS=(ddname,...)]

  [,SEQ=nnn]

  [,REQUEST={STATS                          }]
            {NOSTATS                        }
            {specified-on-option-statement  }

  [,EXEC={STOP}]
         {YES }
         {NO  }
         {0   }

  [,CKPNT={1-999999                         }]
          {specified-on-option-statement    }
          {2000                             }

  [,EXITRTN=exit-module-name]

  [,COPY={1}]
         {2}
```

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

KDSDD=
Defines the VSAM KSDS or ISAM data set of the data base to be reorganized. The ddname must be the same as the name in the DBD that describes this data set. This keyword is used with the DBDDDS keyword for each reorganization of the HISAM data base.

OUTDDS=
Specifies up to two copies for the unload data set. The ddname-1 defines the primary output data set; ddname-2 defines the secondary output data set. If ddname-2 is specified, the COPY keyword must have a value of 2. These copy data sets can reside on either tape or direct access devices.

DBDDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed.

If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

Note: EXEC=STOP must be specified when reorganizing a VSAM data base. This is further explained under "Restrictions" earlier in this chapter.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

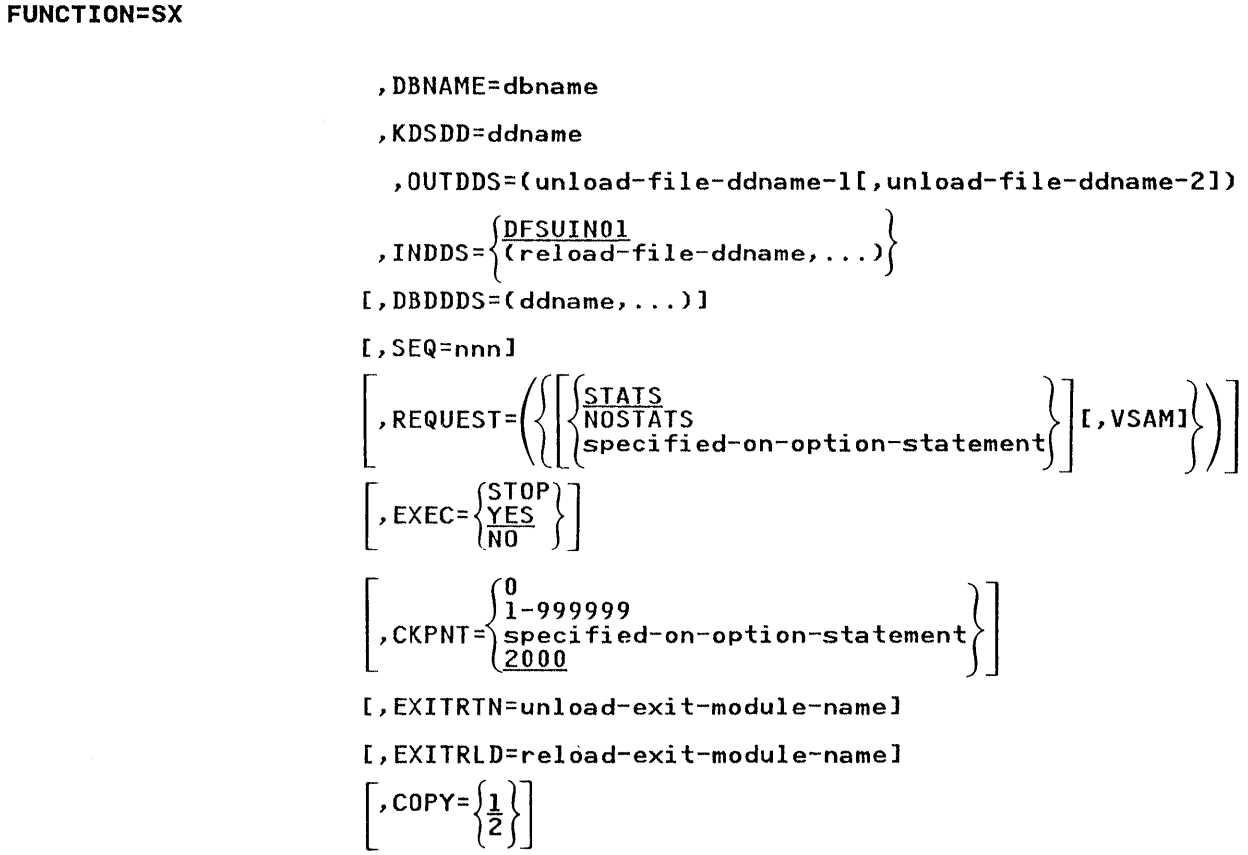
COPY=
Provides for multiple copies of output data sets. COPY can be specified only once for each control statement. COPY=1 produces only one copy and is the default. COPY=2 produces an additional output copy and should be stated if a ddname-2 is specified for the OUTDDS keyword. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.)

The FUNCTION=SX Statement

This control statement causes the UCF to execute the HISAM Reorganization Unload and Reload utilities to reorganize a HISAM data base. The FUNCTION=SX combines the FUNCTION=SU and FUNCTION=SR specifications.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=SX control statement is:



where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

KDSDD=
Defines the VSAM KSDS or ISAM data set of the data base to be reorganized. The ddname must be the same as the name in the DBD that describes this data set. This keyword is used with the DBDDDS keyword for each reorganization of the HISAM data base.

OUTDDS=
Specifies up to two copies for the unload data set. The ddname-1 defines the primary output data set; ddname-2 defines the secondary output data set. If ddname-2 is specified, the COPY keyword must have a value of 2. These copy data sets may reside on either tape or direct access devices.

INDDS=
Specifies the input data set containing the data to be reloaded. This is the data set created from the FUNCTION=SU control statement (HISAM Reorganization Unload utility). If this keyword is omitted, the default is DFSUIN01.

DBDDS=
Allows the user to verify that all data base data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

SEQ=
Links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

VSAM
Specifies that the ISAM/OSAM input reload copy is to be reloaded to a VSAM data set.

The following restrictions apply with respect to using the REQUEST=VSAM keyword for a FUNCTION=SX control statement:

- If an ISAM/OSAM data base is unloaded and the REQUEST=VSAM keyword is specified, the output is in VSAM format. The OS/VS Access Method Services utility must have been run to create the required data sets, and all of the necessary DBD changes must have been made before the HISAM Reload utility can be run.
- REQUEST=VSAM can only be used when making a conversion from ISAM/OSAM to VSAM.
- If a VSAM data base is unloaded, the reloaded data base is VSAM regardless of what is specified for the REQUEST= keyword statement. The original data set must be scratched and reallocated with the OS/VS Access Method Services utility, or a new DSNAME must be created by the Access Method Services utility before the HISAM Reload utility can be run.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

Notes:

1. The EXEC=STOP parameter stays in effect for the entire control statement. A STOP will be performed for both the unload and reload functions, requiring a restart to continue executing subsequent functions.
2. EXEC=STOP must be specified when reorganizing a VSAM data base. This is further explained under "Restrictions" earlier in this chapter.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. The value is equal to the number of root segments and logical records and must be between 1 and 999999. If a CKPNT is not specified, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control at the unload phase of this reorganization to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

EXITRLD=
Specifies an exit routine to be given control at the reload phase of this reorganization to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

COPY=
Provides for multiple copies of output data sets. COPY can be specified only once for each control statement. COPY=1 produces only one copy and is the default. COPY=2 produces an additional output copy and should be stated if a ddname-2 is specified for the OUTDDS keyword. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.)

The FUNCTION=ZB Statement

This control statement causes a zap to data base blocks to correct errors or force abends. Only the VERIFY and REP control statements of the OS/VS2 service aids SPZAP program are supported.

This utility function should be used only by an IBM Field Engineering Program Support Representative or by someone under that person's direction.

The total number of ddnames specified for this function must not exceed the limit of 20.

The format of the FUNCTION=ZB control statement is:

FUNCTION=ZB

```
,DBNAME=dbname

,DBDDDS=ddname

{,VERIFY=nnnnnnnn}                    Note 1
{,REP=nnnnnnnn   }

,VALUE=compare/replace value

,RBNID=dbblock

[,EXEC={STOP}]
       {YES }
       {NO  }

[                 {DU}                    ]
[                 {DR}                    ]
[                 {IM}                    ]
[,RELATE={RV} [,SEQ=nnn]                  ]      Note 2
[                 {SN}                    ]
[                 {SR}                    ]
[                 {SU}                    ]

[,EXITRTN=exit-module-name]
```

Notes:

1. For each complete control statement, the user can specify either the VERIFY or REP keyword; they cannot be specified on the same statement. When specifying the VERIFY keyword on one complete statement and the REP keyword on a separate complete statement, the order of execution is critical; because statements are not automatically sorted by the UCF, make sure that the VERIFY precedes the REP.

   The VERIFY and REP control statements must be included in the same job step. If the REP statement comes in a later step than the VERIFY statement, the REP statement is no longer associated with the VERIFY statement.

2. Use of the RELATE keyword without the SEQ keyword causes the UCF to perform zaps on all of the data bases associated with the related function specified with the RELATE keyword. Use of the SEQ keyword with the RELATE keyword on the same complete control statement restricts the zap to a particular function.

where:

DBNAME=
Specifies the data base to which this function is to be applied. The value "dbname" is the DBD name that exists as a member in the DBD library as well as the actual data base with this name.

DBDDDS=
Defines any ddname of a data base data set required by this function. Only one DBDDDS can be specified per control statement.

VERIFY=
Defines the hexadecimal offset (from the address of the RBA in the RBNID) that the associated VALUE= field is to be compared against. If any verify fails, the entire zap terminates. VERIFY must be coded as 8 hexadecimal digits.

REP=
Defines the hexadecimal offset (relative to zero) that is to be replaced by the associated VALUE= data. REP must be coded as 8 hexadecimal digits.

VALUE=
Defines the data (in hexadecimal representation; for example, C"A" is C1 in the statement) that is to be operated on as a compare field if associated with a VERIFY keyword or as replacement data if associated with a REP keyword. Exactly 8 hexadecimal characters must be specified. This means that 4 bytes of data must be changed or compared for each control statement. This keyword must be specified once for each control statement.

RBNID=
Defines the data base block that is to be operated on by the associated VERIFY= or REP= data (for example, RBNID= 00001000). For ISAM/OSAM-HISAM data sets, this is a relative record number (RRN). For OSAM non-HISAM data sets, this is a relative block number (RBN). For VSAM data sets, this is a relative byte address (RBA) of the first byte in the block. When specified, this keyword must be specified as exactly 8 hexadecimal characters. This value will be packed into a 4-byte field.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

RELATE=
Relates one functional control statement to another. The value "xx" is defined on the related functional control statement and is described under the FUNCTION keyword description. The use of the SEQ keyword further defines the related control statement; if not specified on the same control statement as the RELATE keyword, the zap applies to all of the RELATE specifications comprised in one complete control statement. The RELATE keyword can be used to relate the FUNCTION=ZB control statement to the following functions:

DU	(for DFSURGU0)	SN	(for DFSURGS0)
DR	(for DFSURGL0)	SR	(for DFSURRL0)
IM	(for DFSUDMP0)	SU	(for DFSURUL0)
RV	(for DFSURDB0)		

Sequence=nnn links this zap function to another UCF functional control statement with an identical sequence number. This keyword is not valid if the RELATE keyword is not specified.

EXITRTN=
Allows the user to obtain control at the time the block is in storage to verify that the zap was performed, and, if desired, to compile statistics on the block.

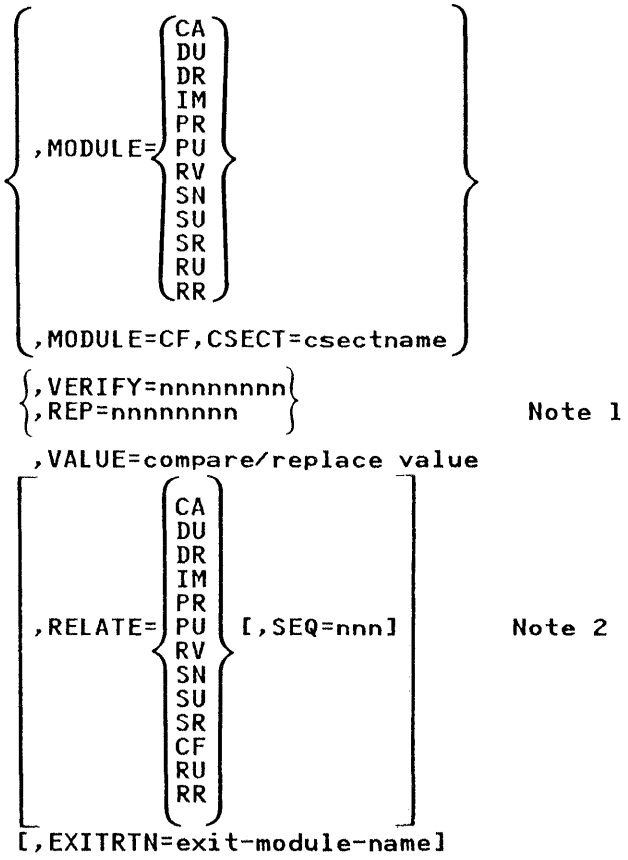
The FUNCTION=ZM Statement

This control statement causes a zap to be applied to certain UCF modules. The zap is applied in storage to correct logic errors or force abends. Only the VERIFY and REP control statements of the OS/VS2 service aids SPZAP program are supported.

This utility function should be used only by an IBM Field Engineering Program Support Representative or by someone under skilled direction.

The format of the FUNCTION=ZM control statement is:

FUNCTION=ZM



Notes:

1. For each complete control statement, the user can specify either the VERIFY or REP keyword; they cannot be specified on the same statement. When specifying the VERIFY keyword on one complete statement and the REP keyword on a separate complete statement, the order of execution is critical; because statements are not automatically sorted by the UCF, make sure that the VERIFY precedes the REP.

2. Use of the RELATE keyword without the SEQ keyword causes the UCF to perform zaps on all modules specified with the RELATE keyword. Use of the SEQ keyword with the RELATE keyword on the same complete control statement restricts the zap to a particular function.

where:

MODULE=
Defines the load module to be zapped. This keyword can be specified only once for each control statement and must specify one of the following modules:

CA	(for DFSUCUM0)	RV	(for DFSURDB0)
DU	(for DFSURGU0)	SN	(for DFSURGS0)
DR	(for DFSURGL0)	SU	(for DFSURUL0)
IM	(for DFSUDMP0)	SR	(for DFSURRL0)
PR	(for DFSURG10)	CF	(for DFSUCF00)
PU	(for DFSURGP0)	RU	(for DFSURUL0)
		RR	(for DFSURRL0)

CSECT=
Defines the CSECT within the load module that is to be zapped. If this keyword is omitted, the load module is changed in the CSECT containing the entry point and is relative to address zero as the entry point offset. (Refer to the IMS/VS SYSGEN listings for the valid CSECT names of the particular load module.) This keyword is valid only with MODULE=CF and is ignored in all other instances.

VERIFY=
Defines the hexadecimal offset (from the address of the RBA in the RBNID) that the associated VALUE= field is to be compared with. If any verify fails, the entire zap terminates. VERIFY must be coded as 8 hexadecimal digits.

REP=
Defines the hexadecimal offset (relative to zero) that is to be replaced by the associated VALUE= data. REP must be coded as 8 hexadecimal digits.

VALUE=
Defines the data (in hexadecimal representation; for example, C"A" is C1 in the statement) that is to be operated on as a compare field if associated with a VERIFY keyword or as replacement data if associated with a REP keyword. Exactly 8 hexadecimal digits must be specified. This means that 4 bytes of data must be changed or compared for each control statement. The VALUE keyword must be specified once for each control statement.

RELATE=
Relates this zap statement to another UCF control statement. This keyword can be specified only on a zap statement.

The value "xx" is defined on the related control statement and is described under the FUNCTION keyword description. The use of the SEQ keyword helps to further define the related functional control statement; if not specified on the same control statement as the RELATE keyword, the zap applies to all of the RELATE specifications comprised in one complete control statement. The RELATE keyword can be used to relate the FUNCTION=ZM control statement to the following functions:

CA	(for DFSUCUM0)	RV	(for DFSURDB0)
DU	(for DFSURGU0)	SN	(for DFSURGS0)
DR	(for DFSURGL0)	SU	(for DFSURUL0)
IM	(for DFSUDMP0)	SR	(for DFSURRL0)
PR	(for DFSURG10)	CF	(for DFSUCF00)
PU	(for DFSURGP0)	RU	(for DFSURUL0)
		RR	(for DFSURRL0)

SEQ=
Links this zap function to another UCF functional control statement with an identical sequence number. This keyword is not valid if the RELATE keyword is not specified.

EXITRTN=
Allow the user to obtain control at the time the zap has been executed.

KEYWORDS SUMMARY TABLES

Figure 80 on page 365 summarizes the use of the keywords with the different functional utilities. The "Multiplicity" column defines the number of times each keyword can appear for each execution of the UCF. The UCF uses a work area for each function statement to examine keyword values. If this fixed-length work area becomes full, the UCF issues a DFS385A message.

KEYWORD	Multi-plicity	FUNCTIONS																	
		OP	CA	DR	DU	DX	IL	IM	PR	PU	RR	RU	RV	SN	SR	SU	SX	ZB	ZM
FUNCTION	n	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SEQ	n	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
COND	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
REQUEST	n	n	2	n	n	n	*	n	n	n	n	n	n	n	n	n	n	*	*
EXEC	n	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	*
CKPNT	n	1	*	1	1	1	1	1	*	1	1	1	1	1	1	1	1	*	*
IDLEN	1	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RECID	n	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
DENAME	n	*	n	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	*
RELATE	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	1
PURGDT	n	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
KDSDD	n	*	*	*	*	*	*	*	*	*	1	1	*	*	1	1	1	*	*
CUMOUT	1	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
CUMIN	n	*	1	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*
ILPGM	n	*	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*
DSDDNAM	n	*	*	*	*	*	d	*	*	*	*	*	*	*	*	*	*	*	*
OUTDDS	n	*	d	*	2	2	1	2	d	*	*	2	*	d	*	2	2	*	*
INDDS	n	*	*	1	*	1	*	*	1	1	1	*	1	*	d	*	d	*	*
LOGIN	n	*	1	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*
LOGOUT	1	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
EXITRTN	n	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SEGNAME	n	*	*	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*
SCANTYP	n	*	*	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*
DBDDS	n	*	5	d	d	d	d	1	*	d	1	1	1	d	1	1	1	1	*
COPY	n	*	*	*	*	*	*	*	*	*	*	1	*	*	*	1	1	*	*
EXTRACT	n	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*	*
IDXIN	n	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*	*
ILPSBNAM	n	*	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*
MODULE	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1
CSECT	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1
VERIFY	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	e	e
REP	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	e	e
VALUE	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	1
MSGNUM	n	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
SICON	n	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*	*
PURGTM	n	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RBNID	n	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	*
EXITRLD	n	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	1	*	*
WF1DDS	n	*	*	1	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*

where:

* means this keyword cannot be used in this function.

n means this keyword can be used any number of times.

d means this keyword can be used any number of times up to the limit of 20; the limit of 20 is determined by adding all "d" ddnames per functional request.

e means that this keyword can be used only one time and only if no other keyword with an "E" designation in this chart has not also been specified on the same control statement. (It is not permissible to specify the VERIFY and REP keywords on the same control statement.)

1,2...5 means this keyword can be used that number of times as a maximum.

Figure 80. UCF FUNCTION Summary Table

Figure 81 shows the minimum keywords that a user must specify when constructing each type of control statement. The presence of an "r" means that the keywords are required.

KEYWORD	FUNCTIONS																	
	OP	CA	DR	DU	DX	II	IM	PR	PU	RR	RU	RV	SN	SR	SU	SX	ZB	ZM
FUNCTION	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
DBNAME			r	r	r	r	r			r	r	r	r	r	r	r	r	
KDSDD											r				r	r		
ILPGM						r												
OUTDDS							r				r				r	r		
INDDS																		
LOGIN																		
SEGNAME													r					
DBDDDS							r					r					r	
ILPSENAM						r												
MODULE																		r
VERIFY																	r	r
REP																	r	r
VALUE																	r	r
RBNID																	r	

Figure 81. UCF-Required Keywords by Function

RETURN CODES

Upon termination of the UCF, the following return codes are passed to register 15.

Code	Meaning	Action
0	Processing was normal.	DFSNJRNL and DFSNCDS data sets can be scratched.
4	Warning messages have been issued, or termination was caused because EXEC=STOP was specified.	Verify that warnings are not errors; if no errors exist take same action as for return code 0. If restart is desired because of errors or EXEC=STOP, the DFSNJRNL and DFSNCDS must have been saved.
8	Serious errors have occurred. UCF terminated.	Correct errors and begin restart processing. The DFSNJRNL and DFSNCDS data sets must be available as DFSOJRNL and DFSOCDS on restart.
12	UCF failed and terminated with an unrecoverable error.	Correct the error and start the UCF from the beginning Do not attempt restart processing The DFSNJRNL and DFSNCDS data sets can be scratched because restart is not possible.

If the UCF terminates with a nonzero return code, all data sets that were in use at the time of termination may be required for restart. The following data sets must be saved under certain conditions:

DFSURWF1:
This data set is created during Initial Load, HD Reorganization Reload, and Data Base Scan. These data sets are required if the UCF terminates while running one of these functions.

DFSURWF3:
This data set is created by the Prefix Resolution utility for the Prefix Update utility. If the UCF terminates while running the Prefix Resolution utility, this data set does not have to have been saved; it is created again when the Prefix Resolution is restarted. The WF3 data set must, however, have been saved for restart if the UCF terminates while running the Prefix Update utility.

EXAMPLES

Example 1

The following example shows the control statement input data set and the minimum JCL required to execute the UCF.

```
//STEP1       EXEC    PGM=DFSRRC00,PARM='ULU,DFSUCF00'
//STEPLIB     DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB    DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSPRINT    DD      SYSOUT=A,DCB=(BLKSIZE=605,LRECL=121,
//              RECFM=FBA)
//IMS         DD      DSN=IMSVS.PSBLIB,DISP=SHR
//            DD      DSN=IMSVS.DBDLIB,DISP=SHR
//DFSNJRNL    DD      DSN=NJRNL,UNIT=SYSDA,SPACE=(CYL,(2,2)),
//              DISP=(,KEEP),DCB=(RECFM=VB,BLKSIZE=1008,
//              LRECL=300)
//DFSOJRNL    DD      DUMMY
//DFSNCDS     DD      DSN=NCDS,UNIT=SYSDA,SPACE=(CYL,(2,2)),
//              DISP=(,KEEP)
//DFSOCDS     DD      DUMMY
//DFSRDER     DD      DUMMY
//DFSYSIN     DD      *
      UCF CONTROL STATEMENT INPUT
/*
//DFSCNTRL    DD      DSN=CNTRL,UNIT=SYSDA,SPACE=(CYL,(2,2)),
//              DCB=BLKSIZE=80
//*OTHER JCL AS REQUIRED BY
//*UCF CONTROL STATEMENTS
//*SPECIFIED BY THE USER.
```

Example 2

The following example shows the JCL used to execute UCF in a restart situation. STEP1 shows the restart through use of the parameter field in the EXEC card. The DFSYSIN DD statement is specified as DD DUMMY in this run.

```
//STEP1       EXEC    PGM=DFSRRC00,PARM='ULU,DFSUCF00,,,0001'
//STEPLIB     DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB    DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSPRINT    DD      SYSOUT=A
//IMS         DD      DSN=IMSVS.PSBLIB,DISP=SHR
//            DD      DSN=IMSVS.DBDLIB,DISP=SHR
//DFSNJRNL    DD      DSN=NJRNL2,UNIT=SYSDA,DISP=(,KEEP),
//              SPACE=(CYL,(2,2))
//DFSOJRNL    DD      DSN=NJRNL,DISP=(OLD,KEEP)
//DFSNCDS     DD      DSN=NCDS2,UNIT=SYSDA,DISP=(,KEEP),
//              SPACE=(CYL,(2,2)),DCB=BLKSIZE=1600
//DFSOCDS     DD      DSN=NCDS,DISP=(OLD,KEEP)
//DFSRDER     DD      DUMMY
//DFSYSIN     DD      DUMMY,DCB=BLKSIZE=80
//DFSCNTRL    DD      DSN=CNTRL,UNIT=SYSDA,SPACE=(CYL,(2,2)),
//              DCB=BLKSIZE=80
//*
//*OTHER JCL AS REQUIRED TO
//*IDENTIFY DATA SETS REQUIRED
//*FOR THE ACCESS METHOD BEING RESTARTED.
```

Example 3

The following example shows the JCL used to execute the UCF in a restart situation through use of a control statement.

```
//STEP1        EXEC    PGM=DFSRRC00,PARM='ULU,DFSUCF00'
//STEPLIB      DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB     DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSPRINT     DD      SYSOUT=A
//IMS          DD      DSN=IMSVS.PSBLIB,DISP=SHR
//             DD      DSN=IMSVS.DBDLIB,DISP=SHR
//DFSNJRNL     DD      DSN=NJRNL1,UNIT=SYSDA,DISP=(,KEEP),
//              SPACE=(CYL,(2,2))
//DFSOJRNL     DD      DSN=NJRNL,DISP=(OLD,KEEP)
//DFSNCDS      DD      DSN=NCDS1,UNIT=SYSDA,DISP=(,KEEP),
//              SPACE=(CYL,(2,2)),DCB=BLKSIZE=1600
//DFSOCDS      DD      DSN=NCDS,DISP=(OLD,KEEP)
//DFSRDER      DD      DUMMY
//DFSYSIN      DD      *
     FUNCTION=OP,COND=RESTART
/*
//DFSCNTRL     DD      DSN=CNTRL,UNIT=SYSDA,SPACE=(CYL,(2,2)),
//              DCB=BLKSIZE=80
//*
//*OTHER JCL AS REQUIRED TO
//*IDENTIFY DATA SETS REQUIRED
//*FOR THE ACCESS METHOD BEING RESTARTED.
```

Example 4

The following example shows the use of the UCF in a Change Accumulation run. STEP1 shows the JCL for the Change Accumulation execution run.

There are two UCF control statements illustrated in this example. The resemblance of these to the input to the Change Accumulation utility is shown below:

```
ID        006       004      010
DB1 *OTHER 743652359DDNAME1   DDNAME2   DDNAME3
DB0URDBNAME743652359DDNAME4   DDNAME5   DDNAME6
//STEP1       EXEC    PGM=DFSRRC00,PARM='ULU,DFSUCF00'
//STEPLIB     DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB    DD      DSN=IMSVS.RESLIB,DISP=SHR
//DFSPRINT    DD      SYSOUT=A
//IMS         DD      DSN=IMSVS.PSBLIB,DISP=SHR
//              DD    DSN=IMSVS.DBDLIB,DISP=SHR
//DFSNJRNL    DD      DSN=NJRNL,UNIT=SYSDA,SPACE=(CYL,(1,1)),
//               DISP=(,KEEP)
//DFSOJRNL    DD          DUMMY
//DFSNCDS     DD      DSN=NCDS,UNIT=SYSDA,SPACE=(CYL,(1,1)),
//               DISP=(,KEEP),DCB=BLKSIZE=1600
//DFSOCDS     DD      DUMMY
//DFSRDER     DD      DUMMY
//DFSYSIN     DD      *
FUNCTION=CA,DBNAME=*OTHER,REQUEST=STATS,RECID=1,PURGDT=74365      *
      PURGTM=2359,CUMOUT=DFSUCUMN,CUMIN=DFSUCUM0,LOGIN=DFSULOG,   *
      LOGOUT=DFSUDD1,DBDDS=(DDNAME1,DDNAME2,DDNAME3),EXEC=YES
FUNCTION=CA,DBNAME=URDBNAME,RECID=0,PURGDT=74365,PURGTM=2359,     *
      IDLEN=10,DBNUM=04,DDNUM=06,DBDDDS=(DDNAME4,DDNAME5,DDNAME6) *
/*
//DFSCNTRL    DD      DSN=CNTRL,UNIT=SYSDA,SPACE=(CYL,(1,1)),
//               DCB=BLKSIZE=80
//DFSULOG     DD      DSN=LOGINPUT,DISP=(OLD,PASS)
//DFSUDD1     DD      DSN=NEWLOG,UNIT=SYSDA,DISP=(,KEEP),
//               SPACE=(CYL,(2,2)),DCB=(RECFM=VB,LRECL=300,
//               BLKSIZE=1008)
//DFSUCUMN    DD      DSN=UCMN,DISP=(,KEEP),UNIT=SYSDA,
//               SPACE=(CYL,(2,2)),DCB=(RECFM=VBS,LRECL=3200,
//               BLKSIZE=3208)
//DFSUCUM0    DD      DUMMY,DCB=BLKSIZE=100
//SYSUDUMP    DD      SYSOUT=A
//SYSOUT      DD      SYSOUT=A
//SORTLIB     DD      DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01    DD      UNIT=SYSDA,SPACE=(CYL,(2,1))
//SORTWK02    DD      UNIT=SYSDA,SPACE=(CYL,(2,1))
//SORTWK03    DD      UNIT=SYSDA,SPACE=(CYL,(2,1))
//SORTWK04    DD      UNIT=SYSDA,SPACE=(CYL,(2,1))
//SORTWK05    DD      UNIT=SYSDA,SPACE=(CYL,(2,1))
//SORTWK06    DD      UNIT=SYSDA,SPACE=(CYL,(2,1))
/*
```

Example 5

The following example shows the UCF control statements necessary to perform a module zap during a Change Accumulation run. For a description of the Change Accumulation control statements, refer to Example 4. The displacements, CSECT name, and values are for illustrative purposes only.

In this example, the RELATE keyword on the zap module statements ties VERIFY and REP to FUNCTION=CA.

```
FUNCTION=CA,DBNAME=*OTHER,REQUEST=STATS,RECID=1,PURGDT=74365,      *
      PURGTM=2359,CUMOUT=DFSUCUMN,CUMIN=DFSUCUMO,LOGIN=DFSULOG,    *
      LOGOUT=DFSUDD1,DBDDDS=(DDNAME1,DDNAME2,DDNAME3),EXEC=YES,    *
      SEQ=002
FUNCTION=ZM,MODULE=CA,VERIFY=00000000,VALUE=47E0C1C1,RELATE=CA
FUNCTION=ZM,MODULE=CA,REP=00000000,VALUE=47F04040,RELATE=CA
```

PART 4. IMS/VS SYSTEM LOG UTILITIES

Part 4 has two chapters that describe the utilities used for restart, recovery, and analysis of the system log data.

Chapter 9, "Log Maintenance Utilities," describes a utility used to produce a usable log data set from one that contains read errors or to recover log data that was lost as a result of a system failure. Another utility is described that copies data from online log data sets (or batch system log data sets) to system log data sets. Optional output data sets may be created. Control statements and examples are provided.

Chapter 10, "Log Data Formatting Utilities," describes the utilities used for IMS/VS system log analysis and the types of reports that are produced. Control statements and examples and report examples are included.

CHAPTER 9. LOG MAINTENANCE UTILITIES

The information placed on the IMS/VS log is used for many purposes such as providing statistics for accounting, restarting IMS/VS, and recovering data bases. All input messages IMS/VS receives and all output messages it sends are recorded on the IMS/VS log. All messages processed, the processing time, and the number and type of data base references made are also recorded. This information is used to supply statistics about message volume by communication line and terminal.

LOG DATA SETS

Batch systems are able to log to either tape or DASD whereas online systems log only to DASD.

IMS/VS log data is recorded in the following data sets:

- Online Log Data Set (OLDS)
- Write Ahead Data Set (WADS)
- System Log Data Set (SLDS)

These data sets are described in the following sections.

ONLINE LOG DATA SET (OLDS)

Log records created by an online system are written to DASD data sets called OLDS. OLDSs contain all the log records required for restart and recovery. IMS/VS uses the OLDS in a wraparound fashion. When an OLDS is filled, it can be archived to the SLDS by the Log Archive utility (later in this chapter).

If dual logging of the OLDS is an installation requirement, then a pair of data sets (primary and secondary) must be assigned to the OLDS. The ddnames for the OLDS begin with the character string DFSOLP for the primary data set and DFSOLS for the secondary data set. A unique numeric suffix (00 through 99) called an 'OLDS identifier' completes the 8-character ddname. Single or dual logging for the OLDS is determined from the presence of DD statements during system initialization.

WRITE AHEAD DATA SET (WADS)

A WADS is a small direct access data set containing a copy of log write-ahead data log records that are in OLDS buffers but have not yet been written to the OLDS. When logging to DASD (required for online processing), fixed-length blocks make direct retrieval easier. A WADS allows large fixed-length blocks (in variable blocked format) to be written to the OLDS without the requirement to rewrite blocks. When log data has been written to OLDS, the WADS is reused. If a system failure occurs, the log data in the WADS is used to close the OLDS. The close process will occur as part of an emergency restart or as an option of the Log Recovery utility (later in this chapter).

Dual WADS logging is supported to provide backup in the event of a read error while closing the OLDS from the WADS. Also, online support is provided for spare WADSs. When a write error is detected, a spare WADS replaces the WADS that encountered the error.

SYSTEM LOG DATA SETS (SLDS)

As mentioned previously, the OLDS are used in a wraparound fashion by the online IMS/VS system. After an OLDS is filled, its data is copied by the Log Archive utility into an SLDS. The log created by a batch IMS/VS system is also an SLDS.

OLDS FORMAT

The OLDS is a BSAM data set consisting of fixed-length blocks. However, the format declared to BSAM is VB (variable blocked). Each block consists of three parts:

1. A 4-byte block descriptor word
2. Variable-length blocked records
3. A variable-length dummy record

The variable-length dummy record is padded to ensure that each OLDS record is the same fixed length. The format of the OLDS block is shown in Figure 82.

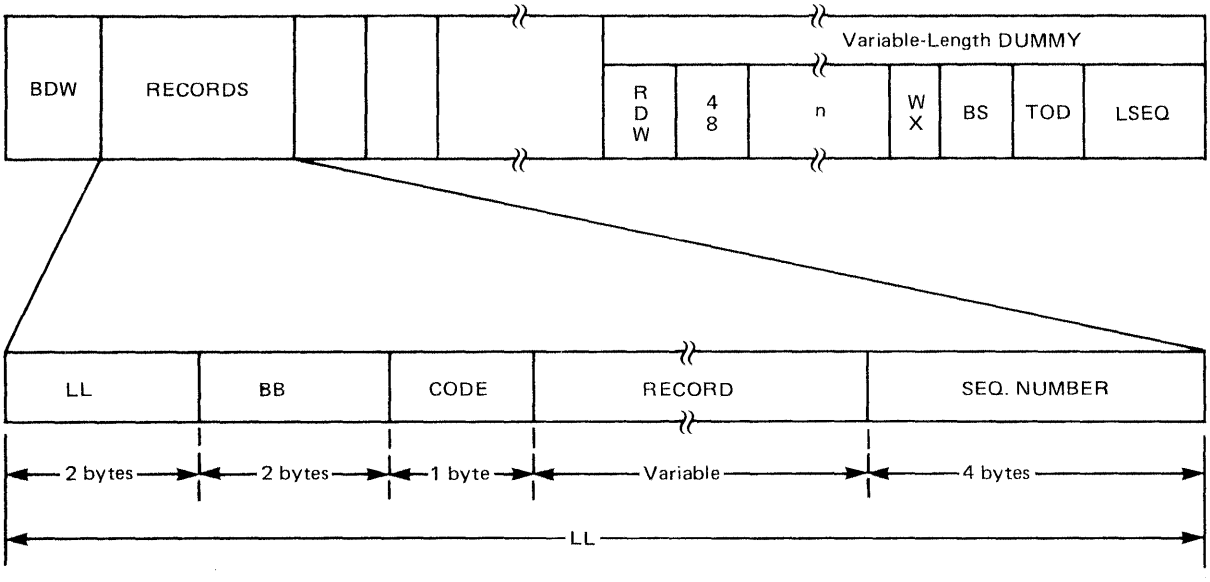


Figure 82. OLDS Block Format

Notes to Figure 82:

BDW
is a 4-byte block descriptor word

RECORDS
is the variable-length blocked records

where:

- LL is a halfword binary number representing the total length of the logical record.
- BB is a halfword used by OS/VS.
- CODE is a 1-byte log code identifying the logical record type.
- RECORD is the variable-length record.

- SEQ NUMBER is a 4-byte log record sequence number.

DUMMY
is the variable-length dummy record.

where:

- RDW is a 4-byte record descriptor word.
- X'48' is the 1-byte log type code (dummy record).
- n is from 0 to n unused bytes that cause the record to exactly fill the block.
- WX is a 2-byte WADS matrix.
- BS is a 4-byte log block sequence number.
- TOD is 4 bytes for time of day.
- LSEQ is a 4-byte log record sequence number.

When messages are logged, control information, in the form of the message prefix, is logged with each message received or sent. In this prefix are the destination or source, date and time, and an input or output sequence number.

LOG RECOVERY UTILITY (DFSULTR0)

OVERVIEW

Use the Log Recovery utility (DFSULTR0) to produce a usable log data set from a log data set that contains read errors or that was not properly terminated. Both OLDSs and batch or online SLDSs may be recovered using DFSULTR0.

This utility has three modes of operation:

- CLS mode
- DUP mode
- REP mode

CLS mode is used to close an OLDS from the WADS or from the next OLDS. DUP mode is used to create an interim log containing error ID records, or a closed batch SLDS containing an end-of-file mark. REP mode reads the interim log, replaces the error ID records with user-specified data, and creates a new log. CLS mode processes only OLDSs. DUP mode and REP mode process either SLDSs or OLDSs. Because operation of these modes differ for OLDS and SLDS, they are described individually under "OLDS Recovery" and "SLDS Recovery" below.

OLDS Recovery

An OLDS must be closed before it can be archived or used as input to any utility. The OLDS in use is closed automatically during normal shutdown or during emergency restart, but will have to be closed using the Log Recovery utility when an emergency restart fails, or when the OLDS was not closed because a write error was detected.

The Log Recovery utility recovers the following types of errors:

- An I/O error while reading the input log data set
- An error in the log record or log block length
- A sequence error in the log record, the log block, or the OLDS write time stamp

SLDS Recovery

An SLDS must be closed before it can be used as input to any utilities or IMS/VS restart. The Log Recovery utility closes an SLDS created by a batch IMS/VS system. The types of errors the utility detects are:

- An I/O error while reading the input log data set
- An error in the log record or log block length
- A log record sequence error

INPUT AND OUTPUT

Single Logging

Using a single log for input, CLS mode:

1. Reads the input log
2. Produces a usable log that is closed

Using a single log for input, DUP mode:

1. Reads the input log
2. Creates an interim log
3. Produces a character and hexadecimal listing of error blocks

Using the interim log produced by DUP mode, REP mode:

1. Reads the interim log
2. Copies good blocks to the output log
3. Replaces error blocks with good ones based on user-specified control statements
4. Produces a usable log that is closed

Dual Logging

Using dual logs for input, CLS mode:

1. Reads the input logs
2. Produces a usable log that is closed

If dual system log input is used and errors at the same position on both input logs are not encountered, the log produced by DUP mode is correct and REP mode is not required. In the following discussion, the terms 'primary' and 'secondary' are used to identify the two logs of a dual log data set.

- DUP mode—In DUP mode, this utility reads the primary log and copies the contents onto a new system log. If an error block is encountered, DUP mode positions a read operation on the secondary log where the log error was encountered. DUP

mode then reads the secondary log and copies the contents onto the same new system log. If an error is now encountered on the secondary log (but not at the same position) DUP mode positions a read operation on the primary log where the error was encountered. This process continues until a complete new system log is produced. Figure 83 illustrates DUP mode and REP mode using dual logging.

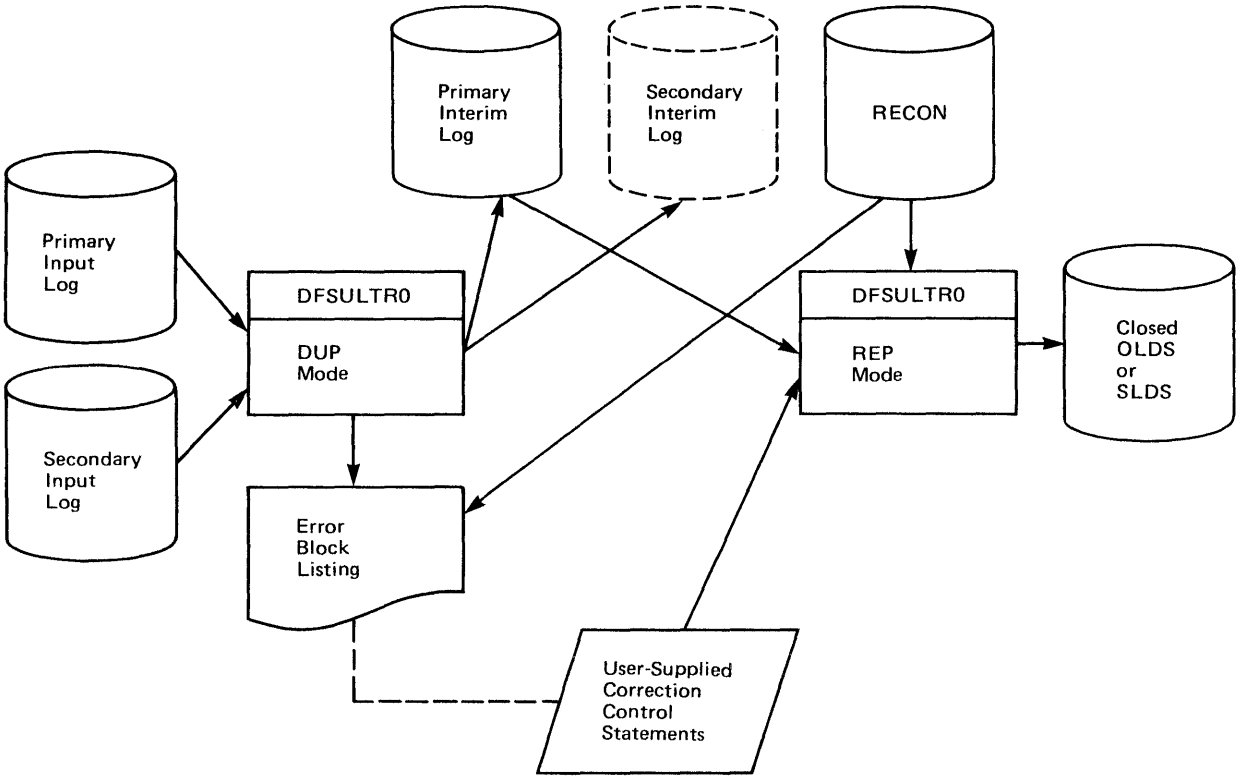


Figure 83. DUP Mode and REP Mode When Dual Logging is Used

If DUP mode encounters an error on both logs in the same position, it copies both error blocks onto the interim log and uniquely identifies the error blocks. The interim log data set contains all valid log blocks, error blocks, and error ID records. DUP mode then produces a character and hexadecimal listing of the error blocks to be used as a guide for creating the user-specified control statements required by REP mode.

- REP mode—Reads the interim log created by DUP mode, copies good blocks, and replaces error blocks with good ones based on user-specified control statements. The output log data set is a usable OLDS.

Interim Log Error ID Record

Figure 84 on page 379 illustrates the error ID record on the interim log produced from dual log input. In this example, BLK2 of both the primary and secondary logs has errors. On the interim log, the first error ID is for BLK2B and the second error ID is for BLK2A. During REP mode, BLK2A or BLK2B is replaced with a good block based on user-specified control statements. This example also shows the valid log after REP mode execution.

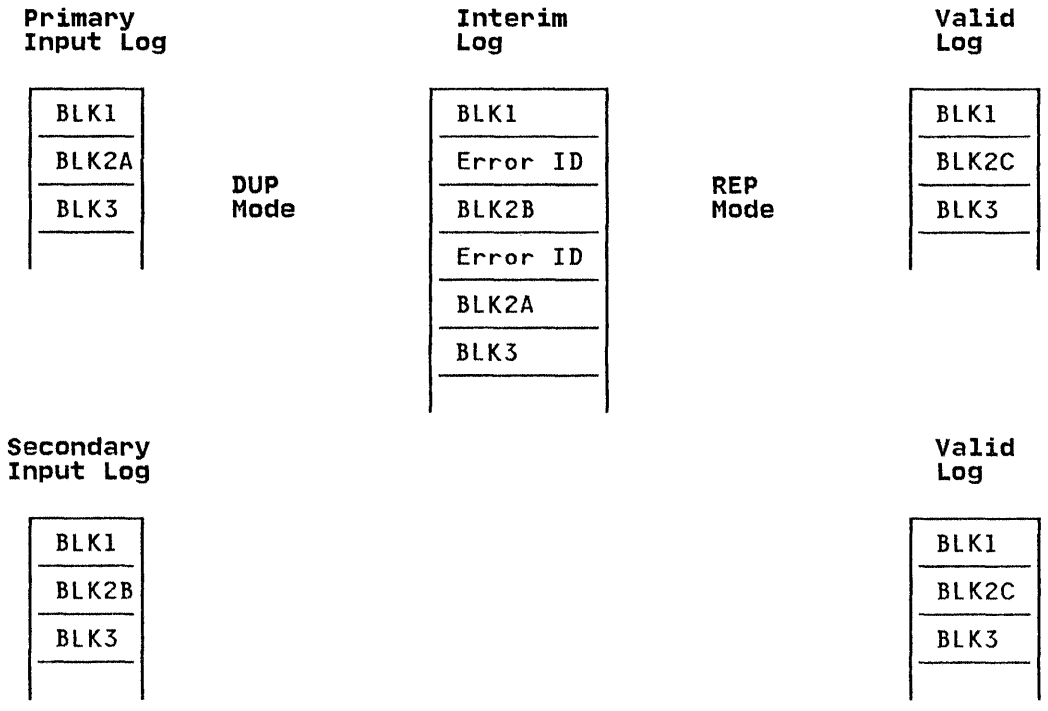


Figure 84. Error ID Records

Error Block Listing (SYSPRINT)

As mentioned above, the error block listing contains the errors found during execution of CLS mode and DUP mode. It also contains verification messages resulting from REP mode followed by a dump of the data record. The content of the error block listing is discussed in more detail below:

CLS MODE AND DUP MODE ERROR LISTING:

- I/O ERROR ON ddddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn BLOCK LENGTH CHANGED FROM X'1111'
  (A SYNAD exit was taken because of an I/O error on the input log, when only one log data set was input.)

- RECORD IN ERROR ON ddddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn BLOCK LENGTH CHANGED FROM X'1111'
  (A block length error was detected and was changed to a block length in the DCB.)

- RECORD IN ERROR ON ddddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn RECORD LENGTH ERROR IS FOUND
  (A record length error was detected, but the block length was good.)

- RECORD IN ERROR ON ddddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn LOG SEQ ERROR ssssssss to ssssssss
  (A log record sequence number was not in ascending sequence.)

- RECORD IN ERROR ON ddddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn BLK SEQ ERROR ssssssss to ssssssss
  (A log block sequence number was not in ascending sequence. This only applies to OLDSs. A block sequence number is in the suffix of each OLDS block.)

- RECORD IN ERROR ON ddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn TIME OF DAY IN OLDS SEQUENCE ERROR
  (A time stamp in the suffix of the OLDS log block was not in ascending order. This only applies to OLDSs.)

- RECORD IN ERROR ON ddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn SEQ# ERROR IN THE ALTERNATE LOG
  (During dual logging, DFSULTR0 switched logging to the alternate log after a log record sequence number error was found in the current log. However, the alternate log also had the log record sequence number error. The dump following this message is the BLOCK in the alternate log that corresponds to an error block in the current log.)

- RECORD IN ERROR ON ddddddd BLOCK #bbbbbb
  ERROR-ID=Xnnnnn LOG SEQ ERROR I/O ERR TO ssssssss
  (A log record sequence number error was detected, because the previous block had an I/O error.)

  ddddddd - ddname
  bbbbbb - block number
  ERROR-ID - corresponds to the error ID in the interim log data set
  llll - block length specified in the error block
  Xnnnnn - Annnnn or Bnnnnn (Error-ID)
  ssssssss - a log record sequence number or log block sequence number

REP MODE VERIFICATION MESSAGES:

During REP mode processing, a valid replacement of data on the interim log data set will cause the following message to be printed:

DATA REPLACED IN RECORD Axxxxx ... replacement data text...

where xxxxx is the error ID.

An error in the control statement format will cause the following message to be printed:

ERROR IN CONTROL STATEMENT FORMAT ... text of control statement...

DUMP OF DATA RECORD:

The dump of the data record following the verification messages is a hexadecimal representation of the record. The hexadecimal representation is printed in four lines per print line of the data record. The first line consists of the position within the block in error (starting with 1), and the EBCDIC representation of the bytes. The second line indicates the first byte of each log record, using an asterisk. The third line consists of the zone half representation. The fourth line consists of the digit half representation. The printed output will be seen as:

```
000001           q   RRE  b    1st line (EBCDIC representation)
          *                    2nd line (first byte of a log record)
        2000020049 00DDC40809  3rd line (high-order hexadecimal digit)
        00000D0008 029954024F  4th line (low-order hexadecimal digit)
```

Active Region Messages

When WADS is specified in CLS mode, the active PSBs at the time of the system failure will be printed. A line is printed for each PSB active at the time of failure. If backout is required for the PSB, data base names are listed under the PSB line in the output. The format of this output is shown below:

```
*******   ACTIVE REGIONS AT THE TIME OF ABEND   ******

   PSB (TRAN CODE) - pppppppp PGM gggggggg mmmmmm
   DBNAME - dddddddd
   DBNAME - dddddddd
   PSB (TRAN CODE) - pppppppp PGM gggggggg mmmmmm
   DBNAME - dddddddd
```

where:

pppppppp	is the PSB name or transaction code
gggggggg	is the program name
dddddddd	is the data base name
mmmmmm	is the message BACKOUT REQUIRED; BACKOUT NOT REQUIRED; or BACKOUT MAY BE REQUIRED

JCL REQUIREMENTS

The JCL required to run DFSULTR0 is explained below. Examples of JCL using different modes appear later in this chapter.

EXEC

Invokes the Log Recovery utility (DFSULTR0). Its format is:

```
//STEP EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii,
//              DBRC=ddd,PSBLIST=ppp,CICS=ccc'
```

iiiiiiii

is the IMSID of the on-line system that created the input OLDS. IMSID= is required for CLS mode (except when PSBLIST=YES is specified). IMSID= is required for DUP mode with OLDS input and DBRC=YES (specified or defaulted). IMSID= is ignored if it is specified but not needed.

ddd

is N | NO | Y | YES.

The DBRC= default **IS NOT** established by the IMSCTRL macro during IMS/VS system definition.

DBRC=NO may be specified to explicitly declare that DBRC **IS NOT** to be used for this execution of this utility.

DBRC=YES may be specified to explicitly declare that DBRC **IS** to be used for this execution of this utility.

DBRC=YES is required (and the default) for CLS mode (except when PSBLIST=YES is specified and it defaults to DBRC=NO).

DBRC=YES is optional for DUP and REP modes. If DUP mode is run with DBRC active then REP mode should also be run with DBRC active.

If CICS=YES is specified then DBRC=NO will default.

If PSBLIST=YES is specified in DUP mode with SLDS input then DBRC=YES will default.

Except as already noted, DBRC=YES will default.

ppp
is Y | YES.

PSBLIST=YES is used when a previous execution of this utility issued one or both of the following messages:

- DFS3272I X'47' LOG RECORD NOT FOUND. ACTIVE PSB MESSAGES NOT GENERATED.

  To get active region messages, the Log Recovery utility must be rerun in CLS mode (see Example 12).

- DFS3279I INCOMPLETE DEDB LOG CHAIN MAY REMAIN IN PREVIOUS LOG DATA SET.

  To remove the incomplete DEDB log chain from the prior log, the Log Recovery utility must be rerun. If the previous log is still available as an OLDS, rerun the job in CLS mode (see Example 12).

  If the previous OLDS has already been archived into an SLDS, it is also necessary to run a Log Recovery utility in DUP mode (see Example 13).

PSBLIST=YES should not be run with an OLDS that is not closed. An incomplete listing may result. PSBLIST=NO is the default, and can be requested only by omitting the PSBLIST= parameter completely.

ccc
is Y | YES.

CICS=YES is used when the IEFRDER input is a CICS log. This parameter indicates that special sequence checking is to be performed and is required when processing a CICS log (see Examples 4, 7, and 10).

CICS=YES should not be specified along with PSBLIST=YES. CICS=NO is the default, and can only be specified by omitting the CICS= parameter completely.

Note: DBRC=NO is the default when CICS=YES is specified. If DBRC is being used then specify DBRC=YES and make the RECON data sets available to this utility.

If a parameter is to be allowed to default, the complete parameter (including the keyword) must be omitted from the PARM field.

If no input parameters are specified, the default will be IMSID=(not specified), DBRC=YES, PSBLIST=NO, and CICS=NO.

DD Statements

The DD statements described below should only be supplied if they are required for a given execution of the Log Recovery utility.

If single logging was used and DBRC was active, single logs must be presented as input to the Log Recovery utility and only single logs may be created as output from DUP and REP mode. Otherwise, DBRC abends will result.

If dual logging was used and DBRC was active, dual logs must be presented as input to the Log Recovery utility. Otherwise, incorrect DBRC RECON updates may result. Likewise, if dual logs were presented as input, dual logs must be created as output from DUP and REP mode. Care should be taken to ensure that primary and secondary DSNAMEs are correctly specified on the DD statements.

OLDS input must be specified using the DFSOLP (and DFSOLS) DD statement. SLDS input must be specified using the IEFRDER (and IEFRDER2) DD statement. CICS input must be specified using the IEFRDER DD statements.

DFSOLP (and DFSOLS) DD statements may not be specified in an execution that also contains an IEFRDER (and IEFRDER2) DD statement. DFSWADSn DD statements may not be specified in an execution that also contains a DFSNOLP (and DFSNOLS) DD statement. DFSWADSn and/or DFSNOLP (and DFSNOLS) may not be specified in an execution that also contains an IEFRDER (and IEFRDER2) DD statement. DFSPOLP (and DFSPOLS) DD statements may not be specified in an execution that also contains a DFSNOLP (and DFSNOLS) DD statement.

Refer to the Examples later in this chapter for examples of valid DD statement combinations.

STEPLIB DD
points to IMSVS.RESLIB, which contains this utility's modules.

SYSPRINT DD
defines the system messages data set.

SYSUDUMP DD
defines the dump data set.

Note: SYSUDUMP statements are not included in the examples later in this chapter.

DFSOLP DD
defines the primary, or only, input OLDS.

DFSOLS DD
defines the secondary input OLDS. Include this statement only when dual OLDSs are used.

DFSWADSn DD
defines the WADS data set, where n can be 0 through 9. All WADSs used during online execution can be specified, but only those that were in use by the online system at the time of failure are required. This DD statement is required when closing an OLDS from a WADS. If no WADS were in use by the on-line system, then supply no DFSWADSn DD statements.

DFSNOLP DD
defines the primary, or only, next-OLDS. The next-OLDS is the OLDS written by the online IMS/VS system immediately after the OLDS having a write error.

DFSNOLS DD
defines the secondary next-OLDS. Include this statement only when dual OLDSs are used.

DFSPOLP DD
defines the primary OLDS data set that the IMS/VS online subsystem used before the specified OLDS data set which is being closed. If there is no prior OLDS data set, this DD statement should not be used. This DD statement is used only when an OLDS data set is being closed from the WADS.

DFSPOLS DD
defines the secondary OLDS data set that the IMS/VS online subsystem used before the specified OLDS data set that is being closed. Include this statement only when dual prior OLDS are used.

IEFRDER DD
defines the primary, or only, input SLDS or CICS log. All input SLDS or CICS logs for DUP mode should have the same block size. (See Example 3 for multivolume SLDS considerations when running DUP.) Any CICS log used for

input needs the RECFM=VB coded on the DCB subparameter for this DD statement.

IEFRDER2 DD
defines the secondary input SLDS. Include this statement only when dual logs are used. Omit this statement if you do not want the data sets. Do not use DD DUMMY or DSNAME=NULLFILE as unpredictable results can occur.

NEWRDER DD
defines the primary, or only, output data set for the new or interim log.

NEWRDER2 DD
defines the secondary output data set for the new or interim log. The contents of NEWRDER and NEWRDER2 are identical. Omit this statement if you do not want the data sets. Do not use DD DUMMY or DSNAME=NULLFILE as unpredictable results can occur.

RECON1 DD
Defines the first DBRC RECON data set. This statement is not required if dynamic allocation is used.

RECON2 DD
defines the second DBRC RECON data set. This statement is not required if dynamic allocation is used.

RECON3 DD
defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using. This statement is not required if dynamic allocation is used.

SYSIN DD
defines the control data set containing the log recovery input control statements.

UTILITY CONTROL STATEMENTS

CLS Mode-Close an OLDS from the WADS or NEXT OLDS

The format of this control statement is:

Position	Description
1-3	indicates CLS mode. DBRC is required for CLS mode. When closing from the WADS, if a prior OLDS is available, the suffix from the last block written to the prior OLDS (the block sequence number is passed to DBRC at OLDS switch and stored in the RECON) is obtained, and the block suffix is used to establish a basis for sequence checking the OLDS being closed. Further, note that when closing from the WADS, either EOF or encountering the first error causes an attempt to close the OLDS from the WADS. If a sequence error is found, CLS mode will fail. A listing that contains the block at the first error may be produced (see Example 1). When closing from the next-OLDS, the sequence number of the first block of the next-OLDS (BSN) is determined. The input OLDS is closed when block BSN-1 is found on the input OLDS. If either EOF or an error is encountered before block BSN-1 is found, CLS mode will fail (see Example 2).

5	NOWADS suppresses the use of WADS when closing the OLDS. **Warning: NOWADS should only be used when WADS is unavailable. Avoid using this keyword if at all possible; you will lose system integrity.**

DUP Mode-Recover an OLDS, SLDS, or CICS Log-Create an Interim Log

The format of this control statement is:

Position	Description
1-3	indicates DUP mode.
5-14	ERRC=nnnnn is used to terminate DUP mode after a predefined number of I/O or sequence errors are detected on the input log data set. nnnnn specifies the number of errors (00000 through 99999). If no value is specified or the keyword is omitted, the default is 99999. This field must contain 5 digits, with leading zeros. If ERRC=00000 is specified, DUP mode is terminated and the interim log is closed when the first error is encountered. The error ID record and error blocks are not written on the interim log. REP mode is not required. A listing may be produced that contains the block at the the first error. If ERRC>00000 is specified, DUP mode is terminated when either EOF is encountered or ERRC is reached (ERRC is tested before each block read). If errors are found, error ID records and error blocks are written on the interim log and REP mode is required. A listing that contains the errors found is produced. A typical use of ERRC=00000 is to close an SLDS or CICS log without having to run REP mode (see Examples 3 and 4). Specifying ERRC>00000 is usually used when recovering an OLDS, SLDS, or CICS log (see Examples 5, 6, and 7).

REP Mode-Recover an OLDS, SLDS, or CICS Log-Create a New Log

This mode reads the interim log created by DUP mode, copies good blocks, and replaces error blocks with good ones based on the REP control statements. (Only the primary input data set is read during REP mode). The output log data set is a new OLDS, SLDS, or CICS log. At least one control statement is required but any number can be included (see Examples 8, 9, and 10).

The three formats of this control statement are:

Position	Description
1-3	indicates REP mode.
5-14	SEQ=xnnnnn indicates the identification number of the block to be changed. The number is provided in the listing output by DUP mode. See "Error Block Listing (SYSPRINT)" earlier in this chapter for a description of the content of the error block listing.
16-25	POS=pppppp indicates the starting position, relative to 1, of the data to be replaced. SKIP indicates the output log will not contain this block of data.

CLOSE indicates the output log will be closed immediately before this error block.

27-80 DAT=dd...dd is 2 to 50 hexadecimal characters (0 through 9, A through F) representing the replacement data.

Note: The REP mode "CLOSE" option should not be confused with the process of closing an OLDS from the WADS or next-OLDS using CLS mode.

The following rules apply to use of the REP statement:

- At least one control statement must be supplied.
- Unless the log is closed at a prior block, each error block identified in the DUP mode output must have at least one control statement supplied for it.
- When multiple REP statements are provided, the identification numbers (SEQ=) must be in ascending block number sequence.
- If a block is identified as being in error even though the data is good, a control statement must be supplied for the block. Replace the first 4 bytes of the good block with the existing data. This is usually the case for the first block following an I/O error.
- If dual logs are used in DUP mode, supply a statement for only one of the two blocks in error, either Annnnn or Bnnnnn. The block not selected is ignored and is not written to the output log.
- If the log being recovered is an OLDS which has not been properly closed from either the WADS or next OLDS, then the Log Recovery utility must be rerun in CLS mode using the output of REP mode as input.

ERROR PROCESSING

The Log Recovery utility provides the following return codes:

Code	Meaning
0	Successful completion
4	Successful completion—this condition code is accompanied by one or both of the following messages: **DFS3272I** X'47' LOG RECORD NOT FOUND. ACTIVE PSB MESSAGES NOT GENERATED. **DFS3279I** INCOMPLETE DEDB LOG CHAIN MAY REMAIN IN PREVIOUS LOG DATA SET.
8	Unsuccessful completion

These return codes can be tested by the COND= parameter on the EXEC statement of a later job step.

Descriptions of all error messages issued by DFSULTR0 are provided in IMS/VS Messages and Codes Reference Manual.

EXAMPLES

Example 1

The following example shows how to close an OLDS from the WADS using CLS mode. The input data set(s) is closed in place. The DBRC RECON data set(s) is updated with the "close time."

```
//EXAMPL01  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - DBRC=NO is not valid.
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is covered in Examples 11 and 12
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is not valid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  Primary   OLDS to be closed
//DFSOLS    DD ........  Secondary OLDS to be closed
//DFSPOLP   DD ........  Primary prior OLDS
//DFSPOLS   DD ........  Secondary prior OLDS
//DFSWADSn  DD ........  WADS used by on-line system
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
CLS
```

Note: If no WADS were in use when the input OLDS or prior OLDS was created, remove the DFSWADSn DD statement(s).

If no prior OLDS are available, remove the DFSPOLP (and DFSPOLS) DD statement.

Example 2

The following example shows how to close an OLDS from the next-OLDS using CLS mode. The input data set is closed in place. The DBRC RECON data set(s) is updated with the "close time" and the "ERROR" flag is turned off.

```
//EXAMPL02  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - DBRC=NO is not valid
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is not valid
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is not valid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  OLDS to be closed from next-OLDS
//DFSNOLP   DD ........  next-OLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
CLS
```

Example 3

The following example shows how to close an SLDS created by IMS/VS batch, using DUP mode and ERRC=00000. The input data set(s) is copied to, and closed in, the output data set(s). The input SLDS information in the DBRC RECON is replaced by the output data set(s) information.

```
//EXAMPL03  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*        (PARM NOT REQUIRED - SEE NOTES BELOW)
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is covered in Example 13
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is covered in Example 4
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  Primary   SLDS to be closed
//IEFRDER2  DD ........  Secondary SLDS to be closed
//NEWRDER   DD ........  Primary   output SLDS
//NEWRDER2  DD ........  Secondary output SLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
DUP ERRC=00000
```

Note: When ERRC=00000 is specified, NEWRDER (and NEWRDER2) is closed when EOF or the first error is encountered on IEFRDER (and IEFRDER2). If the execution is successful, REP mode is not required. If the execution is unsuccessful, DUP mode should be rerun with ERRC>00000 and REP mode is required.

The first error may not be the correct closing point.

See Example 6 for DUP mode with ERRC>00000.

See Example 9 for REP mode.

If the input SLDS (IEFRDER, IEFRDER2) is a multiple volume tape data set, only the last volume needs to be specified on the DD statement. In addition, the data set name (DSN) on the output DD statement (NEWRDER, NEWRDER2) should be the same as the input. If the execution is successful, only the volume information is replaced in the DBRC RECON. ERRC=00000 is required.

Example 4

The following example shows how to close a CICS log using DUP mode and ERRC=00000. The input data set is copied to, and closed in, the output data set.

If DBRC is active (DBRC=YES specified and RECONs available), the input CICS log information in the DBRC RECON is replaced by the output data set information.

```
//EXAMPL04  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='CICS=YES'
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=NO,PSBLIST=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is invalid
//* NOTE - CICS=YES is required
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  CICS log to be closed
//NEWRDER   DD ........  output CICS log
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//*                      (required if DBRC=YES is specified)
//SYSIN     DD *
DUP ERRC=00000
```

Note: If ERRC=00000 is specified, NEWRDER is closed when EOF or the first error is encountered on IEFRDER. REP mode is not required if the execution is successful. If the execution is unsuccessful, DUP mode should be rerun with ERRC>00000 and REP mode is required.

The first error may not be the correct closing point.

See Example 7 for DUP mode with ERRC>00000.

See Example 10 for REP mode.

Example 5

The following example shows how to use DUP mode as the first of two steps in the recovery of an OLDS. The input data set(s) is copied to an interim data set(s). Interim log records are created in the DBRC RECON.

```
//EXAMPL05  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is invalid
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is invalid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  Primary   OLDS to be recovered
//DFSOLS    DD ........  Secondary OLDS to be recovered
//NEWRDER   DD ........  Primary   interim data set
//NEWRDER2  DD ........  Secondary interim data set
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
DUP ERRC=NNNNN
```

Note: If ERRC>00000 is specified (the default is 99999), error blocks are written to the output data set(s), and a listing is produced for the blocks in error. REP mode is required to correct the errors and to remove error blocks. If no errors are found and the execution is successful, REP mode is not required.

When ERRC=00000 is specified, NEWRDER (and NEWRDER2) is closed when EOF or the first error is encountered on DFSOLP (and DFSOLS). If the execution is successful, REP mode is not required. If the execution is unsuccessful, DUP mode should be rerun with ERRC>00000 and REP mode is required.

If the log being recovered is an OLDS which has not been properly closed from either the WADS or next OLDS, then the Log

Recovery utility must be rerun in CLS mode using the output of REP mode as input (or the output of DUP mode if no errors were detected).

See Example 8 for REP mode.

Example 6

The following example shows how to use DUP mode as the first of two steps in the recovery of an SLDS. The input data set(s) is copied to an interim data set(s). Interim log records are created in the DBRC RECON.

```
//EXAMPL06  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*        (PARM NOT REQUIRED - SEE NOTES BELOW)
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is covered in Example 13
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is covered in Example 7
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  Primary   SLDS to be recovered
//IEFRDER2  DD ........  Secondary SLDS to be recovered
//NEWRDER   DD ........  Primary   interim data set
//NEWRDER2  DD ........  Secondary interim data set
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
DUP ERRC=NNNNN
```

Note: If ERRC>00000 is specified (the default is 99999), error blocks are written to the output data set(s) and a listing is produced for the blocks in error. REP mode is required to correct the errors and to remove error blocks. If no errors are found and the execution is successful, REP mode is not required.

See Example 3 for DUP mode with ERRC=00000.

See Example 9 for REP mode.

Example 7

The following example shows how to use DUP mode as the first of two steps in the recovery of a CICS log. The input data set is copied to an interim data set.

If DBRC is active (DBRC=YES specified and RECONs available), interim log records are created in the DBRC RECON.

```
//EXAMPL07  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='CICS=YES'
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=NO,PSBLIST=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is invalid
//* NOTE - CICS=YES is required
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  CICS log to be recovered
//NEWRDER   DD ........  interim data set
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//*                      (required if DBRC=YES is specified)
//SYSIN     DD *
DUP ERRC=NNNNN
```

Note: If ERRC>00000 is specified (the default is 99999), error blocks are written to the output data set(s) and a listing is produced for the blocks in error. REP mode is required to correct the errors and to remove error blocks. If no errors are found and the execution is successful, REP mode is not required.

See Example 4 for DUP mode with ERRC=00000.

See Example 10 for REP mode.

Example 8

The following example shows how to use REP mode as the second of two steps in the recovery of an OLDS. The input data set(s) is copied to a new OLDS data set(s). During the copy process, error blocks are removed and the blocks in error are corrected as directed by the REP control statements. The interim data set information in the DBRC RECON is deleted. The original OLDS information in the DBRC RECON is replaced by the output set(s) information.

```
//EXAMPL08  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is invalid
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is invalid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  Primary   interim data set
//DFSOLS    DD ........  Secondary interim data set
//NEWRDER   DD ........  Primary   recovered OLDS
//NEWRDER2  DD ........  Secondary recovered OLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
REP ..........
```

Note: See the Control Statements section for an example of the formats of REP.

If the log being recovered is an OLDS which has not been properly closed from either the WADS or next OLDS, then the Log Recovery utility must be rerun in CLS mode using the output of REP mode as input.

See Example 5 for DUP mode.

Example 9

The following example shows how to use REP mode as the second of two steps in the recovery of an SLDS. The input data set(s) is copied to a new SLDS data set(s). During the copy process, error blocks are removed and the blocks in error are corrected as directed by the REP control statements. The interim data set information in the DBRC RECON is deleted. The original SLDS information in the DBRC RECON is replaced by the output data set(s) information.

```
//EXAMPL09  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*        (PARM NOT REQUIRED - SEE NOTES BELOW)
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES,PSBLIST=NO,CICS=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is invalid
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is covered in Example 13
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  Primary   interim data set
//IEFRDER2  DD ........  Secondary interim data set
//NEWRDER   DD ........  Primary   recovered SLDS
//NEWRDER2  DD ........  Secondary recovered SLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
REP .........
```

Note: See the Utility Control Statements section for an example of the formats of REP.

See Example 6 for DUP mode.

Example 10

The following example shows how to use REP mode as the second of two steps in the recovery of a CICS log. The input data set is copied to a new CICS log data set. During the copy process, error blocks are removed and the blocks in error are corrected, as directed by the REP control statements.

If DBRC is active (DBRC=YES specified and RECONs available), the interim data set information in the DBRC RECON is deleted. The original CICS log information in the DBRC RECON is replaced by the output data set information.

```
//EXAMPL10  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='CICS=YES'
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=NO,PSBLIST=NO
//* NOTE - PSBLIST=NO cannot be specified explicitly
//* NOTE - PSBLIST=YES is invalid
//* NOTE - CICS=YES is required
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  interim data set
//NEWRDER   DD ........  recovered CICS log
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//*                      (required if DBRC=YES is specified)
//SYSIN     DD *
REP .........
```

Note: See the Utility Control Statements section for an example of the formats of REP.

See Example 7 for DUP mode.

Example 11

The following example shows how to generate a listing of "active PSBs" after having received message DFS3272I (X'47' LOG RECORD NOT FOUND. ACTIVE PSB MESSAGES NOT GENERATED). CLS mode is used.

```
//EXAMPL11  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='PSBLIST=YES'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=NO,CICS=NO
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is not valid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  next prior primary   OLDS
//          DD ........  closed     primary   OLDS
//DFSOLS    DD ........  next prior secondary OLDS
//          DD ........  closed     secondary OLDS
//DFSWADSn  DD ........  WADS used by on-line system
//*                            (see note below)
//SYSIN     DD *
CLS
```

Note: When PSBLIST=YES is specified, CLS mode will not close the OLDS.

If there are no valid WADS data sets, the WADS DD statements should be commented out, removed completely, or specified as DD DUMMY.

Example 12

The following example shows how to remove incomplete DEDB log chains from the next-prior-OLDS after receiving message DFS3279I (INCOMPLETE DEDB LOG CHAIN MAY REMAIN IN PREVIOUS LOG DATA SET) while closing an OLDS. CLS mode is used. The next-prior-OLDS is updated in place. An "ACTIVE PSB" listing is also generated.

```
//EXAMPL12  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='PSBLIST=YES'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=NO,CICS=NO
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is not valid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  next prior primary   OLDS
//          DD ........  closed     primary   OLDS
//DFSOLS    DD ........  next prior secondary OLDS
//          DD ........  closed     secondary OLDS
//DFSWADSn  DD ........  WADS used by on-line system
//*                            (see note below)
//SYSIN     DD *
CLS
```

Note: When PSBLIST=YES is specified, CLS mode will not close the OLDS.

If the next-prior-OLDS has also been archived, see Example 16.

If there are no valid WADS data sets, the WADS DD statements should be commented out, removed completely, or specified as DD DUMMY.

Example 13

The following example shows how to remove incomplete DEDB log chains from the SLDS after receiving message DFS3279I (INCOMPLETE DEDB LOG CHAIN MAY REMAIN IN PREVIOUS LOG DATA SET) while closing an OLDS. In this case, the next-prior-OLDS has also been archived. DUP mode is used and DBRC is required. A new SLDS is created to replace the one in error. An "ACTIVE PSB" listing is not generated.

```
//EXAMPL13  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='PSBLIST=YES'
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES,CICS=NO
//* NOTE - CICS=NO cannot be specified explicitly
//* NOTE - CICS=YES is not valid
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  SLDS containing prior primary   OLDS
//IEFRDER2  DD ........  SLDS containing prior secondary OLDS
//NEWRDER   DD ........  New primary   SLDS
//NEWRDER2  DD ........  New secondary SLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (may be dynamically allocated)
//SYSIN     DD *
DUP ERRC=00000
```

Note: Specification of ERRC=00000 is required. It is assumed that there is nothing wrong with the SLDS.

If the execution is successful, REP mode is not required. If the execution is unsuccessful, recover the SLDS and then rerun this job.

If an RLDS was created when the input SLDS was created, it must be deleted from the DBRC RECON (use DBRC commands to accomplish this). Deleting the RLDS (and the successful execution of this job) will cause the new SLDS to be used when DBRC generates JCL for data base recovery.

LOG ARCHIVE UTILITY (DFSUARC0)

The Log Archive utility (DFSUARC0) is used to produce an SLDS from a filled OLDS, a batch IMS/VS SLDS, or a CICS/VS log. The utility runs as an OS/VS batch job and multiple log archive jobs can execute concurrently. When dual output is requested, the SLDS consists of primary and secondary data sets.

OLDS ARCHIVE

The online IMS/VS system writes log records to an OLDS in a wraparound fashion. When the OLDS is filled, it can be copied to an SLDS using the Log Archive utility. The SLDS may be on DASD, MSS, or tape.

IMS/VS notifies DBRC whenever an OLDS is filled and/or closed. DBRC updates the RECON data set to indicate that the OLDS is available to be archived.

The archive utility is able to archive multiple OLDSs to a single SLDS as long as the OLDSs being archived were created consecutively by IMS/VS. The JCL supplied to the utility defines which and how many OLDSs are to be archived. The GENJCL facility of DBRC allows you to specify which OLDSs should be included in the created JCL or to specify that all OLDSs not yet archived should be included. If all the specified OLDSs are archived successfully, DBRC updates the RECON data set to indicate that the OLDSs are now available for reuse by the online system. If the archive utility fails, the same archive job should be run again.

If automatic archiving is not specified, you must create the JCL to generate the utility. If you specified automatic archiving, IMS/VS will invoke the DBRC GENJCL function to generate JCL for the utility when the specified number of OLDSs have been filled or closed. (See IMS/VS Data Base Recovery Control: Guide and Reference for more information.) If the DBRC DD statement for the GENJCL output is directed to the internal reader, the archive jobs will be automatically started.

BATCH DASD LOG DATA SET ARCHIVE

An IMS/VS batch system writes log records on an SLDS that may be on tape or DASD. This allows an IMS/VS batch user to log to DASD, create an SLDS, and later copy that SLDS to MSS, DASD, or tape. The input data set can be either single or dual. When the input is from a DASD SLDS created with DBRC present, the archive utility will notify DBRC to update the existing SLDS record(s) with the new SLDS information. You must create the JCL for the archive job of a batch SLDS.

CICS/VS LOG ARCHIVE

CICS/VS logs are treated as SLDSs, and may be used as input to the Log Archive utility. You must create the JCL for the archive job of a CICS/VS log.

OTHER FUNCTIONS

Besides copying log records to an SLDS, the Log Archive utility has the following additional optional functions. These functions can be specified in control statements.

CREATING AN RLDS (RECOVERY LOG DATA SET): You can request creation of an output data set containing all the log records needed for DB recovery. The output data set is referred to as a recovery log data set (RLDS). If the input dataset contains records for DB recovery, the RLDS will be known to DBRC and will be used in place of the SLDS by GENJCL when creating JCL for DB recovery, change accumulation and DSLOG. If the input dataset contains no records needed for DB recovery, the RLDS will be a null dataset. In this case DBRC will record the dataset name and volume serial number of the SLDS, in place of the RLDS DSNAME and volume serial number, and will then use the SLDS for GENJCL instead of the null RLDS. This RLDS will be known to DBRC and will be used (in place of the SLDS) by GENJCL when creating JCL for DB Recovery, Change Accumulation, and DSLOG.

OMITTING LOG RECORDS ON SLDS: Generally, the SLDS should contain all the log records from the OLDS, but if you want to omit some types of log records from the SLDS, these log records must be specified in an SLDS control statement, using the NOLOG parameter. The SLDS must contain those records that may be needed for data base recovery and IMS/VS restart. The Log Archive utility will issue an error message and terminate if a required record type is specified to be omitted.

COPYING LOG RECORDS INTO USER DATA SETS: The Log Archive utility can copy selected log records into multiple user data sets directly. In SYSIN control statements, you can specify the log records to be selected and the ddname of the data set into which the records are to be written.

SPECIFYING USER EXIT ROUTINES: The user can specify multiple user exit routines for the archive utility. The archive utility passes control to each user exit routine at initialization, input log read, and termination time. User exit routines can process the log records or create a data set.

SPECIFYING FORCED END OF VOLUME: To ensure that corresponding volumes in a dual SLDS on tape contain the same records (and consequently are interchangeable), the number of blocks to be written on a volume can be specified. EOV will be forced to both SLDSs when the specified number of log blocks have been written. Figure 85 shows an overview of the Log Archive utility.

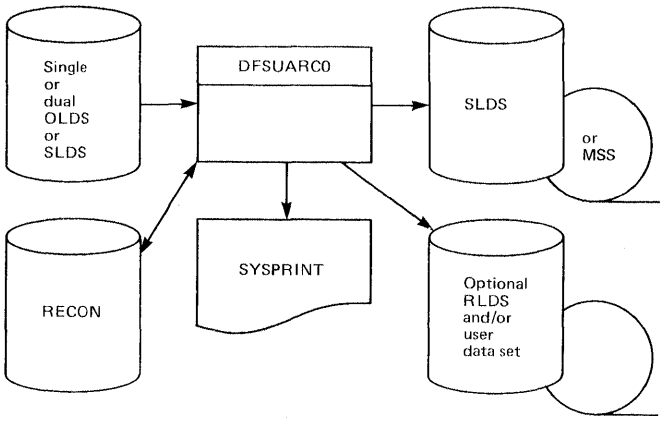


Figure 85. Overview of the Log Archive Utility

OLDS INPUT

The OLDS used for input must have been successfully closed. The status in RECON must be 'ARCHIVE NEEDED'.

An error in a single OLDS causes the archive job to be terminated. The Log Recovery utility should be run to recover the OLDS, and the archive utility can be rerun.

If dual OLDSs were used as input during IMS/VS online execution, both should be used as input to the archive utility. Then if an error in the primary OLDS is encountered, the archive utility will switch to the secondary OLDS. If the record is found in the secondary OLDS, the archive job continues. If an error is encountered in the same block, the archive job will terminate. The Log Recovery utility should be run to recover the OLDS, and the archive utility can be rerun. If one dual OLDS is not available, for example the status is not ARCHIVE NEEDED, only the available OLDS is used as input. The unavailable OLDS is ignored.

If multiple OLDSs are specified as the input OLDS, they must have been created consecutively. The OLDSs created by different IMS/VS system executions cannot be input at one time.

For OLDSs created between recovery points (a recovery point results at every /DBR or /DBD command which forces an OLDS switch), the archive utility performs as follows: If at least one of the SLDSs and RLDSs is placed on DASD, the output data sets will be closed and the archive job will be terminated just after the recovery point. If all SLDSs and RLDSs are placed on tape, "Force EOV" will be performed on all SLDSs and RLDSs and the archive job will continue using the next volume for the SLDS and RLDS.

DBRC verifies the input OLDS. If there is an error in the OLDS specifications, the Log Archive utility will be terminated with an error message.

SLDS INPUT

The SLDS used for input is the SLDS on DASD created by an IMS/VS batch system. The SLDS must have been successfully closed.

Dual SLDSs may be used as input. If an error is encountered in the primary SLDS, the Log Archive utility will switch to the secondary SLDS. If the record is found on the secondary SLDS, the archive job will continue. An error in a single SLDS or errors in the same block in dual SLDSs causes the archive job to be terminated. The Log Recovery utility should be run to recover the SLDS and the archive utility can be rerun.

A CICS/VS log can also be input as an SLDS. For a CICS/VS log, there is only a primary SLDS, and the SLDS must have been successfully closed.

PROGRAM OUTPUT

In addition to the SLDS, the optional RLDS, and the user data set produced as output, the Log Archive utility also produces a listing in SYSPRINT. SYSPRINT contains the following:

- A listing of control statements
- A listing of checkpoint time stamp IDs
- A listing of descriptive messages
- A listing of the result of the archive

Below are examples of the SYSPRINT output:

Listing of Control Statements:

```
********LOG ARCHIVE UTILITY CONTROL STATEMENT********
SLDS -
   NOLOG(10,16,5F,67,69) FEOV(08000)
COPY  DDNOUT1(DATASET1) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(16) FLDLEN(1) COND(E)) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(50) FLDLEN(1) COND(E)) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(51) FLDLEN(1) COND(E)) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(52) FLDLEN(1) COND(E))
EXIT  NAME(UEXIT01)
```

Listing of Checkpoint Log Records:

```
  USER CHECKPOINT RECORD - yy.ddd hh:mm:ss.t CHKPT-id region-id prg-name (v1)(v2)
SYSTEM CHECKPOINT RECORD - yy.ddd hh:mm:ss.t chkpt-id (v1)(v2)
                           CHECKPOINT XXX  (RESTART TTT)
```

When checkpoint log records (X'18' and X'4001') are found, this message shows where they are and the contents of the log records. Date, time, and checkpoint ID are shown for both. Region-id and program name are for X'18' records; checkpoint request type is for X'4001' records, where XXX is the type of checkpoints requested in English. Also shown is the volume serial of the output primary SLDS volume (v1) and, if dual output, the secondary SLDS volume (v2) to which the checkpoint is copied. Restart type is also given for the first X'4001' record where TTT is the type of restart performed in English.

Listing of Descriptive Messages:

```
*** END OF VOLUME FORCED ON SLDS. PRIMARY(volser) SECONDARY(volser) ***
*** WRITE ERROR ON SLDS|USER|RLDS ddname ***
*** OUT-OF-SPACE on SLDS|USER|RLDS ddname ***
*** NO RECORD FOUND FOR SLDS|USER|RLDS ddname ***
```

A dump of the block in error will occur with the following messages: (Refer to "DUMP of Data Record" in the section on the Log Recovery utility.)

```
*** LOG RECORD SEQUENCE NUMBER ERROR IN OLDS ddname BLOCK# nnnnn ***
    kkkkk TO mmmmm
    (kkkkk and mmmmm are the 4-byte hexadecimal log sequence numbers
    of the consecutive records at which the sequence error was detected)
*** READ ERROR IN ddname BLOCK# nnnnn ***
*** TIME OF DAY SEQUENCE NUMBER ERROR IN OLDS ddname BLOCK# nnnnn ***
    kkkkk TO mmmmm
    (kkkkk and mmmmm are the 4-byte hexadecimal OLDS time-of-day values
    extracted from the consecutive blocks)
*** LENGTH ERROR IN ddname BLOCK# nnnnn ***
```

Listing of the Result of the Archive:

```
*** LOG ARCHIVE UTILITY (DFSUARC0)                  **hh:mm yy.ddd **
       COPIED LOG RECORDS

FROM     DDNAME=ddname VOLSER=volser          DDNAME=ddname VOLSER=volser
           (for primary input)                   (for secondary input)
                 .                    .
                 .                    .
TO PRIMARY SLDS DSNAME=dsname
    VOLSER =   volser  volser  volser .............
TO SECONDARY SLDS DSNAME=dsname
    VOLSER =   volser  volser  volser .............

SLDS DOES NOT CONTAIN THE FOLLOWING LOG RECORDS:
  'xx' 'xx' 'xx' 'xx' ......

TO PRIMARY RLDS   DSNAME=dsname
     VOLSER =   volser  volser  volser .............
TO SECONDARY RLDS DSNAME=dsname
     VOLSER =   volser  volser  volser .............
```

JCL REQUIREMENTS

The Log Archive utility executes as a standard OS/VS job. A job statement, an EXEC statement, and DD statements that define input and output are required.

EXEC
defines the program to be run and optional execute parameters. Its format is:

```
//STEP EXEC PGM=DFSUARC0
                PARM= 'ssss, DBRC=ddd, CICS=ccc'
```

ssss
indicates a 1- to 4-character IMS/VS subsystem identifier and must be specified if the input data set is an OLDS. This is the same value as specified for IMSID for the online IMS/VS system that created the data in the OLDS.

ddd
is N | NO | Y | YES.

DBRC=NO may be specified to explicitly declare that DBRC **is not** to be used for this execution of the utility.

DBRC=YES may be specified to explicitly declare that DBRC **is** to be used for the execution of this utility.

If DBRC= is not specified, it will default to YES.

ccc
is Y | YES.

CICS=YES is used when the DFSSLDSP input is a CICS log. This parameter indicates that special sequence checking is to be performed and is required when processing a CICS log.

CICS=NO is the default, and can only be specified by omitting the CICS= parameter completely.

If a parameter is to be allowed to default, the complete parameter (including the keyword) must be omitted from the PARM field.

If no input parameters are specified, the default is DBRC=YES and CICS=NO.

STEPLIB DD
points to IMSVS.RESLIB and/or the program library that contains the Log Archive program and any user exit routines.

DFSOLPnn DD (for primary OLDS)
DFSOLSnn DD (for secondary OLDS)
describes the OLDS used for input. Optionally, you can specify dual OLDSs. In the case of dual OLDSs, the suffixes of the primary and secondary OLDS must match. The value of nn (the suffix) can be 00 through 99 and must be the same ddname that was used when the log data was created by an online execution. All the OLDSs used as input had to be used consecutively during an online execution but, in DD statements, they can be specified in any sequence. You can specify 2 through 99 read buffers.

DFSSLDSP DD (for primary input SLDS)
DFSSLDSS DD (for secondary input SLDS)
specifies the batch SLDS or CICS/VS log used for input. Optionally, for a batch SLDS, the user can specify a dual SLDS. An SLDS used for input is mutually exclusive with an OLDS used for input. You can specify 2 through 99 read buffers.

DFSSLOGP DD (for primary output SLDS)
DFSSLOGS DD (for secondary output SLDS)
defines the SLDS used for output. Its format will depend on the device type being used. If the SLDS is on DASD, you must ensure sufficient space is allocated to contain the log being archived. The SLDS block size may be specified and can be different from the input data set block size. If not specified, the block size of the input data set will be used. The secondary SLDS is optional and specifies dual archiving. If the input is a batch SLDS or CICS/VS log and Log Archive is run with DBRC present, dual output can only be created if dual SLDS records are already known to DBRC. If dual SLDSs are being created, they do not have to have the same block size. However, if FEOV has been specified, it will be ignored unless the block size of both data sets are equal and both are allocated to tape. If tape is specified, it must have a standard label. You can specify 2 through 99 write buffers.

ddname DD (for RLDS and/or user output data set)
defines a user data set and/or recovery log data set (RLDS). If the data set is on DASD, you must ensure

sufficient space is allocated to contain the records to be copied to it. The data set will be created with RECFM=VB. The block size may be specified and can be different from the block size of the input data set. If not specified, the block size of the input data set will be used. If dual data sets are being created, they do not have to have the same block size. You can specify 2 through 99 write buffers.

SYSPRINT
defines the output message data set.

SYSDUMP
defines the dump data set.

SYSIN DD
specifies the control statements. This statement is used only if you have control statements.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

These RECON data set ddnames should not be used if you are using dynamic allocation.

CONTROL STATEMENTS

All control statements are optional. The control statements are required when:

- Using user exit routines
- Creating a RLDS
- Splitting certain records off into a user data set
- Eliminating certain records from being copied to the SLDS
- Forcing duplicate tape output volumes

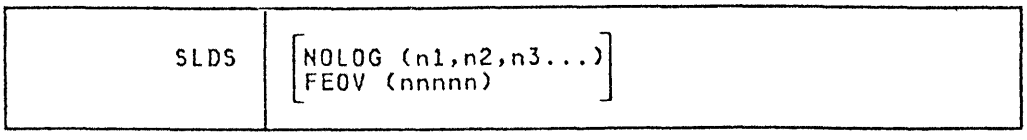
There are three types of control statements, and each statement consists of an operation code and parameters. The rules for using the control statement are:

1. Control statements can be placed in columns 1 to 72 in free format. Parameters can be in any sequence.
2. Each operation code and parameter must be separated with a blank, a comma, or a comment.
3. Multiple lines may be used for a control statement. Continuation characters (+ and -) may be placed between columns 1 and 72. If (+) is used, the lines are concatenated without a blank. If (-) is used, the lines are concatenated with a blank.
4. The value of any parameter must be specified between single parentheses.

SLDS Statement

An SLDS statement specifies log record types that will not be written to the SLDS. It also specifies that end-of-volume should be forced for tape output volumes. If omitted, all log records are copied to the SLDS. Only one SLDS control statement is allowed.

The format of the SLDS control statement is:



NOLOG
defines the log record types that are not to be copied to the SLDS. The value of a NOLOG subparameter should be specified in hexadecimal, for example, SLDS NOLOG (19,1A,1B).

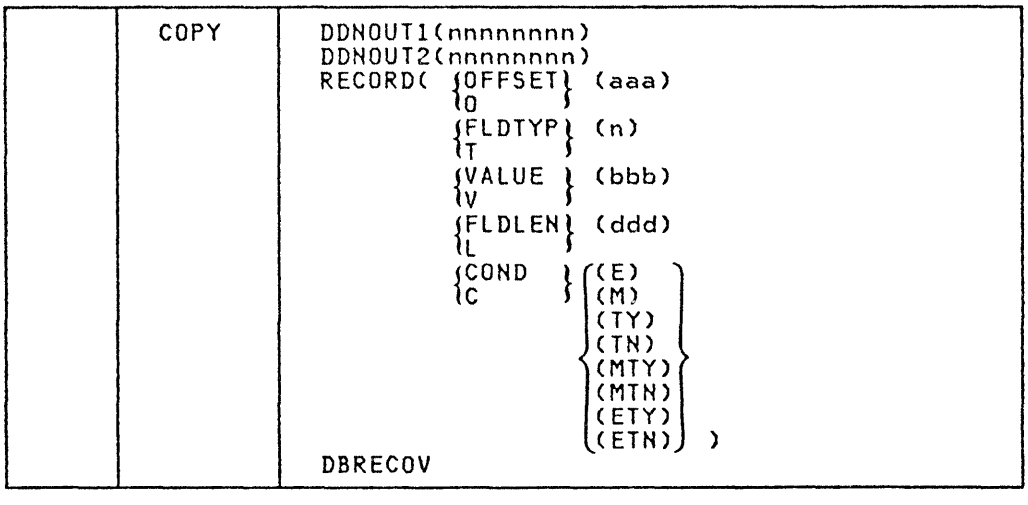
Note: The SLDS must always contain those records that may be needed for data base recovery and for system restart. The Log Archive utility will issue an error message and terminate if a required record type is specified to be omitted.

FEOV
specifies duplicate output tape volumes. This parameter is only applicable in a dual tape SLDS environment and ensures corresponding volumes in a multivolume data set contain the same records (and consequently are interchangeable).

nnnnn indicates the number of blocks to be written to a tape SLDS. Each time the blocks have been written, a FEOV will be issued for both the primary and secondary SLDSs. The block number should be specified in 5 decimal digits. If the block sizes of both SLDSs are not equal, the FEOV parameter is ignored.

COPY Statement

The COPY statement is used to specify that an RLDS or a user data set is to be created during archive. The format for the COPY statement is:



DDNOUT1
DDNOUT2
identifies the ddnames of the data sets. DDNOUT2 only applies if dual copies are being created. The DD statements must be included in the JCL. nnnnnnnn is a ddname value.

RECORD
identifies the conditions for selecting a record to be written to the specified data set.

OFFSET or O
defines the beginning of the field to be tested in the record. The default is position one of the record.

aaa is the value and can be in the range from 1 up to and including the length of the record under test. Maximum value is 32767 bytes. No checking is performed to determine if the logical record length is exceeded. The value specified in the OFFSET keyword is always expressed as relative to byte 1.

FLDTYP or T
defines the type of data in the VALUE field. A value of X or C must be specified.

X defines the data to be treated as hexadecimal character pairs. The test data is packed, two bytes into one, to form hexadecimal equivalents. X is the default.

C defines the data to be treated as EBCDIC.

VALUE or V
Value can be specified in hexadecimal with X or in EBCDIC with C. In the case of EBCDIC, the value can be specified between quotation marks. This quotation mark notation is required when the character string contains a separator of blank or comma. Any characters can be specified within the quotation marks. (Double quotation marks within quotation marks stand for a single quotation mark.) If a minus sign is the last nonblank character, it is assumed that the value is continued on the next line.

bbb value cannot exceed 255 EBCDIC or 510 hexadecimal characters. The length of this field is determined by the FLDLEN value and not by the number of "nonnull" characters in this field.

FLDLEN or L
defines the number of characters to be used from the test field.

ddd represents the actual number of bytes to be used, not the number of characters specified in VALUE. The acceptable range of values for this field is 1 to and including 255. The default is 1.

COND or C
defines the type of test and its relationship to other tests in the group. The default is COND(E).

E marks the last (or only) element in a test series. Any record control statements appearing after this form a new series of tests. This allows various tests to be performed on each record and each test series can be used on different fields within the record.

M indicates this is a multifield test; more than one test is to be made on each input record. All tests in this series must be satisfied before final output selection and processing of this record can begin.

T causes the VALUE byte to be used as a "test under mask" value, instead of a compare field. Only the first byte (two hexadecimal characters if FLDTYP(X)) of the VALUE field will be used. If FLDTYP(C) is used, the hexadecimal equivalent of the EBCDIC character will be the test value. If this parameter is used, the FLDLEN keyword must not be specified and a default length of one will be assumed.

Y indicates that there must be a bit in the record test field for each corresponding bit of the test byte for the "test under mask."This is equivalent to a "branch if ones" test.

N indicates that there must not be a bit in the record test field for any of the correpsonding bits of the test byte for the "test under mask." This is equivalent to a "branch if zeros" test.

MT defines a "test under mask" option but with the properties of a multifield test. This parameter must be used for a multifield test that starts with a "test under mask" value.

ET signifies that a multifield test series ends with a "test under mask" condition.

DBRECOV
indicates that all log records needed for data base recovery should be copied to the specified output data set. This output data set will be known to DBRC and will be used by the GENJCL process in lieu of the created SLDS when creating JCL for DB Recovery, Change Accumulation, or DSLOG. This output data set is called the recovery log data set (RLDS). If there are no records needed for DB recovery, the RLDS will be a null data set. In this case, DBRC will record the DSNAME and volume serial number of the SLDS in place of the RLDS DSNAME and Volume serial number and will use the SLDS for GENJCL instead of the null RLDS.

Note: DDNOUT1 is a required parameter on a COPY control statement. As many RECORD parameters as required can be specified in a COPY control statement. If no RECORD parameter is specified, all log records will be copied to the specified data set. On a given COPY statement, the RECORD parameter and the DBRECOV parameter are mutually exclusive. Multiple COPY control statements may be specified, but only one COPY statement with the DBRECOV parameter is allowed. Two COPY statements must not specify the same output data set.

EXIT Statement

An EXIT statement specifies a user exit routine is to be used.

The format of the EXIT statement is:

```
EXIT   NAME (nnnnnnnn)
```

NAME
specifies the member name of the user exit. The user exit routine will be accessed with a LOAD from the archive utility program; preferably link-edited into JOBLIB/STEPLIB.

Note: Multiple EXIT control statements or multiple NAME parameters may be specified.

ERROR PROCESSING

The Log Archive utility provides the following return codes:

Code	Meaning
0	Archive processing completed successfully.
4	Archive processing completed successfully, but not all OLDS were archived. A recovery point was encountered and end of job was forced. Archive should be rerun for the remaining unarchived OLDS. See sysprint messages.
8	Archive processing completed unsuccessfully. Message DFS3263I indicates the reason.
U3274	ABEND -- DBRC internal failure. Message DFS3274I plus various DSPxxxxx messages indicate the reason.

Descriptions of all error messages issued by DFSUARC0 are provided in IMS/VS Messages and Codes Reference Manual.

EXAMPLES

EXAMPLE 1

The following example is the JCL for the Log Archive utility using the COPY control statement to create an RLDS:

```
//ARCHIVE  JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//ARC1     EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
//*-------------------------*
//*   COPY FROM 3 OLDS TO A SLDS                          *
//*   RLDS AND A USER DATA SET ARE ALSO CREATED           *
//*-------------------------*
//DFSOLP00 DD  DSN=OLP900,DISP=SHR,DCB=(BUFNO=20)
//DFSOLP01 DD  DSN=OLP901,DISP=SHR,DCB=(BUFNO=20)
//DFSOLP02 DD  DSN=OLP902,DISP=SHR
//*-------------------------*
//DFSSLOGP DD  DSN=SLDSP.D82001.N001,UNIT=TAPE,
//       VOL=(,,,99),DISP=(,KEEP),LABEL=(,SL)
//*-------------------------*
//RLDSDD1  DD  DSN=RLDSP.D82001.N001,UNIT=TAPE,
//       VOL=(,,,99),DISP=(,KEEP),LABEL=(,SL)
//*-------------------------*
//USERDD1  DD  DSN=USER.D82001.N001,UNIT=3350,
//       VOL=USER01,DISP=(,KEEP),SPACE=(CYL,5)
//*-------------------------*
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//RECON3   DD  DSN=RECON3,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
 COPY DDNOUT1 (RLDSDD1) DBRECOV
 /*-------------------------*
 /*  THIS USER DATA SET CONTAINS                        */
 /*  X'A5', X'A6', AND X'A7' LOG RECORDS                */
 /*-------------------------*
 COPY DDNOUT1 (USERDD1) -
  RECORD (0(5) T(X) V(A5) L(1) C(E)) -
  RECORD (0(5) T(X) V(A6) L(1) C(E)) -
  RECORD (0(5) T(X) V(A7) L(1) C(E))
 EXIT NAME (UEXIT01)
/*
```

EXAMPLE 2

The following example is the JCL for the Log Archive utility using FEOV to ensure consistency in the SLDS.

```
//ARCHIVE2 JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//ARC2    EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
//*-------------------------*
//*   COPY FROM 2 OLDS TO DUAL SLDS                    *
//*-------------------------*
//DFSOLP02 DD  DSN=OLP902,DISP=SHR
//DFSOLP00 DD  DSN=OLP900,DISP=SHR
//DFSOLS00 DD  DSN=OLS900,DISP=SHR
//DFSOLS02 DD  DSN=OLS902,DISP=SHR
//*-------------------------*
//DFSSLOGP DD  DSN=SLDSP.D82001.N001,UNIT=TAPE,
//      VOL=(,,,99),DISP=(,KEEP),LABEL=(,SL)
//DFSSLOGS DD  DSN=SLDSS.D82001.N001,UNIT=TAPE,
//      VOL=(,,,99),DISP=(,KEEP),LABEL=(,SL)
//*-------------------------*
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
  SLDS FEOV (08000)
 /*-------------------------*
 /*  THE SLDS ARE FORCED EOV AFTER 8000 LOG BLOCKS    */
 /*  ARE WRITTEN.                                     */
 /*-------------------------*
/*
```

EXAMPLE 3

The following example is the JCL for the Log Archive utility to copy a CICS log. Note that DBRC=N is the default if CICS is specified and DBRC=Y must be coded if DBRC is to be used with a CICS log.

```
//ARCHIVE3 JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//*------------------------*
//*   SPECIFY CICS AND DBRC IN PARM                    */
//*------------------------*
//ARC3     EXEC PGM=DFSUARC0,PARM='DBRC=Y,CICS=Y'
//STEPLIB DD DSNAME=IMSVS.RESLIB,DISP=SHR
//*------------------------*
//*   COPY FROM CICS LOG TO SLDS                       */
//*------------------------*
//DFSSLDSP DD  DSNAME=CICS1A,DISP=SHR
//*------------------------*
//DFSSLOGP DD  DSN=SLDSP.D82001.N001,UNIT=TAPE,
//       VOL=(,,,99),DISP=(,KEEP),LABEL=(,SL)
//*------------------------*
//RECON1   DD  DSNAME=RECON1,DISP=SHR
//RECON2   DD  DSNAME=RECON2,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
```

CHAPTER 10. LOG DATA FORMATTING UTILITIES

This chapter describes the (1) IMS/VS Statistical Analysis utility, (2) File Select and Formatting Print Program, (3) IMS/VS Log Transaction Analysis utility, (4) IMS/VS Log Merge utility, and (5) IMS/VS Fast Path Log Analysis utility.

OVERVIEW

The IMS/VS Statistical Analysis utility program modules reside in IMSVS.RESLIB and can be used for analyzing the information on any of the IMS/VS system logs except those from a batch region. Any nonrecoverable and canceled messages are not used. The utility consists of modules DFSISTS0, DFSIST20, DFSIST30, and DFSIST40. These modules must be run in sequence. To run the IMS/VS Statistical Analysis utility on a selected portion of an IMS/VS system log, you must use the IMS/VS Log Transaction Analysis utility to create a new log, tailored to your specifications.

The File Select and Formatting Print Program (DFSERA10) is used with IMS/VS and its related data bases. Its primary function is to assist in the examination and display of data from the IMS/VS log data set.

The IMS/VS Log Transaction Analysis utility program (DFSILTA0) collects information about individual occurrences of IMS/VS transactions based on records in the IMS/VS system log data set. (Logs from a batch region cannot be used.) Any nonrecoverable and canceled messages are not used.

The IMS/VS Fast Path Log Analysis utility (DBFULTA0) is used to prepare statistical reports for Fast Path.

Note: Only the File Select and Formatting Print Program supports CICS/VS.

IMS/VS STATISTICAL ANALYSIS UTILITY

The flow of the IMS/VS Statistical Analysis utility is shown in Figure 86 on page 409.

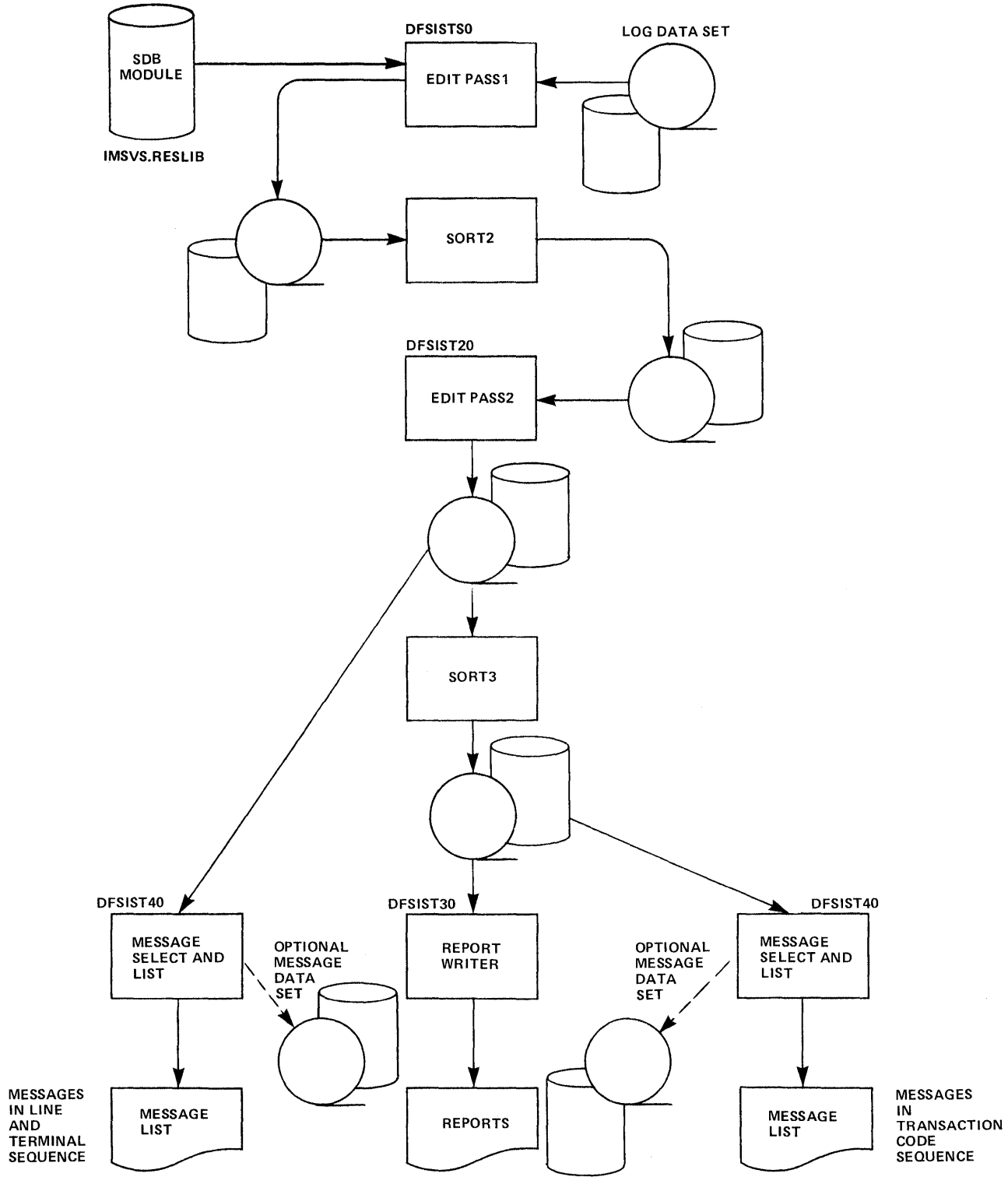


Figure 86. Statistical Analysis Utility Program Flow

Log Records

The log records used by the IMS Statistical Analysis utility are:

01 Input message ready to be put on the destination message queue.

02 Command processed or /LOG command entered.

03 Output message segment ready to be put on the destination message queue.

06 IMS/VS has been started, a date change has occurred, or IMS/VS has been stopped or a FEOV was issued.

07 An application program has been terminated.

33 Message queue manager released a record.

34 Message canceled and a portion of that message has been previously logged.

35 A message has been put on the destination message queue.

36 A message has been taken off the destination message queue.

LOG TYPE 01: Log Type 01 record is written when a message has been completely received by communications and is ready to be put on the destination queue. The destination queue is either a SMB (Scheduler Message Block) or CNT (Communications Name Table). The SMB destination means a transaction code was entered by the terminal operator and an application program will be scheduled. A CNT means a message switch will be done. The terminal operator entered an LTERM and a message. No application program is necessary in this case. The message will be queued for output directly on the LTERM named in the input message.

LOG TYPE 02: This record is created after a command has been successfully completed and before the command completion message is sent. If the command was a /LOG or the command must be reprocessed at restart time, a 02 record is written (for example, /ASSIGN). Type 02 log records are not included in the statistics utilities reports, but can be processed by a user-written routine link-edited with DFSISTO0. The information that may be useful to a user would be the /LOG records. These /LOG records were entered by either the remote terminal operator or the master terminal operator.

LOG TYPE 03: When a segment of a message has been created by an application program and is ready to be put on the destination message queue, the 03 record is written. The destination message queue could be either on SMB or CNT. If an SMB is the destination, then a "program to program" message switch was called for by the application program. If the segment was destined for a CNT, the application program is sending an output message to an LTERM.

In a type 03 record, the date and time fields, PDATE and PTIME, are carried forward from the 01 record. When the statistics utilities are run, the 03 records and 36 records are correlated to determine response time. The time reflected will be from the time the message was put on the input queue (obtained from the 03 record) until the message was released from the output queue (obtained from the associated 36 record).

LOG TYPE 06: Type 06 records are written when IMS/VS is started (during initialization), when IMS/VS terminated (immediately prior to closing the log data set), and when a FEOV is issued.

LOG TYPE 07: This record is the application accounting record of the system. The 07 record is written when an application program has terminated in a message processing or batch-message

processing region.

LOG TYPE 33: When a message is taken off the input message queue or output message queue, the 33 record is written.

LOG TYPE 34: When a message has been canceled and a portion of that message has already been logged, a 34 record is written.

LOG TYPE 35: When a message (input or output) has been put on the destination queue, the 35 record is written.: If the message is very long and requires more than one input message buffer, the record will have the date and time in it. This is because the date and time in 01 record would be invalid under this condition.

LOG TYPE 36: When a message has been sent in its entirety and the message is ready to be released from the queue, a 36 record is written. On all devices except display devices, the message is ready to be released from the queue as soon as the last segment has been successfully sent to the terminal. Display devices are a little different. If the display output is only a single page in length, then the message will be dequeued after the last segment has been successfully sent. The following description is for multiple pages of display output.

The PAGDEL option selected on the TERMINAL macro at system definition time will determine when the message is ready to be released from the queue. If option=PAGEDEL (or PAGEDEL=YES) is specified, the message will be dequeued when a question mark, PA2 key, or a new input transaction has been entered from the terminal. Options=NPGDEL (or PAGEDEL=NO) requires a question mark or PA2 key to be entered to cause the message to be taken off the output queue and the 36 record to be written.

The effect of options=NPGDEL (or PAGEDEL=NO) on response time, as the Statistical Analysis utility views it, can be dramatic. If the terminal operator leaves the current message displayed for a long period or powers off the video device, the message is not removed from the output queue and the 36 record is not written, until terminal operations begin again. Consequently, response time could appear to require many hours or even days.

SORT AND EDIT PASS1 (DFSISTS0)

The functions of SORT and EDIT PASS1 are to:

- Select from the system logs those records used by the statistics programs (logs from batch regions do not contain the desired records, and so cannot be used)
- Sort message and queue manager log records so that all segments of a multisegment message appear together, and enqueue and dequeue records associated with the messages to which they refer
- Edit the records so that, when sorted, the input message, and all output sent as a result of that input, are contiguous Any nonrecoverable and canceled messages are not used.

Concatenation of logs from multiple systems is permitted.

The JCL for SORT and EDIT PASS1 must contain a JOBLIB or STEPLIB statement for the library containing the utility program (IMSVS.RESLIB).

The ',NOTXT' parameter causes the program to ignore the text of the X'01' (input message) and X'03' (output message) log records, thereby reducing the volume of all sort passes. If this parameter is used, Message Select and List (DFSIST40) cannot be run. The DFSISDBX suffix is no longer used and will be ignored if it is specified.

See "Omitting Log Records on SLDS" in Chapter 9 for information on how to use the Log Archive utility to delete log records.

EDIT PASS2 (DFSIST20)

The function of EDIT PASS2 is to take from system messages the records to be used to produce the statistical reports. If DFSIST40 (Message Select and Copy) is not run as part of the statistics job stream, approximately 40% of the output of DFSIST20 can be eliminated by coding NOTXT on DFSISTSO or NOLOG in the SLDS control statement of the Log Archive utility.

REPORT WRITER (DFSIST30)

The function of DFSIST30 is to produce the final statistical reports.

The different types of statistical reports are described below:

1. Messages Queued But Not Sent—by Destination.

   - The output message (X'03') appears on the log, but no record (X'36') appears to indicate that the message was sent to the terminal. Sorted by symbolic terminal name.

2. Messages—Program To Program—by Destination.

   - An output message (X'03') was sent to an SMB. Sorted by destination.

3. Line and terminal report shows the line and terminal loading by time of day. (Could be used to determine the line and terminal utilization, peak traffic periods, etc.)

   - Counts input messages (R), X'01', to IMS/VS from each LTERM, and output messages (S), X'03', to each LTERM from IMS/VS. The report is arranged in line number, relative terminal sequence. A message switch counts as two messages; one from the originating terminal, one to the destination terminal. A broadcast message counts as one message from the originating terminal, and one message each to the destination terminals.

   Note: The next four reports deal with transaction codes. If an output message was generated by a command from a different terminal, the input prefix data is replaced by the message "THIS OUTPUT NOT RESULT OF INPUT." An X'03' message generated by the system, independent of terminal input, would have a transaction code of IMSSYS. If an output message was generated by an input message that was not on the log or by a command from the same terminal (for example, DISPLAY), the transaction code is NOTAVA; otherwise, the transaction code can be found in the generating X'01' log record.

4. Messages Queued But Not Sent—by Transaction Code.

   - The output message (X'03') appears on the log, but no record (X'36') appears to indicate that the message was sent to the terminal. Sorted by transaction code.

5. Messages—Program To Program—by Transaction Code.

   - An output message (X'03') was sent to an SMB. Sorted by transaction code.

6. Transaction Report.

   - This report shows loading by transaction code and by time of day. The time for input messages to IMS/VS from each logical terminal is indicated by "R"; the time for output messages to each logical terminal from IMS/VS is indicated by "S." The report counts the same messages as the "Line and Terminal Report" described in 3 above. Input is sorted by transaction code.

   - The transaction code column may contain the following entries:

     - (NOSORC) - The output message was generated by a command.

     - (NOTAVA) - The output message was generated by an input message that was not on the input log.

     - (IMSSYS) - The output message was generated by IMS/VS.

7. Transaction Response Report.

   - Measures two response times. The first line is response time from complete receipt of the input message (enqueue time X'35') until the response message to the terminal is successfully dequeued (X'36'). The second line is response time from complete receipt of the input message (enqueue time X'35') until the response message to the terminal is started (GU time X'31').

     Note: There can be multiple responses from a single transaction, and they can include any output messages from program-to-program switch transactions that are a result of the original input message.

   - Percentile report shows shortest response, longest response, and 25th, 50th, 75th, and 95th percentile response. A response time within the nth percentile is greater than or equal to n% of the total number of response times processed for that transaction code. For example, a 04.3S number under the '75% RESPONSE' column means that 75% of the total responses for that transaction were equal to or less than 04.3 seconds.

8. Application Accounting Report.

   - Provides sufficient data to allow machine charges to be distributed among application programs and/or transaction codes.

   - The following information is contained in this report:

     - Counts of all requests to Data Language/I

     - Amount of processor task time

     All requests for services from Data Language/I, either for access to messages or data bases, are counted. These counts are accumulated by program, by transaction code within program, and by priority within transaction code.

     Counts of messages processed, and of "get uniques," are included. (Will be different because of "get unique" issued on which end-of-file is returned.)

     Task time is set when a request for scheduling is made. (The value is the maximum time for each transaction, multiplied by the maximum number of transactions.) Remaining time is requested prior to the next request for scheduling. (This time is the actual time the program executed. It does not include any wait time for

accessing data. This time can also be incorrect if the application program is a BMP and issues a TTIMER or STIMER macro.)

Average Processor Time is the total message processor time, divided by the number of messages. It is not rounded. The final average processor time is a recalculated average.

Number of Bad Completion Codes reflects the number of times an application program terminated abnormally, or returned with other than zero in register 15.

9. IMS/VS Accounting Report.

   - Shows start and stop times for the IMS/VS control region.

10. Operating Information.

    - Reports (Items 1 through 7, above) produced, either with or without date control.
    - Program determines if input was sorted on date.
    - Control break occurs whenever date changes, totals printed, and new report started.
    - If not sorted on date, all the log data sets should be processed at one time for a period (such as one week) to produce one summary report.
    - The LINECNT=XX parameter can be included on the EXEC statement for the REPORT WRITER (DFSIST30). This is the only parameter expected, and it is optional. If it is not included, the default line count is 36.
    - Printing of the different statistical reports is not optional; they are all generated.

MESSAGE SELECT AND COPY OR LIST (DFSIST40)

The execution of the Message Select and Copy or List program is optional; it can be executed as a separate step in the same job with the statistical reports, or it can be run independent of the statistical reports.

This program takes output from the second edit program, DFSIST20, before it is sorted (when in line-and-terminal sequence), or after sorting (in transaction-code sequence). Input to this program is specified on the IMSLOGI DD statement, described later in this chapter. To have messages printed in the sequence they occurred (that is, each input message associated with its output message), the input to this program must be &&ED34IN.

Message Select and Copy or List selects messages based on control statements read from SYSIN. Messages selected are printed and/or copied onto an output data set. If a DD statement named IMSLOGO is included, an output data set is created. If a DD statement named IMSLOGP is included, messages selected are printed. Both DD statements can be included in a single run.

UTILITY CONTROL STATEMENTS

All control statements begin in position 1, with a keyword identifying that control statement. Following the keyword is a series of parameters, enclosed within parentheses and separated by commas. Control statements cannot be continued beyond position 71. Multiple control statements with the same keyword, starting in position 1, are permitted. Within parentheses, all

parameters are positional; missing parameters must be indicated by commas. Messages are selected if they fulfill at least one of the criteria specified by the control statement.

A group of names can be indicated by terminating the parameter with an *. Example: INV* causes the names INV, INVENTORY, INVA, and INVB to be selected.

The name parameter ALL may be used to select all names rather than a specified name.

Transaction Code Control Statement

An example of the format of the transaction code control statement is:

```
TRANS CODE=(TRANSCOD,I,O),(TRANSA,I),(INV*,,O),(ALL,I,O)
```

- The first parameter is a transaction code of from 1 to 8 characters.
- The second parameter is I, to indicate that input messages with this code are to be selected.
- The third parameter is an O, to indicate that output messages resulting from this code are to be selected.
- A transaction code of ALL indicates selection of all transaction codes.
- An asterisk within the transaction code causes only characters preceding the asterisk to be compared with the corresponding number of characters from the input transaction code to determine selection. This may also be used to select groups of transaction codes.

Symbolic Terminal Name Control Statement

Examples of the symbolic terminal name control statement are:

```
SYM NAME=(TERMA,I,O),(TERM*,I),(TERMINV,,O,ALL)
SYM NAME=(TERMPAY,I,O,TERMA)
SYM NAME=(ALL,,O,TERMA)
```

- The first parameter is a symbolic terminal name of from 1 to 8 characters.
- The second parameter, I, selects input from this terminal.
- The third parameter, O, selects output generated by input from this terminal.
- The output parameter, O, may be further qualified with another symbolic terminal name. This causes only output to that symbolic name which resulted from inputs from the preceding name to be selected. If ALL is specified, all output resulting from the terminal represented by the preceding name is selected.

Hardware Terminal Address Control Statement

An example of the format of the hardware terminal address control statement is:

```
TERM ADDR=(3,1,I,O),(42,3,,O,21,A),(1,ALL,I,O)
```

- Selection by hardware terminal name is similar to selection by terminal symbolic name, except that, instead of symbolic name, line number and relative terminal number are specified.

- The first parameter is the line number or ALL.
- The second parameter is the relative terminal number on the line or ALL.
- The third and fourth parameters are I and O for selection of input to and output from this terminal.
- Output may be further qualified (similar to symbolic terminal output).
- If an output message was queued but not sent, it is not selected, even if ALL is specified. The SYM NAME control statement must be used.

Time Control Statement

An example of the format of the time control statement is:

```
TIME=(74014,1620,74015,1900)
```

- The first parameter is starting date—year (YY) and day of year (DDD).
- The second parameter is starting time—hours (HH) and minutes (MM).
- The third parameter is ending date—year (YY) and day of year (DDD).
- The fourth parameter is ending time—hours (HH) and minutes (MM).
- If the time control statement is included, only messages specified by a transaction code statement or a terminal control statement and falling within the specified times are selected.

Nonprintable Character Control Statement

The format of the nonprintable character control statement is:

```
NON PRINT=HEX
```

- If this control statement is included, nonprintable characters are printed in hexadecimal, on two lines, with one hexadecimal character above the other. If this statement is not included, nonprintable characters appear as blanks.

JCL REQUIREMENTS

The JCL for execution of the IMS/VS Statistical Analysis utility is given in Figure 87 on page 420. Also see the appropriate operating system Sort/Merge program manual.

Note: The LRECL and BLKSIZE for the log can be calculated as follows:

LRECL = (larger of 1032 or RECLNG operand of the MSGQUEUE macro for IMSVS.LGMSG) + 16.

BLKSIZE = larger of LRECL + 4 or BLKSIZE operand on the IEFRDERDD statement in the "IMS" cataloged procedure.

//JOBLIB DD
Describes the program library containing the utility programs. Its format is:

```
//JOBLIB DD DSN=IMSVS.RESLIB,DISP=SHR
```

// EXEC
Invokes the initial selection and sort process in the statistics program jobstream. If DFSIST40 will not be run, 'NOTXT' improves performance. Its format is:

```
//ST1 EXEC PGM=DFSISTS0[PARM=',NOTXT']
```

//LOGIN DD
Describes the input log data set. Multiple volumes and data sets can be concatenated within one statistics program run, if desired. Its format is:

```
//LOGIN DD DSN=IMSLOG,DISP=OLD,VOL=SER=XXXXXX,UNIT=TAPE
```

where XXXXXX is the volume serial number of the log data set being processed. (For this example, assume that LRECL=3964 and BLKSIZE=3968.)

//SORTWK01-32 DD
Describes the sort program's work data sets. The space defined can vary. The number of data sets must be at lease three. They can be on either tape or disk. For a disk sort, the format is:

```
//SORTWKnn DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
```

//SORTLIB DD
Describes the library containing the sort program's modules. Its format is:

```
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
```

//SORTOUT DD
The sort program's output data set is not used; however, the DCB information must appear on the DD statement. Its format is:

```
//SORTOUT DD DUMMY,DCB=*.LOGIN
```

//LOGOUT DD
Describes the output of DFSISTS0. This data set is normally a temporary one. It serves as input to the next step (a sort). If you want to break the statistics program into multiple jobs, you must modify this statement accordingly. Its normal format is:

```
//LOGOUT DD DSN=&&EDIT1,DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(CYL,(5,5)),DCB=(RECFM=VB,
// BLKSIZE=3996,LRECL=3992,BUFNO=3)
```

BLKSIZE and LRECL can be changed here and in subsequent steps. LRECL must be at least as large as LRECL for LOGIN above, plus 28 bytes for additional prefix information. Space is, of course, a function of the volume of input.

//SYSPRINT DD
Describes the output data set for control messages. Its format usually is:

```
//SYSPRINT DD SYSOUT=A
```

//SYSOUT DD
Describes the message output data set. Its format is:

```
//SYOUT DD SYSOUT=A
```

// EXEC
Executes the sort program. Efficiency can be improved by providing as much main storage as possible. Sort computer storage size needed based on number of work data sets allocated. Its format is:

```
//ST2 EXEC PGM=SORT,REGION=256K
```

//SYSOUT DD
Describes the message output data set. Its format is:

```
//SYSOUT DD SYSOUT=A
```

//SORTIN DD
Describes the input data set to the sort. It is the data set described by the DD statement LOGOUT in the previous step. Its format is:

```
//SORTIN DD DSN=&&EDIT1,DISP=(OLD,DELETE)
```

//SORTLIB DD
Describes the library containing the sort program's modules. Its format is:

```
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
```

//SORTOUT DD
Describes the output data set to the sort. Its format is:

```
//SORTOUT DD DSN=&&EDIT1S,DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(CYL,(5,5)),DCB=*.SORTIN
```

Note: BLKSIZE and LRECL are the same as for EDITDCB1 above.

//SORTWK01-32
These describe sort work data sets. The number of sort work data sets may vary according to normal sort requirements (for example, tape sort or disk sort). The format for a disk sort is:

```
//SORTWKnn DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
```

//SYSIN DD
Describes the sort's control data set normally in the input stream. It therefore is normally a DD * statement. (Followed by the sort control statement.)

SAMPLE SORT CONTROL STATEMENT:

```
SORT FIELDS=(5,1,CH,A,9,4,PD,A,13,36,CH,A),SIZE=XXXX
```

//EXEC
Invokes the second statistics edit module, DFSIST20. If DFSIST40 (Message Select and Edit) is not going to be run, a parameter field of NOTXT should be specified on DFSISTS0. Its format is:

```
//ST3 EXEC PGM=DFSIST20
```

//EDITDCB1 DD
Describes the input data set to DFSIST20 (output of sort in previous step). Its format is:

```
//EDITDCB1 DD DSN=&&EDIT1S,DISP=(OLD,
// DELETE)
```

//EDITDCB2 DD
Describes the output of DFSIST20. Its format is:

```
//EDITDCB2 DD DSN=&&EDIT2,DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(CYL,(5,5)),
// DCB=(RECFM=VB,BLKSIZE=4016,LRECL=4012,
// BUFNO=3)
```

Note: LRECL must be 20 bytes larger than SORTOUT in the previous step. BLKSIZE must be increased proportionately.

//SYSPRINT DD
Describes the output data set for control messages. Its format usually is:

```
//SYSPRINT DD SYSOUT=A
```

The next step is again a sort, and all DD statements, with the exception of SORTIN and SORTOUT, are the same as the previous sort.

//SORTIN DD
Refers to the output of DFSIST20. Its format is:

```
//SORTIN DD DSN=&&EDIT2,DISP=(OLD,DELETE)
```

//SORTOUT DD
Describes the output of the sort that serves as input to DFSIST30 and/or DFSIST40. Its format is:

```
//SORTOUT DD DSN=&&ED34IN,UNIT=SYSDA,

    DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
    DCB=*.SORTIN
```

The normal sort control statement is as shown below.

```
SORT FIELDS=(5,1,CH,A,13,40,CH,A),SIZE=XXXX
```

To sort on date and produce reports under date control, the statement would be as follows:

```
SORT FIELDS=(5,1,CH,A,9,4,PD,A,13,40,CH,A),SIZE=XXXX
```

// EXEC
Invokes DFSIST30 (Report Writer). Its format is:

```
//ST5 EXEC PGM=DFSIST30[,PARM='LINECNT=XX']
```

If LINECNT is not specified, the default is 36.

//EDITDCB2 DD
Describes the input to the report writer (the output of the previous sort). Its format is:

```
//EDITDCB2 DD DSN=&&ED34IN,DISP=(OLD,PASS)
```

//PRINTDCB DD
Describes the output of the report writer, normally the output stream. It may be blocked or unblocked, because I/O is performed using QSAM, with QSAM acquiring the buffers. Its format is:

```
//PRINTDCB DD SYSOUT=A,DCB=(BLKSIZE=133,
// LRECL=133,RECFM=FA)
```

//SYSPRINT DD
Describes the output data set for control messages. Its format usually is:

```
//SYSPRINT DD SYSOUT=A
```

// EXEC
Invokes Message Select and Edit (DFSIST40). This step is optional, and a number of options apply. (See "Message Select and Copy or List.") This format of the statement is:

```
//ST6 EXEC PGM=DFSIST40[,PARM='LINECNT=XX']
```

If LINECNT is not specified, the default is 36.

//IMSLOGI DD
Describes the input to Message Select and Copy or List (DFSIST40). This program uses as input the same data set as DFSIST30 (that is, output of the second sort) or the output of Edit Pass 2 directly. The format of this statement (assuming use of sorted input) is:

```
//IMSLOGI DD DSN=&&ED34IN,DISP=(OLD,DELETE)
```

//IMSLOGP DD
Describes the program's output. The format of the statement is the same as the PRINTDCB statement for DFSIST30.

//IMSLOGO DD
This optional DD statement can be used if you want to create a data set composed of your messages. Its format is:

```
// IMSLOGO DD DSN=OUTPUT,UNIT=tttt,
// DISP=(NEW,KEEP),DCB=(RECFM=VB,
// LRECL=4012,BLKSIZE=4016)
```

//SYSPRINT DD
Describes the output data set for control messages. Its format usually is:

```
//SYSPRINT DD SYSOUT=A
```

//SYSIN DD
Describes the control data set for DFSIST40. It is normally a DD * statement. See the preceding section entitled "Transaction Code Control Statement"for the control statement format.

Figure 87 is an example of the JCL for the Statistical Analysis utility program. This is a full statistics, job-stream example with sorting by date that produces reports under date control. BLKSIZE and LRECL in all data sets are dependent on the input log.

```
//STATS     JOB 1,NAME,MSGCLASS=A,MSGLEVEL=1,PRTY=8
//JOBLIB    DD DSN=IMSVS.RESLIB,DISP=SHR
//ST1     EXEC PGM=DFSISTS0
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//LOGIN     DD UNIT=TAPE,DISP=OLD,VOL=SER=LOGTAP,DSN=IMSLOG
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT   DD DUMMY,DCB=*.LOGIN
//LOGOUT    DD DSN=&&EDIT1,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(5,5)),
//             DCB=(RECFM=VB,BLKSIZE=3996,LRECL=3992,BUFNO=3)
/*
//ST2     EXEC PGM=SORT,REGION=256K
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSOUT    DD SYSOUT=A
//SORTIN    DD DSN=&&EDIT1,DISP=(OLD,DELETE)
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT   DD DSN=&&EDIT1S,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(5,5)),
//             DCB=*.SORTIN
//SYSIN     DD *
 SORT    FIELDS=(5,1,CH,A,9,4,PD,A,13,36,CH,A),SIZE=E200
/*
```

Figure 87 (Part 1 of 2). JCL for the Statistical Analysis Utility Program

```
//ST3      EXEC PGM=DFSIST20
//EDITDCB1   DD DSN=&&EDIT1S,DISP=(OLD,DELETE)
//EDITDCB2   DD DSN=&&EDIT2,UNIT=SYSDA,DISP=(NEW,PASS),
//              SPACE=(CYL,(5,5)),
//              DCB=(RECFM=VB,BLKSIZE=4016,LRECL=4012,BUFNO=3)
//SYSPRINT   DD SYSOUT=A
/*
//ST4      EXEC PGM=SORT,REGION=256K
//SORTLIB    DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSOUT     DD SYSOUT=A
//SORTIN     DD DSN=&&EDIT2,DISP=(OLD,DELETE)
//SORTWK01   DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02   DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03   DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT    DD DSN=&&ED34IN,UNIT=SYSDA,DISP=(NEW,PASS),
//              SPACE=(CYL,(1,1)).
//              DCB=*.SORTIN
//SYSIN      DD *
 SORT    FIELDS=(5,1,CH,A,9,4,PD,A,13,40,CH,A),SIZE=E200
/*
//ST5      EXEC PGM=DFSIST30
//EDITDCB2   DD DSN=&&ED34IN,DISP=(OLD,PASS)
//PRINTDCB   DD SYSOUT=A,DCB=(BLKSIZE=133,LRECL=133,RECFM=FA)
//SYSPRINT   DD SYSOUT=A
/*
//ST6      EXEC PGM=DFSIST40
//IMSLOGP    DD SYSOUT=A,DCB=(BLKSIZE=133,LRECL=133,RECFM=FA)
//IMSLOGI    DD DSN=&&ED34IN,DISP=(OLD,DELETE)
//SYSPRINT   DD SYSOUT=A
//SYSIN      DD *
TRANS CODE=(ALL,I,O)
NON PRINT=HEX
/*
//
```

Figure 87 (Part 2 of 2). JCL for the Statistical Analysis Utility Program

STATISTICS REPORTS EXAMPLES

Following is a list of types of statistics reports produced by the Report Writer (DFSIST30). Examples of each of the reports follow. (The report date, which is in the upper right corner of these examples, will not appear unless a sort by date is specified.)

- Messages queued but not sent (by destination).
- Messages—program to program (by destination).
- Line and terminal.
- Messages queued but not sent (by transaction code).
- Messages—program to program (by transaction code).
- Transaction.
- Transaction response.
- Application accounting.
- IMS/VS accounting.
- Messages. (This report is produced by DFSIST40, Message Select and Copy.)

```
    MESSAGES  --  QUEUED BUT NOT SENT          DATE 03/06/82

                  TOTAL
DESTINATION       MESSAGES

ELEANOR               1
SW1050                1
T2741N1               1
T2742N3               1
```

```
    MESSAGES -- PROGRAM TO PROGRAM            DATE 03/06/82

                  TOTAL
DESTINATION       MESSAGES

PA10                  107
```

RELATIVE TERMINAL NUMBER

LINE AND TERMINAL REPORT DATE 03/06/76 PAGE 1

LINE RTN	R/S	TOTAL MESSAGES	TOTAL CHARACTERS	AVG SIZE	HOURLY DISTRIBUTION 00-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-24
001 01																		
WTOR	S	4	76	19	0	4	0	0	0	0	0	0	0	0	0	0	0	0
003 01																		
MODEL2	S	120	6,183	51	0	120	0	0	0	0	0	0	0	0	0	0	0	0
010 01																		
TST3270	R	120	3,120	26	0	120	0	0	0	0	0	0	0	0	0	0	0	0
013 01																		
T2741	S	117	5,482	46	0	117	0	0	0	0	0	0	0	0	0	0	0	0
	R	118	3,168	26	0	118	0	0	0	0	0	0	0	0	0	0	0	0
025 01																		
ELEANOR	S	312	14,052	46	0	312	0	0	0	0	0	0	0	0	0	0	0	0
	R	313	8,238	26	0	313	0	0	0	0	0	0	0	0	0	0	0	0
02																		
T2741N1	S	418	19,528	46	0	418	0	0	0	0	0	0	0	0	0	0	0	0
	R	419	10,694	25	0	419	0	0	0	0	0	0	0	0	0	0	0	0
03																		
T2741N3	S	102	4,602	45	0	102	0	0	0	0	0	0	0	0	0	0	0	0
	R	104	2,804	26	0	103	0	0	0	0	0	0	0	0	0	0	0	0
04																		
SW1050	S	367	16,972	46	0	367	0	0	0	0	0	0	0	0	0	0	0	0
	R	368	9,416	25	0	368	0	0	0	0	0	0	0	0	0	0	0	0
LINE TOTALS	S	1,199	55,154	45	0	1199	0	0	0	0	0	0	0	0	0	0	0	0
	R	1,204	31,162	26	0	1204	0	0	0	0	0	0	0	0	0	0	0	0
SYSTEM TOTALS	S	1,440	66,895	46	0	1440	0	0	0	0	0	0	0	0	0	0	0	0
	R	1,442	37,440	26	0	1442	0	0	0	0	0	0	0	0	0	0	0	0

LOGICAL TERMINAL NAME

SENT

RECEIVED

```
    MESSAGES -- QUEUED BUT NOT SENT          DATE 03/06/82

TRANSACTION     TOTAL
CODE            MESSAGES

(IMSSYS)            5
```

```
    MESSAGES -- PROGRAM TO PROGRAM           DATE 03/06/82

TRANSACTION     TOTAL
CODE            MESSAGES

TA10                107
```

T R A N S A C T I O N R E P O R T D A T E 03/15/76 P A G E 1

TRANSACTION CODE	R/S	TOTAL MESSAGES	TOTAL CHARACTERS	AVG SIZE	HOURLY DISTRIBUTION 00-07	07-08	08-09	09-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-24
(NOSORC)	S	5	296	59	0	0	0	5	0	0	0	0	0	0	0	0	0	0
ADDINV	S	1	57	57	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	R	1	23	23	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ADDPART	S	1	48	48	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	R	1	25	25	0	0	0	1	0	0	0	0	0	0	0	0	0	0
CLOSE	S	2	89	44	0	0	0	2	0	0	0	0	0	0	0	0	0	0
	R	1	29	29	0	0	0	1	0	0	0	0	0	0	0	0	0	0
DISBURSE	S	2	90	45	0	0	0	2	0	0	0	0	0	0	0	0	0	0
	R	1	31	31	0	0	0	1	0	0	0	0	0	0	0	0	0	0
DLETINV	S	1	61	61	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	R	1	24	24	0	0	0	1	0	0	0	0	0	0	0	0	0	0
DLETPART	S	1	52	52	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	R	1	17	17	0	0	0	1	0	0	0	0	0	0	0	0	0	0
DSPALLI	S	1	462	462	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	R	1	16	16	0	0	0	1	0	0	0	0	0	0	0	0	0	0
DSPINV	S	4	1,120	280	0	0	0	4	0	0	0	0	0	0	0	0	0	0
	R	4	95	23	0	0	0	4	0	0	0	0	0	0	0	0	0	0
PART	S	1	140	140	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	R	1	13	13	0	0	0	1	0	0	0	0	0	0	0	0	0	0
SYSTEM	S	19	2,415	127	0	0	0	19	0	0	0	0	0	0	0	0	0	0
TOTALS	R	12	273	22	0	0	0	12	0	0	0	0	0	0	0	0	0	0

T R A N S A C T I O N R E S P O N S E R E P O R T D A T E 03/15/76 P A G E 1

TRANSACTION CODE	TOTAL RESPONSES	LONGEST RESPONSE	95% RESPONSE	75% RESPONSE	50% RESPONSE	25% RESPONSE	SHORTEST RESPONSE
ADDINV	1	00.9S	00.9S	00.9S	00.9S	00.9S	00.9S
	1	00.9S	00.9S	00.9S	00.9S	00.9S	00.9S
ADDPART	1	01.4S	01.4S	01.4S	01.4S	01.4S	01.4S
	1	01.4S	01.4S	01.4S	01.4S	01.4S	01.4S
CLOSE	2	00.7S	00.7S	00.7S	00.7S	00.7S	00.7S
	2	00.7S	00.7S	00.7S	00.7S	00.7S	00.7S
DISBURSE	2	00.6S	00.6S	00.6S	00.6S	00.6S	00.6S
	2	00.6S	00.6S	00.6S	00.6S	00.6S	00.6S
DLETINV	1	00.6S	00.6S	00.6S	00.6S	00.6S	00.6S
	1	00.6S	00.6S	00.6S	00.6S	00.6S	00.6S
DLETPART	1	00.6S	00.6S	00.6S	00.6S	00.6S	00.6S
	1	00.6S	00.6S	00.6S	00.6S	00.6S	00.6S
DSPALLI	1	22.5S	22.5S	22.5S	22.5S	22.5S	22.5S
	1	22.4S	22.4S	22.4S	22.4S	22.4S	22.4S
DSPINV	4	01.1S	01.1S	00.6S	00.5S	00.5S	00.5S
	4	01.0S	01.0S	00.6S	00.5S	00.5S	00.5S
PART	1	07.1S	07.1S	07.1S	07.1S	07.1S	07.1S
	1	07.1S	07.1S	07.1S	07.1S	07.1S	07.1S
TOTAL RESPONSES		14					

A P P L I C A T I O N A C C O U N T I N G R E P O R T D A T E 03/15/76 P A G E 1

PROGRAM NAME	TRANSACTION CODE	PRI	MESSAGE- QTY	GU	COUNTS GN	ISRT*	DATA BASE GU	GN	GNP	GHU	GHN	GHNP	COUNTS ISRT	DLET	REPL	CC OR RC NOT 0	TOT MESS CPU TIME	AVR TIME
DFSSAM02	PART	07	1	1	0	2	1	1	0	0	0	0	0	0	0	0	00.1S	0.106S
DFSSAM03	DSPINV	07	4	4	0	16	8	8	0	0	0	0	0	0	0	0	00.4S	0.104S
DFSSAM04	ADDINV	07	1	1	1	1	0	0	0	0	0	0	1	0	0	0	00.0S	0.083S
	ADDPART	07	1	1	1	1	0	0	0	0	0	0	2	0	0	0	00.0S	0.091S
	DLETINV	07	1	1	1	1	0	0	0	1	0	0	0	1	0	0	00.0S	0.098S
	DLETPART	07	1	1	1	1	1	1	0	1	0	0	0	1	0	0	00.0S	0.097S
	********	**	4	4	4	4	1	1	0	2	0	0	3	2	0	0	00.3S	0.092S
DFSSAM05	CLOSE	07	1	1	0	3	0	0	0	1	0	0	0	0	1	0	00.1S	0.113S
DFSSAM06	DISBURSE	07	1	1	0	3	0	0	0	1	0	0	0	0	1	0	00.0S	0.084S
DFSSAM07	DSPALLI	07	1	1	0	6	1	5	0	0	0	0	0	0	0	0	00.1S	0.158S
SYSTEM TOTALS			12	12	4	34	11	15	0	4	0	0	3	2	2	0	01.2S	0.104S

* INDICATES TOTAL SHOWN IN 100'S

@ INDICATES TOTAL SHOWN IN 10,000'S

I M S ACCOUNTING REPORT D A T E 08/10/76 P A G E 1

START TIME 19:33:04 (START TIME ON FIRST DAY)
I M S DAY 08/10/76**
I M S DAY 08/16/76**
STOP TIME 18:35:16 (STOP TIME ON LAST DAY)

REPORT PERIOD IS FROM 08/10/76 TO 08/16/76.

END OF REPORTS

* Second insert is counted for single user issued insert if all the following conditions are met:
1. New HIDAM Root
2. ISAM/OSAM Index
3. Not Duplicate Key (II status not returned)

**These dates will not appear unless the input to DFSIST30 is sorted with date control.

```
                    M E S S A G E S

OUTPUT SEG=001 LEN=073 *DFS994I *CHKPT 75074/090308**004871********SIMPLE**COLD START COMPLETED. *
INPUT   TRANSACTION   LINE   RELA   SEQ   SYMBOLIC                        OUTPUT   LINE   RELA   SEQ   SYMBOLIC
PREFIX     CODE        NO    TERM   NO    ADDRESS    DATE     TIME        PREFIX    NO    TERM   NO    ADDRESS    DATE     TIME

         THIS OUTPUT NOT RESULT OF INPUT                                  002     01     00000  AA         75.074   09.03.09

OUTPUT SEG=001 LEN=076 *DFS551I MESSAGE REGION STARTED  ID=001  TIME=0903   CLASSES=002,000,000,000 *
INPUT   TRANSACTION   LINE   RELA   SEQ   SYMBOLIC                        OUTPUT   LINE   RELA   SEQ   SYMBOLIC
PREFIX     CODE        NO    TERM   NO    ADDRESS    DATE     TIME        PREFIX    NO    TERM   NO    ADDRESS    DATE     TIME

         THIS OUTPUT NOT RESULT OF INPUT                                  002     01     00001  AA         75.074   09.03.44

OUTPUT SEG=001 LEN=043 * DFS552I MESSAGE PROCESSING REGION STOPPED *
INPUT   TRANSACTION   LINE   RELA   SEQ   SYMBOLIC                        OUTPUT   LINE   RELA   SEQ   SYMBOLIC
PREFIX     CODE        NO    TERM   NO    ADDRESS    DATE     TIME        PREFIX    NO    TERM   NO    ADDRESS    DATE     TIME

         THIS OUTPUT NOT RESULT OF INPUT                                  002     01     00016  AA         75.074   09.04.25

OUTPUT SEG=001 LEN=052 *DFS994I *CHKPT 75074/090415**004871********SIMPLE** *
INPUT   TRANSACTION   LINE   RELA   SEQ   SYMBOLIC                        OUTPUT   LINE   RELA   SEQ   SYMBOLIC
PREFIX     CODE        NO    TERM   NO    ADDRESS    DATE     TIME        PREFIX    NO    TERM   NO    ADDRESS    DATE     TIME

         THIS OUTPUT NOT RESULT OF INPUT                                  002     01     00005  AA         75.074   09.04.16

OUTPUT SEG=001 LEN=052 *DFS994I *CHKPT 75074/090423**004871********SIMPLE** *
INPUT   TRANSACTION   LINE   RELA   SEQ   SYMBOLIC                        OUTPUT   LINE   RELA   SEQ   SYMBOLIC
PREFIX     CODE        NO    TERM   NO    ADDRESS    DATE     TIME        PREFIX    NO    TERM   NO    ADDRESS    DATE     TIME

         THIS OUTPUT NOT RESULT OF INPUT                                  002     01     00015  AA         75.074   09.04.23

INPUT  SEG=001 LEN=013 *PART AN960C10*
OUTPUT SEG=001 LEN=063 * PART=AN960C10          DESC=WASHER                PROC CODE=74 *
OUTPUT SEG=002 LEN=077 * INV CODE=2  MAKE DEPT=12-00  PLAN REV NUM=     MAKE TIME= 63  COMM CODE=14   *
INPUT   TRANSACTION   LINE   RELA   SEQ   SYMBOLIC                        OUTPUT   LINE   RELA   SEQ   SYMBOLIC
PREFIX     CODE        NO    TERM   NO    ADDRESS    DATE     TIME        PREFIX    NO    TERM   NO    ADDRESS    DATE     TIME

         PART          002     01    00004  AA         75.074   09.03.44              002     01     00002  AA         75.074   09.03.51

INPUT  SEG=001 LEN=016 *DSPALLI AN960C10*
```

Indicates a 13-character report message whose text was "PART AN960C10".

Indicates a 140-character message generated by the transaction code "PART" and transmitted to line 2, relative term 1 (logical terminal name AA).

FILE SELECT AND FORMATTING PRINT PROGRAM (DFSERA10)

GENERAL DESCRIPTION

The File Select and Formatting Print Program is used with IMS/VS and its related data bases. Its primary function is to assist in the examination, display, and transfer of data from the IMS/VS log data set. The program can:

- Print or copy an entire log data set;
- Print or copy from multiple log data sets based upon control card input;
- Select and print log records on the basis of sequential position in the data set;
- Select and print log records based upon data contained within the record itself, such as the contents of a time, date, or identification field;
- Allow EXIT routines to special process any selected log records.

These features are selected and controlled by a series of statements that allow the user to define the input and output options, selection ranges, and various field and record selection criteria.

JCL REQUIREMENTS

The File Select and Formatting Print program executes as a standard operating system job and, as such, requires a JOB statement as defined by the user's installation. Additionally, an EXEC and appropriate DD statements to define inputs and outputs are required.

EXEC
This statement must be either of the format PGM=DFSERA10 or included in a cataloged procedure.

STEPLIB DD
This statement defines a DSORG=PO data set containing the EXIT routine modules. If EXIT routines are not used or if the modules reside in LINKLIB, this statement is not required.

SYSPRINT DD
This statement describes the output data set to contain the formatted print records and control messages. It will usually be defined as SYSOUT=A.

DCB parameters specified for this data set are RECFM=FBA and LRECL=133. Block size may be provided on the SYSPRINT DD statement and must be a multiple of 133. The default is 133.

SYSIN DD
This statement describes the input control data set. This file must be in card image format.

input or data DD
This statement defines the input data set(s) to be examined to produce the formatted print records.

These data sets must be standard labeled files, either direct access or tape. They can be of any record format (F, FB, V, VB, VBS, or U), as long as they are of DSORG=PS.

If a file with RECFM=U is used, the DCB BLKSIZE parameter must be specified. These files are processed using QSAM.

Therefore, any file that QSAM supports can be described as input.

The default ddname used is SYSUT1.

output or data DD
This statement defines the optional output data set(s) to contain the selected records.

DFSERA10 sets the RECFM of this data set equal to the RECFM specified for the input data set. This will also be done for LRECL and/or BLKSIZE if not specified.

The default ddname used is SYSUT4.

INPUT AND OUTPUT

All data input is processed using QSAM and can reside on either tape or direct access storage devices. Data set organization must be physical sequential, while the record format can be fixed or variable in length, blocked or unblocked, or of undefined length. Multiple input and output data sets may be used and may reside on different device types.

The data set containing control information must be in card image form. These statements are reproduced on the output print data set in the same format and sequence as they are processed. If error conditions are encountered, error messages are produced following the statement to which they apply.

Output data may be formatted and printed on the SYSPRINT data set, copied to a specified data set unchanged, or both.

Data to be printed is formatted into 32-byte segments and displayed in both hexadecimal and EBCDIC forms, with the hexadecimal relative offset value preceding each segment.

PROGRAM CONTROL

The flow of control for the program passes through two major stages:

- Control statement processing, where construction of record test and selection parameters takes place and control statement errors are diagnosed
- Record selection and output processing, where the input data is read, analyzed, and compared with the selection parameters to determine the applicability of the record for output

During the first phase, parameter statements are read and examined, and the required test or test series is constructed to create a test group. This test group is then used in record selection when control passes to the next phase of the program. In the second phase, the input data records are read, and disposition is decided by the results of each test in the group. When the end of the input data is reached, either by an end-of-file condition being encountered or the indicated record count being satisfied, program control shifts back to phase one, where the next group of tests is constructed.

CONTROL STATEMENTS

Three types of control statements are used to guide the program through the described phases. An additional statement type can be used to provide titles or comments on the output listings. Keyword operands on these statements can be extended to additional statements, to a maximum of 9, by placing a nonblank character in position 72 and continuing the operand in position

16 of the next statement. Each full keyword has a corresponding abbreviated form that may be used.

The CONTROL statement defines the ddnames to be used for the input and output data sets and the beginning and ending limits of the data set to be scanned. Inclusion of this statement is optional if the default operands values are satisfactory.

The OPTION statement defines the test or series of tests to be performed upon the data of the candidate record to determine its qualification for selection. One or more tests can be executed on each logical record by the appropriate number of OPTION statements, creating the logical "OR" function. Records can be analyzed with the logical "AND" function by using the multifield test capability of the COND operand and the necessary number of OPTION statements, creating a test series. The operands COND=M and COND=E are used to denote the beginning and ending, respectively, of a series for multifield testing of a record.

Each OPTION function has its own output processing defaults. If multiple OPTION functions are used to create a multifield test series, final output processing is determined by the OPTION and its associated keywords that are defined along with the COND=E keyword.

The END statement is a delimiter used to separate one group of tests (comprised of one or more OPTION statements), from subsequent groups of tests on the next data set. When an END statement is encountered in the control input stream, the construction of record selection parameters ceases and the processing of input data records starts. Proper use of the END statement allows one execution of the utility program to perform a varied number of tests on one or more IMS/VS log data sets.

The * or COMMENTS statement may be used to include any information deemed helpful by the user to identify tests or data and has no effect on the utility program.

CONTROL Statement

The CONTROL statement is optional. If not specified, the default values cause the SYSUT1 input file to be examined. The optional output data set defined on the SYSUT4 DD statement will only be opened if the OPTION COPY function is specified in the current group of tests. This data set will only be used if COND=E is also specified.

1	10	16
CONTROL	CNTL	[{SKIP | K} = {0 | aaa}] [,{STOPAFT | H} = {16777215 | bbb | EOF | (bbb,E)}] [,{DDNAME | D} = {SYSUT1 | ddname}] [,{DDNOUT | O} = {SYSUT4 | ddname}]

SKIP= or K=
Defines the first record tested. All prior records are ignored.

If this keyword is not specified, a default value of zero is used and causes the first record on the input file to be tested.

aaa
Must be specified in the range of zero to 999 999, and cannot have embedded commas.

STOPAFT= or H=
Defines the last record to be tested. When this value has been reached by counting processed records, the current group of tests is terminated.

If this keyword is not specified, a default value of 16777215 is used.

bbb
Must be specified in the range of 1 to 99 999 999, with no embedded commas. If the value zero is specified, one record is processed.

EOF
Denotes end-of-file condition: allows record processing beyond the stated maximum of 99999999 records.

E
Causes records to be counted for test sequence termination only if they satisfy selection criteria. Otherwise, all records read (after the SKIP value) are counted.

DDNAME= or D=
Identifies the input data set for the current group of tests. A corresponding DD statement must be supplied.

If this keyword is not specified, a default of SYSUT1 is used and the appropriate DD statement must be supplied.

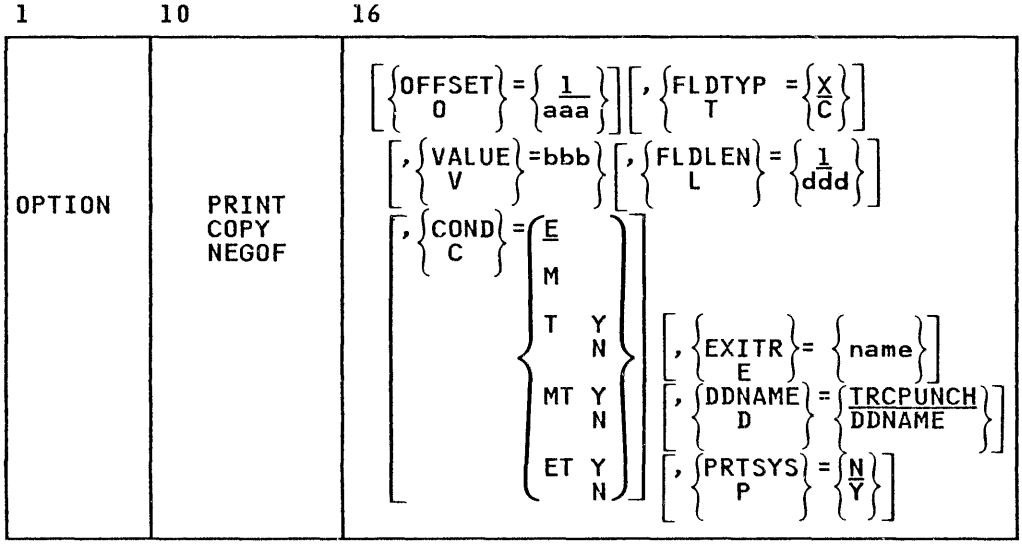
ddname
Must be the ddname of the input file if the default of SYSUT1 is not used.

DDNOUT= or O=
Identifies the optional output data set for the current group of tests.

This keyword is used in conjunction with the OPTION COPY function and is only required if a ddname other than the default of SYSUT4 is required. DDNOUT or the presence of SYSUT4 will not cause this data set to be used; this data set will be used only if OPTION COPY is specified with COND=E.

OPTION Statement

The OPTION statement causes one set of tests to be constructed. One or more OPTION functions may be specified in any combination desired, to further define the selection criteria and output processing to be performed against each input record. Except for "EXITR" and "DDNAME" keyword operands, all records processed by phase 2 of this program will either be displayed on the SYSPRINT data set or transferred to the specified output data set when all keyword operands are omitted.



Options

Each option has two distinct functions:

1. Determines starting position for OFFSET keyword
2. Determines output processing to be performed

If individual options are combined to form a multifield test, the use of OFFSET remains unchanged; however, output processing is determined by the OPTION coded with the 'COND=E' keyword.

PRINT

This parameter causes all selected records to be displayed on the SYSPRINT data set.

COPY

This parameter causes all selected records to be transferred to the specified output data set. These records may also be displayed on the SYSPRINT data set by use of the PRTSYS keyword.

NEGOF

This parameter causes the OFFSET keyword value to be used as a negative offset from the end of the log record. All records selected using this function will be displayed on the SYSPRINT data set.

Keywords

The following keywords are all optional:

OFFSET= or O=

This keyword is used to define the location of the first byte of the field to be tested in the record. The default is position one of the record.

aaa

This value can be in the range from one up to and including the length of the record under test. Maximum value is 32767 bytes, and no checking is performed to determine if the logical record length is exceeded.

Note: If DSECTs are used to locate values in control records or blocks, you must adjust the starting value for the OFFSET parameters. Most DSECTs start with a relative value of ZERO, while the value specified in the OFFSET keyword is always expressed as relative to byte 1.

FLDTYP= or T=
This keyword is used to define the type of data in the VALUE=field.

X
This parameter defines the data to be treated as hexadecimal pairs. The test data is packed (2 bytes into one to form hexadecimal equivalents). This is the default value.

Example: If VALUE=D9D6D6E3E2C5C7 (14 bytes) is specified with the FLDTYP=X parameter, the resultant VALUE= will look like this: ROOTSET in EBCDIC or D9D6D6E3E2C5C7 in hexadecimal; in either case, the length is only 7 bytes.

C
This parameter defines the data to be treated as EBCDIC. The test data is used as punched in the card, with no alterations.

VALUE= or V=
This keyword defines those characters that comprise the test field. If FLDTYPE=X was specified, this data must be entered as hexadecimal character pairs. For a "test under mask" condition, a single pair must represent the hexadecimal value for the test. If FLDTYP=C was specified, this data must be entered as EBCDIC characters. If the character of blank or comma is to be included in this operand, FLDTYP=X must be used with the appropriate hexadecimal equivalent.

bbb
This value can not exceed 255 EBCDIC or 510 hexadecimal characters. The length of this field is determined by the FLDLEN= keyword value and not by the number of "nonnull" characters in this field.

FLDLEN= or L=
This keyword defines the number of characters to be used from the test field.

ddd
This value represents the actual number of bytes to be used, not the number of characters specified in the VALUE= keyword. The acceptable range of values for this field is 1 to and including 255. The default is 1.

COND= or C=
This keyword defines the type of test and its relationship to other tests in the group. If this keyword is not specified, the default is COND=E.

E
This parameter marks the last (or only) element in a test series. Any OPTION control statements appearing after this form a new series of tests. This allows various tests to be performed on each record and each test series can be used on different fields within the record. Final output processing is determined by the OPTION function defined with this keyword value.

M
This parameter indicates that this is a multifield test. That is, more than one test is to be made on each input record. All tests in this series must be satisfied before final output selection and processing of this record can begin.

T

This parameter causes the VALUE= byte to be used as a "test under mask" value, instead of a compare field. Only the first byte (two hexadecimal characters if FLDTYP=X) of the VALUE= field will be used. If FLDTYP=C is used, the hexadecimal equivalent of the EBCDIC character will be the test value. If this parameter is used, the FLDLEN= keyword must not be specified and a default length of 1 will be assumed.

Y

This parameter indicates that, for the "test under mask" to be considered satisfied, there must be a bit in the record test field for each corresponding bit of the test byte. This is equivalent to a "branch if ones" Test.

N

This parameter indicates that, for the "test under mask" to be considered satisfied, there must not be a bit in the record test field for any of the corresponding bits of the test byte. This is equivalent to a "branch if zeros" test.

MT

This parameter defines a "test under mask" OPTION as described above in the T discussion but with the properties of a multifield test as described in the M discussion. Because the T parameter assumes a default value of 1,e, the MT parameter must be used for a multifield test that starts with a "test under mask" value.

ET

This parameter signifies that a multifield test series ends with "test under mask" condition.

EXITR= or E=

This keyword specifies the entry point name of an exit routine to be given control when a candidate record has satisfied all selection criteria for the current test.

If multiple test groups have specified the same exit routine, an attempt will be made to load the routine into storage for each group; therefore, the routine should be reenterable. Upon reaching end of file on input, a final call will be made to the exit routine. You can determine if end of file was reached by checking for zeros in the parm field.

Interface to the exit routine is as follows:

ENTRY:

REGISTERS

R1 contains a pointer to a parameter list.
R13 points to an empty save area.
R14 contains a return address.
R15 contains the exit routine entry address.

PARMLIST

The parameter list consists of two words, the first is a pointer to the candidate record; the second (with the high order bit on) is a pointer to the SYSPRINT data set DCB.

EXIT:

Upon return from the exit routine, register 15 is used to decide whether or not processing is to continue on this record.

A nonzero value indicates that no further processing is to be done on this record, and selection tests will start again against the next input record.

A zero value indicates that this record is required, and output processing will now be determined based upon the last OPTION statement encountered containing the COND=E keyword.

If the EXITR keyword is omitted, processing will continue as though a return code value of zero had been received.

DDNAME= or D=
This keyword is used to allow the user to define the output data set used by the DL/I call trace log record retrieval routine (DFSERA50) whenever it has been specified as the user exit routine. A corresponding DD statement must be supplied.

If this keyword is not specified and DFSERA50 is the exit routine, a default of TRCPUNCH is used and the appropriate DD statement must be supplied.

PRTSYS= or P=
This keyword is used to determine whether or not to display all selected records on the SYSPRINT data set.

N
This parameter indicates that no printing of selected records is to be done.

Y
This parameter indicates that all records transferred to the output data set will also be formatted and printed.

This keyword can only be used with OPTION COPY function. N is the default.

END Statement

When all tests have been defined for the current input file, the END statement must be used to cause execution of those tests to begin.

Positions 10 and on can be used for comments.

1	10	16
END	[comments]	

COMMENTS Statement

The COMMENTS statement is optional and, if used, causes its contents to be displayed on the SYSPRINT data set.

1	10	16
*		

EXAMPLES

The following examples illustrate some of the ways in which DFSERA10 can be used. Most of the examples refer to the IMS/VS log data set; however, this program can be used with any data set that can be processed using QSAM.

For clarity, all option keywords are specified in full form, and many were coded where the default could be taken. Use of the short form and keyword defaults would greatly reduce the input required. Each example also makes use of the "COMMENTS" statement to describe the functions being performed.

Example 1 shows the JCL and control statements required to print or copy all log records from an IMS/VS log data set.

```
//EXAMPLE1 JOB
//          EXEC PGM=DFSERA10
//STEPLIB   DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//             DSN=IMSLOG,VOL=SER=123456
//SYSUT4    DD UNIT=SYSDA,SPACE=(TRK,(3,1)),DISP=(NEW,PASS),
//             DSN=EXAMPLE1.COPY1,VOL=SER=IMSPAC
//SYSIN     DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : DEFAULTS                                  *
*                 INPUT = SYSUT1                                    *
*                OUTPUT = SYSPRINT                                  *
*  SELECTION QUALIFIERS :                                           *
*            1. DEFAULT = ALL INPUT RECORDS                         *
*-------------------------------------------------------------------*
OPTION   PRINT
END
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : DEFAULTS                                  *
*                 INPUT = SYSUT1                                    *
*                OUTPUT = SYSUT4                                    *
*  SELECTION QUALIFIERS :                                           *
*            1. DEFAULT = ALL INPUT RECORDS                         *
*-------------------------------------------------------------------*
OPTION   COPY
END
*-------------------------------------------------------------------*
/*
```

Example 2 shows two ways of selecting and printing all log records of a specific type.

```
//EXAMPLE2 JOB
//          EXEC PGM=DFSERA10
//STEPLIB   DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//LOGIN     DD UNIT=TAPE,DISP=(OLD,KEEP,LABEL=(,SL),
//             DSN=IMSLOG,VOL=SER=123456
//SYSIN     DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*            1. TYPE X'16' IN 5TH BYTE = (ALL /SIGN ON/OFF)         *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
END
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*            1. TYPE X'16' IN 5TH BYTE = (LOG RECORD TYPE)          *
*            2. FLAG X'01' IN 6TH BYTE = 1 (/SIGN ON - ONLY)        *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=6,FLDTYP=X,VALUE=01,COND=ETY
END
*-------------------------------------------------------------------*
/*
```

Example 3 shows how to print or copy two log record types, each containing a field value (USERID) common to both, but residing at different offsets depending upon the record type.

```
//EXAMPLE3 JOB
//         EXEC PGM=DFSERA10
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//LOGIN    DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//            DSN=IMSLOG,VOL=SER=123456
//LOGOUT   DD UNIT=SYSDA,SPACE(TRK,(3,1)),DISP=(NEW,PASS),
//            DSN=EXAMPLE3.COPY1,VOL=SER=IMSPAC
//SYSIN    DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*             1. LOG RECORD TYPE X'16'                              *
*                USERID IN 9TH BYTE (FROM BEGINNING OF RECORD)      *
*             2. LOG RECORD TYPE X'50'                              *
*                USERID IN 12TH BYTE (FROM END OF RECORD)           *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=9,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=E
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=M
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=E
END
*-------------------------------------------------------------------*
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = LOGOUT                                   *
*  SELECTION QUALIFIERS :                                           *
*             *  THE SAME AS FOR THE 'PRINT' AND 'NEGOF' OPTIONS    *
*             ABOVE, BUT SINCE THE 'COPY' OPTION DEFINES AN OUTPUT*
*             DATA SET OTHER THAN SYSPRINT, THIS OPTION MUST BE     *
*             CODED WITH THE 'COND=E' KEYWORD.                      *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN,DDNOUT=LOGOUT
OPTION   PRINT OFFSET=9,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=E
END
*-------------------------------------------------------------------*
/*
```

Example 4 selects all specified log record types, each containing a common userid value, and both print and transfer these records to the specified output data set.

```
//EXAMPLE4 JOB
//         EXEC PGM=DFSERA10
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//LOGIN    DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//            DSN=IMSLOG,VOL=SER=123456
//LOGOUT   DD UNIT=SYSDA,SPACE=(TRK,(3,1)),DISP=(NEW,PASS),
//            DSN=EXAMPLE4,COPY1,VOL=SER=IMSPAC
//SYSIN    DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = (SYSPRINT AND LOGOUT)                    *
*                  *  SINCE MULTIFIELD TESTS ARE BEING USED,        *
*                  AND CONSIST OF MULTIPLE OPTION FUNCTIONS,        *
*                  FINAL OUTPUT PROCESSING OF THE SELECTED RECORD   *
*                  IS BASED UPON THE 'COPY' OPTION AND 'PRTSYS=Y'   *
*                  KEYWORD BEING CODED WITH 'COND=E'.               *
*  SELECTION QUALIFIERS :                                           *
*             1. USERID = USERBBBB                                  *
*             2. LOG RECORD TYPES (X'16',X'50',X'51',X'52')         *
*-------------------------------------------------------------------*
CONTROL  CNTL   DDNAME=LOGIN<DDNOUT=LOGOUT
OPTION   PRINT  OFFSET=9,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
OPTION   NEGOF  OFFSET=12,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=E
OPTION   NEGOF  OFFSET=12,FLDTYP=C,VALUE=USERBBBB,GLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=51,FLDLEN=1,COND=E
OPTION   NEGOF  OFFSET=12,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=52,FLDLEN=1,COND=E
END
*-------------------------------------------------------------------*
/*
```

Example 5 copies selected log records to individual output data sets in one execution of DFSERA10. All selected records are printed.

```
//EXAMPLE5 JOB
//         EXEC PGM=DFSERA10
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//            DSN=IMSLOG,VOL=SER=123456
//LOGOUT1  DD UNIT=SYSDA,SPACE=(TRK,(3,1)),DISP=(NEW,PASS),
//            DSN=EXAMPLE5.COPY1,VOL=SER=IMSPAC
//LOGOUT2  DD UNIT=SYSDA,SPACE=(TRK,(3,1)),DISP=(NEW,PASS),
//            DSN=EXAMPLE5.COPY2,VOL=SER=IMSPAC
//LOGOUT3  DD UNIT=SYSDA,SPACE=(TRK,(3,1)),DISP=(NEW,PASS),
//            DSN=EXAMPLE5.COPY3,VOL=SER=IMSPAC
//SYSIN    DD *
*--------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                  *
*                  INPUT = DEFAULT (SYSUT1)                          *
*                 OUTPUT = SYSPRINT AND (LOGOUT1,LOGOUT2,LOGOUT3)    *
*  SELECTION QUALIFIERS :                                            *
*             1. LOG RECORD TYPE X'16'                               *
*             2. USERIDS = (USERAAAA,USERBBBB,USERCCCC)              *
*--------------------------------------------------------------------*
CONTROL  CNTL   DDNOUT=LOGOUT1
OPTION   COPY   OFFSET=9,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
END
*--------------------------------------------------------------------*
CONTROL  CNTL   DDNOUT=LOGOUT2
OPTION   COPY   OFFSET=9,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
END
*--------------------------------------------------------------------*
CONTROL  CNTL   DDNOUT=LOGOUT3
OPTION   COPY   OFFSET=9,FLDTYP=C,VALUE=USERCCCC,FLDLEN=8,COND=M
OPTION   COPY   PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
END
*--------------------------------------------------------------------*
/*
```

Example 6 shows the JCL and control statements required to print record 158 of an OSAM image copy data set and all type X'50' records on a log data set that refer to this block number (assuming unblocked OSAM).

```
//EXAMPLE6 JOB
//         EXEC PGM=DFSERA10
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//            DSN=IMSLOG,VOL=SER=123456
//IMAGFILE DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//            DSN=OSAMIMAG,VOL=SER=456789
//SYSIN    DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = IMAGFILE                                 *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*           1. OSAM RBN = 0000009E (RECORD NO. 158)                 *
*-------------------------------------------------------------------*
CONTROL  CNTL  STOPAFT=(1,E),DDNAME=IMAGFILE
OPTION   PRINT OFFSET=1,FLDTYP=X,VALUE=0000009E,FLDLEN=4,COND=4
END
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : DEFAULTS                                  *
*                  INPUT = SYSUT1                                   *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*           1. LOG RECORD TYPE X'50'                                *
*           2. DATA BASE NAME = DATABAS1                            *
*           3. FLAG X'04' IN 7TH BYTE = 0 (OSAM DATA SET)           *
*           4. OSAM RBN = 0000009E                                  *
*-------------------------------------------------------------------*
OPTION  PRINT OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=M
OPTION  PRINT OFFSET=25,FLDTYP=C,VALUE=DATABAS1,FLDLEN=8,COND=M
OPTION  PRINT OFFSET=7,FLDTYp=X,VALUE=04,COND=MTN
OPTION  PRINT OFFSET=43,FLDTYP=X,VALUE=0000009E,FLDLEN=4,COND=E
END
*-------------------------------------------------------------------*
/*
```

Log Type X'67' Record Format and Print Module (DFSERA30)

This module formats Type X'67' log records. It is an exit routine to the File Select and Formatting Print Program (DFSERA10). Because this routine formats log records, it passes a return code to DFSERA10. This return code tells DFSERA10 that the log record has been processed and, therefore, requires no additional processing.

The program specifically formats trace and general purpose subrecord types (X'00' and X'01') and SNAP subrecord types (X'FD' and X'FF'). Other log records are formatted in OS/VS dump format.

For trace and SNAP subrecord types, the program creates log record leader information followed by a formatted printout of each element within the log record.

The control statements required to properly format type X'67' log records using this exit routine follow.

```
Col. 1          10      16                                          72
     CONTROL    CNTL
     OPTION     PRINT   OFFSET=5,FLDLEN=2,                           X
                        VALUE=67aa,COND=E,                           X
                        EXITR=DFSERA30
     END
```

where: aa is the subrecord type and

aa=01, specifies TRACE subrecord type
=FD, specifies SNAP subrecord type
=FF, specifies ABEND subrecord type

The following is sample output. AE9004 is the storage address of the LXB at the time the log record was created. The second column of each line is the relative offset from the LXB.

```
DFSERA30 -- FORMATTED LOG PRINT
•
•
•
INTERNAL TRACE RECORD
•
•
•
LXB
 AE9004  000000   807F0BC9 00093660   00AE9350 00AE92B0   00091E90 00AE991C  17000000 7F0C0000
 AE9024  000020   80000000 520821CE   0008229C 000820C6   80082194 012141CE  60000054 0A000000
 AE9044  000040   30000005 022140C6   600000CE 09000000   30000005 47000000  20000001 00000000
 AE9064  000060   00000000 00000000   00000000 00000000   00000000 00000000  00000000 00000000
 AE9084  000080 TO AE90C4  0000C0    SAME AS ABOVE
 AE90E4  0000E0   00000000 0C419317   F1044193 17F10441   9337E218 D243F510  A314A8C3 419101A2
 AE9104  000100   02F30C41 93179101   A502F004 F30C4193   17F10441 93170000  00000000 00B66218
```

Program Isolation Trace Record Format and Print Module (DFSERA40)

The program isolation (PI) trace format and print module receives type '67FA' log records as an exit routine from the file select and formatting print program (DFSERA10) and formats the records on the SYSPRINT data set. These log records are produced by the PI (program isolation) trace, trace PI enqueue and dequeue calls to DFSLRH00, and also by DL/I calls to the DL/I analyzer. The DL/I analyzer processes all DL/I calls. When tracing is active, the DL/I analyzer calls are traced. The standard ENQ/DEQ call is invoked by the DFSLR macro instruction. PI tracing is executed by the /TRACE command in an IMS/VS online environment or by the OPTIONS card with LOCK=OUT specified.

In a data sharing environment, if the PI trace is active and being logged, the PI trace logger is activated by the IMS/VS lock manager (DFSLMGR0) and exits to the IMS/VS resource lock manager (IRLM).

The PI trace record format and print module is loaded during the execution of the file select and formatting print program and must reside in the LINKLIB or in a JOBLIB or STEPLIB data set.

The control statements required for this exit routine are:

```
Col. 1          10        16                                          72
     CONTROL    CNTL
     OPTION     PRINT     OFFSET=5,FLDLEN=2,VALUE=67FA,               X
                          COND=E,EXITR=DFSERA40
     END
```

The following is a sample output (the spacing of fields is altered):

```
DATE:   05/11/82
MODULE PST  TIME (*=ET) CALLR  ACT    LEV  WC  WFC SEQN  FDBK  RC  PC  ID= (RBA    DMB DCB SUF) CLS  TOKEN     COMMENTS

LRHC   C1                      GZIDB                 0ABE
LRHC   C1                      RZIDP                 0AC1
PIEX   01   23:36:22.472  DLI   TENQ   LPD  CC  00   0AC3  0000            48107505  8007  01

LRHC   01                      TTLKX                 0AC4
PIEX   C1   23:36:22.472  DLI   ENQ    UPD  00  00   0AC5  0000            00000658  8006  01        00722C50
LRHC   C1                      GRIDX                 0AC6
PIEX   01   23:36:22.474  APP   ENQ    SHR  00  00   0ACA  0000            00000694  8006  01    C   00722CDC
LRHC   C1                      GCCMX                 0ACB
DLAC   01   23:36:22.493         GU                  0ACE                        8                   DL/I CALL
PIEX   C1   23:36:22.493  DLI   TDEQ   UPD  00  00   0ACF  0000            00000658  8006  01
LRHC   01                      RRIDX                 0AD0
LRHC   C1                      GZIDB                 0AD1
LRHC   01                      RZIDP                 0AD4

PIEX   C1   23:36:22.495  DLI   ENQ    LPD  CC  00   0AD6  0000            00001108  8006  01        00722C50
LRHC   01                      GRIDX                 0AD7
PIEX   01   23:36:22.496  DLI   TDEQ   UPD  CC  CC   0ADA  0000            00001108  8006  01
LRHC   C1                      RRIDX                 0ADB
DLAC   03   23:36:23.614         GU                  0ADE                        1                   DL/I CALL
LRHC   C3                      GZIDB                 0AE4
LRHC   03                      RZIDB                 0AE7
PIEX   C3   23:36:23.735  DLI   TENQ   UPD  00  00   0AE9  0000            48107105  8007  01
LRHC   03                      TTLKX                 0AEA
PIEX   03   23:36:23.736  DLI   ENQ    UPD  00  00   0AEB  0000            00000408  8006  01        00722C50

LRHC   C3                      GRIDX                 0AEC
PIEX   03   23:36:23.737  APP   ENQ    SHR  00  00   0AEO  CCCC            CCC00428  8C06  01    C   00722C14
LRHC   C3                      GCCMX                 0AEI
DLAC   C3   23:36:23.834         GU                  0AES                        2                   DL/I CALL
PIEX   03   23:36:23.835  DLI   TDEQ   LPD  CC  00   0AEA  0000            00000408  8006  01
LRHC   C3                      RRIDX                 0AEB
LRHC   03                      GZIDB                 0AEC
LRHC   C3                      RZIDP                 0AEF
PIEX   03   23:36:23.838  DLI   ENQ    LPD  00  00   0BC1  0000            00001108  8006  01        00722C50
LRHC   C3                      GRIDX                 0BO2

PIEX   03   23:36:23.840  DLI   TDEQ   UPD  00  00   0BC5  0000            00001108  8006  01
LRHC   C3                      RRIEX                 0BO6
DLAO   02   23:36:27.257         GHU                 0BOE                        4                   DL/I CALL
PIEX   02   23:36:27.257  DLI   TDEC   LPD  00  00   0B10  0000            CC000B7C  8006  01
LRHO   02                      RRIEX                 0B11
LRHC   C2                      GZIDB                 0E12
LRHO   02                      RZIDB                 0B15
PIEX   C2   23:36:27.263  DLI   TENC   LPD  CC  CC   0B17  0000            48107105  8007  01
LRHO   02                      TTLKX                 0B18
PIEX   C2   23:36:27.263  DLI   ENC    LPD  00  CC   0B19  0000            00000408  8006  01        00722CA0

LRHC   C2                      GRIEX                 0B1A
PIEX   02   23:36:27.265  DLI   TENQ   UPD  C1  CO   0B1E  1800  04        00000428  8006  01        00722C14
PIEX   03   23:36:45.079  APP   DEC    SHR  CC  CC   0B34  CCCO                            C
LRHO   03                      RQCML                 0B35
PIEX   C2    .  0:17.850* DLI   UNK    RD                0B37              6F
LRHC   02                      TTLKL                 0B38        04
DLAC   C2   23:36:45.982         GHU                 0B3A                        5                   DL/I CALL
PIEX   02   23:36:45.982  DLI   TDEQ   UPD  CO  CO   0B3B  0000            00000408  8006  01
LRHC   C2                      RRIEX                 0B3C
LRHO   02                      GZIDB                 0B3D

LRHC   C2                      RZIDB                 0E40
PIEX   02   23:36:45.986  DLI   TENQ   UPD  CC  CO   0B42  0000            48107505  8007  01
LRHC   C2                      TTLKX                 0B43
PIEX   C2   23:36:45.986  DLI   ENC    UPD  CO  CC   0B44  0000            00000658  8C06  01        00722CA0
LRHO   02                      GRIDX                 0B45
```

Explanation of Column Headings

DATE

Specifies the date PI trace was started. The TIME field is relative to this date.

MODULE

Specifies the module that issued the DFSLR call to DFSLRH00 or the module that called the IRLM or DFSFXC10. The four characters selected come from the xxxx portion of the full module name DFSxxxx0.

PST
Specifies the program specification table (PST) number (from PSTPSTNR).

TIME
Specifies the time of the call as HHH:MM:SS.UUU, where UUU is milliseconds, relative to the date on which tracing was started. If the return code (RC) is 04 and PI trace timing was active at the time of the call, the next record for this PST in this report will show the elapsed time of the enqueue wait in this field. The time will be indicated as MM:SS.UUU*, with the "*" indicating it is an elapsed time.

CALLR
Specifies the type of caller (DLI,FP,APP).

ACT
Specifies the action requested.

LEV
Specifies the level of control for this call.

RD read only

SH share

UPD update

EXC exclusive

WC
Number of PSTs that hold this resource in a state that caused this PST to wait.

WCF
Number of PSTs waiting for this PST to release this resource.

SEQN
Specifies the sequence number of the corresponding internal trace.

FDBK
Is 2 bytes of feedback information from either DFSFXC10 or the IMS/VS resource lock manager (IRLM).

RC
Specifies the return code from DFSFXC10 or the IMS/VS resource log manager (IRLM).

00 Successful completion
04 Caller must IMS wait for control of the requested resource
08 Deadlock; request has been disallowed. This transaction causes an internal pseudoabend, a backout, and automatic rescheduling.
0C Invalid call

PC
Specifies the PST post code following the enqueue wait. This field is only present when RC is 04 and the TIME field has an "*" at the end.

60 Deadlock has occurred. This transaction causes an internal pseudoabend, a backout, and automatic rescheduling.
6F Control of the resource has been obtained.

ID=
Specifies an 8-byte identification of the resource being enqueued or dequeued. It contains a 4-byte RBA, a 2-byte DMB number, a 1-byte DCB number, and a 1-byte SUF (suffix) field.

CLS
For APP types of callers, specifies the Q-command code class requested. For LMGR traces, specifies the CLASS parameter.

TOKEN
The address of the control block enqueued or locked on this call or, if the type of call is an unlock or DEQ call, the address of the control block being passed to the lock manager.

COMMENTS
Specifies 'DL/I CALL' if a trace is requested from DFSDLA00. Other comments are for LMGR traces.

DL/I CALL IMAGE CAPTURE ROUTINE (DFSERA50)

If trace data is sent to the IMS/VS log data set, it can be retrieved using utility DFSERA10 and special DL/I call image capture routine DFSERA50. DFSERA50 will deblock, format, and number the DL/I call image capture records to be retrieved. To use DFSERA50, a DD card must be inserted defining a sequential output data set in the DFSERA10 input stream. The default DD name for this DD card is TRCPUNCH. The card must specify BLKSIZE=80.

The following examples of DFSERA10 input control statements in the SYSIN data set may be used to retrieve DL/I call image capture data from the log data set:

- Print all DL/I call image capture records.

Column 1 Column 10

```
OPTION   PRINT OFFSET=5,VALUE=5F,FLDTYP=X
```

- Print selected DL/I call image capture records by PSB name.

Column 1 Column 10

```
OPTION   PRINT OFFSET=5,VALUE=5F,COND=M
OPTION   PRINT OFFSET=25,FLDTYP=C,FLDLEN=8,VALUE=psbname,COND=E
```

- Format DL/I call image capture records (in format acceptable as input for the DL/I test program DFSDDLT0).

Column 1 Column 10

```
OPTION   PRINT OFFSET=5,VALUE=5F,COND=M
OPTION   PRINT EXITR=DFSERA50,OFFSET=25,FLDTYP=C,                x
               VALUE=psbname,FLDLEN=8,DDNAME=OUTDDN,COND=E
```

Note: The DDNAME= parameter is used to name the DD card used by DFSERA50. The data set defined on the OUTDDN DD statement will be used instead of the default TRCPUNCH DD statement. For this example, the DD appears as:

```
//OUTDDN DD ...,DCB=(BLKSIZE=80),...
```

IMS/VS TRACE TABLE RECORD FORMAT AND PRINT MODULE (DFSERA60)

The IMS/VS trace table record format and print module is an exit routine. It receives type '67FA' log records from the File Select and Formatting Print Program (DFSERA10) and formats the records on the SYSPRINT data set. These log records are

produced when you specify (in the OPTIONS statement for the DFSVSAMP data set or DFSVSMnn PROCLIB member) that trace table(s) is to be written to the log.

DFSERA60 is loaded during execution of DFSERA10 and must reside in the LINKLIB or in a JOBLIB or STEPLIB data set.

The control statements required to invoke DFSERA60 follow:

```
Col. 1        10    16
     CONTROL  CNTL
     OPTION   PRINT OFFSET=5,FLDLEN=2,VALUE=67FA,          x
                    COND=E,EXITR=DFSERA60
     END
```

Explanation of Column Headings

See the assembly of macro IDLIVSAM TRACENT or the microfiche for an explanation of the format of the trace entry.

IMS/VS LOG TRANSACTION ANALYSIS UTILITY (DFSILTA0)

The IMS/VS Log Transaction Analysis utility (DFSILTA0) collects information about individual occurrences of IMS/VS transactions, based on records in the IMS/VS system log data set. (Log data sets from a batch region do not contain the desired records, and so cannot be used.) The information collected includes transaction identification, source, message processing program, dependent region, priority, and the class of the transaction. Any nonrecoverable and canceled messages are not used.

DFSILTA0 also accumulates the time that each transaction is received, the time of the message get unique (GU) call, the time the message processing program (MPP) is terminated, the time the output message was placed on the output queue, and the time the output message starts to the terminal. From these times, DFSILTA0 calculates total response time, time on the input queue, processing time, and time on the output queue. This information enables an installation to find bottlenecks in the system and to evaluate whether or not transaction classes have been assigned correctly. If there is a need to run the IMS/VS Statistical Analysis utility on a smaller portion of the IMS/VS system log data set, DFSILTA0 can provide a new log tailored to an installation's specifications. DFSILTA0 is put into IMSVS.RESLIB by IMS/VS system definition.

If the user wishes to run the Log Transaction Analysis utility against two or more IMS system logs that are communicating using multiple systems coupling, the IMS/VS Log Merge utility (DFSLTMG0) must be run first. The system ID field reflects the order of input to DFSLTMG0.

DFSILTA0 creates a queue entry in a GETMAIN storage pool for each transaction that falls within the specified times or checkpoints. These queue entries are not freed nor are they reused until all the log records necessary to complete an entry on the log transaction analysis report have been found on the log.

Because of this method of processing, if a large number of transactions are enqueued but not processed for any reason, a degradation in performance can be expected in the form of an increase in storage usage and processor time.

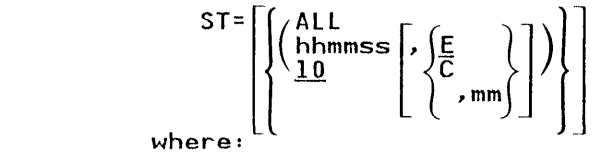
Control statements for the sort program should be examined to determine whether they'll need to be changed. This is because provision for 256 dependent regions has increased the length of the dependent region ID field for the IMS/VS Log Analysis Report.

PROGRAM INPUTS

There are three types of input to DFSILTA0:

- IMS/VS log data set—is required input.
- Report title card—provides descriptive information for the optional title data set.
- Parameters—there are two optional keyword parameters: ST= and OUT=. These specify what portion of the log data set is to be examined for transactions, and what outputs are to be produced. Parameters can be specified in any combination and should be separated from each other by a comma.

Parameter Formats and Descriptions



where:

ST=
Specifies starting and ending times. If the ST parameter is omitted, the default is the first checkpoint encountered.

ALL
Specifies the complete log data set.

hhmmss
Specifies hour, minute, and second. Begin selecting transactions that originated after the first checkpoint occurring at or after this time. The default is to process 10 minutes from this time.

C
Specifies the number of checkpoints to be processed before selection of transactions stops.

mm
Specifies the number of minutes to select transactions.

E
Specifies to end of data set from the specified start time. E is the default.

The Log Transaction Analysis utility scans records between checkpoints. Records before the first checkpoint on an intermediate log data set would only be analyzed by reference to a checkpoint on a previous log.

OUT= NOLOG
NOREPORT

where:

OUT=
Specifies the desired output. If the OUT= keyword is not specified, the DFSILTA0 defaults are to produce both a log data set and a report from the log.

NOLOG
Specifies that a new IMS/VS log data set is not to be produced.

NOREPORT
Specifies that no report is to be produced.

PROGRAM OUTPUTS

DFSILTA0 produces:

- A new IMS/VS log data set, if specified.
- A detailed report in input sequence (NOREPORT is not specified).
- A report, on disk, that can be sorted to produce a sequenced report.
- A heading report (NOREPORT is specified).

The starting position and length of the field names on the Detailed Report Format are used in the optional sort step to produce sequenced reports.

IMS/VS Log Analysis Report

Figure 88 on page 450 describes the format of the IMS/VS Log Analysis Report. Figure 89 on page 452 is an example of an IMS/VS Log Analysis Report.

Detailed Report Format

Identification	Starting Position	Length
Sequence Number	1(Note 1)	5
Transaction Code	7	8
Priority of Transaction (PR)	16	1
Class of Transaction (CL)	18	3
Input Relative Line	22	6
Input Relative Terminal	29	3
Output Relative Line	33	6
Output Relative Terminal	40	3
Processing Type (PT)	44	1 (Note 2)
Program Name	46	8
Dependent Region ID	55	3
Time of SMB Enqueue (Transaction received)	59	7 (Note 3)
Time of Message Schedule or GU	68	7 (Note 3)
Time of CNT Enqueue (Message put on output queue)	77	7 (Note 3)
Time of Program End or Next Message GU	86	7 (Note 3)
Time of CNT GU (Output message starts to terminal)	95	7 (Note 3)
System IDs	103	3 (Note 9)
Time in Input Queue	106	5 (Notes 4 & 5)
Time Processing	113	5 (Notes 4 & 6)
Time in Output Queue	120	5 (Notes 4 & 7)
Total Time	127	5 (Notes 4 & 8)

1. Starting position 1 is a carriage control character that will alter the starting positions of the fields when producing a report on disk.

2. Processing Types (PT):
   - A - Program Abend or Unconnected Transaction
   - B - Processing Restarted
   - S - Send/Receive Processing
   - T - Transmit Only Processing
   - C - Conversational Send/Receive Processing
   - D - Transmit Only Conversational Processing
   - P - Program Switch Send/Receive Processing
   - Q - Transmit Only Program Switch Processing
   - X - Conversational Program Switch, Send/Receive Processing
   - Y - Transmit Only Conversational Program Switch Processing
   - M - Message Switch
   - F - /FORMAT entered (Transaction Code Field has MODNAME)
   - O - Region Occupancy (A region is occupied by a program that is processing transactions that existed in the input queue before the start checkpoint was encountered or a program scheduled by an unrecoverable message.)

3. Time HHMMSST

4. Time SSSST or OVRFLOW. (If the total seconds exceeds the field size, OVRFLOW is printed.)

5. Input queue time is from SMB enqueue to message schedule.

6. If the wait for input (WFI) system option is used, the time processing field will also include the wait time between transactions.

Figure 88 (Part 1 of 2). IMS/VS Log Analysis Report Format

7. Output queue time is from CNT enqueue to CNT GU.
   The field will be blank if output queue time is 0.

8. The two ways to obtain total time are as follows:

   - (MSG-END) - (SMB-ENQ) = TOTAL when CNT-GU is not available
   - (CNT-DEQ) - (SMB-ENQ) = TOTAL when CNT-GU is available

9. This field will be blank unless the tape used for input to
   DFSILTA0 was created by using the Log Tape Merge Utility (DFSLTMG0).
   If the tape was created using DFSLTMG0, this field will consist of
   three numbers, each with a value of 1 through 9.
   Each number represents an xx suffix of a logxx DD statement used
   as input to DFSLTMG0.
   The first number represents where the transaction was input,
   the second represents where the transaction was processed,
   and the third represents where the output for the transaction
   was sent.

Figure 88 (Part 2 of 2). IMS/VS Log Analysis Report Format

Figure 89. IMS/VS Log Analysis Report


```
*** IMS LOG DATA SET FIRST ACCOUNTING RECORD DATA
IMS STARTED   AT TIME =  7.18.26.6 , DATE = 79.264

SPECIFIED START TIME IS  7.19.44.0

CALCULATED END  TIME IS  7.24.44.0

PROCESSED 00160 TRANSACTIONS THIS RUN
REPORT DATA SET CONTAINS 00167 COMPLETE RECORD SETS

    JOB       STEP     DB       CLASSES
    NAME      NAME     ID   ************
  MPP83     MPP83SC6   13    1
  MPP85     MPP85      15    1
  MPP86     MPP86      16    1
  MPP84TS1  MPP84      14    1
  MPP87     MPP87      17    1
  MPP88     MPP88SC2   18    1
  MPP8A     MPP88A     1A    1
  MPP89     MPP89      19    1

PRELOG NORMAL END OF JOB
```


SEQ NBR	TRANS CODE	P R	C L	***IN*** PLINE/PT	**OUT*** PLINE/RT	P T	PGM NAME	DR ID	SNG*ENQ HHMMSST	MSG*SCHED HHMMSST	CNT*ENQ HHMMSST	MSG*END HHMMSST	CNT*GU HHMMSST	SYS IN Q ID SSSST	PROC SSSST	OUT Q SSSST	PAGE TOTAL SSSST
00001	QE2	1	1	0000B3 001	0000B3 001	C	PROGQE2	15	728066	728066	728080	728080	728080		0	14	723
00002	QE2	1	1	00004E 001	00004E 001	C	PROGQE2	16	728113	728113	728118	728118	728119		0	5	1 683
00003	SC2			000067 001	000067 001	F					729184		729184				
00004	DE1			00009E 001	00009E 001	F					729184		729184				
00005	HR1			0000B1 001	0000B1 001	F					729190		729190				
00006	PS3			000042 001	000042 001	F					729188		729188				
00007	TS1	1	1	000053 001	000053 001	S	PROGTS1	14	729184	729184	729190	729190	729190		0	6	22
00008	QE1			0000B6 001	0000B6 001	F					729202		729202				
00009	SC5			0000A2 001	0000A2 001	F					729205		729205				
00010	SC6	1	1	00003E 001	00003E 001	S	PROGSC6	13	729192	729193	729202	729202	729206		1	9	4 22
00011	QE1			00008D 001	00008D 001	F					729206		729206				
00012	IT2			0000A5 001	0000A5 001	F					729206		729207				1
00013	CHKPT	0002	**********	**********	**********				729240	729240	729240	729240	729240				
00014	DE2	1	1	00001E 001	00001E 001	P	PROGDE2	1A	728104	728104	728113	728114	729219		0	10	706 746
00015	PS2			000068 001	000068 001	F					729230		729230				
00016	QE4	1	1	0000B3 001	0000B3 001	C	PROGQE4	16	729189	729190	729204	729204	729204		1	14	61
00017	SC2	1	1	000067 001	000067 001	S	PROGSC2	18	729211	729211	729226	729227	729228		0	16	39
00018	SC2			000023 001	000023 001	F					729219		729220				1
00019	SC4	1	1	000080 001	000080 001	S	PROGSC4	19	729183	729184	729222	729222	729222		1	38	68
00020	QE4	1	1	00004E 001	00004E 001	C	PROGQE4	16	729196	729204	729210	729210	729210		8	6	56
00021	QE2	1	1	000070 001	000070 001	C	PROGQE2	17	728142	728142	728148	728149	728148		0	7	711
00022	DE2	1	1	0000A9 001	0000A9 001	P	PROGDE2	19	728003	728004	728012	728012	729236		1	8	824 854

JCL REQUIREMENTS

EXEC
Executes the IMS/VS Log Transaction Analysis utility, DFSILTA0.

```
//STEP0    EXEC    PGM=DFSILTA0,PARM='ST=(hhmmss,,mm),
//                 OUT=NOLOG'
```

This sample produces a report but no log data set.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

```
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
```

HEADING DD
Describes the heading output data set.

```
//HEADING  DD  SYSOUT=A
```

PRINTER DD
Describes the printed report output data set.

```
//PRINTER  DD  SYSOUT=A
```

REPORT DD
Describes the report output data set. This data set may be passed to a sort step. Report entry headings and any checkpoint records are not included in this data set.

```
//REPORT  DD  DSN=&&REPORT,DISP=(,PASS),UNIT=SYSDA,
//            SPACE=(CYL,(1,1))
```

LOGIN DD
Describes the input log data set.

```
//LOGIN  DD  DSN=IMSVS.LOG,DISP=OLD,VOL=SER=XXXXXX,
//           UNIT=YYYY
```

LOGOUT DD
Describes the optional log data set. This log data set can be used as input to the IMS/VS Statistical Analysis utility.

The LOGOUT data set content is identical to that of LOGIN within the interval specified, except that the type 6 record at the beginning of LOGIN is recopied.

```
//LOGOUT  DD  DSN=IMSVS.LOGOUT,DISP=(,PASS),
// VOL=SER=XXXXXX,UNIT=TAPE,DCB=(RECFM=VB,LRECL=6004,
//            BLKSIZE=6008)
```

TITLE DD
Describes the optional title data set. This allows for the inclusion of descriptive information on each page of the printer output data set.

```
//TITLE  DD  *
  * * * Descriptive information
```

The next sort step is optional.

EXEC
Executes the sort program.

```
//STEP1  EXEC  PGM=SORT
```

SYSOUT DD
Describes the message output data set for the sort.

```
//SYSOUT DD SYSOUT=A
```

SORTIN DD
Describes the input data set to the sort. It is the data set described by the DD statement REPORT in the previous step.

```
//SORTIN  DD  DSN=&&REPORT,DISP=(OLD,DELETE)
```

SORTOUT DD
Describes the output data set to the sort. It is used for printing a sequenced report.

```
//SORTOUT  DD SYSOUT=A
```

SORTWK01-12|DDs
Describe the sort program's work data sets. At least three data sets must be used; they can be tape or disk. For disk:

```
//SORTWKnn  DD  UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
```

SYSIN DD
Describes the sort program's control data set. For a control data set in the input stream, the format is:

```
//SYSIN  DD  *
```

Sample Sort control statement to provide a report sequenced by message GU schedule time within region:

```
SORT FIELDS=(67,7,CH,A,55,2,CH,A),SIZE=E500
```

IMS/VS LOG MERGE UTILITY (DFSLTMG0)

If the user wants to run the Log Transaction Analysis utility (DFSILTA0) against two or more IMS/VS system logs from systems that are communicating using multiple systems coupling, the IMS/VS Log Merge utility must be run first.

The Log Merge utility can merge up to nine input IMS/VS system logs (LOG01, LOG02,..., LOG09) from separate IMS/VS systems running during the same time span. This utility produces one log as input to the Log Transaction Analysis utility.

The Log Merge utility (DFSLTMG0) module is placed in IMSVS.RESLIB by IMS/VS system definition.

PROGRAM INPUTS

- Logs from up to nine separate IMS/VS MSC systems. A log from any single system may consist of a series of logs concatenated in time sequence.
- SYSIN control statements

Control Statement Format

- START

  If a specified start time is wanted, which must be relative to the time field in LOG01, this statement must be present.

COL	LENGTH	VALUE
6	1	blank
7	Variable	yyddd,hhmmss or, hhmmsstt where any trailing digits may be omitted.

- STOP

  If a specified stop time is wanted, which must be relative to the time field in LOG01, this statement must be present.

COL	LENGTH	VALUE
1	4	STOP
5	1	blank
6	Variable	yyddd,hhmmss or, hhmmsstt where any trailing digits may be omitted.

- Log Record Selection

  If the user wishes to merge only certain types of log records, this control statement is used. The format is free-form, starting in column 1. Any of the keywords listed below can be used, in any combination desired, with the following syntax restrictions:

  - BLANK, following a keyword will terminate processing of this control card.
  - COMMA, following a keyword will cause processing of this control card to continue.

Keyword	Meaning
ALL	All log record types are wanted (this is the default if no control cards are present).
MSG	Selects all log records necessary for DFSILTA0; X'01', X'03', X'06', X'07', X'08', X'3X' series, X'40', X'42', X'47'.
3X	Selects all log records within the range; X'30' to X'3F'.
XX	Where XX is the log record type wanted.

  (See the chapter "Data Areas and Records Format" in IMS/VS Program Logic Manual.)

PROGRAM OUTPUTS

DFSLTMG0 produces a merged data set of log records made between the times specified with DATE/TIME. This time is local time for the system of the data set containing log record until a link record is encountered; after that link, all merge times are relative to the time for that log designated LOG01. Merged output should not be used as input to the Data Base Recovery utility.

JCL REQUIREMENTS

This example produces log records needed by DFSILTA0 between 8:30 a.m. and 10:30 a.m. on Julian date 75332 from two IMS/VS system logs.

EXEC
Executes the IMS/VS Log Merge utility DFSLTMG0.

```
//STEP0 EXEC PGM=DFSLTMG0
```

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

```
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
```

PRINT DD
SYSPRINT data set for control statements and error messages.

```
//PRINT DD SYSOUT=A
```

LOG01 DD
Describes the first input log data set.

```
//LOG01 DD DSN=IMSVS.LOGA,DISP=OLD,
//         VOL=SER=XXXXXX,UNIT=TAPE
```

LOG02 DD
Describes the second input log data set.

```
//LOG02 DD DSN=IMSVS.LOGB,DISP=OLD,
//         VOL=SER=XXXXXX,UNIT=TAPE
```

LOGOUT DD
Describes the output data set.

```
//LOGOUT DD DSN=IMSVS.LOGOUT,DISP=(,PASS),
//          VOL=SER=YYYYYY,UNIT=TAPE,
//          DCB=(RECFM=VBS,LRECL=6000,BLKSIZE=6008)
```

SYSIN DD
Describes the control statement data set.

```
//SYSIN DD *
```

Sample control cards

```
START 75332,0830
STOP 75332,1030
MSG
```

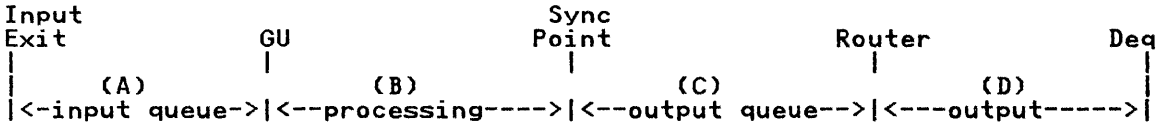
IMS/VS FAST PATH LOG ANALYSIS UTILITY (DBFULTA0)

The IMS/VS Fast Path (FP) Log Analysis utility is an offline utility that prepares statistical reports for Fast Path. The reports are based on data recorded on the IMS/VS system log. The utility produces five reports:

- Detail Listing of Exception Transactions
- Overall Summary by Transaction Code
- Summary of Exception Detail by Transaction Code
- Summary of MPP and BMP Transactions by PSB Name
- Recapitulation of the Analysis Report

These reports are useful for system installation, tuning, and trouble shooting. This utility is not related to the IMS/VS DC Monitor or the IMS/VS Log Transaction Analysis utility.

For a Fast Path transaction, the following four time intervals are computed:



(A) Input queue time—period from input exit to the GU of the application program

(B) Processing time—period from the GU (I/O PCB) of the application program to sync point

(C) Output queue time—period from sync point to entry to the output router

(D) Output time—period from output router entry to dequeue time

The maximum time interval that can be recorded on the logs is 65.535 seconds. If the time interval for A, B, or C exceeds 65.535 seconds, it will be recorded as 65.535 and displayed on the report as 65.5 seconds.

The four time intervals are computed and inserted into reserved fields in Fast Path log records and are thus made part of the normal logging procedure. Time intervals (A) and (B) appear in the input message (log record ID X'5901') and the output message (ID X'5903') log records respectively. Time intervals (C) and (D) appear in the dequeue log record (ID X'5936'). Sync point time is the boundary between intervals (B) and (C).

ADDITIONAL INFORMATION AND DEPENDENCIES

The Fast Path Log Analysis report includes additional performance-related data items. The Fast Path log records contain the following information:

- Input message record (log record ID X'5901')
  - The routing code for the transaction
  - The input terminal's LTERM name for the transaction
  - The balancing group queue count
- Sync point record (ID X'5937')
  - The number of DEDB calls made
  - The number of I/Os incurred for DEDB calls (ID X'5937' plus count of certain ID X'5950's)
  - The number of MSDB calls made
  - The number of control interval (CI) contentions
  - The number of overflow buffers used
  - The number of waits for data base buffer

The time intervals (A), (B), (C), and (D) plus the nine additional performance-related items above are combined with other logged information to produce the Detail Listing of Exception Transactions report.

INPUT

The Fast Path Log Analysis utility uses the following input:

- An IMS/VS input recovery log data set
- A control data set that contains the execution parameters

OUTPUT

The Fast Path Log Analysis utility processing consists of the following two steps: (1) constructing Fast Path transaction detail records (FPTDR), and (2) analyzing the FPTDRs and printing the reports.

The basic unit of output from this utility is the FPTDR: one FPTDR is constructed for each Fast Path transaction processed. An FPTDR is a 122-byte EBCDIC logical record consisting of all the data associated with a given transaction (compiled from one or more log records) and a sequence number that indicates the order in which this transaction entered sync point processing. The last log record that can supply data for each FPTDR is the dequeue record for that transaction.

The Fast Path Log Analysis utility uses the FPTDRs to form the following three output data sets:

- Total Traffic, normally a tape or direct access data set that contains every FPTDR. This data set can be passed to a subsequent job step for sorting and printing, or for additional analysis.
- Exception Traffic, normally a direct access or tape data set that contains only those FPTDRs that were exceptional and thus appear in the Detail Listing of Exception Transactions report. This data set can be passed to a subsequent job step for sorting and printing, or for additional analysis.
- Formatted Reports, normally a printer output data set that consists of several reports formed by various combinations of transaction detail records. These reports are described below.

The Total Traffic and Exception Traffic data sets are provided to make it convenient for the user to post process performance data formatted by the utility without using the log data set. For example, inspection of reports can indicate that the Total Traffic data set should be sorted and printed in physical line number and terminal sequence to analyze a problem possibly related to line activity.

Records are written to the Total Traffic and Exception Traffic data sets in the order in which they are completed—in the order of dequeue records for the normal transaction sequence. However, the sequence number assigned for each transaction is determined by the order in which the transaction enters sync point processing.

FORMAT OF TOTAL TRAFFIC AND EXCEPTION TRAFFIC DATA SETS

The Fast Path Log Analysis utility gathers the Fast Path transaction detail records that are written to the Total Traffic and Exception Traffic data sets. A single logical record is written for each FPTDR. Data set organization is fixed blocked, LRECL=122, and BLKSIZE specified by the user in the //SYSUT1 and //SYSUT2 DD statements. The logical records are written in order of the dequeue record associated with each transaction.

The data code is EBCDIC for all characters. The format of each logical record is:

Offset (Decimal)	Length (Bytes)	Contents, Meaning
	5	Sequence number field, assigned in sync point order
	1	Blank
	8	Transaction code, left adjusted, blank fill or PSB name for transactions which are not message-driven.
14	1	Blank
15	10	Sync point clock times in hours, minutes, seconds and tenths of seconds in the format hhmmsstt
25	1	Blank
26	1	Sync failure reason code character, or blank A MSDB verify failure B MSDB arithmetic overflow C DEDB sequential dependent area full D DEDB sequential dependent insert caused buffer overflow E DEDB sequential dependent buffer overflow three times F DEDB area not available for use G Dynamic MSDB area full H MSDB required segment not found L ROLB call O Out of resources R Resource deadlock S Out of space in data sets U Application program abend
27	1	Blank
28	8	Routing code. Identification of the balancing group
36	1	Blank
37	8	Logical terminal name for the input transaction
45	1	Blank
46	4	Relative line number for input, edited decimal integer
50	1	Blank

Offset (Decimal)	Length (Bytes)	Contents, Meaning
51	4	Relative physical terminal number, edited decimal integer
55	1	Blank
56	2	Number of transactions on queue at sync point, edited decimal integer
58	1	Blank
59	4	Input queue time, in seconds and tenths of seconds in the edited decimal format 'SS.T'
63	1	Blank
64	4	Processing time, in seconds and tenths of seconds in the edited decimal format 'SS.T'
68	1	Blank
69	4	Output queue time, in seconds and tenths of seconds in the edited decimal format 'SS.T'
73	1	Blank
74	5	Total transit time, in seconds and tenths of seconds in the edited decimal format 'SS.T'
79	1	Blank
80	5	Output time, in seconds and tenths of seconds in the edited decimal format 'SS.T'
85	1	Blank
86	3	Input message length in bytes, edited decimal integer
89	1	Blank
90	3	Output message length in bytes, edited decimal integer
93	2	Blanks
95	3	Number of DEDB calls, edited decimal integer
98	3	Blanks
101	3	Number of DEDB GETS and PUTS, edited decimal integer
104	2	Blanks
106	3	Number of MSDB calls, edited decimal integer
109	1	Blank

Offset (Decimal)	Length (Bytes)	Contents, Meaning
110	3	Number of buffers used minus the normal allocation. Edited decimal integer. If negative, the following column contains a minus sign (-).
113	1	Blank or minus sign (-)
114	3	Number of control interval contentions, edited decimal integer
117	1	Blank or *; * indicates this transaction had waited for and then was allowed to exceed a buffer allocation
118	1	Blank
119	3	Number of waits for data base buffers, edited decimal integer

The zeros on the left of edited fields are suppressed; however, there is always at least a single nonblank digit to the left of a decimal point.

Fields that are unused (for example, the output time field of a record that has no dequeue information) are set to blanks.

Decimal integer fields that contain overflow values are indicated by the value of all 9's. For example, an overflow value for input queue time appears as 99.9. This method of indicating overflow causes overflowed fields to sort high.

DETAIL LISTING OF EXCEPTION TRANSACTIONS REPORT

The user defines, with input parameters, what is considered to be an exceptional transit time value for each online Fast Path transaction. Transit time is defined as the sum of intervals A, B, and C (defined earlier in this section). Output time D is not included for this purpose. Any transaction with a transit time that exceeds the specified exceptional value is included in an output report titled "Detail Listing of Exception Transactions Report" and can be written on the SYSUT2 data set. Figure 90 on page 463 is an example of a Detail Listing of Exception Transactions report.

The column headings of the Detail Listing of Exception Transactions report are:

- SEQ NO.: sequence in which this transaction entered sync point processing. Five print positions are provided for this column; therefore, if there are more than 99 999 transactions during the specified analysis period, the sequence number wraps to 0.
- TRANCODE OR PSB: the transaction code, or PSB name if the transaction is not message driven.
- SYNC POINT TIME: the clock time at sync point processing.
- S F: sync failure reason code character for transactions which fail sync processing. A nonblank character in this column indicates sync failure and, for this case, the columns described below are blank. The meaning of nonblank codes A through F is in Figure 90 on page 463. Information relating to sync failures is obtained from type X'5938' log records.

- ROUTING CODE: identification of the balancing group. This column is blank for transactions that are not message driven.
- LOGICAL TERMINAL: the input LTERM name for this transaction.
- PHYSICAL LINE TERM: relative line number and relative physical terminal for input.
- Q CT: the number of transactions on queue (LBG) when this transaction entered sync point processing.
- IN-Q: time interval A, input queue time. This column and ROUTING CODE through Q CT columns above are blank for transactions that are not message driven.
- PROC: time interval B, processing time. This column and OUTQ through MSG-LEN OUT are blank for transactions that are not message driven.
- OUTQ: time interval C, output queue time.
- TOTAL: the sum of time intervals A, B, C. This is the transit time as defined for the utility. The magnitude of this sum exceeds the exception value for the transaction code.
- OUT TIME: time interval D, output time (to dequeue).
- MSG-LEN IN: the input message length in characters.
- MSG-LEN OUT: the output message length in characters.
- DEDB CALL: the number of DL/I calls to DEDB during processing.
- DEDB I/O: the sum of GETs and PUTs for the DEDB calls.
- MSDB CALL: the number of MSDB calls during this processing.
- DB BUF: the algebraic value of the difference between the number of data base buffers used and the number of buffers in the normal allocation. A positive value indicates the number of overflow buffers used; a negative value indicates the number unused of the normal allocation.
- CI CONT: the number of control interval contentions during this processing. An asterisk immediately to the right of this column indicates that this transaction has waited for and then was allowed to exceed a buffer allocation limit.
- BUF WTS: the number of waits for data base buffer during this processing.

Exceptional transit time values can be specified separately for each Fast Path transaction code. A global value can be specified that applies to all other unspecified transaction codes.

The Detail Listing of Exception Transactions report includes detail records for transactions that are not message driven. They are also sequenced according to sync point processing time. Data in the ROUTING CODE through MSG-LEN OUT columns are not applicable to transactions that are not message driven and are left blank.

The Detail Listing of Exception Transactions report includes transactions for which a dequeue log record is not found. For these transactions, the output queue time and therefore the total transit time are unknown and are not formatted. This condition is marked in the report by the characters NO DEQ under the TOTAL column.

DETAIL LISTING OF EXCEPTION TRANSACTIONS: PAGE 3

SEQ NO.	TRANCODE OR PSB	SYNC POINT TIME	S E	ROUTING CODE	LOGICAL TERMINAL	PHYSICAL LINE	LTERM	Q CT	TRANSIT TIME (SEC). IN-Q	PROC	QUID	DIAL	OUT TIME	MSG-LEN IN	OUT	DEDB... CALL	IZO	MSDB CALL	DB BUF	CI CONT	BUF WTS
1	TCODE1	10:01:01.1		SAVING	TELLER1	1111	1	0	0.1	0.3	0.7	1.1	2.6	25	36	7	1	9	0	1	0
5	TCODE1	10:05:05.0	A																		
6	TCODE1	10:06:06.0	D																		
7	TCODE2	10:07:07.0	C																		
8	TCODE2	10:08:08.0	H																		
9	TCODE3	10:09:09.0	E																		
10	TCODE3	10:10:10.0	F																		
11	TCODE7	10:11:11.0	G																		
12	PROG1	10:12:12.0														9	2	11	0	1	1
13	PROG3	10:13:13.0														12	2	13	2	0	2
14	PROG1	10:14:14.0														4	1	4	0	0	0
3	TCODE1	10:03:03.0		SAVING	TELLER3	2222	3	3	0.3	0.6	1.2	2.1	30.0	40	56	16	1	12	4	1	2
4	TCODE3	10:04:04.0		WITHDRAW	TELLER4	3333	4	4	0.4	0.8	0.6	1.8	40.0	28	86	13	1	10	3-	2	0
17	TCODE4	11:17:17.0		TRUST	TELLER17	3333	17	0	0.4	1.4	0.9	2.7	60.7	73	139	14	2	18	2	3*	2
25	PROG2	13:27:27.0														18	0	14	0	2	0
26	PROG4	14:28:28.0	B																		
18	TCODE4	11:18:18.0		TRUST	TELLER18	3333	18	1	0.6	1.2	0.8	2.6	39.5	60	119	10	1	12	1-	0*	1
19	TCODE5	11:19:19.0		LOAN	TELLER19	2222	19	98	65.5	65.5	65.5	196.6	999.9	999	999	999	999	999	999	999	999
21	TCODE2	11:22:22.0		CHECKING	TELLER22	2222	22	9	0.1	0.6	1.1	1.8	13.5	22	49	12	2	10	1	1	2
22	TCODE3	11:23:23.0		WITHDRAW	TELLER23	4444	23	2	0.2	0.2	1.2	1.6	2.1	36	48	10	2	8	2	0	3
23	TCODE3	12:25:25.0		WITHDRAW	TELLER25	3333	25	0	0.2	0.2	1.3	1.7	2.0	18	27	4	0	18	0	0	0
24	TCODE2	12:26:26.0		CHECKING	TELLER26	2222	26	1	0.4	0.5	2.8	3.7	3.0	20	25	3	0	16	2-	0	0
27	TCODE7	0:32:32.0		AUTOCASH	TELLER32	4444	32	0	0.1	0.2	1.1	1.4	5.0	16	27	10	2	12	0	0	0
28	TCODE7	1:33:33.0		AUTOCASH	TELLER33	4444	33	0	0.1	0.3	1.2	1.6	10.5	18	30	12	2	8	1	0	1
29	TCODE7	2:34:34.0		AUTOCASH	TELLER34	4444	34	1	0.2	0.4	1.4	2.0	9.8	16	26	14	1	10	1-	1	2
16	TCODE3	10:16:00.6		WITHDRAW	TELLER16	4444	16	3	0.1	0.5		NO DEQ		22	44	14	1	6	1	2	1
15	TCODE2	10:15:15.0		CHECKING	TELLER15	3333	15	9	0.2	0.3		NO DEQ		13	26	10	2	4	0	0	0

Figure 90. Detail Listing of Exception Transactions Report

The Detail Listing of Exception Transactions report also includes transactions for which sync failure occurred. For these transactions, only sequence number, transaction code, sync point clock time, and failure reason code character are given.

OVER-ALL SUMMARY BY TRANSACTION CODE REPORT

A summary report is produced, by transaction code, for every message-driven Fast Path transaction read from the log by the Fast Path Log Analysis utility. The column headings for this report are:

- TRANS CODE: the transaction code.
- NO. OF TRANS: the number of occurrences of the transaction code for which a transit time value was computed.
- TRANSIT TIME: the average and maximum values of transit time intervals.
- LENG OF INPUT: the average and maximum values of input message length.
- LENG OF OUTPUT: the average and maximum values of output message length.
- NO. of DEDB CALL: the average and maximum numbers of DEDB calls per processing interval.
- NO. OF DEDB I/O: the average and maximum numbers of GET and PUT for DEDB per processing interval.
- NO. OF MSDB CALL: the average and maximum numbers of MSDB calls per processing interval.
- AVG BUF: for this transaction, the average value of the difference between the number of buffers used and the normal allocation. A positive value indicates overflow buffers. A negative value indicates the number less than normal allocation.
- SYNC FAIL: the total number of occurrences of this transaction code that failed sync point processing.
- CI CONT: the total number of times control interval contentions occurred for this transaction code.
- BUF WTS: the total number of waits for data base buffers for this transaction code.

The averages are computed using the number of occurrences of the transaction code for which a transit time value was computed. Figure 91 on page 465 is an example of the Over-All Summary by Transaction Code report.

OVERALL SUMMARY BY TRANSACTION CODE: PAGE 5

TRANS CODE	NO. OF TRANS	TRANSIT TIME IN SEC. (IN-Q THRU OUT-Q) AVG	TRANSIT TIME IN SEC. (IN-Q THRU OUT-Q) MAX	LENG OF INPUT MESSAGE (CH) AVG	LENG OF INPUT MESSAGE (CH) MAX	LENG OF OUTPUT MESSAGE (CH) AVG	LENG OF OUTPUT MESSAGE (CH) MAX	NO. OF DEDB CALL AVG	NO. OF DEDB CALL MAX	NO. OF DEDB I/O AVG	NO. OF DEDB I/O MAX	NO. OF MSDB CALL AVG	NO. OF MSDB CALL MAX	AVG BUF	..TOTAL NO. OF.. SYNC FAIL	CI CONT	BJF DIS
TCODE7	5	1.7	2.0	16	18	27	30	12	14	2	2	10	12	0.0	1	1	3
TCODE2	3	2.1	3.7	25	34	33	49	8	12	1	2	12	16	0.3	2	1	3
TCODE3	3	1.7	1.8	27	36	53	86	9	13	1	2	12	18	0.3-	2	2	3
TCODE5	1	196.6	196.6	999	999	999	999	999	999	999	999	999	999	999.9	0	1000	1000
TCODE4	2	2.7	2.7	66	73	129	139	12	14	2	2	15	18	0.5	0	3	3
TCODE6	1	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0.0	0	0	0
TCODE1	2	1.6	2.1	32	40	46	56	12	16	1	1	11	12	2.0	2	2	2

Figure 91. Over-All Summary by Transaction Code Report

SUMMARY OF EXCEPTION DETAIL BY TRANSACTION CODE REPORT

A summary report is produced for that subset of transactions that exceed the exception criteria specified by the analysis control parameters. The format of this report is identical to that of the Over-All Summary by Transaction Code report. Transactions for which a dequeue record was not found are not included in this summary. Records from transactions that are not message driven are also not included in this summary. Figure 92 on page 467 is an example of the Summary of Exception Detail by Transaction Code report.

Figure 92. Summary of Exception Detail by Transaction Code Report

SUMMARY OF EXCEPTION DETAIL BY TRANSACTION CODE: PAGE 4

TRANS CODE	NO. OF TRANS	TRANSIT TIME IN SEC. (IN-Q THRU OUT-Q) AVG	TRANSIT TIME IN SEC. (IN-Q THRU OUT-Q) MAX	LENG OF INPUT MESSAGE (CH) AVG	LENG OF INPUT MESSAGE (CH) MAX	LENG OF OUTPUT MESSAGE (CH) AVG	LENG OF OUTPUT MESSAGE (CH) MAX	NO. OF DEDB CALL AVG	NO. OF DEDB CALL MAX	NO. OF DEDB I/O AVG	NO. OF DEDB I/O MAX	NO. OF MSDB CALL AVG	NO. OF MSDB CALL MAX	AVG BUF	..TOTAL NO. OF.. SYNC FAIL	CI CONF	BUF WTS
TCODE7	3	1.7	2.0	16	18	27	30	12	14	2	2	10	12	0.0	1	1	3
TCODE2	2	2.8	3.7	21	22	37	49	8	12	1	2	13	16	0.5-	2	1	2
TCODE3	3	1.7	1.8	27	36	53	86	9	13	1	2	12	18	0.3-	2	2	3
TCODE5	1	196.6	196.6	999	999	999	999	999	999	999	999	999	999	999.9	0	1000	1000
TCODE4	2	2.7	2.7	66	73	129	139	12	14	2	2	15	18	0.5	0	3	3
TCODE1	2	1.6	2.1	32	40	46	56	12	16	1	1	11	12	2.0	2	2	2

SUMMARY OF MPP AND BMP TRANSACTIONS BY PSB NAME REPORT

A summary report is produced for all transactions from MPP and BMP regions that entered sync point processing during the interval specified for the analysis. Data is summarized by PSB name. The headings for this report have the same meaning as those given for the Summary of Exception Detail by Transaction Code report. Figure 93 on page 469 is an example of the Summary of MPP and BMP Transactions by PSB Name report.

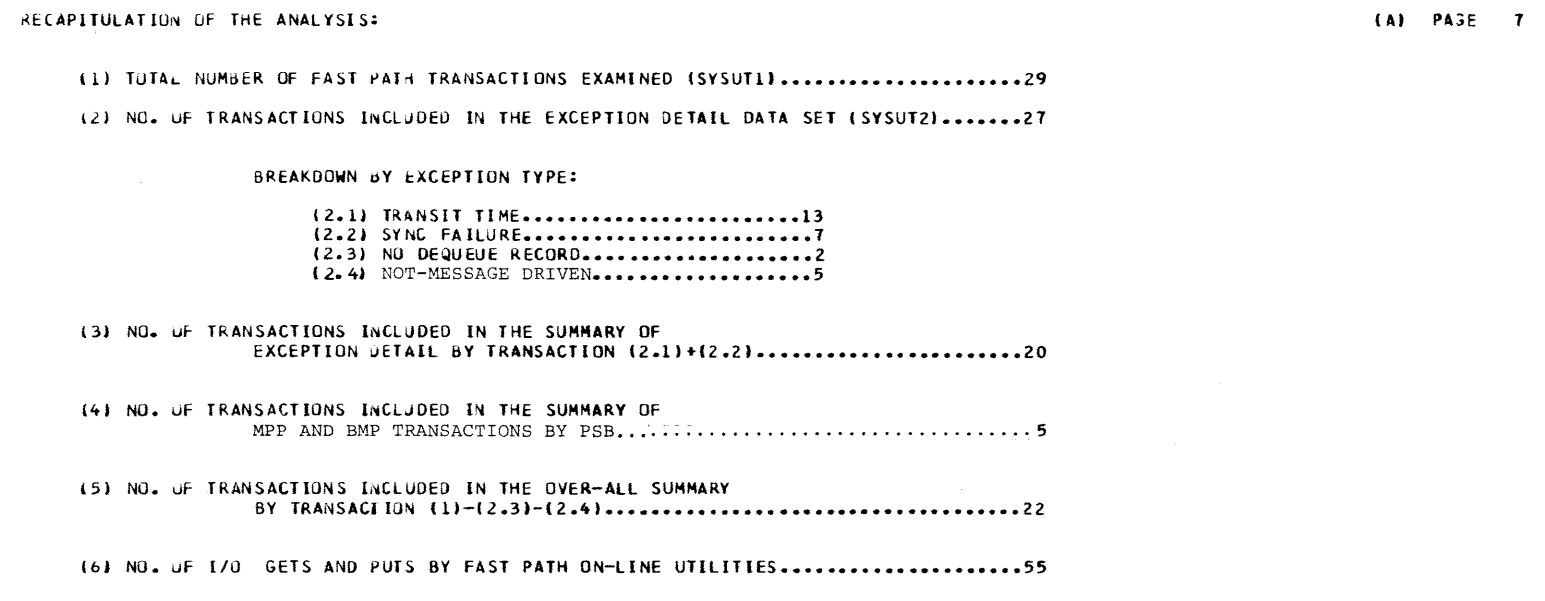
SUMMARY OF MPP AND BMP TRANSACTIONS BY PSB NAME:

PSB NAME	NO. OF TRANS	NO. OF DEDB CALL AVG	NO. OF DEDB CALL MAX	NO. OF DEDB I/O AVG	NO. OF DEDB I/O MAX	NO. OF MSDB CALL AVG	NO. OF MSDB CALL MAX	AVG BUF	..TOTAL NO. OF.. SYNC FAIL	CI CONT	BUF WTS
PROG4	0	0	0	0	0	0	0	0.0	1	0	0
PROG2	1	18	18	0	0	14	14	0.0	0	2	0
PROG1	2	7	9	2	2	8	11	0.0	0	1	1
PROG3	1	12	12	2	2	13	13	2.0	0	0	2

Figure 93. Summary of MPP and BMP Transactions by PSB Name Report

RECAPITULATION OF THE ANALYSIS REPORT

The first five items in this report are self-explanatory. The last item indicates the total number of input/output operations (sum of GETs and PUTs) by Fast Path online utilities during the specified analysis period. This count is obtained by accumulating a statistic field in the Fast Path X'5953' type log records. Figure 94 on page 471 is an example of the Recapitulation of the Analysis Report.



```
RECAPITULATION OF THE ANALYSIS:                                    (A)  PAGE   7

     (1) TOTAL NUMBER OF FAST PATH TRANSACTIONS EXAMINED (SYSUT1).....29

     (2) NO. OF TRANSACTIONS INCLUDED IN THE EXCEPTION DETAIL DATA SET (SYSUT2).....27

                    BREAKDOWN BY EXCEPTION TYPE:

                         (2.1) TRANSIT TIME.....13
                         (2.2) SYNC FAILURE.....7
                         (2.3) NO DEQUEUE RECORD.....2
                         (2.4) NOT-MESSAGE DRIVEN.....5


     (3) NO. OF TRANSACTIONS INCLUDED IN THE SUMMARY OF
                    EXCEPTION DETAIL BY TRANSACTION (2.1)+(2.2).....20


     (4) NO. OF TRANSACTIONS INCLUDED IN THE SUMMARY OF
                    MPP AND BMP TRANSACTIONS BY PSB.....5


     (5) NO. OF TRANSACTIONS INCLUDED IN THE OVER-ALL SUMMARY
                    BY TRANSACTION (1)-(2.3)-(2.4).....22


     (6) NO. OF I/O  GETS AND PUTS BY FAST PATH ON-LINE UTILITIES.....55
```

Figure 94. Recapitulation of the Analysis Report

CONTROL STATEMENTS

Control statements in the SYSIN data set control the Fast Path Log Analysis utility. The user can specify the time period of Fast Path execution for which the analysis is to be performed. This is expressed as the starting time (clock time) and/or an ending time. Transactions whose sync point time stamps fall within this interval are processed. If no interval is specified, the entire log data set is processed.

After the log is processed up to the end time specified, scanning continues to find dequeue records related to transactions that were processed during the specified analysis time interval.

Multivolume log data sets are processed by specifying multiple volumes in the //LOGTAPE DD statement or by concatenation of DD statements.

Transit Time Exception Specification

You can limit the volume of printed output produced by specifying for each transaction code a transit time value that is considered to be exceptional. Occurrences of transaction transit times that are less than the exceptional value do not appear in the Detail Listing of Exception Transactions report. A different exception transit time can be specified for each unique transaction code. A global value can also be specified for all transaction codes that are not individually specified. A separate summary report is produced for those transactions that exceed the exception criteria.

A detail report of all the transactions processed from the log data set can be produced either by not specifying an exceptional transit time (default=0) or by printing the total FPTDR data set in a subsequent job step.

An upper limit can be placed on the number of transactions that are printed in the Detail Listing of Exception Transactions report. This limit can be used to prevent the production of unexpectedly large output listings.

ANALYSIS PARAMETER STATEMENT FORMATS

All statements begin in column 1. The statements can appear in any order and are listed in the SYSPRINT data set for verification.

Starting Date Specification

The user can specify the date of the earliest transaction to be processed in Julian format. Transactions with an earlier date are ignored. If the starting time is also specified, transactions with an earlier sync point time on that day are also ignored. The format is:

STARTDAY=YYDDD (last two digits of the year and the sequential number of the day, running from 1 to 366)

The default value is the date IMS was started, from the type 42 log record.

Ending Date Specification

The user can specify the date of the latest transaction to be processed in Julian format. Transactions with a later date are ignored. If the ending time is also specified, transactions with a later sync point time on that day are also ignored. The format of this parameter is as follows:

ENDDAY=YYDDD (last two digits of the year and the sequential number of the day, running from 1 to 366)

The default value, if ending time is specified, is the date IMS was started from the type 42 log record. If ending time is less than starting time, the default is one day later. If neither ending date nor ending time are specified, the entire dataset is processed.

Starting Time Specification

The user can specify the time of the earliest transaction to be processed. Transactions with an earlier sync point time are ignored. The format is:

START=HH:MM:SS (in hours, minutes, and seconds for a 24-hour clock)

The default value is 00:00:00, which causes the analysis to begin with the first transaction on the log data set.

Ending Time Specification

The user can specify the sync point time of the latest transaction to be processed. Transactions with a later sync point time are ignored. The format is:

END=HH:MM:SS (in hours, minutes, and seconds for a 24-hour clock)

The default value is such that the analysis ends with the last transaction on the log data set.

The date on the log data set is not explicitly specified by a parameter statement. The data is implicit with the specification for the log data set that is in the JCL Requirements section. The Julian date is read from the log header record when execution begins, and this date is printed as part of the parameter summary for verification.

Exceptional Transit Time Specification

The user can specify a time interval for each Fast Path transaction that is to be considered exceptional for reporting purposes. The format is:

TT (TRANCODE)=SS.T (in seconds and tenths of seconds)

The transaction code, up to 8 characters, is enclosed in parentheses. As many as 100 individual transaction codes can be specified. A global value of exceptional transit time is specified as follows:

TT(*)=SS.T (in seconds and tenths of seconds)

This value applies to all transaction codes that are not individually specified. Individual specification overrides the global value. The default value for the global exceptional transit time is 0. A practical upper limit of exceptional transit time is 65.5 seconds. This limitation results from the field size used to express the time intervals (A), (B), and (C) in the Fast Path log records.

Not Message-Driven Option

The user can specify transactions which are not message-driven to be considered exceptions and are to be included in the Detail Listing of Exception Transactions report. The accepted formats are:

NON-MESSAGE

or

NOT-MESSAGE

Note that "nonmessage-driven" transactions are no longer supported, so in this case "NOT-MESSAGE" means transactions not message-driven.

Detail Listing of Exception Transactions Report Size Limitation

The user can limit the number of lines printed in the Detail Listing of Exception Transactions report. After this limit is reached, the analysis continues; however, no further transactions are printed in the Detail Listing of Exception Transactions report. The format is:

MAXDETAIL=n

where n is an integer of no more than seven digits. The default value is 1000. The limitation of printed output lines does not affect the number of exception detail records that are written to the exception detail traffic data set (SYSUT2).

Printed Page Line Count Specification

The user can specify the number of lines printed per page for the printed reports. The format is:

LINECNT=n

where n is an integer greater than 5. The value specified applies to titles and headers so that 6 is the minimum allowable value. The default value is 55 lines per page.

Each parameter statement is listed in the SYSPRINT data set exactly as it is read for verification. Figure 95 on page 475 is an example of parameter statements read from the SYSIN data set and how they are listed in the SYSPRINT data set. After all parameter statements are read, the utility prints a summary display of either the parameters supplied or the default values that are used for parameters not specified. Figure 96 on page 476 is an example of the parameter display. Date information is obtained from the log buffer control record (log record code X'42').

```
LOG ANALYSIS FOR IMS/VS FAST PATH                                    PAGE    2

SPECIFIED INPUT PARAMETERS:

          ANALYSIS START TIME:  10:00:01     DATE:   76348
                    END TIME:  03:00:00     DATE:   76349

          NOT-MESSAGE DRIVEN TRANSACTION DETAIL REQUESTED.

          A MAXIMUM OF     500 EXCEPTIONAL TRANSACTIONS WILL BE LISTED.

          TRANSIT TIME EXCEPTION VALUES:
                                          EXCEPTION VALUE IN SEC.
                    TRANSACTION CODE      (IN-Q THRU OUT-Q)

                        *GLOBAL*                  1.0
                        TCODE1                    0.5
```

Figure 95. Specified Input Parameters

```
                                                                PAGE    1
LOG ANALYSIS FOR IMS/VS FAST PATH

THE FOLLOWING PARAMETER CARDS WERE READ FROM SYSIN:

      START=10:00:04
      END=03:00:00
      NOT-MESSAGE
      MAXDETAIL=500
      LINECNT=50
      TT(*)=1.0
      TT(TCODE1)=0.5
```

Figure 96. Parameter Display

JCL REQUIREMENTS

EXEC
Executes the Fast Path Log Analysis utility.

```
//EXEC PGM=DBFULTA0
```

STEPLIB DD
Describes the program library that contains the DBFULTA0 load module.

```
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
```

SYSPRINT DD
Describes the data set that receives the printed output of DBFULTA0—reports, messages, and parameter statement images. This DD statement is required.

```
//SYSPRINT DD SYSOUT=A
```

SYSUT1 DD
Describes the data set that receives the total traffic output of DBFULTA0. This is a sequential data set consisting of every Fast Path transaction detail record formed by DBFULTA0. Each FPTDR is EBCDIC and LRECL=122 bytes. This DD statement is optional and is omitted if the total traffic data set is not required. Block size specification is optional. The default value for BLKSIZE is 1220.

```
//SYSUT1 DD DSN=&&TOTAL,DISP=(,PASS),UNIT=SYSDA,
//   SPACE=(CYL,(1,1)),DCB=BLKSIZE=2440
```

SYSUT2 DD
Describes the data set that receives the exception traffic output of DBFULTA0. This is a sequential data set consisting of the Fast Path transaction detail records that are exceptions. It is a copy of the Detail Listing of Exception Transactions report with headings and carriage control characters suppressed. The logical record length is 122 bytes. If this DD statement is omitted, this data set is not produced. Block size specification is optional. The default value of BLKSIZE is 1220. The MAXDETAIL parameter specification does not apply to this data set.

```
//SYSUT2 DD DSN=&&EXCEP,DISP=(,PASS),UNIT=SYSDA,
//   SPACE=(CYL,(1,1)),DCB=BLKSIZE=1220
```

LOGTAPE DD
Describes the input log data set. This DD statement is required.

```
//LOGTAPE DD DSN=IMSVS.LOG,DISP=OLD,VOL=SER=XXXXXX,
//   UNIT=xxxx
```

SYSIN DD
Describes the input control data set. This data set is used to specify execution parameters. This DD statement is optional. The following is a sample control data set:

```
//SYSIN DD *
START=09:59:59    24 hour notation,
END=12:00:00      note colons
LINECNT=45        lines per page for reports
NOT-MESSAGE       include transactions not message-driven
MAXDETAIL=5000    exceptions limit detail listing
TT(*)=15.0
TT(TCODE1)=3.0
TT(TCODE2)=2.5
TT(TCODE3)=1.0
```

MESSAGES AND CODES

User abend codes are not generated.

The following return codes are produced:

Code	Meaning
0	Successful completion of analysis
4	Analysis prematurely ended, partial results produced
8	Unable to perform analysis
12	Unable to open ddname SYSPRINT

See IMS/VS Messages and Codes Reference Manual for an explanation of the messages generated by this utility.

PART 5. PERFORMANCE REPORTING AND SERVICE

Part 5 has two chapters that describe the utilities used to produce performance-related summary reports; to copy messages onto a system output device; and to verify compatibility of system definitions for IMS/VS systems in a multisystem environment.

Chapter 11, "Performance Reporting Utilities," describes the utilities that organize, format, and print performance-related reports. Control statements and examples and report examples are included.

Chapter 12, "System Service Utilities," describes the print utility that copies messages produced by the online control program from the online data sets to a system output device. This chapter also describes the utility that verifies the consistency and compatibility among IMS/VS system definitions in a multisystem environment.

CHAPTER 11. PERFORMANCE REPORTING UTILITIES

DATA BASE (DB) MONITOR REPORT PRINT PROGRAM (DFSUTR30)

The DB Monitor Report Print program (DFSUTR30) is an offline utility that organizes and prints performance-related data collected by the DB Monitor (DFSMNTB0) during execution of the IMS/VS DB system. For detailed information on output from each of the reports and information on how to read them, see IMS/VS Data Base Administration Guide. This section describes operation of the DB Monitor.

RESTRICTIONS: The DFSUTR30 program depends on the data records on the data set produced by DFSMNTB0. The accuracy of reported times and statistics reflected in the reports depends on those in the data set. Records of various events are expected in pairs—a start-event record and an end-event record; events are not counted and reported unless both are received.

Note: Monitor reports will have invalid data when used with CICS/VS multithreading.

INPUT TO DFSUTR30

The DB Monitor Report Print program runs in batch mode, with one job step for each monitor trace period.

JCL Requirements

JOB
Initiates the job.

EXEC
Specifies the program name (PGM=DFSUTR30,REGION=256K).

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

SYSPRINT DD
Specifies the output data set, usually SYSOUT=A.

SYSUT1 DD
Specifies the input data set which is a labelled data set written by monitor module DFSMNTB0. It can be a separate data set (ddname= and dsname=IMSMON) or the system log (dsname=IMSLOG).

ANALYSIS DD
Specifies the Analysis Control data set. This file must be in card image format. Use DD DUMMY for default parameters—no distribution report, and input is the first trace interval on the tape.

Analysis Control Data Set: The Analysis Control data set has three record types that allow the user to request the Distribution Appendix report, to redefine distribution intervals, and to specify which trace interval is to be processed.

- To generate the Distribution Appendix report, specify either DISTRIBUTION or DIS, beginning in column 1.
- To override the default distribution intervals, specify control statements in the form Dn n1,n2,...

- To denote which trace interval (other than the first, which is the default) on the input data set is to be processed, specify FILE=nn, or FILE=n, beginning in column 1.

Note that the FILE= specification does not refer to OS files. It refers to trace intervals recorded within an OS file.

JCL EXAMPLE (DFSUTR30)

The following example shows the JCL for a complete set of reports on the first trace interval from a tape with a serial number of IMSDA1:

```
//TRACE    JOB  (969,6014),CHAPMAN,MSGLEVEL=(1,1),CLASS=A
//STEP1    EXEC PGM=DFSUTR30,REGION=256K
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=A
//SYSUT1   DD   DSN=IMSMON,VOL=SER=IMSDA1,UNIT=TAPE,DISP=(OLD,KEEP)
//ANALYSIS DD   *
DISTRIBUTION
/*
```

If the distribution for D13 were to be modified, and the second trace interval specified, the Analysis Control data set would have to be modified as follows:

```
//ANALYSIS DD  *
DISTRIBUTION
FILE=2
D13 ,,2,4,6,8,10,12
```

Note that the first two intervals would not be changed in the example but would remain as 0 and 1, respectively.

DATA COMMUNICATION (DC) MONITOR REPORT PRINT PROGRAM (DFSUTR20)

The DC Monitor Report Print program (DFSUTR20) is a batch program that takes the data collected by the DC Monitor (DFSMNTR0) and prints summary reports and distribution displays of the data. The report formats and the nature of information in the reports are identical or similar to those printed by the DB Monitor Print Program (DFSUTR30). The DC Monitor Report Print program does not support CICS/VS.

For detailed information on output from the reports and an explanation of how to read them, see IMS/VS System Administration Guide.

If the Monitor does not collect certain types of information usually found in a particular report, that report, or the section of that report that would normally contain the information, is not produced. For example, if no checkpoints occur, only the headings for checkpoint are printed.

In any report for which data is captured at the start and end of the Monitor trace interval, the report will display the data captured at these intervals, and their difference. Since data for these reports is needed at both intervals, these reports will NOT be generated if the IMS/VS control region is terminated prior to the Monitor trace.

Most of the terms used in reports printed by the DB Monitor Report Print program (DFSUTR30) also appear in reports printed by the DC Monitor Report Print program (DFSUTR20). An explanation of the terms that apply to DFSUTR20 is given in IMS/VS System Administration Guide.

REPORT SELECTION: By means of the Analysis Control data set (described later under "Input to DFSUTR20"), the user can select a subset of reports to be printed. The time required for the processing and the printing of reports and the storage required to produce the reports are a function of the reports that are requested. In the Analysis Control data set, specify:

DLI if a call summary report is desired.

ONLY DLI if only a call summary report is desired.

DIS or DISTRIBUTION if a Distribution Appendix is required for the reports requested.

If none of the above options is selected, all reports except the Call Summary report and the Distribution Appendix report will be printed.

INPUT TO DFSUTR20

The Monitor Report Print program runs as a batch program, with a tape sequential data set as input. The contents of this data set were created by the IMS/VS Monitor module (DFSMNTR0) during IMS/VS online execution.

JCL REQUIREMENTS

// JOB
Initiates the job.

// EXEC
Specifies the program name (PGM=DFSUTR20, REGION=512K).

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

//SYSPRINT DD
Specifies the output data set that is to contain the reports and control messages. It is usually coded as SYSOUT=A. The DCB parameters for this data set are RECFM=FBA and LRECL=133. BLKSIZE may be provided on the SYSPRINT DD card and must be a multiple of 133. If the BLKSIZE is not provided, a default value of 133 will be used.

//SYSUT1 DD
Specifies the input data set to be analyzed. It is a labelled sequential data set written by the monitor module DFSMNTR0. (The ddname and dsname are IMSMON in the IMS/VS procedure.)

//ANALYSIS DD
Specifies the Analysis Control data set. This file must be in card image format.

Analysis Control Data Set: The Analysis Control data set determines which reports are to be printed and allows for distribution redefinition for the Distribution reports. See "Specifying Distribution Redefinitions" for information on how to respecify the distributions.

- If you want only the Call Summary report printed, include the following statement anywhere in the Analysis Control data set for this run. The statement starts in card image column 1.

```
1
ONLY DLI
```

- To generate the Call Summary report, include the following statement anywhere in the Analysis Control data set for this run. If this statement is not included, the default option is taken; that is, all reports except the Call Summary report are printed. The statement starts in card image column 1.

```
1
DLI
```

- To generate the optional Distribution Appendix report, include the following statement anywhere in the Analysis Control data set. If this statement is not included, only the summary reports are printed. The statement starts in card image column 1.

```
1
DIS
```

Specifying Distribution Redefinition: The general format for specifying a user redefinition of a distribution is:

```
Dn     n1,n2...
```

where

Dn
starts in column 1 and is the distribution identifier (ID).

n1 through n9
are each 8 digits or less, and each is a positive number between 0 and 99 999 999.

Each redefinition may occupy more than one card, if necessary. The format for continuation cards follows the OS/VS rules:

- The last value on the first card must be followed by a comma and at least one blank.
- The first value on the continuation card cannot start before column 2 and not after column 10.
- Comments can be punched onto the cards if they are preceded by at least one blank.

Assume that the distribution for region elapsed execution time is identified as D1 and has a default definition of:

```
0 1 2 3 30 300 3000 30000 3000000 30000000 INF
```

It can be redefined to be:

```
0 1 2 5 30 40 50 60 3000000 30000000 INF
```

This redefinition is accomplished by the following record in the Analysis Control data set:

```
D1 1,2,5,30,40,50,60,3000000,30000000
```

Because the numbers are positional parameters, the same redefinition could have been obtained by specifying the following:

```
D1 ,,5,,40,50,60
```

Refer to "Using Frequency Distributions from DC Monitor Output" in IMS/VS System Administration Guide for the default values for each distribution identifier. The section also explains which distributions are used by the various reports.

JCL EXAMPLE (DFSUTR20)

The following JCL produces a complete set of reports, including the Call Summary report from a tape with a serial number of IMSDA1.

```
//TRACE JOB (969,6014),CHAPMAN,MSGLEVEL=(1,1),CLASS=A
// EXEC PGM=DFSUTR20,REGION=512K
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1 DD DSN=IMSMON,UNIT=TAPE,VOL=SER=IMSDA1,DISP=(OLD,KEEP)
//ANALYSIS DD *
DLI CALL REPORT
DISTRIBUTION
/*
```

If the distribution for D30 and D2 were to be modified, the JCL would have to be modified as follows:

```
      .
      .
      .
//ANALYSIS DD *
DLI CALL REPORT
DISTRIBUTION
D30 8000,24000,50000,75000
D2 1000,2000,3000,4000,5000,6000,7000,8000,9000
/*
```

IMS/VS PROGRAM ISOLATION TRACE REPORT UTILITY PROGRAM (DFSPIRP0)

The Program Isolation Trace logs all PI enqueue/dequeue requests in X'67FA' PI trace log records when PI tracing is active. The PI Trace Report utility prints a report from these X'67FA' log records that shows only those enqueue requests that required a wait (the resource was not immediately available). The report can be restricted to a time period by specifying a PRINT control statement. The report shows:

- Requested resource (DMB name, DCB number, and 4-byte hexadecimal ID (RBA)).
- Time of the enqueue request (time of call).
- Elapsed time of the wait (how long the requesting task had to wait for the resource to become available); if PI trace timing is in effect, use of the /TRACE ALL command. Note that no elapsed wait time for Fast Path is recorded.
- Names of the requesting and holding transactions.
- Total number of waits by ID, DCB, and DMB

If PI trace timing was in effect, the PI trace log records (X'67FA') of enqueue requests for any ID appear in the log data set in the same order in which the ID was acquired. Thus the requesting transaction of an enqueue request is considered to be the holding transaction of the next enqueue request for the same ID if the latter required a wait. If timing was not in effect during PI tracing, it is possible, although not likely, that the PI trace log records may not be in the same order in which the ID was acquired. This can occur if an enqueue request acquires an ID and, just before it is added to the PI trace logger buffer, a higher priority task interrupts with an enqueue request for the same ID. This second task will be required to wait, but its PI trace log record will be ahead of the record corresponding to the holding task. This cannot occur if timing is in effect because the log records used by the trace report utility are added to the log buffer only after the resource has been acquired.

UTILITY CONTROL STATEMENT

An optional utility control statement can be used to specify the starting and stopping times desired for the report. The control statement is printed in the program output. The format is:

```
PRINT START=HHMM,STOP=HHMM,DATE=MM/DD
```

PRINT must be coded in the operation field. Keywords can be entered in any order, separated by a comma. Keywords cannot be repeated. Data can be entered from columns 1 through 71 with one or more spaces before and after PRINT. A space following a parameter indicates the end of data; anything after is considered a comment.

If only PRINT is specified, or if a control statement is omitted (SYSIN data set not provided), or if a blank control statement is supplied, the entire log data set is searched. If only PRINT and START are specified, the log data set is searched to the end from the start time. If only PRINT and STOP are specified, the log data set is searched from the beginning until the stop time.

START and STOP times are relative to the date specified or, if DATE is omitted, relative to the date on which PI tracing started, as recorded in the PI trace log records. If only PRINT and DATE are specified, the log data set is searched for records beginning at 00:00:00 on that date. If DATE and STOP are specified without START, the log data set is searched from 00:00:00 on that date until the stop time relative to that date is encountered.

HH specifies hours and must be two numeric digits from 00 through 99 relative to the date specified or, if DATE is omitted, the date on which PI tracing started. MM of START or STOP specifies minutes and must be two numeric digits from 00 through 59. DD specifies the day of the month and must be one or two numeric digits from 1 through the maximum number of days for the specified month. MM/DD must be within 12 days of the date PI tracing was started.

JCL REQUIREMENTS

//JOBLIB DD
Describes the program library containing the utility program. Its format is:

```
DSN=IMSVS.RESLIB,DISP=SHR
```

// EXEC
Invokes the utility program. Its format is:

```
//PITRACE EXEC PGM=DFSPIRP0,REGION=150K
```

//LOGTAPE DD
Describes the log input data sets. Multiple data sets are concatenated if the log extends over more than one data set. Its format is:

```
//LOGTAPE DD DSN=nnnn,DISP=OLD,VOL=SER=xxxxxx,
// UNIT=YYYY
```

where nnnn is the data set name, xxxxxx is volume serial, and YYYY is the input type of the log data set.

//PRINT DD
Describes the report data set. Its format is:

```
//PRINT DD SYSOUT=A
```

//SORTLIB DD
Describes the sort program library. Its format is:

```
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
```

//SYSOUT DD
Describes the message output data set for sort. Its format is:

```
//SYSOUT DD SYSOUT=A
```

//SORTWK01-32 DD
Describes the sort program's work data sets. The space defined may vary. The number of data sets must be at least three. These data sets usually reside on a direct access device, but a tape volume can be used instead. For disk sort, the format is:

```
//SORTWKnn DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
```

SYSDA must equal one type of disk storage because the sort program does not allow work areas across mixed device types.

//SYSPRINT DD
Describes the system messages data set. The format is:

```
//SYSPRINT DD SYSOUT=A
```

//SYSIN DD
Describes the data set containing the optional utility control statement (described in next section). If it is included in the input stream, this DD statement is normally DD *. This statement can be omitted if the optional utility control statement is also omitted.

JCL EXAMPLE

The following example requests a report of all enqueues that required a wait and were requested between 0900 and 0930 on the date that PI tracing was started.

```
//PITSTATS  JOB  MSGLEVEL=1
//JOBLIB    DD   DSN=IMSVS.RESLIB,DISP=SHR
//          EXEC PGM=DFSPIRP0,REGION=150K
//LOGTAPE   DD   DSN=IMSLOG,UNIT=TAPE,VOL=SER=XXXXXX,DISP=OLD
//PRINT     DD   SYSOUT=A
//SORTWK01  DD   UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02  DD   UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03  DD   UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK04  DD   UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK05  DD   UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK06  DD   UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT   DD   SYSOUT=A
//SYSPRINT  DD   SYSOUT=A
//SORTLIB   DD   DSN=SYS1.SORTLIB,DISP=SHR
//SYSIN     DD   *
            PRINT START=0900,STOP=0930
/*
```

If the report is required to be from 11:30 P.M. until 1:10 A.M. the control statement would be:

```
PRINT START=2330,STOP=2510
```

The following examples would all be identical if PI tracing had first been started on 11/25.

```
PRINT START=1000,STOP=1530,DATE=11/27
PRINT START=3400,STOP=3939,DATE=11/26
PRINT START=5800,STOP=6330
```

The report heading date is the date from the log when PI tracing was first started. The time of the calls and the start and stop times in the report are relative to this date.

SPOOL SYSOUT PRINT UTILITY PROGRAM (DFSUPRT0)

When a communication line is defined for SPOOL SYSOUT during system definition, a utility program is provided to copy messages produced by the online control program from its set of data sets to a system output device. Both the spool data sets and the system output device are processed using QSAM. Blocking factors for spool data sets are determined by the online control program. System output device blocking can be specified through JCL on the SYSPRINT DD statement.

Condition codes returned by the program are:

0	Successful completion.
4	No data sets allocated for printing.
8	SYSPRINT DD statement missing.
12	I/O error on SYSPRINT data set.

The Spool Sysout Print Utility program does not support CICS/VS.

JCL REQUIREMENTS

// EXEC
This statement may be in the form PGM=DFSUPRT0 or activate a procedure which contains the required JCL statement. A region size of 30K bytes is usually adequate for execution.

//STEPLIB DD
Defines the library containing the print utility. This is usually DSNAME=IMSVS.RESLIB,DISP=SHR.

//SYSPRINT DD
Defines the system output device to which output is directed. The record format is VBM. If either no block size or a block size less than 141 is specified, the default block size of 141 is assumed. Any block size valid for QSAM and greater than 141 can be specified. Any logical record length may also be specified. If none is specified, the default is 4 less than the block size (137 if block size default of 141 is used).

//SPOOLnn DD
This (where nn is any valid alphameric identifier) describes the spool data set to be printed. This is normally DSNAME=IMSVS.SYSnn, where 'nn' is assigned by system definition. DCB information either should not be coded or, if coded, must specify RECFM=VBM.

A sample print procedure is contained under "Member Name IMSWTnnn" in IMS/VS System Programming Reference Manual.

Output from the print utility will include a page of status information, similar to that pictured below, followed by the contents of the spool data sets indicated as FULL printed in chronological sequence.

When the print utility is started, it should complete before a /START LINE command is issued to make the SPOOL SYSOUT available.

DFSUPRT0 EXAMPLE

```
    DFSUPRTO - SYSOUT PRINT UTILITY    TIME  11:14:3   DATE    74.172

DDNAME        STATUS      CREATED TIME          DATE       DATASET NAME
SPOOL1         FULL        12:42:415           74.176      IMSVS.SYSNN
SPOOL2         FULL        19:08:537           74.176      IMSVS.SYSNN
SPOOL3         FULL        13:05:011           74.175      IMSVS.SYSNN
SPOOL4         FULL        19:57:417           74.176      IMSVS.SYSNN
```

where:

DDNAME
Is the user-provided DDNAME.

STATUS
Is FULL—if data set is to be printed.

Is INUS—if being filled online.

Is AVAL—if not being used.

CREATED TIME
Is time of creation (24-hour clock) (HH:MM:SST).

DATE
Is Julian date of creation (YY.DDD).

DATASET NAME
Is the DSNAME of the assigned data set.

System messages included in a spool data set always have unprintable control characters (typically the new-line symbol, X'15'). If a UCS printer is used as a SYSOUT device, these may print as extraneous alphabetic characters if fold-mode operation is specified in response to the UCS parameter request.

MULTIPLE SYSTEMS VERIFICATION UTILITY

The Multiple Systems Verification utility (DFSUMSV0) is provided for use with IMS/VS Multiple Systems Coupling when MTM, CTC, BSC, or VTAM is used. The Multiple Systems Verification utility cannot detect errors associated with Intersystem Communication (ISC). (Consult "Administration of Multiple Systems" in IMS/VS System Administration Guide.) The Multiple Systems Verification utility does not support CICS/VS.

This utility is used to verify the consistency and compatibility of system definitions for IMS/VS systems in a multisystem environment. The Multiple Systems Verification utility cannot verify compatibility of the log write-ahead option between systems, because the option is specified when IMS/VS is started. The Multiple Systems Verification utility cannot detect errors associated with directed routing. It should be run before attempting online executions to verify all defined links and routing paths. From 2 to 255 IMS/VS systems may be verified in any one execution of the verification utility. The use of this utility points out errors that can prevent the IMS/VS systems from performing properly; otherwise, the user must manually verify the compatibility of system definitions.

See the IMSCTRL macro in IMS/VS Installation Guide for the parameters needed to generate the multisystem control block and verification utility.

If the MSVERIFY parameter is specified with the SYSTEM keyword on the IMSCTRL macro, only the IMS/VS multisystem control block and verification utility will be generated.

Before attempting to execute this utility, the user must resolve all IMS/VS definition errors in all the multisystem control blocks to be included in the total multisystem configuration. After all system definitions are complete, the user is responsible for link-editing all the multisystem control blocks from all the IMS/VS multisystem definitions into IMSVS.RESLIB or some other user-specified library. The utility will then have access to the multisystem control blocks. For problem determination, assembly listings of all the IMS/VS multisystem control blocks should be available when the utility is executed. Without the assembly listings, it is difficult for the user to resolve inconsistencies and incompatibilities displayed as a result of the multisystem verification process.

This utility processes in two phases:

- Input validation
- Multisystem control block verification

INPUT VALIDATION

The verification utility requires as input one or more control statements within a SYSIN data set. The control statements must contain the 1- to 3-digit suffix supplied on the MSVID keyword of the IMSCTRL macro. Each control statement can contain one or more such suffixes, specified in any sequence. The input statement scan ends when a blank position is encountered; if position 1 is blank, the input statement is treated as a comment statement. If more than one suffix is specified in a control statement, each suffix following the first one must be separated from the preceding suffix by a comma. Only the significant digits of a suffix need be specified.

Each suffix in a control statement must be complete in that statement and cannot be continued in the next control statement.

After the input has been validated, this phase prints a list of the valid multisystem control block names. The utility then determines if the multisystem control block names are in the IMSVS.RESLIB PDS directory. The utility prints any control block names not found in the directory. If any errors have been detected up to this point, the utility terminates execution with a return code of 12. If no errors have been found, the utility loads the multisystem control blocks into main storage.

MULTISYSTEM CONTROL BLOCK VERIFICATION

The utility verifies the following specific portions of each multisystem control block:

- Partner IDs and assigned physical links
- Logical link paths
- Remote and local transaction attributes
- Presence of corresponding logical terminals

Logical Links and Physical Links

The partner IDs in the logical link definitions are verified to ensure that a partner ID:

- Is not referenced in only one system
- Is referenced in only two systems

Each partner ID, as defined with the PARTNER keyword on the MSLINK macro, is checked against every other partner ID in every other multisystem control block. Appropriate messages are printed if any errors are found. Logical links in error are treated as undefined in subsequent steps of the verification process.

When a partner ID is verified and there is also an MSPLINK (physical link) defined for this logical link in both systems, the physical link attributes are verified for type and buffer size. There are the following types of physical links:

- Main-storage-to-main-storage
- Channel-to-channel
- BISYNC (For BISYNC, the CONTROL=NO/YES keyword must be NO in one system and YES in the other system.)
- VTAM

If any physical link incompatibility is found, the attributes of both physical link definitions are displayed to assist the user in determining the error. If the MSPLINK is defined for only one logical link, an information message is printed, indicating that the other end is undefined.

Remote SYSID to Local SYSID Paths

The SYSID table entries for a SYSID are verified across all multisystem control blocks for that SYSID number to determine if any path errors exist. A path error is an incomplete path between a multisystem control block in which the SYSID is defined as remote and the multisystem control block in which the SYSID is defined as local.

An incomplete path can occur for the following reasons:

- A SYSID in a multisystem control block for an intermediate system does not contain an address of an MSNAME block (undefined SYSID).
- The MSNAME block is assigned to a logical link block that has a path back to itself without a local SYSID (loop condition).
- The MSNAME block is associated with a logical link block that has an invalid partner ID. The partner ID is invalid if it is not defined in any other multisystem control block or if it is defined in more than one other multisystem control block.
- A SYSID number is defined as local in more than one system.
- A SYSID number is not defined as local in any system.

SYSIDs are scanned in numeric order until all logical link paths are verified. After a path is verified, the utility may display warning messages. These messages identify which logical link numbers this SYSID number-MSNAME should not be assigned to because an invalid path to the local SYSID would result.

Remote and Local Transaction Attributes

Each multisystem control block is scanned for remote transaction definitions. Each remote transaction definition references a remote and local SYSID. Each remote transaction code is compared with the transaction codes in the system where the remote SYSID is defined as local. If no matching transaction code is found, an error message is printed. If a match is found, the attributes are verified in the two multisystem control blocks.

The following transaction code attributes must be consistent between systems:

- Local
- Remote
- Recoverable
- Nonrecoverable
- Conversational
- Fixed scratch pad area
- Fixed scratch pad area length
- Noninquiry
- Inquiry
- Single segment
- Multisegment
- Non-Fast Path

If there is a discrepancy in the transaction attributes, an error message is printed. The attributes specified for the transaction in both systems are displayed.

The return from the local transaction is checked to ensure a path exists out of the local system back to the corresponding remote transaction. If an error exists, a message is printed.

Presence of Corresponding Logical Terminals

Each multisystem control block is scanned for remote logical terminal definitions (LTERMs). Each LTERM is associated with an MSNAME block. Each MSNAME block contains remote and local SYSID definitions.

When a remote LTERM is found, the utility checks to ensure that a corresponding LTERM is defined with the same name in the system where the remote SYSID for the MSNAME is defined as local. If no corresponding LTERM definition is found, an error message is printed. If an LTERM is found, verification continues.

The utility checks the return path from the destination LTERM to ensure that a path exists out of the local system back to the corresponding remote LTERM. If an error exists, a message is printed.

The return path to that system is the SYSID defined as local in the MSNAME block in the multisystem control block in which the LTERM was defined as remote.

After all verification work is complete, the utility prints a path map as an aid in visualizing the configuration of systems.

RETURN CODES

When the verification is complete, a message is printed showing the highest return code generated during the verification process.

- Return code of 0 means only information and warning messages are printed.
- Return code of 12 means that errors were detected that must be resolved before multisystem execution.

JCL REQUIREMENTS

Before the verification utility can be executed, the multisystem control block modules for all systems to be verified must be loaded into IMSVS.RESLIB or some other user-specified library. Below is a sample job stream to accomplish this:

```
//MSADD    JOB  (JOB STATEMENT PARAMETERS)
//STEP1    EXEC PGM=IEWL,REGION=200K,
                PARM='SIZE=(200K),NCAL,LET,REUS,XREF,LIST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSVS.RESLIB,DISP=OLD
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(100,10))
//SYSLIN   DD *
  PLACE OBJECT MODULES HERE
  NAME DFSMSXXX(R)
/*
```

The user must provide two or more object modules. A NAME statement identifying the preceding multisystem control block module (replace XXX with the 3-digit control block name suffix) must follow each object deck. If a library other than IMSVS.RESLIB is used, modify the //SYSLMOD statement accordingly.

The JCL required to execute the verification utility follows:

```
//MSVERIFY JOB  (JOB STATEMENT PARAMETERS)
//STEP1    EXEC IMSMSV
//SYSIN    DD   *
     INPUT FOR IMS/VS MULTISYSTEM VERIFICATION UTILITY
/*
```

If PARM=ALL is specified in the EXEC statement for the utility execution, the information message DFS2327I is printed as part of the utility output. This message warns the user not to do a //ASSIGN of the SYSID/MSNAME to the logical link referenced, because the assignment does not provide a path to the local SYSID at this SYSID level.

UTILITY CONTROL STATEMENTS

Sample input data:

    1,255,6,009,80,02,198 are valid.

    0,256,0040,NYC,1A,0001,FF,5,5 are invalid.

Each invalid entry is printed, and the type of error identified.

For example:

```
0       Is not in the range from 1 to 255
256     Is not in the range from 1 to 255
0040    More than 3 digits
NYC     Nonnumeric
5,5     Duplicate input data
```

Valid input for three systems:

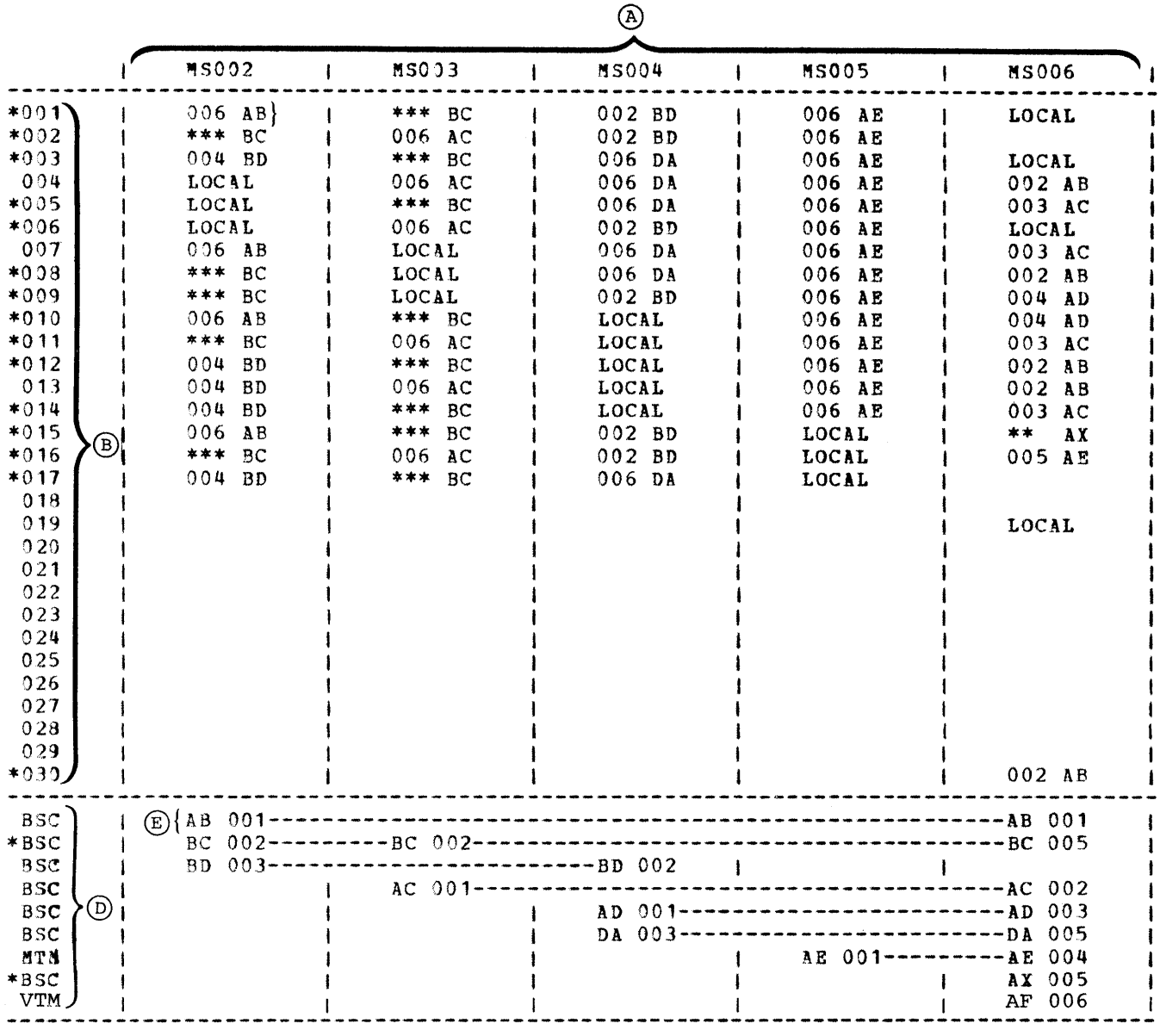
```
            Position 1

STATEMENT   1,5,255
   or
STATEMENT   001,005,255
   or
STATEMENT   255,01,5

            OR

STATEMENT   1        DFSMS001 NYC
   and
STATEMENT   255      DFSMS255 LA
   and
STATEMENT   05       DFSMS005 CHICAGO
```

OUTPUT MESSAGES AND PATH MAP

The verification utility output includes information, warning, and error messages, and a path map. The messages are defined in IMS/VS Messages and Codes Reference Manual. Because an error can cause other error conditions, messages with lower numbers should be analyzed and corrected first. The path map is a summary, in matrix format, of the routing paths verified by the utility. It is produced for the first 18 or fewer systems (in ascending sequence by multisystem control block name) being verified. If more than 18 systems are being verified, verification for all occurs, appropriate messages for all are provided, but the path map is only provided for the first 18.

Figure 97 on page 495 is a sample path map for a configuration of five systems with multisystem control block names of DFSMS002, DFSMS003, DFSMS004, DFSMS005, and DFSMS006. (For the sake of simplicity, these names will be used to refer to the specific systems in the rest of this discussion.)

A path map has two parts. The top part relates SYSIDs to specific systems. The bottom part relates partner IDs to logical links between specific systems. The contents of the path map are keyed by circled letters that are defined after Figure 97.




```
     *    = AN ERROR WAS DETECTED ON THIS LINE
     **   = NO PARTNER FOR ID
     ***  = MULTI-PARTNERS FOR ID
DFS2399I JOB TERMINATED - RETURN CODE 12
```

Figure 97. Sample Multisystem Path Map

Letter	Meaning
A	The top row contains the multisystem control block names (not including the DFS prefix) of the systems verified by execution of this utility.
B	The first column of the top part contains all SYSIDs defined in the multisystem configuration. An asterisk preceding the SYSID number indicates an error exists on this line of the matrix.
C	Each entry in the top part relates the SYSID (column B) to the above multisystem control block name (row A). • Most entries contain the 3-digit suffix of the multisystem control block name of the system defined as logically linked to the above system (row A) and the 2-character partner ID defined for this logical link. • A blank entry, such as SYSID 002 for DFSMS006, indicates this SYSID was not defined for this system. Note that all entries for SYSIDs 020 through 029 are blank; this means these SYSIDs were not specified in any of the multisystem control blocks. • An entry specifying LOCAL, such as SYSID 004 for DFSMS002, identifies this system as the system in which this SYSID is defined as local. • Errors are identified by asterisks. One asterisk preceding the SYSID indicates that one or more errors were found for this printed line. If the suffix portion of an entry is replaced by two asterisks (**), such as SYSID 015 for DFSMS006, the verification utility found no partner for this system. Three asterisks (***) indicate that more than two partners were found, such as SYSID 002 for DFSMS002. If a SYSID is defined as local for more than one system, the printed line is identified by one asterisk and more than one entry on that line specifies LOCAL. SYSID 006 contains an example of this error.
D	The first column of the bottom part contains the physical link type: BSC = binary synchronous communication line CTC = channel-to-channel adapter MTM = main-storage-to-main-storage VTAM = ACF/VTAM session type 6 This column entry is either blank, if no physical link is defined, or assigned depending on the first physical link encountered, for the logical link identified by E. An asterisk preceding the link type (or alone if no physical link is defined) indicates an error exists for this printed line.
E	Identifies the logical link in terms of partner ID and relative logical link number. The partner IDs relate directly to those in the top part of the chart. The verification utility relates partner systems by connecting them with a dashed line. In Figure 97 on page 495 , partnerships BC and AX are in error. These errors were identified in the top part of the chart but are more clearly demonstrated in the bottom part. Partner ID BC has more than two definitions; AX has just one definition.

PART 6. IMS/VS FAST PATH

Part 6 has two chapters that describe the utilities used to assist in the creation, maintenance, and recovery of IMS/VS Fast Path main storage data bases (MSDBs) and data entry data bases (DEDBs).

Note: CICS/VS does not support Fast Path data bases. These utilities cannot be used with CICS/VS.

Chapter 13, "IMS/VS Fast Path Main Storage Data Base (MSDB) Utilities," describes the offline utilities that initialize and load, maintain, and reconstruct MSDBs. The input and output data sets that the utilities use and produce are described. Control statements and examples are included.

Chapter 14, "IMS/VS Fast Path Data Entry Data Base (DEDB) Utilities," describes the two offline utilities that initialize the DEDBs and aid in the DEDB recovery process and the three online utilities that maintain and reorganize DEDBs. The input and output data sets that the utilities use and produce are described. Control statements, commands, and examples are included.

CHAPTER 13. IMS/VS FAST PATH MAIN STORAGE DATA BASE (MSDB) UTILITIES

IMS/VS Fast Path provides two offline utilities that are used to initialize, alter, and recover main storage data bases (MSDBs). These utilities are:

- MSDB Maintenance utility (DBFDBMA0)
- MSDB Dump Recovery utility (DBFDBDR0)

The design information for Fast Path in IMS/VS Data Base Administration Guide is prerequisite reading for this chapter.

INTRODUCTION TO THE MSDB UTILITIES

The user should be aware of the following items in using the Fast Path DB utilities:

- The MSDB offline utility will run in an OS/VS batch region.
- Each MSDB utility must be executed in a separate job step. The program cannot switch from one utility to another within the same job step.

The MSDB utilities use a variety of input data sets (explained later in this chapter) depending on the particular function being performed; but the primary output from all functions of both utilities is a sequential data set containing MSDBs. This data set, defined on the MSDBINIT DD statement (or MSDBOLD and MSDBNEW in the Maintenance utility), can become input for a system startup or restart process that requires the MSDBs to be loaded.

THE MSDBINIT DATA SET

All MSDBs defined to an IMS/VS system are contained in a single sequential data set defined by the MSDBINIT DD statement. The MSDBs are written in sequence based on DBD name and key. MSDBINIT contains variable length records, each of which contains one MSDB segment preceded by a prefix. The prefix contains the following data:

- Record length—2 bytes
- MSDB name (same as DBD name)—8 bytes
- MSDB type—1 byte
- Key length—1 byte
- Key—variable length

The prefix is followed by the entire content of the segment. Records for nonterminal-related MSDBs with keys embedded in the segment contain two copies of the key, one in the prefix and one in the segment data.

Figure 98 on page 499 shows the MSDBINIT record format.

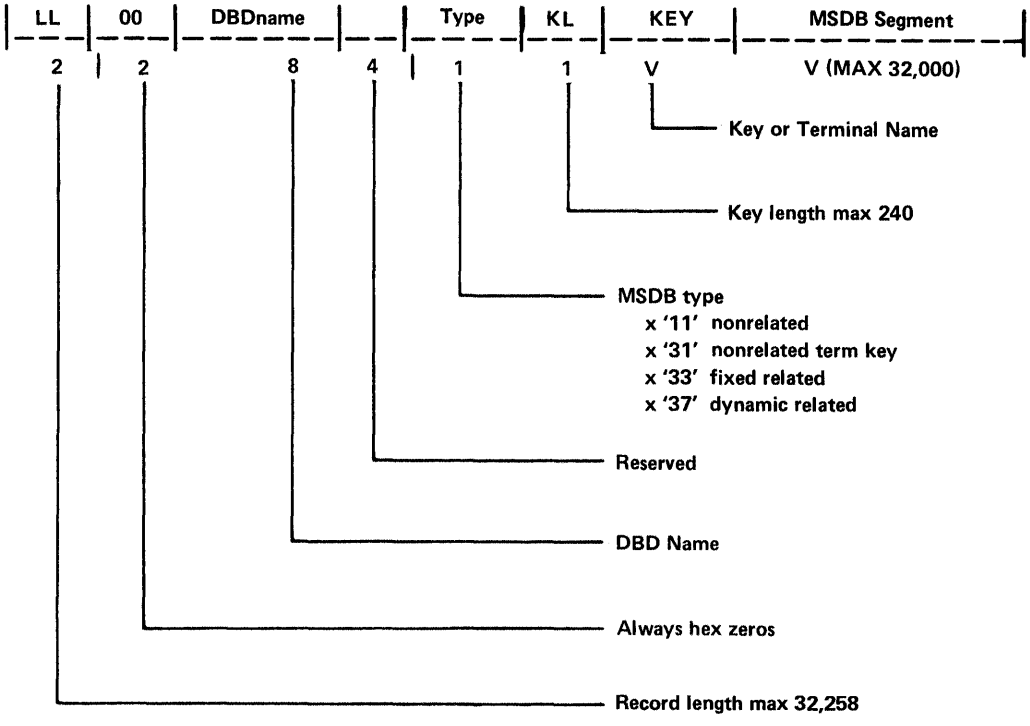


Figure 98. MSDBINIT Record Format

The MSDB Maintenance utility is used to insert, replace, delete, and modify main storage data bases in the MSDBINIT file. All of these actions may be performed (on different MSDBs) in a single run of the utility.

MSDB MAINTENANCE UTILITY (DBFDBMA0)

The MSDB Maintenance utility is an offline utility that runs in an OS/VS batch region. This utility creates an OS/VS sequential data set for MSDB initial load and is used to maintain the MSDBs.

Figure 99 on page 500 shows the input and output data sets used by the MSDB Maintenance utility for initializing and altering MSDBs. The input data sets are:

- A change data set in card format or MSDBINIT record format containing MSDB names, record keys, data, and other information as described in "MSDB Change Data Set."
- A control data set containing a RUN statement and one or more ACTION statements specifying the function to be performed. See "Control Statements."
- An old MSDBINIT data set containing formatted MSDBs produced by a previous execution of the MSDB Maintenance utility, the Dump Recovery utility, or, possibly, a user-written routine.

The MSDB Maintenance utility produces the following output data sets:

- A new MSDBINIT data set containing formatted MSDBs that may be new or altered, depending on the function just performed.
- A message data set.

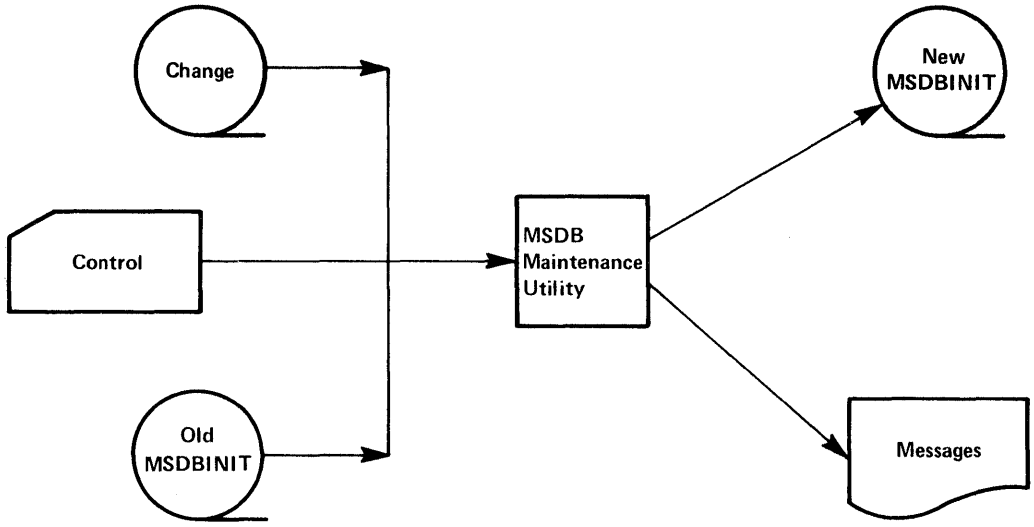


Figure 99. MSDB Maintenance Utility Input and Output Data Sets

The general action of the MSDB Maintenance utility is as follows:

1. Copy records from the old MSDBINIT file to the new one as long as the MSDB name in the old MSDBINIT record is not equal to the name in the control file.
2. When the MSDB name in the old MSDBINIT file is equal to the name in the control file, perform the action(s) requested in the control file.
3. Repeat steps 1 and 2 until the input files are exhausted.

To insert MSDBs:

- The change data set must be supplied, must be either in card format or in MSDBINIT format or both, and must contain data for all segments to be formatted into new MSDBs.
- The control data set must contain ACTION statements specifying MODE=INSERT for each MSDB to be inserted.
- The old MSDBINIT data set, if present, must not contain a record for any MSDB to be inserted.

The MSDBs are read from the change data set, formatted, and written to the new MSDBINIT data set by the utility.

To replace MSDBs:

- The change data set must be supplied and can be either in card format or in MSDBINIT format or both, must name MSDBs that exist on the old MSDBINIT data set, and must supply data for all segments.
- The control data set must contain an ACTION statement specifying MODE=REPLACE for each MSDB to be replaced to appear in the new MSDBs.
- An old MSDBINIT data set must be supplied and must contain at least one record for each MSDB to be replaced.

Except for the MSDBs specified in the change data set, all the MSDBs are copied from the old MSDBINIT and written to the new MSDBINIT. When one of the specified MSDBs is reached, it is read and formatted from the change data set, rather than copied from the old MSDBINIT.

To delete MSDBs:

- The change data set is not required.
- The control data set must contain an ACTION statement specifying MODE=DELETE for each MSDB to be deleted. The named MSDBs must exist on the old MSDBINIT.
- An old MSDBINIT data set must be supplied.

Except for the MSDBs specified in the change data set, all the MSDBs are copied from the old MSDBINIT and written to the new MSDBINIT. The specified MSDBs are merely read over and ignored.

The three functions described above are used to insert, replace, or delete entire MSDBs. The MODIFY function is used to insert, replace, delete, and alter segments within MSDBs. It can be used to alter one or more fields in specified segments or across a range of segments by key.

To modify MSDB segments:

- The change data set can be either in card format or MSDBINIT format or both and must supply the MSDB name, key field name, and, optionally, data for the segment to be modified.
- The control statement data set must contain ACTION statements specifying MODE=MODIFY for each MSDB to be modified.
- An old MSDBINIT data set must be supplied and must contain at least one record for each MSDB to be modified.

The utility compares the change record data set with the old MSDBINIT. If a change record with data is supplied for an existent key, the record is modified as specified by the change records. If a change record without data is supplied for an existent key, the record is deleted. If a change record with data is supplied for a nonexistent key, the record is inserted. Any record for which no change record is supplied is not modified.

INPUT AND OUTPUT

The MSDB Maintenance utility uses the following input:

- A data set (MSDBCTL) that contains control statements (RUN and ACTION statements)
- A change record data set that contains data in a card image (80-byte records) (MSDBCCHG) or in an MSDBINIT record format (MSDBLCHG), unless only deleting MSDBs
- A data set from a previous utility run (MSDBOLD), unless only inserting new MSDBs

The MSDB Maintenance utility produces the following output:

- An MSDBINIT data set (MSDBNEW)
- A printed summary of utility action or error statements (MSDBPRT)
- A punched deck that can be used to update the PROCLIB (MSDBPUN). The deck contains a card for each record in the DBFMSDBX member of the IMSVS.PROCLIB. The cards may be used

as data cards with the IEBUPDTE utility to update the member in the library.

If the MSDBs are so critical to the Fast Path applications that IMS should not run without them, a first card may be placed in the DBFMSDBx member containing the following characters:

```
MSDBABND=Y
```

This must be the first card of the member or an error will occur. This card will cause the IMS control region to be abended if an error occurs during the loading of the MSDBs in system initialization. This card must be added manually to the MSDBPUN data set described in the output of this utility.

JCL REQUIREMENTS

EXEC
Executes the MSDB Maintenance utility. This statement can either be in the form PGM=DBFDBMA0 or specify a procedure that contains the required JCL. The utility requires at least 512K bytes of virtual storage.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

MSDBACB DD
Defines the IMS/VS ACB library. This data set must reside on a direct access device.

MSDBOLD DD
Describes the sequential MSDBINIT data set that contains MSDBs from a previous utility run.

MSDBNEW DD
Describes the sequential data set that contains the new MSDBs after the current utility run.

MSDBCTL DD
Describes a data set that contains control statements.

MSDBCCHG DD
Describes a data set that contains change records in card format. This data set must be sequenced by DBD name and key

MSDBLCHG DD
Describes a data set that contains change records in MSDBINIT record format. This data set must be sequenced by DBD name and key.

MSDBPRT DD
Describes a message data set for printed output.

MSDBPUN DD
Describes an output data set that will contain change statements in IEBUPDTE format for adding or replacing member DBFMSDBX in IMSVS.PROCLIB.

CONTROL STATEMENTS

The MSDB Maintenance utility requires two types of control statements called the RUN statement and the ACTION statement coded in columns 1 through 71. Continuations are not permitted. Comments may be included as illustrated in the examples in "MSDB Maintenance Utility Examples" later in this chapter.

RUN Statement

One RUN statement is used for the entire utility execution. The format of the run statement is:

```
PROC=suffix
```

where:

PROC=
Specifies the 1-byte suffix for the MSDB start procedure (DBFMSDBx).

Action Statement

One action statement is required for each MSDB to be included in the utility execution. Any MSDB in the input MSDBINIT data set that is not referred to in an action statement is written to the new MSDBINIT data set with no change. Action statement keywords can be separated by commas or blanks. The format of the Action statement is:

```
DBN=dbname,MODE={INSERT }{[,INCR=number] },FIXED={Y}
               {REPLACE}{                }      {N}
               {DELETE }
               {MODIFY }
```

where:

DBN=
Specifies the same MSDB name that is specified in the DBDGEN.

MODE=
Designates the action against the specified MSDB:

INSERT
Specifies that a new MSDB is to be built from MSDB change records and placed in MSDBNEW. If INSERT is specified, all change records for the MSDB to be inserted must contain segment data, and the old MSDBINIT file (MSDBOLD), if present, must contain no records with the same MSDB name.

REPLACE
Specifies that all records of this MSDB are to be removed and new records from the change record data set are inserted. If REPLACE is specified, all change records for the affected MSDB must contain segment data.

DELETE
Specifies that the MSDB is to be deleted. If DELETE is specified, a change record cannot appear for the MSDB to be deleted.

MODIFY
Specifies that individual records in the MSDB are to be modified.

The utility compares the change record data set with the old MSDBINIT data set. If a change record with data is supplied for an existent key, the record is modified as specified by the change records. If a change record without data is supplied for an existent key, the record is deleted. If a change record with data is supplied for a nonexistent key, the record is inserted. Any record for which no change record is supplied is not modified. A range of records cannot be inserted using the TO= keyword.

INCR=
Specifies the decimal number of empty segments to be reserved in a dynamic MSDB for future inserts. Change records are not required for a dynamic MSDB. This number should not exceed the number of logical terminals defined.

FIXED=
Specifies if this MSDB is to be page-fixed (Y) or not page-fixed (N). Y is the default.

Notes:

1. A record (key) cannot appear more than once for the same MSDB in the change data set.
2. Dynamic MSDBs can be empty, but other types of MSDBs must have at least one record.

MSDB CHANGE DATA SET

The MSDB change data set contains keywords and operands that specify the changes to be applied to an MSDB. The data set contains variable-length records (segments) that are inserted, replace the old records, or modify individual records within an MSDB. The data set does not contain change records for the DELETE operation. Change records can be supplied in either a MSDBINIT record format data set or in a card format data set or both. Records must be in sequence by MSDB and, within each MSDB, by key.

MSDBINIT Record Format

This format is the same as described in Figure 99 on page 500, except that the field defined as an MSDB segment is optional for the MODIFY operation. For the MODIFY operation:

- Supply a record with an existent key in the KEY field and the desired segment content in the MSDB segment field to replace a segment.
- Supply a record with a nonexistent key in the KEY field and the desired segment content in the MSDB segment field to insert a segment.
- Supply a record with an existent key in the KEY field and no data in the MSDB segment field to delete a segment.

Card Format

Each statement may contain data in columns 1 through 71 of 80-byte records.

Note: Comments may be used by placing an asterisk in column 1 of each comment statement. Remarks may appear in the same statement as keywords as shown in the following examples. The equal sign (=) is a reserved symbol for keywords and may not be used in the remarks.

Each statement may contain the following keywords separated by commas or blanks.

```
DBN=name,KEY={value1}[,TO={value1}]
              {value2}    {value2}

[,FIELD=fieldname][,DATA={value1}]
                         {value2}
                         {value3}
                         {NULL  }
```

where:

DBN=
Specifies the MSDB name. The name must be the same as the name in the DBDGEN.

KEY=
Specifies the key of the record to be acted upon. See the description of value1 and value2 under the DATA keyword.

TO=
Is an optional keyword that enables the manipulation of a range of segment keys within a single statement. The key that TO= refers to must have a higher value than the key that KEY= specifies. TO= is valid only if the action statement specifies MODIFY. See the description of value1 and value2 under the DATA keyword.

FIELD=
Is an optional keyword that specifies which fields are to be modified. FIELD= is valid only for INSERT, REPLACE, and MODIFY. Each FIELD= keyword must be followed by its associated DATA= keyword.

DATA=
Specifies the contents of a field or segment. If a FIELD= keyword precedes DATA=, it specifies the contents of a field. If FIELD= does not precede DATA=, it specifies the entire contents of a segment.

value1
Is a character field within quotation marks. This field can be continued by breaking the field at column 71, inserting a nonblank character in column 72, and continuing the field in column 16 of the next statement.

value2
Is the hexadecimal representation of the character field within quotation marks and is preceded by an 'x'. This field can be continued by breaking the field at column 71, inserting a nonblank character in column 72, and continuing the field in column 16 of the next statement.

value3
Is an arithmetic field supplied as a signed or unsigned decimal number within parentheses instead of quotation marks. The utility converts this number to the format matching the field type. Leading zeros may be omitted.

NULL
Specifies that the entire segment (except for the sequence field) or the specified field is set to a null value.

Character and hexadecimal fields are set to blanks (X'40'). Arithmetic fields are set to a zero value.

If a field is given more than one definition, the arithmetic definition prevails.

RETURN CODES

The following return codes are provided:

Code	Meaning
0	Utility executed successfully.
4	Error—error message printed.
8	Unable to open print data set.

MSDB MAINTENANCE UTILITY EXAMPLES

The following examples show the JCL to execute the MSDB Maintenance utility. The control data set and the change record data set are used as input.

Example 1 creates two new MSDBs.

Example 1

```
//STEP1 EXEC PGM=DBFDBMA0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605)
//MSDBCTL DD *
  PROC=M
  DBN=MSDBLM01 MODE=INSERT FIXED=YES
  DBN=MSDBLM06 MODE=INSERT INCR=4 FIXED=NO
/*
//MSDBACB DD DSN=IMSVS.ACBLIB,DISP=SHR
//MSDBNEW DD DSN=XXXX,DISP=(NEW,CATLG),VOL=SER=IMSDCL,
//           SPACE=(CYL,1),UNIT=SYSDA,
//           DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026)
//MSDBPUN DD SYSOUT=B
//MSDBCCHG DD *
   DBN=MSDBLM01
*                  CREATE TWO NEW MSDBS
*           BUILD NONTERMINAL-RELATED MSDB LM01
       KEY='0001'                                  CHARACTER KEY
               FIELD=FIELDC01 DATA='RP'            CHARACTER DATA
               FIELD=FIELDH01 DATA=X'9999'         HALFWORD DATA (HEX)
               FIELD=FIELDF01 DATA=X'9FFFFFFFF'    FULLWORD DATA (HEX)
               FIELD=FIELDP01 DATA=(1)             DECIMAL DATA (ARITH)
               FIELD=FIELDP02
                              DATA=X'9C'               HEX
               FIELD=FIELDX03 DATA=NULL            NULL FIELD
       KEY='0002'
               FIELD=FIELDC01 DATA='RP'
               FIELD=FIELDH01 DATA=(-1)            HALFWORD TO ARITH -1
               FIELD=FIELDF01 DATA=(-1)            FULLWORD TO ARITH -1
               FIELD=FIELDP01 DATA=(1)             DECIMAL TO ARITH  +1
*                             LET FIELDP02 DEFAULT TO ZERO
*                             LET FIELDP03 DEFAULT TO ZERO
*                             LET FIELDX03 DEFAULT TO BLANKS
*          BUILD DYNAMIC TERMINAL RELATED MSDB LM06 WITH ONE UNUSED SEGMENT
   DBN=MSDBLM06
       KEY='LTERM01 '
               FIELD=FIELDSEQ DATA='0001'
               FIELD=FIELDH01 DATA=(0)
               FIELD=FIELDX03 DATA=NULL
       KEY='LTERM02 '
               FIELD=FIELDSEQ DATA='0002'
               FIELD=FIELDH01 DATA=(0)
               FIELD=FIELDX03 DATA=NULL
       KEY='LTERM03 '
               FIELD=FIELDSEQ DATA='0003'
               FIELD=FIELDH01 DATA=(0)
               FIELD=FIELDX03 DATA=NULL
/*
```

Example 2 deletes one MSDB, modifies a second as described in the comments, and replaces a third. All unmentioned MSDBs and unmentioned segments in MSDBLM04 are copied from MSDBOLD to MSDBNEW.

Example 2

```
//STEP2  EXEC PGM=DBFDBMA0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605)
//MSDBCTL  DD *
  DBN=MSDBLM03 MODE=DEL
  DBN=MSDBLM04 MODE=MOD
  DBN=MSDBLM05 MODE=REP
  PROC=Y
//MSDBACB DD DSN=IMSVS.ACBLIB,DISP=SHR
//MSDBNEW DD DSN=XXXX,DISP=(NEW,CATLG),VOL=SER=IMSDCL,
//           DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//           SPACE=(CYL,1),UNIT=SYSDA
//MSDBOLD DD DSN=IMSVS.MSDBLM01(0),DISP=SHR
//MSDBPUN DD SYSOUT=B
//MSDBCCHG DD *
   DBN=MSDBLM04
*              MODIFY THE MSDB BY DELETING ONE SEGMENT, CHANGING
*              THE TRANSACTION LIMIT IN THE DESIGNATED SEGMENTS,
*              AND COPYING THE REMAINDER OF THE FIELDS FROM THE
*              OLD FILE.
       KEY='LTERM01 '                      DELETE THIS SEGMENT
       KEY='LTERM02 '                      MODIFY 1 FIELD IN THIS SEGMENT.
              FIELD=FIELDP03 DATA=(1500)   SET MAX TRANSACTION LIMIT
       KEY='LTERM03 ',TO='LTERM09 '        MODIFY 1 FIELD IN EACH SEGMENT
*                                          WITHIN THE RANGE OF KEYS
              FIELD=FIELDP03 DATA=(750)    SET MAX TRANSACTION LIMIT
   DBN=MSDBLM05
*              INSERT 2 SEGMENTS, 1 FIELD IN EACH SEGMENT INITIALIZED
*              ALL OTHER FIELDS ARE SET TO NULL VALUES COPIED
       KEY='LTERM03 '
              FIELD=FIELDP01 DATA=(-2)     SET TIME ZONE FACTOR
       KEY='LTERM05 '
              FIELD=FIELDP01 DATA=(+1)     SET TIME ZONE FACTOR
/*
```

Example 3 merges two MSDB files. The MSDBs on MSDBLCHG are merged with the MSDBs on MSDBOLD.

Example 3

```
//*            MERGE TWO MSDB FILES TO COMBINE ALL MSDBS INTO
//*            ONE INITIAL LOAD FILE.
//STEP3  EXEC PGM=DBFDBMA0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605)
//MSDBLCHG DD DSN=MSDBUT32,DISP=SHR
//MSDBOLD  DD DSN=MSDBUT31,DISP=SHR
//MSDBCTL  DD *
  DBN=MSDBLM04 MODE=INSERT          FROM CHANGE FILE
  DBN=MSDBLM05 MODE=INSERT          FROM CHANGE FILE
//MSDBACB  DD DSN=IMSVS.ACBLIB,DISP=SHR
//MSDBNEW  DD DSN=MSDBUT33,DISP=(NEW,CATLG),VOL=SER=IMSDCL,
//            DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//            SPACE=(CYL,1),UNIT=SYSDA
//MSDBPUN  DD SYSOUT=B
/*
```

MSDB DUMP RECOVERY UTILITY (DBFDBDR0)

The MSDB Dump Recovery utility is an offline utility that runs in an OS/VS batch region. This utility uses either the dump data set (MSDBDUMP) or the checkpoint data sets (MSDBCP1 and MSDBCP2) to create a new copy of MSDBINIT.

The format of the MSDB control record, the first record on the MSDB dump or checkpoint data set, was changed between release 2 (1.2.0) and release 3 (1.3.0). For this reason, the level of DBFDBDR0 used to create the MSDB dump or checkpoint data set must be the same.

If the system terminates abnormally and emergency restart is unable to recover the MSDBs, the Dump Recovery utility is used in RECOVERY mode to reconstruct the MSDBs. The MSDBs are reconstructed using the checkpoint data sets and the associated system log. From this input the utility produces a new copy of the MSDBINIT data set. Whenever a new MSDBINIT is needed, containing one or more selected MSDBs, the Dump Recovery utility is used in UNLOAD mode with the MSDBDUMP data set as input.

To unload MSDBs:

- The MSDBDUMP data must be supplied.
- The IMS/VS log data set is not used.
- The control data set must specify UNLOAD.

The MSDBs are unloaded onto the MSDBINIT data set at the same level of change as they were dumped.

To reconstruct MSDBs:

- The MSDB checkpoint data sets (MSDBCP1 and MSDBCP2) must be supplied. The utility will determine which is the older valid image copy and will recover from that point.
- The IMS/VS log data set containing MSDB changes logged since the older checkpoint data set was created.
- The control data set should specify RECOVERY.
- If both the checkpoint data set DD statements specify the same data set, the utility will recover from the checkpoint in that data set.
- Because the format of the MSDBDUMP data set is identical to that of the checkpoint data sets, the user can force recovery from the MSDBDUMP data set by specifying that data set in both checkpoint DD statements.

The MSDBs are reconstructed on the MSDBINIT data set by applying all the logged changes to the image copy data set.

INPUT AND OUTPUT

Figure 100 on page 509 shows the input and output data sets used by the MSDB Dump Recovery utility for unloading and reconstructing MSDBs. The input data sets are:

- An MSDB dump data set (MSDBDUMP) or the MSDB checkpoint data sets (MSDBCP1 and MSDBCP2). The dump data set is created by the /DBDUMP command. A checkpoint data set is written automatically at each checkpoint: the output data set alternates on successive checkpoints between MSDBCP1 and MSDBCP2.
- An IMS/VS system log data set (optional).
- A control data set (optional). See "Control Statements."

The primary output is a new MSDBINIT data set. A message data set is also produced.

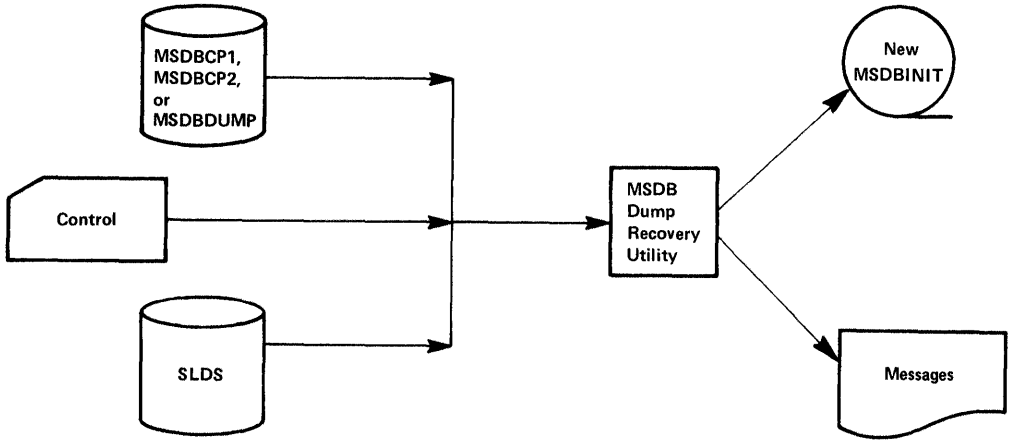


Figure 100. MSDB Dump Recovery Utility Input and Output Data Sets

JCL REQUIREMENTS

EXEC
Executes the MSDB Dump Recovery utility. This statement can either be in the form PGM=DBFDBDR0 or specify a procedure that contains the required JCL. The utility requires at least 512K bytes of virtual storage.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

MSDBDUMP DD
Describes a data set that contains a dump of all or selected MSDBs.

MSDBCP1 DD
Describes a checkpoint data set that contains an image copy of all MSDBs.

MSDBCP2 DD
Describes a checkpoint data set that contains an image copy of all MSDBs.

IEFRDER DD
Describes the data set that contains the old IMS/VS system log, which is used for recovery operations.

This data set may reside on multiple volumes; the volumes containing the checkpoint for the older of the two image copies and all later volumes must be mounted and specified in the DD statement.

MSDBCTL DD
Optional. Describes a file containing control statements in 80-byte records.

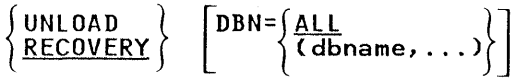
MSDBPRT DD
Describes the message data set. It can reside on tape, direct access, or printer, or be routed through the output stream.

MSDBINIT DD
Describes the data set that will contain the unloaded or reconstructed MSDBs after the utility executes.

CONTROL STATEMENTS

The control statements for the Dump Recovery utility specify the input and whether the utility unloads or reconstructs the MSDBs. These control statements are read from the MSDBCTL data set.

The control statements are free form. The control information is contained in columns 1 through 71. The list of MSDB names can be continued by inserting a comma after the last name of the statement, inserting a nonblank character in column 72, and continuing the list in column 16 of the next statement (columns 1 through 15 must be blank). Comments may be included as illustrated in the examples following this section.



where:

UNLOAD
Specifies that the MSDB dump data set defined by the MSDBDUMP DD statement is to be unloaded onto an OS/VS sequential data set (MSDBINIT). No updates from the log will be applied.

RECOVERY
Specifies that the MSDBs are to be reconstructed from the older MSDB checkpoint data set (MSDBCP1 or MSDBCP2) and the associated IMS/VS system log.

Note: To force a recovery from a particular checkpoint data set, both MSDBCP1 and MSDBCP2 DD statements must specify the same data set.

DBN=ALL
Specifies that all MSDBs in the data set are to be unloaded or reconstructed, according to the operation specified.

DBN=(dbname,...)
Specifies which MSDBs in the data set are to be unloaded or reconstructed, according to the operation specified.

If DBN= is not supplied, DBN=ALL is assumed.

If no control statements are supplied, defaults are RECOVERY and DBN=ALL.

RETURN CODES

The following return codes are provided:

Code	Meaning
0	Utility executed successfully.
4	Error—error message printed.
8	Unable to open print data set.
12	I/O error in print routine.

MSDB DUMP RECOVERY UTILITY EXAMPLES

The following examples show the JCL and utility control statements needed to execute the MSDB Dump Recovery utility.

Example 1 unloads all MSDBs from the MSDBDUMP data set onto the MSDBINIT data set.

Example 1

```
//UN101 EXEC PGM=DBFDBDR0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A
//*                 MESSAGE PRINT FILE
//MSDBINIT DD DSN=IMSVS.MSDBLM02,DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//            VOL=SER=IMSDCL,SPACE=(CYL,1),
//            DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026)
//MSDBDUMP DD DSN=IMSVS.LMDMP,DISP=SHR     MSDB DUMP
//MSDBCTL DD *
*                               UNLOAD ALL MSDB'S
  UNLOAD  DBN=ALL
/*
```

Example 2 reconstructs the MSDBs from the older checkpoint data sets and the IMS/VS system log.

Example 2

```
//RC101 EXEC PGM=DBFDBDR0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A
//*                 MESSAGE PRINT FILE
//MSDBINIT DD DSN=IMSVS.MSDBLM03,DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//            VOL=SER=IMSDCL,SPACE=(CYL,1),
//            DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026)
//MSDBCP1 DD DSN=IMSVS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2 DD DSN=IMSVS.LMCP2,DISP=SHR      CHECKPOINT # 2
//IEFRDER DD DSN=IMSVS.LMLOG,DISP=SHR      IMS OLDS/SLDS
//MSDBCTL DD *
*                               RECOVERY ALL MSDB'S
   RECOVERY DBN=ALL
/*
```

Example 3 unloads one particular MSDB from the MSDBDUMP data set onto the MSDBINIT data set.

Example 3

```
//UN201 EXEC PGM=DBFDBDR0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A
//*                 MESSAGE PRINT FILE
//MSDBINIT DD DSN=IMSVS.MSDBLM02,DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//       VOL=SER=IMSDCL,SPACE=(CYL,1),
//       DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026)
//MSDBDUMP DD DSN=IMSVS.LMDMP,DISP=SHR     MSDB DUMP
//MSDBCTL DD *
*                               UNLOAD ONLY MSDB 05
  UNLOAD  DBN=(MSDBLM05)
/*
```

Example 4 reconstructs one particular MSDB from the older checkpoint data set and the IMS/VS system log.

Example 4

```
//RC202 EXEC PGM=DBFDBDR0
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//MSDBPRT  DD SYSOUT=A
//*                MESSAGE PRINT FILE
//MSDBINIT DD DSN=IMSVS.MSDBLM03,DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//             VOL=SER=IMSDCL,SPACE=(CYL,1),
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026)
//MSDBCP1 DD DSN=IMSVS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2 DD DSN=IMSVS.LMCP2,DISP=SHR      CHECKPOINT # 2
//IEFRDER DD DSN=IMSVS.LMLOG,DISP=SHR      IMS OLDS
//MSDBCTL DD *
*                            RECOVERY ONLY MSDB 05
  RECOVERY DBN=(MSDBLM05)
/*
```

CHAPTER 14. IMS/VS FAST PATH DATA ENTRY DATA BASE (DEDB) UTILITIES

IMS/VS Fast Path provides one offline utility and five online utilities that are used to initialize, maintain, and reorganize data entry data bases (DEDBs). These utilities are:

- DEDB Initialization (offline)
- DEDB Area Data Set Create (online)
- DEDB Area Data Set Compare (online)
- DEDB Sequential Dependent Scan (online)
- DEDB Sequential Dependent Delete (online)
- DEDB Direct Reorganization (online)

"Choosing A Data Base Type" in IMS/VS Data Base Administration Guide is a prerequisite to this chapter.

USER CONSIDERATIONS

The user should be aware of the following items in using the Fast Path DEDB utilities.

- The restart capability is only supported for the DEDB Direct Reorganization utility. If the restart (REST) parameter is used for the DEDB Scan and Delete utilities, the utilities will attempt an initial run from the start of the area defined by the restart parameter. If the restart (REST) parameter is used for the DEDB Area Data Set Create and Compare utilities, it will be ignored.
- Each DEDB utility must be executed in a separate job step. The program cannot switch from one utility or one data base to another within the same job step.
- The IMS/VS batch DB Recovery, batch DB Image Copy, DB Change Accumulation utilities and/or the system log(s) can be used for recovery purposes.

DEDB INITIALIZATION UTILITY (DBFUMIN0)

The DEDB Initialization utility is an offline utility that is executed in an OS/VS batch region. This utility initializes one or more data sets of one or more areas of a DEDB. The DEDB must have been previously allocated using IDCAMS. Because only one DEDB can be specified as a member in an ACBLIB DD statement, only one DEDB can be initialized at a time. When you want to initialize the multiple area data sets of an area, the area must be registered in the DBRC RECON data set and these area data sets must be in an unavailable status in the RECON data set.

After the data sets have been initialized and the DEDB areas have been formatted to the DBDGEN specifications, a user-written program issues INSERT calls to load the data.

The IMS/VS batch DB Image Copy utility should be run. (The Online DB Image Copy utility does not support DEDBs.) There is no facility for recovering the DEDBs until the batch Image Copy utility has been executed. If a system failure occurs before the batch Image Copy utility is run, the data set must be redefined, and the Initialization utility must be rerun.

INPUT AND OUTPUT

The DEDB Initialization utility uses the following input:

- The ACB generated for the specific DEDB (access to the DBD member of ACBLIB is required).
- DBRC RECON data set if area is registered
  - Area is in recovery-needed status
  - Area data set is in unavailable status
- A control data set that specifies which areas are to be initialized.

The utility produces the following output:

- A data set that contains output messages and statistics.
- Formatted areas.
- DBRC RECON data set if area is registered
  - Area is not in recovery-needed status
  - Area data set is in available status

JCL REQUIREMENTS

EXEC
Executes the DEDB Initialization utility and must be in this format:

```
PGM=DBFUMIN0
```

The DBNAME from the DMCB is used as the password for the DEDB areas.

The EXEC statement for the DBFUMIN0 utility allows you to specify either the 'DBRC=N' parameter or no parameter at all, when the batch subsystem has no DBRC interface. Under these circumstances the RECONn DD statements are not required. But, if the SYSGEN IMSCTRL macro parameter has DBRC=FORCE,988 then an error message (DFS0044I DBRC REQUIRED FOR THIS EXECUTION) will be displayed.

When 'DBRC=Y' is specified in the EXEC statement, the RECONn DD statements are required.

When the DBRC parameter in the EXEC statement is not specified, DBRC is set according to the DBRC parameter in the IMSCTRL macro. The default for the batch subsystem in the IMSCTRL macro is NO.

STEPCAT DD
Describes a private VSAM user catalog that is searched first.

A STEPCAT DD statement must be included if the defined areas are cataloged in a user catalog.

STEPLIB DD
Points to IMSVS.RESLIB, which contains the IMS/VS nucleus and required action modules.

ACBLIB DD
Describes the DBD member of the ACBLIB that contains information on the data bases to be initialized. This DD statement must be included.

RECON1 DD
Defines the first DBRC (Data Base Recovery Control) RECON data set.

RECON2 DD
Defines the second DBRC RECON data set. This RECON data set must be included if the area is registered in the DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

These RECON data set ddnames should not be used if you are using dynamic allocation.

SYSPRINT DD
Describes a data set that contains error messages if errors occur during processing and statistics if the utility runs to successful completion.

ddname DD (for DEDBs)
areaname DD
Describes a data set for each area that is to be initialized. If an area is not registered in the RECON data set, the ddname of this statement must be the same as the area name. A DD statement for each area to be initialized is required. If an area is registered in the RECON data set, the ddname of this statement must be the same as the ADS name found in the ADS list of the RECON data set and the area of the ADS must be set as recovery-needed (all ADSs of this area are in unavailable status). A DD statement for each ADS to be initialized is required.

All data sets must be previously defined in the VSAM catalog.

DISP=OLD, UNIT, and VOL parameters must be specified if the DD statement describes a multivolume data set.

CONTROL DD
Describes the input control statement data set. This data set must have a logical record length of 80 bytes and have fixed-length blocks. See "Control Statements" below for an explanation of the required format of this data set.

CONTROL STATEMENTS

The control statements for the DEDB Initialization utility specify the name(s) of the area(s) to be initialized, or that the entire DEDB is to be initialized. These statements must reside in the data set defined by the CONTROL DD statement.

The control statements must be 80-byte records in the following format:

1	14
{AREA=areaname / ALL}	[comments]

where:

AREA=
Specifies the name of an area to be initialized. Area names must be one to eight alphameric characters (A-Z, 0-9, #, @, $). Only one area name can be specified for each control statement and must begin in column 1.

ALL
Specifies that all areas in the DEDB are to be initialized. ALL must be specified starting in column 1 of the first record in the CONTROL data set. If ALL is specified, the remainder of this control statement and all following control statements are ignored.

RETURN CODES

The following return codes are provided:

Code	Meaning
0	Requested initialization was successful.
4	Error in parameters.
8	ACB or DBRC processing error.
12	Error processing data set information.
16	Error during format processing.
20	SYSPRINT error.

Note: See IMS/VS Messages and Codes Reference Manual for explanations of the messages accompanying all nonzero return codes.

DEDB INITIALIZATION UTILITY EXAMPLES

Example 1 shows sample JCL to initialize a DEDB.

DEDB Utility Example 1

```
//INITDB9 JOB
//*
//*       INITIALIZE DEDB DATA BASE       (DATABO1)
//*
//JOBLIB DD DSN=IMSVS.RESLIB,DISP=SHR
//JOBCAT   DD  DSN=DCLCATLG,DISP=SHR
//TESTIT EXEC PGM=DBFUMIN0,REGION=800K,TIME=60
//ACBLIB DD DSN=IMSVS.ACBLIB(DATABO1),DISP=SHR
//*                            DBD FOR DATA BASE
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//RECON3   DD DSN=RECON3,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DB01AR01 DD DSN=DB01AR01,DISP=OLD,VOL=SER=VOL111
//DB01AR02 DD DSN=DB01AR02,DISP=OLD,VOL=SER=VOL111
//DB01AR03 DD DSN=DB01AR03,DISP=OLD,VOL=SER=VOL222
//DB01AR04 DD DSN=DB01AR04,DISP=OLD,VOL=SER=VOL222
//DB01AR05 DD DSN=DB01AR05,DISP=OLD,VOL=SER=VOL333
//DB01AR06 DD DSN=DB01AR06,DISP=OLD,VOL=SER=VOL333
//*                             DATA BASE TO BE INITIALIZED
//SYSUDUMP DD SYSOUT=A
//CONTROL  DD  *
ALL     ALL AREAS INITIALIZED
/*      END OF DBFUMIN0
```

Example 2 shows sample JCL to initialize three areas of a DEDB.

DEDB Utility Example 2

```
//INIT3A  JOB
//*
//*       INITIALIZE DEDB DATA BASE       (DATAB02)
//*
//JOBLIB   DD DSN=IMSVS.RESLIB,DISP=SHR
//JOBCAT   DD DSN=DCLCATLG,DISP=SHR
//TESTIT EXEC PGM=DBFUMIN0,REGION=800K,TIME=60
//ACBLIB   DD DSN=IMSVS.ACBLIB(DATAB02),DISP=SHR
//RECON1   DD DSN=RECON1,DISP=SHR
//RECON2   DD DSN=RECON2,DISP=SHR
//RECON3   DD DSN=RECON3,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DB9AR3   DD DSN=DB9AR3,DISP=OLD
//DB9AR20  DD DSN=DB9AR20,DISP=OLD,VOL=SER=(VOL222,VOL333),UNIT=SYSDA
//DB9AR157 DD DSN=DB9AR157,DISP=OLD
//CONTROL  DD *
AREA=DB9AR3
AREA=DB9AR20
AREA=DB9AR157
/*   END OF DBFUMIN0
```

DEDB ONLINE UTILITIES

The DEDB online utilities run in the DB Utility Type Dependent Region. This region is used exclusively for IMS/VS Fast Path online utilities and is not available to the Fast Path user.

The DEDB online utilities (create, compare, scan, delete, and reorganization) can run concurrently with Fast Path online applications potentially accessing the same area within a DEDB, but cannot concurrently access the same control interval. As soon as a conflict occurs, the latest requestor waits unless a potential deadlock situation occurs. In this case, there is an algorithm to determine which requestor is terminated. A DEDB online utility cannot run concurrently with another Fast Path utility against the same area.

Three commands provide information essential to the basic operation of the online utilities:

- TYPE indicates the specific utility to be executed.
- ERRORACTION describes the procedure for handling parameter errors.
- GO specifies how much parameter information to scan before executing the utility.

These commands, as well as the other DEDB online utility commands, are described later in the chapter.

The online utilities all invoke the following procedure:

```
//FPUTIL PROC DBD=,SOUT=A,RGN=500K,OPT=N,DIRCA=000,
//            REST=00,STIMER=1,TLIM=1,PSB=DBF#FPU0,OBA=00,PRLD=
//IFP   EXEC  PGM=DFSRRC00,REGION=&RGN,
//      PARM=(IFP,&DBD,&PSB,&REST,&OBA,&OPT,&TLIM,&DIRCA,&PRLD,&STIMER)
//STEPLIB  DD DSN=IMSVS.RESLIB,DISP=SHR
//PROCLIB  DD DSN=IMSVS.PROCLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT
//*        UTILITY OUTPUT MESSAGES ARE DIRECTED TO SYSPRINT          */
//*        BY THE FAST PATH SCAN UTILITY                             */
```

DEDB AREA DATA SET CREATE UTILITY (DBFUMRI0)

The DEDB Area Data Set Create utility is an online utility that creates one or more copies from multiple DEDB area data sets during online transaction processing without stopping the area. The DEDB Initialization utility is not required. During the copying process, application programs may continue. Two or more data sets with defective CI(s) can be used by this utility to produce a copy free of defective CI(s). WRITEs requested from application programs are performed to both the AVAILABLE and the new data sets. Prior to starting this utility, the new data sets must be defined to DBRC using the DBRC INIT.ADS command. The new area data set (ADS) must be unavailable in the DBRC RECON data set.

The data set is defined by the VSAM definition. The size of the control interval (CI) must be identical to those defined for the other data sets of the same area. The allocated space must be equal to or greater than those defined for the other data sets of the same area.

If a read error is detected in a CI in an available data set, another data set in the same area will be used. If all the available data sets have read errors in the same CI, this utility terminates. If a write error is detected in the new data set, this utility terminates.

This utility cannot be restarted but must be rerun from the beginning. This utility can be stopped before processing has completed by using the /STOP REGION command.

INPUT AND OUTPUT

The DEDB Area Data Set Create utility uses the following input:

- A data set that contains the input parameters supplied by commands
- DBRC RECON data set

The DEDB Area Data Set Create utility produces the following output:

- One or more copies of AVAILABLE data sets
- A data set that contains output messages and statistics

JCL REQUIREMENTS

EXEC
Executes the DEDB Data Set Create utility. This statement can either be in the form PGM=DFSRRC00 or specify a procedure that contains the required JCL.

STEPLIB DD
Describes the library that contains the DEDB Data Set Create utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. Required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement. See "Summary of DEDB Online Utility Commands" later in this chapter for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

DEDB AREA DATA SET CREATE UTILITY EXAMPLE

These examples show the JCL and utility control statements for executing the DEDB Area Data Set Create utility.

Example 1 (creating 1 new area data set)

```
//CREATE1  JOB  MSGLEVEL=(1,1)
//UTL1     EXEC FPUTIL,DBD=DEDBJN23
//*
//*        DEDB=JN23 IS THE TARGET DATA BASE
//*FOR THE DEDB AREA DATA SET CREATE UTILITY
//*
//SYSIN DD
TYPE CREATE
AREA DB23AR3
DDNAME DB23AR33
GO
/*
```

Example 2 (creating 2 new area data sets)

```
//CREATE2  JOB  MSGLEVEL=(1,1)
//UTL2     EXEC FPUTIL,DBD=DEDBJN23
//*
//*        DEDBJN23 IS THE TARGET DATA BASE
//*        FOR THE DEDB AREA DATA SET CREATE UTILITY
//*
//SYSIN DD
TYPE CREATE
AREA DB23AR3
DDNAME DB23AR33
DDNAME DB23AR34
GO
/*
```

Example 3 (creating 5 new are data sets)

```
//CREATE4  JOB  MSGLEVEL=(1,1)
//UTL4     EXEC FPUTIL,DBD=DEDBJN23
//*
//*        DEDBJN23 IS THE TARGET DATA BASE
//*        FOR THE DEDB AREA DATA SET CREATE UTILITY
//*
//SYSIN DD
TYPE CREATE
AREA DB23AR3
DDNAME DB23AR33
DDNAME DB23AR34
DDNAME DB23AR35
DDNAME DB23AR36
DDNAME DB23AR37
GO
/*
```

DEDB AREA DATA SET COMPARE UTILITY (DBFUMMH0)

The DEDB Area Data Set Compare utility is an online DEDB utility that compares the physical records (CIs) of two or more area data sets (ADSs) of an area. The utility compares the CIs of:

- Root addressable part and independent overflow part

- Reorganization unit of work (UOW)
- Sequential part if the sequential dependent segment is defined in the AREA statement at DBDGEN time

This utility does not compare:

- The control CIs (first and second CI of an area)
- All CIs in a residual part when the space defined at DBDGEN is smaller than that defined by the VSAM definition

In case of unequal comparison, full dumps of up to 10 unmatched records are printed out onto the media specified by the SYSPRINT DD statement and comparison processing continues until the end of the area data sets.

At the end of the comparing process, a message is printed on a SYSPRINT data set that tells the number of unmatched CIs and the number of identical CIs. This utility also checks the error queue element (EQE) in each specified area data sets and prints the EQE status of an area on the SYSPRINT data set.

The comparison of each CI is done only when the to-be-compared CI count is equal to or greater than 2. The to-be-compared CI count is decreased by 1 each time an I/O error or an EQE is detected.

The DEDB Area Data Set Compare utility terminates if:

- The to-be-compared area data set or area is stopped by a /STOP AREA or /STOP ADS command
- The to-be-compared area data set or area is stopped by an internal stop command
- The to-be-compared area data set count is decreased to 1

This utility can be stopped immediately by a /STOP REGION command.

INPUT AND OUTPUT

The DEDB Area Data Set Compare utility uses the following input:

- A data set that contains the input parameters supplied by commands

The DEDB Area Data Set Compare utility produces the following output:

- A printed dump of up to 10 unmatched records on the SYSPRINT data set
- A message showing the number of unmatching CIs and the number of identical CIs, also printed out on the SYSPRINT data set
- An error queue element (EQE) status report on the SYSPRINT data set

JCL REQUIREMENTS

EXEC
Executes the DEDB Data Set Compare utility. This statement can either be in the form PGM=DFSRRC00 or specify a procedure that contains the required JCL.

STEPLIB DD
Describes the library that contains the DEDB Data Set Compare utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. Required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement. See "Summary of DEDB Online Utility Commands" later in this chapter for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

DEDB AREA DATA SET COMPARE UTILITY EXAMPLE

These examples show the JCL and utility control statements for executing the DEDB Area Data Set Compare utility.

Example 1 (comparing 2 area data sets)

```
//UTL101  EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATA BASE FOR
//*       THE DEDB AREA DATA SET COMPARE UTILITY
//*
//SYSIN   DD
TYPE COMPARE
AREA DB23AR2
DDNAME DB23AR21
DDNAME DB23AR22
GO
/*
```

Example 2 (comparing 7 area data sets)

```
//UTL102  EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATA BASE FOR
//*       THE DEDB AREA DATA SET COMPARE UTILITY
//*
//SYSIN   DD
TYPE COMPARE
AREA DB23AR2
DDNAME DB23AR21
DDNAME DB23AR22
DDNAME DB23AR23
DDNAME DB23AR24
DDNAME DB23AR25
DDNAME DB23AR26
DDNAME DB23AR27
GO
/*
```

Example 3 (comparing all available area data sets of an area)

```
//UTL103  EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATA BASE FOR
//*       THE DEDB AREA DATA SET COMPARE UTILITY
//*
//SYSIN   DD
TYPE COMPARE
AREA DB23AR2
GO
/*
```

The following is an example of the error queue element (EQE) status report:

```
AREA EQE STATUS

         RBA      ADS01    ADS02    ADS03    ADS04
 1. 00000800        *
 2. 00001400*                           *
 3. 00001800                 *
 4. 00001C00        *                   *
 5. 00002000        *                             *
 6. 00002400                                      *
 7. 00002800        *        *         *          *
 8. 00002C00                                      *
 9. 00003000                                      *
10. 00003400                                      *

 EQE COUNT          5        2         2          7

 NO DATA AVAILABLE FOR CI STARTING AT 00000800
```

The following is a sample printed dump of the unmatched records:

```
UNMATCHED CI AT     AREA=AREA01    RBA=00000800

ADS02    0000  F0F1F2F3.................C1C2C3C4  *0123......ABCD*
ADS03    0000  F0F1F2F3.................A1C2C3C4  *0123.......BCD*
ADS04    0000  F0F1F2F3.................C1C2C3C4  *0123......ABCD*

ADS02    0020  F4F5F6F7.................C5C6C7C8  *4567......EFGH*
ADS03    0020  00000000.................C5000000  *.........E...*
ADS04    0020  F4F5F6F7.................C5C6C7C8  *4567......EFGH*

ADS02    03C0  F4F5F6F7.................C5C6C7C8  *4567......EFGH*
ADS03    03C0  00000000.................C5000000  *.........E...*
ADS04    03C0  F4F5F6F7.................C5C6C7C8  *4567......EFGH*

ALL      03E0  D1D2D3D4.................E2E3E4E5  *JKLM......STUV*

DFS3753I  COMPARISON NOT PERFORMED FOR ADS=ADS01
          REASON: NO DATA AVAILABLE DUE TO EQE
```

DEDB SEQUENTIAL DEPENDENT SCAN UTILITY (DBFUMSC0)

The DEDB Sequential Dependent Scan utility is an online utility that runs in the DB Utility Type Dependent Region. This utility scans and copies dependent segments to a sequential data set for offline processing by user-written programs. The scan utility can be stopped at any time at the end of an area by using the IMS/VS /STOP REGION command. The utility can be used to maintain data in historical order by time, to do a statistical analysis of the data, or to obtain information to prepare for a DEDB reorganization.

A user exit routine must be specified if certain segments within a range (STARTRBA and STOPRBA) are to be copied. An exit routine can change the segment content and length in place. The length cannot exceed the block size of the SCANCOPY DD data set minus 8 bytes or the utility terminates with an error message that indicates the utility work area is unusable. If an exit routine is not specified, the scan utility takes the default and the segment contents pass through the specified range unchanged. If a range limit is not specified, all dependent segments are scanned and copied. See "DC User Exit and Edit Routines" in IMS/VS System Programming Reference Manual for information on how to write a user exit routine. This chapter also describes the DBFUMSE0 module that is supplied as the default exit routine.

RANGE LIMITS

There are three ways to specify the range of sequential dependent segments to process. The starting location (STARTRBA) can be specified using the STARTRBA, STARTROOT, or STARTSEQ command. The stopping location can be specified using the STOPRBA, STOPROOT, or STOPSEQ command. If a range is not specified, the scan starts with the oldest current sequential dependent segment in the area and ends with the newest one.

STARTRBA is used to specify the 4-byte RBA of the segment or the 8-byte combination of cycle number and RBA. The cycle number is returned by the POS call and is the number that DEDB uses as a prefix to the RBA to get a value that is used only once within the life of the area.

If the cycle number is specified as a nonzero value, the scan utility checks to see that it matches the current cycle number for the RBA in question. If the cycle number is not the same as the current cycle number for that RBA or the RBA is not in the range in which sequential dependents are currently being stored, an error message is printed and no data is scanned. If the cycle number is not specified or is specified as zero, the current cycle number for that RBA is used. If the start RBA and the stop RBA are specified without cycle numbers, the stop RBA could actually be lower than the start RBA because of the wraparound from the highest RBA back to the lowest RBA. However, the cycle number for the stop RBA would be higher. If an RBA is specified that is not the address of the beginning of a segment, such as an address inside of a segment, the scan starts with the next segment.

The other two ways to specify the start of the scan assume that marker segments are used. A marker segment is a special dependent segment that has a unique field value. Marker segments are maintained in the sequential dependent segments by running an application program that inserts them as sequential dependents.

The root segment that has an inserted marker segment must be specified to the utility. This is done by specifying the key of the root segment on the STARTROOT command. If the sequential dependent is not the most recently inserted sequential dependent for the root, information must be specified for the utility to use to build an SSA to search for the desired dependent. An SSA is built by specifying a field name, a comparison operator, and

a field value on the STARTSEQ command. The comparison rules are the same as for a normal DEDB SSA. The field value specified must be the same as for a normal DEDB SSA. The field value specified must be the same length as the field in the DEDB. If the root does not exist in the indicated area or the indicated segment is not found, an error message is printed and no data is scanned.

An example of the use of marker segments is:

> In a user installation that does not operate on a 24-hour basis, a user may want to process each day's transactions. The user selects particular root segments to which special sequential dependent segments are to be added. The dependent segments have a field value that contains the day's date.
>
> The user runs an application program at the end of each day that inserts the dependent segments. These dependent segments are marker segments that mark the end of each day's processing and can be used to control the scan range. The utility scans on the unique field value (date) of the marker segments.

The three ways to specify the stop RBA are exactly the same as the ways to specify the start RBA, except that STOPRBA, STOPROOT, and STOPSEQ commands are substituted for STARTRBA, STARTROOT, and STARTSEQ. The last segment scanned is the last segment that begins at or before the stop RBA. If the stop RBA is the same as the start RBA, then no more than one segment is scanned. If the stop RBA refers to an earlier segment than the start RBA, an error message is printed and no data is scanned.

Start RBA and stop RBA do not have to be specified by the same methods.

INPUT AND OUTPUT

The DEDB Sequential Dependent Scan utility uses the following input:

- A data set that contains the input parameters supplied by commands

The DEDB Sequential Dependent Scan utility produces the following output:

- A data set that contains a copy of dependent segments (A user exit may be employed to limit this data set to specific segments.)
- A data set that contains output messages and statistics

JCL REQUIREMENTS

EXEC
Executes the DEDB Sequential Dependent Scan utility. This statement can either be in the form PGM=DFSRRC00 or specify a procedure that contains the required JCL.

STEPLIB DD
Describes the library that contains the DEDB Sequential Dependent Scan utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. Required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement. See "Summary of DEDB Online Utility Commands" later in this chapter for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

SCANCOPY DD
Describes the scan output data set that contains dependent segments in variable-length, blocked records. This data set cannot be the SYSOUT data set.

Note that one SCANCOPY data set is produced, and it contains output from a scan of either one area or multiple areas.

DEDB SCAN UTILITY EXAMPLE

This example shows the JCL and utility control statements for executing the DEDB Scan utility.

```
//SCAN2 EXEC  FPUTIL,RGN=500K,
//       DBD=DEDBJN03,REST=00
//*       DBD=DBDNAME AS TARGET DATA BASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//SCANCOPY DD DSN=SCAN203,VOL=SER=IMSDCL,DISP=(NEW,PASS,DELETE),
//            SPACE=(TRK,5),UNIT=SYSDA,
//            DCB=(NCP=5,BLKSIZE=2048)
//SYSIN    DD *
*-------------------------------------------------------------*
*          USE THE STOPROOT WITH STOPSEQ TO LIMIT THE         *
*          RANGE OF THE SCAN UP TO THE LAST DAY ACCUMULATION  *
*          OF SEGMENTS.                                       *
*_____________________________________________________________*
*             SET ERROR OPTION
 ERROR SCANRUN
*             ONLINE SEQ DEP SCAN UTILITY
 T S
*             THE TARGET DATA BASE IS
*   DBDNAME DEDBJN03
*             THE TARGET DEDB AREA IS
    AREA DB3AREA0
*             THE BUFFER ARE NOT TO BE FIXED
 FIXOPT NO
*             THE NO. OF BUFFERS IS
 STOPROOT=C'R301102A'
*             STOP ON SEQ DEP SEGMENT
 STOPSEQ FIELD=SDFLDDAT,OP='<=',VALUE=C'273'
*                                          JULIAN DATE
             THE EXIT PROGRAM NAME IS
 EXIT EXITLDM3
```

DEDB SEQUENTIAL DEPENDENT DELETE UTILITY (DBFUMDL0)

The DEDB Sequential Dependent Delete utility is an online utility that runs in the DB utility type dependent region. This utility logically deletes sequential dependents of an area of a DEDB within a specified limit. After the dependent segments are deleted, the utility resets the segment boundaries. After successful completion of a utility run, the freed space in the area is available for reuse.

RBA LIMITS

There are three ways to specify the limit of sequential dependent segments to delete. The STOPRBA, STOPROOT, and STOPSEQ commands can be used to specify a stop RBA. If a dependent segment limit (stop RBA) is not specified, the delete utility deletes all the current dependent segments in the area starting with the oldest segment.

STOPRBA is used to stop at the 4-byte RBA of the segment or the 8-byte combination of cycle number and RBA. The cycle number is the number that the DEDB uses as a prefix to the RBA to get a value that is used only once within the life of the area.

If the cycle number is specified as a nonzero value, the delete utility checks to see that it matches the current cycle number for the RBA in question.

If the cycle number is not the same as the current cycle number for that RBA or the RBA is not in the range in which sequential dependents are currently being stored, an error message is printed and no data is deleted. If the cycle number is not specified or is specified as zero, the current cycle number for that RBA is used. If an RBA is specified that is not the address of the beginning of a segment, such as an address inside of a segment, the deletion starts with the next segment.

The other two ways to specify the stop RBA assume that marker segments are used. A marker segment is a special dependent segment that has a unique field value. Marker segments are maintained in the sequential dependent segments by running an application program that inserts them as sequential dependents.

The root segment that has an inserted marker segment must be specified to the utility. This is done by specifying the key of the root segment on the STOPROOT command. If the sequential dependent is not the most recently inserted sequential dependent for the root, information must be specified for the utility to use to build an SSA to search for the desired dependent. An SSA is built specifying a field name, a comparison operator, and a field value on the STOPSEQ command. The comparison rules are the same as for a normal DEDB SSA. The field value specified must be the same length as the field in the DEDB. If the root does not exist in the indicated area or the indicated segment is not found, an error message is printed and no data is deleted.

INPUT AND OUTPUT

The DEDB Sequential Dependent Delete utility uses the following input:

- A data set that contains the input parameters supplied by commands

The DEDB Sequential Dependent Delete utility produces the following output:

- An area with freed space
- A data set that contains output messages and statistics

JCL REQUIREMENTS

EXEC
Executes the DEDB Sequential Dependent Delete utility. This statement can either be in the form PGM=DFSRRC00 or specify a procedure that contains the required JCL.

STEPLIB DD
Describes the library that contains the DEDB Sequential Dependent Delete utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. Required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement. See "Summary of DEDB Online Utility Commands" in this chapter for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains messages and statistics.

DEDB SCAN AND DELETE UTILITIES EXAMPLES

This example shows the JCL and another set of utility control statements for executing the DEDB Scan and Delete utilities.

```
//SCAN  EXEC  FPUTIL,RGN=500K,
//       DBD=DEDBJN02,REST=00
//*       DBD=DBDNAME AS TARGET DATA BASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//SCANCOPY DD DSN=SCAN202,VOL=SER=IMSDCL,DISP=(NEW,PASS,DELETE),
//            SPACE=(TRK,5),UNIT=SYSDA,
//            DCB=(NCP=5,BLKSIZE=2048)
//SYSIN   DD *
*_____________________________________________________________________*
*          THE STOPRBA  COMMAND WILL LIMIT THE RANGE OF THE           *
*          SCAN.                                                      *
*_____________________________________________________________________*
*             SET ERROR OPTION
 ERRORACTION SCANRUN
*             ONLINE SEQ DEP SCAN UTILITY
  TYPE SCAN
*             THE TARGET DATA BASE IS
* DBDNAME DEDBJN02
*             THE TARGET DEDB AREA IS
 AREA DB2AREA1
*             THE BUFFER AREA TO BE FIXED
   FIXOPT YES
*                      STOP ON RBA
 STOPRBA  X'29E98' LAST SEQ DEP RBA (102A03-2)
*             THE EXIT PROGRAM NAME IS
 EXIT EXITLDM3
//DELETE EXEC  FPUTIL,RGN=500K,
//       DBD=DEDBJN02,REST=00
//*       DBD=DBDNAME AS TARGET DATA BASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//SYSIN   DD *
*_____________________________________________________________________*
*            DELETE  DEDB JN02 USING THE 'STOPROOT' TO LIMIT          *
*            THE SCOPE OF THE DELETE.                                 *
*_____________________________________________________________________*
*             SET ERROR OPTION
 ERRORACTION       SCANRUN
             ONLINE SEQ DEP DELETE UTILITY
  TYPE  DELETE
*             THE TARGET DATA BASE IS
* DBDNAME                                                  DEDBJN02
*             THE TARGET DEDB AREA IS
 AREA DB2AREA1
*             STOP ON ROOT SEGMENT KEY
 STOPROOT=C'R211304D'
*                        LAST ROOT WITH A SEQ DEP SEGMENT
```

DEDB DIRECT REORGANIZATION UTILITY (DBFUMDR0)

The DEDB Direct Reorganization utility is an online utility that runs in the DB dependent type region. The utility removes external storage fragmentation and sequences the root and direct dependent segments in the control intervals (CIs).

Control intervals in the root portion of each area are grouped into units of work (UOW). The Reorganization utility reorganizes one UOW at a time. There are two phases within the reorganization of one UOW:

1. Build phase (rebuilds the UOW)
2. Copy phase (copies the rebuilt UOW into the original location and frees all independent overflow control intervals that are no longer used by the UOW)

The Reorganization utility uses the standard resource management of IMS/VS Fast Path. A control interval that is being reorganized is held exclusively by the Reorganization utility and therefore cannot be accessed by any other program during these two phases.

BUILD AND COPY PHASES

The Reorganization utility uses one "reorganization UOW" in each area as temporary storage for reorganization. During the build phase, the utility copies one UOW's root segments and their direct dependent segments in logical order to the reorganization UOW and updates any affected chain pointers in the reorganization UOW. If the reorganization UOW cannot hold all the segments of the chains, the utility allocates independent overflow space and fills it with the remaining segments. If the overflow is insufficient, the utility will terminate. An attempt is made to put all dependent segments into the same control interval as the root. The utility does not change the UOW being reorganized until it has been completely rebuilt in the reorganization UOW, at which time the utility sets the "copy phase reached" indicator (in the DMAC).

During the copy phase, the utility writes the reorganization UOW control intervals back into their corresponding locations within the UOW being reorganized. The "copy phase reached" indicator is reset after the rebuilt UOW has been successfully copied into the proper UOW location in the DEDB. Any independent overflow CIs that are no longer needed are then freed.

I/O ERROR HANDLING AND AREA STATUS

The following summarizes the I/O error handling by the Reorganization utility and the area status after an I/O error.

Utility	Action	Area Status
BUILD Phase		
Read error	Terminates	Usable
Write error	Terminates	Usable
COPY PHASE		
Read error	Terminates	Stopped (recovery-needed)
Write error	Continues	Usable

When the reorganization utility finds an I/O error while in BUILD phase, the utility terminates and the target area is usable. When the reorganization utility finds a read error while in COPY phase, the utility stops the area, puts the area in a recovery-needed status in the DBRC RECON data set, and terminates. When the reorganization utility finds a write error while in COPY phase, the utility continues the reorganization process and the target area is usable.

Conversion of DEDBs

The DEDB Direct Reorganization utility can be used to convert IMS/VS Version 1 Release 2 Fast Path DEDBs to IMS/VS Version 1 Release 3 Fast Path DEDBs. To use this utility, specify the TYPE, AREA, and GO control statements. STARTUOW or STOPUOW control statements should not be specified.

UOW RANGE LIMITS

The STARTUOW command enables the user to reorganize a part of the root addressable portion of the area by specifying at which UOW to start reorganizing. The STOPUOW command specifies the last UOW to reorganize. The first UOW in the area is number 0, the second one is number 1, the third one is number 2, etc. If STARTUOW is not specified, the utility starts reorganizing with the first UOW in the area and continues to the specified STOPUOW. If STOPUOW is not specified, the utility starts at the specified start UOW and processes until the end of the area. If both the start UOW and the stop UOW are specified, the start UOW must have a lower value than the stop UOW. If an invalid value for either UOW is specified, an error message is printed and no data is reorganized. The UOW number can be entered in either decimal or hexadecimal format.

RECOVERY

The IMS/VS Change Accumulation utility and the system log(s) can be used for recovery purposes. Every data change is logged on the system log, and data integrity is maintained for every log record.

RESTART

If the system or utility terminates during the build phase, reorganization can be restarted at the beginning of the build phase on the UOW which was being processed. However, if the system terminates during the copy phase, the copy phase is completed during the next open process starting with the UOW which was processing.

If data space is not freed because of system termination, the next utility run on the area is considered a restart, and the space is reclaimed.

The /STOP REGION command causes an orderly termination of the utility. A message indicating which UOW was last reorganized is written to the SYSPRINT data set. The operator then knows where to restart the utility (the next UOW). Integrity is maintained throughout the utility execution and restart procedure.

For a description of the 'REST=' parameter, refer to the IMS/VS System Programming Reference Manual under 'FPUTIL'.

INPUT AND OUTPUT

The DEDB Direct Reorganization utility uses the following input:

- A data set that contains input parameters supplied by the commands

The DEDB Direct Reorganization utility produces the following output:

- An area that contains compressed and logically sequenced UOWs
- A data set that contains output messages

JCL REQUIREMENTS

EXEC
Executes the DEDB Direct Reorganization utility. This statement can either be in the form PGM=DFSRRC00 or specify a procedure that contains the required JCL.

STEPLIB DD
Describes the library that contains the DEDB Direct Reorganization utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. This statement is required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS/VS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statements. See "Control Statement" later in this chapter for a complete description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains messages.

RETURN CODES

The following return codes are provided by the online utilities:

Code	Meaning
0	Utility executed as requested
4	Sysprint error or, if issued by DBFUMDR0, processing stopped at user request
8	Error in parameter analysis or, if issued by DBFUMDR0, processing stopped because of lack of resources.
12	Conflict on AREA
16	Open or I/O error on DBFUMSC0 O/P data set
20	I/O error on AREA
24	Processing stopped because of lack of resources
40	Unable to allocate sufficient space from either the reorganization or independent overflow areas

Note: Error messages accompany all nonzero return codes. If a return code of 20 or 24 is returned from DBFUMDR0, the utility must be restarted to maintain data base integrity.

DEDB DIRECT REORGANIZATION UTILITY EXAMPLE

This example shows the JCL and utility control statements for executing the DEDB Direct Reorganization utility.

```
//ORGDB01 EXEC  FPUTIL,RGN=500K,
//        DBD=DEDBJN01,REST=00
//*       DBD=DBDNAME AS TARGET DATA BASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//SYSIN   DD *
*********************************************************************************
*               ONLINE DEDB ROOT REORGANIZATION UTILITY
*********************************************************************************
 TYPE REORG
*                 SET ERROR OPTION
 ERROR HALT
*               THE TARGET DATA BASE IS
*DBDNAME DEDBJN01
*               THE TARGET AREA IS
 AREA DB1AREA1
 GO
*               THE TARGET DATA BASE IS
*DBDNAME DEDBJN01
*               THE TARGET AREA IS
 AREA DB1AREA2
 GO
```

UTILITY CONTROL STATEMENTS

The DEDB online utilities use QSAM to read the SYSIN data set. The input can be blocked or unblocked, fixed length or variable length. The records are interpreted as lines of input statements or commands, and the characters are in EBCDIC. The records can be 80 characters long, but can be as long as 120 characters. The records can be shorter than 80 characters if the necessary data fits onto the input line.

SUMMARY OF DEDB ONLINE UTILITY COMMANDS

Figure 101 is a summary of the DEDB online utility commands and operands and their synonyms. More detailed information on commands follows.

COMMAND	OPERATOR	SCAN UTILITY	DELETE UTILITY	REORG UTILITY	CREATE UTILITY	COMPARE UTILITY
AREA		R	R	R	R	R
BUFNO		O	O	O	N/A	N/A
DDNAME		N/A	N/A	N/A	R	O
ERRORACTION		O	O	O	N/A	N/A
	STOP	O	O	O	N/A	N/A
	SCAN	O	O	O	N/A	N/A
	SCANRUN	O	O	O	N/A	N/A
EXIT		O	N/A	N/A	N/A	N/A
FIXOPT		O	O	O	N/A	N/A
	NO	O	O	O	N/A	N/A
	YES	O	O	O	N/A	N/A
GO		O	O	O	O	O
STARTRBA		O	N/A	N/A	N/A	N/A
STARTROOT		O	N/A	N/A	N/A	N/A
STARTSEQ		O	N/A	N/A	N/A	N/A
	FIELD	O	N/A	N/A	N/A	N/A
	OP	O	N/A	N/A	N/A	N/A
	VALUE	O	N/A	N/A	N/A	N/A
STARTUOW		N/A	N/A	O	N/A	N/A
STOPRBA		O	O	N/A	N/A	N/A
STOPROOT		O	O	N/A	N/A	N/A
STOPSEQ		O	O	N/A	N/A	N/A
	FIELD	O	O	N/A	N/A	N/A
	OP	O	O	N/A	N/A	N/A
	VALUE	O	O	N/A	N/A	N/A
STOPUOW		N/A	N/A	O	N/A	N/A
TYPE		R	R	R	R	R
	COMPARE	N/A	N/A	N/A	N/A	R
	CREATE	N/A	N/A	N/A	R	N/A
	DLET	N/A	R	N/A	N/A	N/A
	REORG	N/A	N/A	R	N/A	N/A
	SCAN	R	N/A	N/A	N/A	N/A

Figure 101. Summary of DEDB Online Utility Commands

KEY
O=Optional
R=Required
N/A=Not applicable

COMMAND FORMAT

Parameters are specified in free-form. Except for the 120-character maximum, fields are not restricted to any particular columns.

Each statement must begin on a new line. The command name begins with the first nonblank character and is ended by a blank or an equal sign. An asterisk as the first character indicates a comment.

If the command requires operands, the operand field begins with the next character that is either nonblank or not an equal sign. If the command allows multiple operands, the operands are separated by commas. The operand field is ended by a blank or by the end of the line. However, characters that are part of an EBCDIC value (specified as a character string within quotation marks) do not count as ending commas or as ending blanks.

Characters following the operand field on a line are treated as comments.

COMMAND CONTINUATION

The operand field can be continued between operands by using a dangling comma as follows:

```
STARTSEQ  OP='FIELD=FLD1,       (first line)
          VALUE=X'C4C5C2'       (continuation line)
```

A quoted string can be continued by using a closing quotation mark, a comma, and a reopening quotation mark as follows:

```
STARTROOT X'C1C2C3C4C',         (first line)
          '5C6C7',              (second line)
          'C8C9'                (lastline)
```

Comments can be included on each line with a blank between the dangling comma and the comments.

COMMAND DESCRIPTIONS

The input parameters to the DEDB online utilities are supplied by the following commands.

TYPE
Specifies either the scan, delete, or reorganization utility as the type of run. This command is required and must be included before the first GO command or before the end of the SYSIN file. The TYPE command should be specified only once within a job step because the program cannot switch from one utility to another within the same job step.

ERRORACTION
Specifies the action the specified utility should take after detecting an error and printing an error message. This command is optional and can be specified as many times as desired. If ERRORACTION is not included, the STOP operand is the default.

STOP
Specifies that the utility stop immediately.

SCAN
Specifies that the utility continue scanning the input for errors.

SCANRUN
Specifies the same processing as the SCAN operand, except that the utility ignores a detected error after the next GO command is encountered.

GO
Is used to separate a series of requests. It must be specified to process more than one area, to use more than one exit routine, or to process more than one range of data within a single job step.

AREA
Identifies the area to be processed by the name that is on the control region DD statement. This command must be repeated after each GO or RUN command.

The name must be 1 to 8 characters. The first character must be alphabetic, and the rest of the characters must be alphabetic or numeric. The alphabetic characters are A-Z, @, #, and $. The numeric characters are 0-9.

DDNAME
Specifies the area data set to be created or compared.

For the create utility, the area data set specified on a DDNAME statement must be in an unavailable status in the DBRC RECON data set and the maximum allowable number of the DDNAME statements is 6.

For the compare utility, the area data set specified on a DDNAME statement must be in an available status and the maximum allowable number of the DDNAME statements is 7.

EXIT
Identifies the user exit routine by load module name. This command is optional and is used only with the scan utility.

The name must be 1 to 8 characters. The first character must be alphabetic, and the rest of the characters must be alphabetic or numeric. The alphabetic characters are A-Z, @, #, and $. The numeric characters are 0-9.

BUFNO
Specifies the number of buffers to use to read and/or write the DEDB.

For the utilities other than the Reorganization utility, a minimum of 7 buffers is required. If BUFNO is not specified, the default is taken. The default number of buffers is the number of CIs per UOW plus 7.

For the Reorganization utility, a minimum number of buffers is calculated as follows:

```
Minimum number of buffers = number of
  hierarchic levels of DEDB + 2 + 10
```

If BUFNO is not specified, the default is taken. The default number of buffers is the number of CIs per UOW plus the number of hierarchic levels of the DEDB + 2 + 10

FIXOPT
Specifies whether (yes) or not (no) page-fixed buffers are used. Page-fixed buffers are used for better performance. If FIXOPT is not specified, YES is the default.

STARTRBA
Specifies up to 8 bytes (16 digits) of sequential dependent address information. The low-order 4 bytes specify the relative byte address within the area to start processing sequential dependents. The high-order 4 bytes are optional and are used to supply a cycle number. STARTRBA is an optional command and is used with the scan utility.

A hexadecimal value is a value of the form X'hex digits', where the valid hexadecimal digits are 0-9 and A-F.

STARTROOT
Specifies the root key field value for the root used to find the start RBA. The STARTROOT command is optional and is used with the scan utility.

The value must be a hexadecimal value, a character value, or a packed value. The value begins with a letter (X, C, or P) and continues with a quoted string, such as X'2A1B", C'AIN''T', or P'-00199'. Values are evaluated for both length and content. For example, X'00' (1 byte) is not the same as X'0000' (2 bytes).

STARTSEQ
Specifies the sequential dependent segment used to find the start RBA. The STARTSEQ command is optional and is used with the scan utility.

FIELD=(name2)
Is a field name as is specified in an SSA.

The name must be 1 to 8 characters. The characters must be alphabetic or numeric. The alphabetic characters are A-Z, @, #, and $. The numeric characters are 0-9.

OP=(operator)
Is a comparison operator as is specified in an SSA.

VALUE=(value)
Is a field value as is specified in an SSA.

The FIELD, OP, and VALUE fields are used, together with the STARTROOT data, to set up a POS call to find the start RBA.

STOPRBA
Specifies the stop RBA. The rules are the same as described for specifying the starting RBA using the STARTRBA command. This optional command is used with either the scan or delete utilities.

The hexadecimal value is a value of the form X'hex digits', where the valid hexadecimal digits are 0-9 and A-F.

STOPROOT (value)
Specifies the root key field value. This optional command is used with either the scan or delete utilities.

The value must be a hexadecimal value, character value, or a packed value. The value begins with a letter (X, C, or P) and continues with a quoted string, such as X'2A1B', C'AIN''T', or P'-00199'. Values are evaluated for both length and content.

STOPSEQ
Specifies the sequential dependent segment used to find the stop RBA. This optional command is used with either the scan or delete utilities.

FIELD=(name2)
Is a field name as is specified in an SSA.

The name must be 1 to 8 characters. The characters must be alphabetic or numeric. The alphabetic characters are A-Z, @, #, and $. The numeric characters are 0-9.

OP=(operator)
Is a comparison operator as is specified in an SSA.

VALUE=(value)
Is a field value as is specified in an SSA.

The FIELD, OP, and VALUE fields are used, together with the STOPROOT data, to find the STOP RBA in the same way STARTSEQ data is used to find the START RBA.

STARTUOW
Specifies the first unit of work (UOW) to reorganize. This command is optional and is used with the reorganization utility.

STOPUOW
Specifies the last unit of work (UOW) to reorganize. This command is optional and is used with the reorganization utility.

The DEDB online utilities recognize alternative spellings that can be used as abbreviations or synonyms for the commands and keywords. Figure 102 shows these abbreviations and synonyms.

Commands	Abbreviations/Synonyms
AREA	A,AREANAME
BUFNO	BUFFERNUMBER,BUFFNO, BUFN
DDNAME	DD,DDN
ERRORACTION	ERR,ERROR
EXIT	USEREXIT
FIXOPT	FIX,FIXOPTION
GO	RUN
STARTRBA	F,FA,FIRST,FIRSTA,FIRSTRBA,FRBA,STARTA
STARTROOT	FIRSTR,FIRSTROOT,FR,FROOT,STARTR
STARTSEQ	FIRSTS,FIRSTSEQ,FS,FSEQ,STARTS
STARTUOW	FIRSTUOW,FIRSTW,FUOW,FW,STARTW
STOPRBA	K,KEEP,KEEPA,KEEPRBA,KRBA,L,LAST, LASTA,LASTRBA,LRBA,STOPA
STOPROOT	KEEPR,KEEPROOT,KR,KROOT,LASTR, LASTROOT,LR,LROOT,STOPR
STOPSEQ	KEEPS,KEEPSEQ,KS,KSEQ,LASTS, LASTSEQ,LS,LSEQ,STOPS
STOPUOW	LASTUOW,LASTW,LUOW,LW,STOPW
TYPE	T

Keywords	Abbreviations/Synonyms
COMPARE	CM,COM,COMP
CREATE	C,CR
DLET	D,DELETE,DL
FIELD	F
NO	N,OFF
OP	O,OPERATOR
REORG	DR,R
SCAN	SKIP (as ERRORACTION operand)
SCAN	S,SC (as TYPE operand)
SCANRUN	SKIPRUN
STOP	END,HALT,QUIT
VALUE	V,VAL
YES	ON,Y

Figure 102. DEDB Online Utility Command and Keyword Abbreviations and Synonyms

INDEX

A

B

C

D

G

H

L

M

N

O

P

R

S

T

X

IMS/VS Version 1
Utilities Reference Manual
SH20-9029-9

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Please use pressure sensitive or other gummed tape to seal this form.

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

SH20-9029-9

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | Please do not staple | Fold and tape

IBM®

SH20-9029-9

IMS/VS Version 1 Utilities Reference Manual (File No. S370-50) Printed in U.S.A. SH20-9029-9

IBM®

SH20-9029-09