SH20-9029-3

Program Product

IMS/VS Version 1
Utilities Reference Manual

Program Number 5740-XX2

Release 1.2

IBM

Fourth Edition (May 1976)

This edition replaces the previous edition (numbered SH20-9029-2) and its technical newsletter (numbered SN20-9118) and makes them obsolete.

This edition applies to Version 1 Release 1.2 of IMS/VS, program number 5740-XX2, and to all subsequent releases unless otherwise indicated in new editions or technical newsletters.

Technical changes are summarized under "Summary of Amendments" following the list of figures. Each technical change is marked by a vertical line to the left of the change.

Information on the Interactive Query Facility has been moved to the *IMS/VS System Programming Reference Manual;* information on the DL/I Test program has been moved to the *IMS/VS Application Programming Reference Manual;* and information on Insert, Delete, and Replace Rules in Chapter 1 has been deleted. Duplicate information on these rules is in the *IMS/VS System/Application Design Guide.*

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/370 Bibliography,* GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, P. O. Box 50020, Programming Publishing, San Jose, California 95150. All comments and suggestions become the property of IBM.

UTILITY INDEX

GENERATION UTILITIES

DATA BASE UTILITIES

IMS/VS SYSTEM LOG UTILITIES

PERFORMANCE REPORTING AND SERVICE UTILITIES

HOW TO USE THIS PUBLICATION

This publication is designed for system programmers, application programmers, system analysts, and computer operators whose responsibilities require a knowledge of how to execute the Information Management System/Virtual Storage (IMS/VS) 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.
- "Guide to the IMS/VS Publications" chart.

These topics are explained below.

ORGANIZATION OF THE PUBLICATION

In addition to this section, a table of contents, and a list of figures, this publication contains:

- Summary of Amendments for Version 1, Release 1.2; for Version 1, Modification Level 1.1; and for Version 1, Modification Level 1, which are summaries of the major technical changes reflected in this publication for each release.
- Four parts, each consisting of several chapters that explain related utilities.

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

  1. Data Base Description (DBD) Generation
  2. Program Specification Block (PSB) Generation
  3. Application Control Blocks (ACB) Maintenance Utility

  Part 2, "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:

  4. Data Base Reorganization/Load Utilities
  5. Data Base Recovery Utilities
  6. Utility Control Facility

Part 3, "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:

7. Log Maintenance Utilities

8. Log Data Formatting Utilities

Part 4, "Performance Reporting and Service Utilities," has two chapters that describe the utilities used to produce performance-related summary reports and to copy messages onto a system output device. The chapters are:

9. Performance Reporting Utilities

10. System Service Utility

- "Index," a subject index to this publication.

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 System/Application Design Guide, SH20-9025
IMS/VS Application Programming Reference Manual, SH20-9026

ASSOCIATED PUBLICATIONS

The additional publications associated with this publication are:

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
IMS/VS Message Format Service User's Guide, SH20-9053
IMS/VS Advanced Function for Communications, SH20-9054
IMS/VS Program Logic Manual, Volume 1 of 3, LY20-8004
IMS/VS Program Logic Manual, Volume 2 of 3, LY20-8005
IMS/VS Program Logic Manual, Volume 3 of 3, LY20-8041
IMS/VS Low-Level Code/Continuity Check in DL/I: Program Reference and Operation Manual, SH20-9047

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 Access Method Services, GC26-3826
OS/VS JCL Reference, GC28-0618
OS/VS DOS/VS VM/370 Assembler Manual, GC33-4010

GUIDE TO USING THE IMS/VS PUBLICATIONS

The chart concluding this section provides a guide for using the IMS/VS publications. It is divided into three parts, each dealing with a specific IMS/VS component -- Data Base System, Data Communication feature, and Interactive Query Facility (IQF) feature. For each component, one or more functional areas is identified. For each functional area, one or more tasks are specified, and the IMS/VS manual or manuals that contain major information regarding this task are noted. The titles of the IMS/VS manuals are abbreviated as follows:

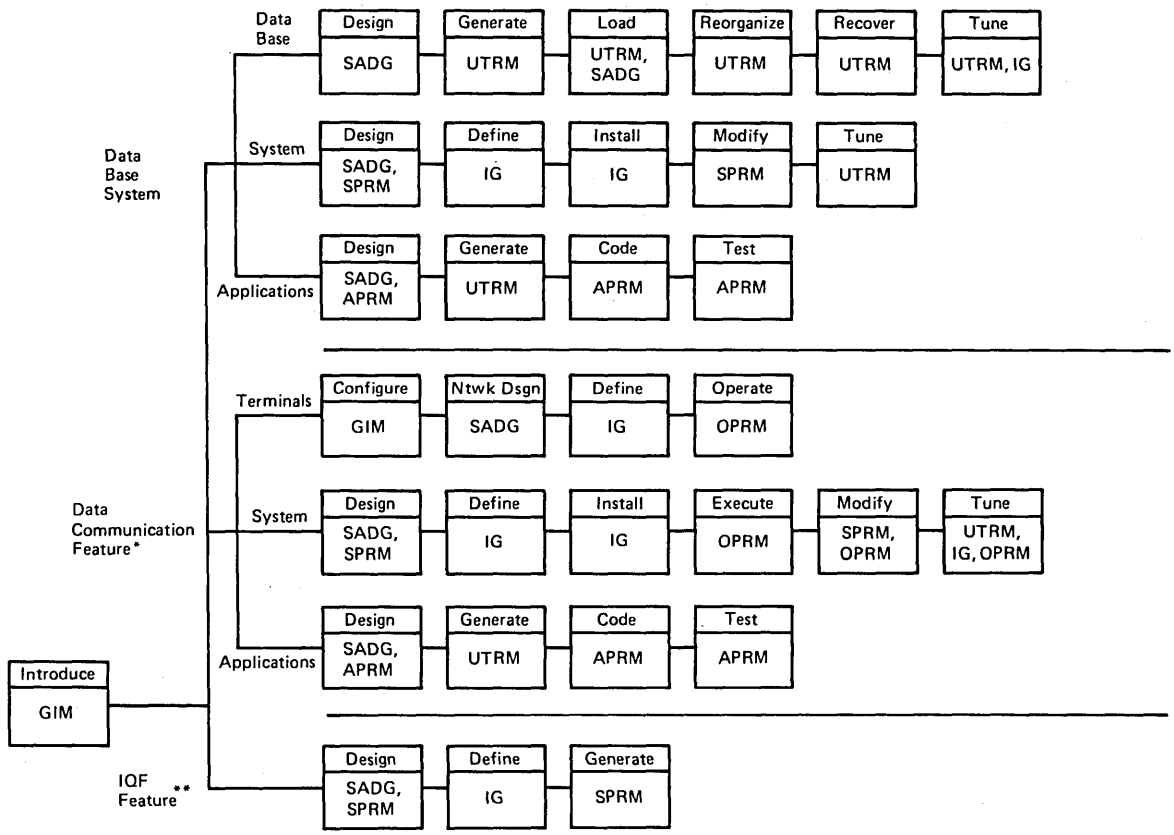
Abbreviation	Full Manual Title
GIM	IMS/VS General Information Manual
SADG	IMS/VS System/Application Design Guide
IG	IMS/VS Installation Guide
SPRM	IMS/VS System Programming Reference Manual
APRM	IMS/VS Application Programming Reference Manual
UTRM	IMS/VS Utilities Reference Manual
OPRM	IMS/VS Operator's Reference Manual

Five IMS/VS manuals are not referred to on the chart:

- IMS/VS Messages and Codes Reference Manual: This manual supports essentially all tasks noted on the chart.
- IMS/VS Program Logic Manual: This manual supports essentially all tasks noted on the chart.
- IMS/VS Low Level Code/Continuity Check in DL/I: Program Reference and Operation Manual: This manual supports the Data Base System when the LLC/CC function is used.
- IMS/VS Message Format Service User's Guide: This manual supports the Data Communication feature when MFS is used.
- IMS/VS Advanced Function for Communications: This manual supports the Data Communication feature when either the IBM 3600 Finance Communication System or IBM 3790 Data Communication System is used.

The IQF section on the chart refers only to IMS/VS system library manuals that contain information on IQF. Additional IQF information can be found in:

- IQF General Information Manual, GH20-1074
- IQF Language Guide, GH20-1222
- IQF Terminal User's Reference Guide, GH20-1223



*References for the DC feature are in addition to those for the DB System.

**References for this feature are in addition to those for the DC feature.

Figure P-1. Guide to the IMS/VS Publications

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

VERSION 1, RELEASE 1.2

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

TECHNICAL CHANGES

- 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 Pass1 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 the IMS/VS System Programming Reference Manual.
- Information on the DL/I Test program (formerly Appendix C) has been moved to the IMS/VS Application Programming Reference Manual.
- Information on Insert, Delete, and Replace Rules in Chapter 1 has been deleted. Duplicate information is in the 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

PART 1. GENERATION UTILITIES

Part 1 has three 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, and INDEX 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 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 the IMS/VS System/ Application Design Guide. In particular, the chapter "Data Base Design Considerations" of the Design Guide is a prerequisite to this chapter.

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. 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 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 relationships between segments.

HISAM DBD Generation

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

- One to ten 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 ten 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 ten 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) 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.

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:

  a. The sequence field key of the root segment occurrence it indexes.
  b. 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 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.
- 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 16	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 1-1 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 six types of DBD generations.

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

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

** Maximum of 28 for a secondary index data base.

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

Figure 1-1. Summary of DBD Generation Statements

Figure 1-2 shows a summary of 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	HDAM	HISAM	HIDAM
DBD	NAME=	(dbname1		R	O	R	R	R	R	R			
		,dbname2,...)			R				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			
	RMNAME =	(mod					R						
		,anch					O						
		,rbn					O						
		,bytes)					O						
	,PASSWD=	yes		O	O	O	O		O	O	O	O	O
DATASET	LOGICAL									R			
	or DD1=	ddname1		R	R	R	R	R	R				
	,DEVICE=	device		R		R	R	R	R				
	,MODEL=	model		O		O	O	O	O				
	,DD2=	ddname2		R	O								
	,OVFLW=	ddname3	1			R			R				
	,BLOCK=	(blkfact1	2	O	O	O	O	O	O				
		,blkfact2)	2	O		O			O				
		or size	2				O	O					
	,SIZE=	(size1	2		O	O	O	O	O				
		,size2)				O			O				
	,RECORD=	(reclen1	2	O	O	O			O				
		,reclen2)	2	O	O	O			O				
	,RECFM=	,recfm1			R								
	,SCAN=	cyls					O	O					
	,FRSPC=	(fbff					O	O					
		,fspf)					O	O					

KEY
O = Optional
R = Required
X = Use for logical relationships
* Notes referenced are in part 4 of this figure.

Figure 1-2 (Part 1 of 4). 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	HISAM	HDAM	HIDAM	INDEX	LOGICAL	HDAM	HISAM	HIDAM
SEGM	NAME=	segname1		R	R	R	R	R	R			
	,PARENT=	0	3	O	O	O	O	O	O			
		or										
		((segname2	4	R	R	R	R		R			
		,[SNGL / DBLE])				O	O					
		,(lpsegname								X	X	X
		,[VIRTUAL / PHYSICAL]								X	X	X
		,dbname1))								X	X	X
	,BYTES=	(max bytes		R	R	R	R	R				
		,min bytes)	5		O	O	O					
	,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])	6	O	O	O	O	O				
	,SOURCE=	((segname							R	X		X
		,[KEY / DATA]							O	X		X
		,dbname)							R	X		X
		(,segname	7						R			
		,[KEY / DATA]							O			
		,dbname))	7						R			
	,COMPRTN=	(routinename	5		O	O	O					
		,[KEY / DATA]	5		O	O	O					
		,INIT)	5		O	O	O					

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

* Notes referenced are in part 4 of this figure.

Figure 1-2 (Part 2 of 4). 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	HDAM	HISAM	HIDAM
LCHILD	NAME=	(segname1	5/8			R	R	R	R		X	X	X
		,dbname)	5/8			R	R	R	R		X	X	X
	[,POINTER= ,PTR=]	SNGL	9						O		X	X	X
		DBLE									X	X	X
		NONE									X	X	X
		INDX	10				O	R					
		SYMB	11			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				
		,SEQ	12	O		O	O	O	O				
		[,U / M])		O		O	O	O	O				
		or systrelfldname				O	O	O					
	,BYTES=	bytes		R		R	R	R	R				
	,START=	startpos		R		R	R	R	R				
	,TYPE =	[X / P / C]		O		O	O	O	O				
XDFLD	NAME=	fldname	5			R	R	R					
	,SEGMENT=	segname				O	O	O					
	,CONST=	char	13			O	O	O					
	,SRCH=	list1	13			R	R	R					
	,SUBSEQ=	list2	13			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			
FINISH				R	R	R	R	R	R	R			
END				R	R	R	R	R	R	R			

KEY

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

* Notes referenced are in part 4 of this figure.

Figure 1-2 (Part 3 of 4). Summary of Statements and Operands Used by Data Base Type

Notes:

1. Required if ISAM/OSAM are the OS access methods. If VSAM is the OS access method for an index data base, and keys of all index segments are unique, OVFLW need not be specified. OVFLW is invalid for a simple HISAM data base.

2. When not specified by the user, DBDGEN generates value used.

3. The PARENT=keyword must be omitted, or PARENT=0 must be specified for the root segment type of a data base.

4. Required on SEGM statements for all dependent segment types.

5. VSAM is the prerequisite for variable length segments, segment edit/compression and secondary indexing. Variable length segments, segment edit/compression and secondary indexing are invalid for a simple HISAM data base.

6. Required when a segment type does not have a unique sequence field.

7. Used to define a concatenated segment type. Not allowed for a simple HISAM data base.

8. Required for HIDAM primary index, optional for a secondary index.

9. Required for primary index of HIDAM data base.

10. 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.

11. 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.

12. A unique sequence field is required for the root segment type of HISAM, HIDAM, and the primary HIDAM index data bases.

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

Figure 1-2 (Part 4 of 4). Summary of Statements and Operands Used by Data Base Type

DBD Generation Input Deck Structure

For DBD generation, an OS/VS job step accepts ten types of control statements from the SYSIN job stream arranged in a specific order. Each control statement is described in detail in the following sections.

Figure 1-3 shows the rules for structuring a DBD generation input deck. The order of statements shown is required for all six types of DBD generation. 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.

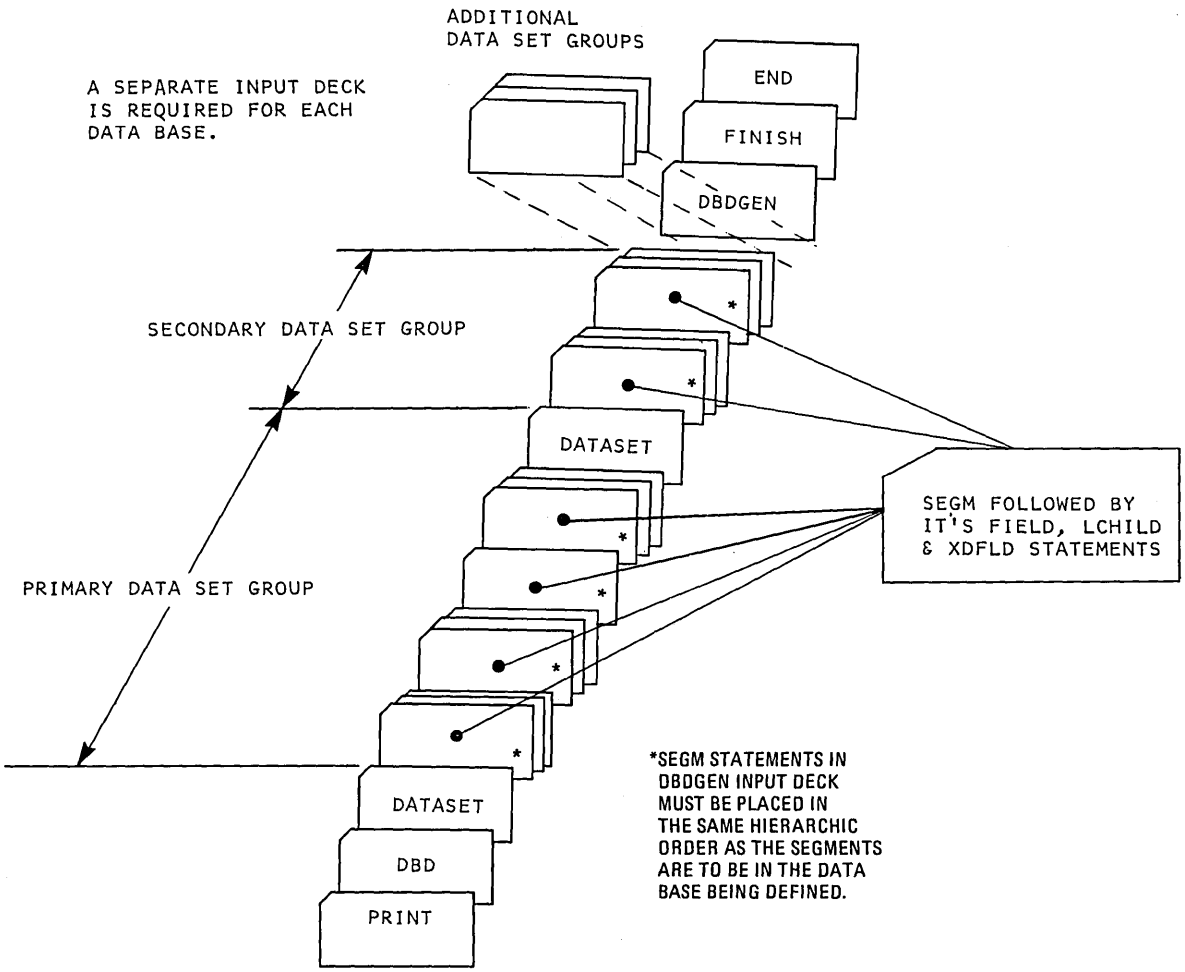


Figure 1-3. DBDGEN Input Deck Structure

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 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, these syntax conventions apply:

   a. Words written in all capital letters must appear exactly as written.

   b. Words written in lowercase letters are to be replaced by a user-specified value. Valid user-specified values are numeric values or 1- to 8-character alphameric names.

   c. 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 sixteen on the continuation statement.

   d. The symbols [ ], { }, and ,... are used as an aid in defining the operands of a control card. These symbols are not coded; they act only to indicate how a control statement may be written.

      [ ] indicates optional operands. The operand enclosed in the brackets (for example, [VL]) may or may not be present, depending on whether or not the associated option is desired. If more than one item is enclosed in brackets, one or none may be coded.

      { } indicates that a choice of an operand parameter must be made. One of the operand parameters from the vertical stack within braces must be coded.

      ,... indicates that more than one set of parameters may be designated in the same operand.

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. When a data base PCB relates to this DBD generation, this operand value must be the name used in the DBDNAME= operand on a data base PCB statement within a PSB generation. For a DBD generation of a shared secondary index, only one of the data set names may be used; the other names will be added as aliases.

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.

ASSEMBLY LISTING

An OS/VS assembly language listing of the DBD macro expansion created by DBD generation execution is provided. The inclusion of an assembly 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 assembly listing will provide some of the parameters necessary to define the data sets of the data base to VSAM. 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
+*,*
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,*
+*,     DEFINE CLUSTER (NAME(DDI3I1) -
+*,*            INDEXED KWYS (10,6) -
+*,*            RECORDSIZE (678,678) -
+*,*            CONTROLINTERVALSIZE (2048))
+*,*
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
+*,*
+*,*    DEFINE CLUSTER (NAME(DDI3O1) DDI3O1 NONINDEXED -
+*,*            RECORDSIZE (678,678) -
+*,*            CONTROLINTERVALSIZE (2048))
+*,*
+*,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

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 1-4. Refer to the table in the figure for an explanation of the converted values.

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

```
DBD NAME=DI41SK01,ACCESS=(HISAM,VSAM)
DATASET DD1=DDI3I1,DEVICE=2314,OVFLW=DDI3O1
SEGM NAME=SEGA1,PARENT=0,BYTES=40
FIELD NAME=(FLDA1,SEQ,U),BYTES=10,START=1,TYPE=C
SEGM NAME=SEGB1,PARENT=((SEGA1)),BYTES=10,FREQ=9
FIELD NAME=(FLDB1,SEQ,U),BYTES=8,START=2,TYPE=C
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
```

Example:

Output from DBD generation contains the statement:

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

Convert the values to binary and decimal representations:

```
 Byte 0   Byte 1    Byte 2    Byte 3
  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. Non-sequenced inserts at current position.

Byte 2 - Two four-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.

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 non-sequential last
	..01....	Insert non-sequential first
	..11....	Insert non-sequential here at current position
	....10..	Replace physical
	....01..	Replace virtual
	....11..	Replace logical
	10	Delete physical
	01	Delete virtual
	11	Delete logical
	00	Bi-virtual delete
2	.XX..XXX	Reserved
	1.......	Segment is paired
	...1....	Segment type requires type 64 processing
	....1...	Segments 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 1-4. Segment Flag Codes

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 the 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 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:

STATEMENT TYPE	KEYWORD	OPERAND	NOTES *	operands used by data base type							operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	HDAM	HISAM	HIDAM
DBD	NAME=	(dbname1		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			
	RMNAME =	(mod					R						
		,anch					O						
		,rbn					O						
		,bytes)					O						
	,PASSWD=	yes		O	O	O	O	O	O		O	O	O

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

where:

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
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.

RMNAME=
specifies information used to manage data stored in an HDAM data bases primary data set group. This operand is only valid when ACCESS=HDAM is specified. The parameters of this operand are defined below. A randomizing module controls root segment placement in or retrieval from an 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, an 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 HDAM data base. An example of a randomizing module is in the 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 one to five.

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.

rbn

specifies the maximum relative block number value that the user wishes to allow a randomizing module to produce for this data base. 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.

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. 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.

PASSWD=

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 Operating System access method. (The default is NO.)

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.

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

DATASET STATEMENT

Each DATASET statement defines a data set group within a data base. At least one DATASET statement is required for each DBD generation. The maximum number depends on the type of data base being defined. HSAM, SHSAM, GSAM, SHISAM, index and logical data bases can have only one data set group. HISAM, HDAM, and HIDAM data bases can be divided into 1 to 10 data set groups subject to rules that follow.

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 the paragraph titled "Use of the Label Field" for this exception.

Rules for Dividing a Data Base into Multiple Data Set Groups

HISAM, HDAM and HIDAM data bases can be divided into a maximum of ten 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 1-5.

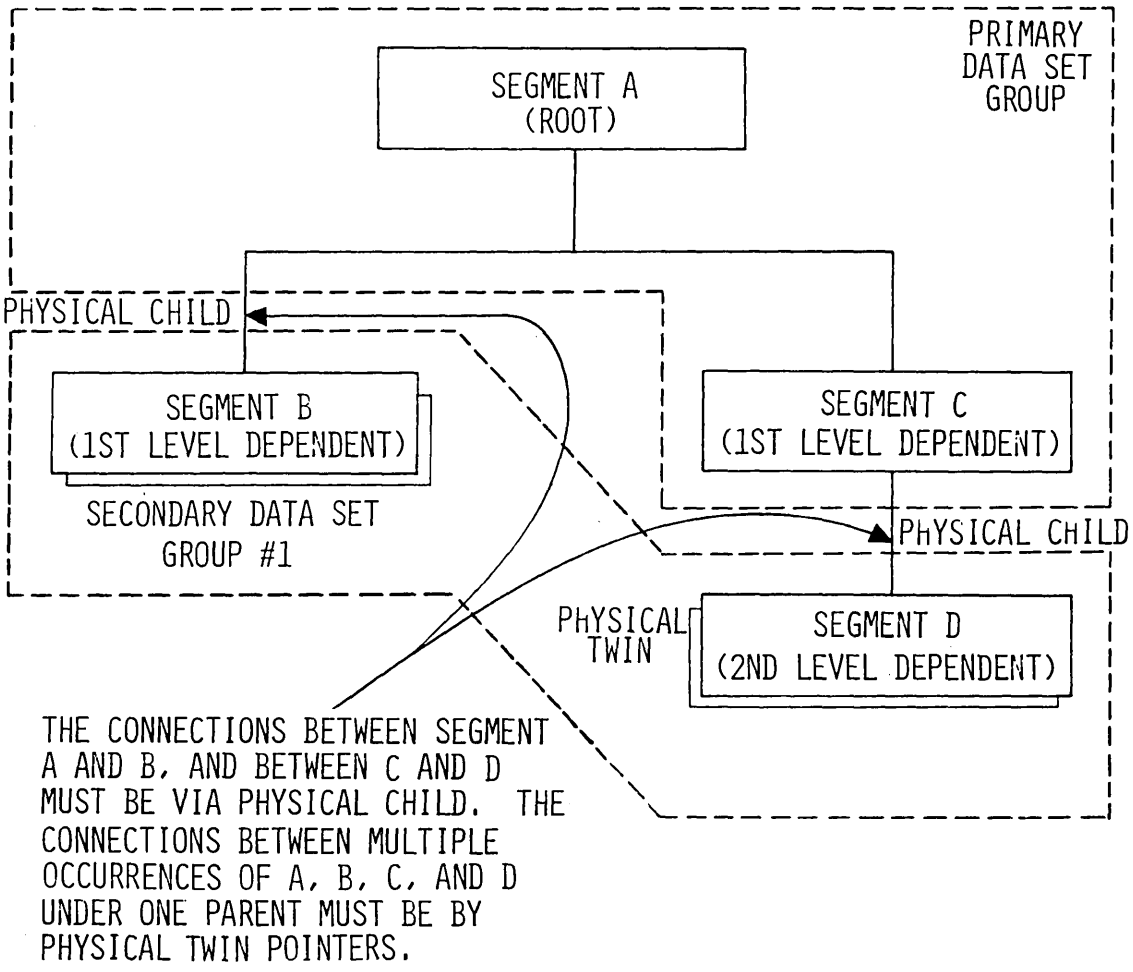


Figure 1-5. Connections through Physical Child and Physical Twin Pointers

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 hierarchical 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 hierarchical 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 can not 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 1.5, 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

DSG1      DATASET       DD1=PRIMARY,DEVICE=2314,BLOCK=1648

          SEGM          NAME=SEGMENTA,BYTES=100

DSG2      DATASET       DD1=SECOND,DEVICE=2314,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 format of the DATASET statement is:

STATEMENT TYPE	KEYWORD	OPERAND	NOTES	operands used by data base type							operands used for logical relationships		
				HSAM	GSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL	HDAM	HISAM	HIDAM
DATASET	LOGICAL									R			
or	DD1=	ddname1		R	R	R	R	R	R				
	,DEVICE=	device	3	R		R	R	R	R				
	,MODEL=	model		O		O	O	O	O				
	,DD2=	ddname2		R	O								
	,OVFLW=	ddname3	1			R			R				
	,BLOCK=	(blkfact1	2	O	O	O	O	O	O				
		,blkfact2)	2	O		O			O				
		or											
		size	2				O	O					
	,SIZE=	(size1	2		O	O	O	O	O				
		,size2)				O			O				
	,RECORD=	(reclen1	2	O	O	O			O				
		,reclen2)	2	O	O	O			O				
	,RECFM=	,recfm	4		R								
	,SCAN=	cyls					O	O					
	,FRSPC=	(fbff					O	O					
		,fspf)					O	O					

KEY

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

Notes:

1. Required if ISAM/OSAM are the OS access methods. If VSAM is the OS/VS access method for an index data base, and keys of all index segments are unique, OVFLW need not be specified. OVFLW is invalid for a simple HISAM data base.

2. When not specified by the user, DBDGEN generates value used.

3. DEVICE is invalid for a GSAM data base.

4. RECFM is only valid for a GSAM data base.

where:

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 illegal, and this must be the only DATASET statement for the DBD generation. The SEGM statements that follow this statement can only have 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 1-6.

HSAM	GSAM	HISAM	HIDAM HDAM	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	ddname of primary data set	operand is invalid

Figure 1-6. 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 control interval or block lengths do not exceed the usable track length of the storage device specified. A list of the valid entries for this operand follows (one of the underlined values must be chosen):

Device Name	Device=
Disk Facility	2314, 2305, 2319, 3330, 3340, or 3350
Tape	2400, 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 32,000 bytes.

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

DD2=

specifies the 1- to 8-character alphameric ddname of the output data set required for an HSAM or simple HSAM data base. For GSAM, this operand is optional, and is used to specify the ddname of the output data set. If it is omitted, ddname1 is assumed.

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 non-unique 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 since all index segments are inserted in the key sequenced data set of the index.

BLOCK=

is used to specify the blocking factors to be used for data sets in each data set group for HSAM, GSAM, HISAM, and INDEX data bases, as shown in Figure 1-7.

For HISAM and INDEX data bases that use ISAM/OSAM as the OS/VS access methods, the BLOCK=operands 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 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 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 is not a multiple of 512 and is less than 8192 bytes, the resulting control interval size is the product rounded 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 is in the range of 8192 to 30270 bytes but is not a multiple of 2048, the resulting control interval size is the product rounded up to the next higher multiple of 2048.

For HDAM and HIDAM data bases, the SIZE 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 SIZE operand, DBDGEN adds space for root anchor points, free space elements, and data set 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.

HSAM	GSAM	HISAM	HIDAM HDAM	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 reclen1 data set logical record length. reclen2 is overflow data set logical record length.	BLOCK= size, or SIZE= size of OSAM or VSAM data set in data set group. RECORD= is ignored.	BLOCK= blkfact1 is primary data set blocking factor. blkfact2 is overflow blocking factor. RECORD= reclen1 is primary is primary 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.

Figure 1-7. Use of BLOCK= and RECORD= Operands

SIZE=

is used to override DBDGENs computation of control interval or block size. When used, no overhead is added to the values specified. 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. For VSAM data sets in the range of 8192 to 30720 bytes (maximum allowed size), the values specified 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, 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. For ISAM/OSAM the size value specified must be an even multiple of the record parameter plus overhead 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.

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

size ≥ MAXSEG + VSAM CONTROL

Where:

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

VSAM CONTROL= 7 bytes for ISAM/OSAM data sets, 12 bytes for unblocked BSAM data sets.

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 RMNAME=rbn operand is specified:

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

  size ≥ (RAPs*4) + FSE + 2 + OVERHEAD,
  and
  size ≥ MAXSEG + FSE + OVERHEAD

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

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

  where:

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

  FSE= 4 bytes for a free space element

  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 + FSE + VSAM CONTROL

where:

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

FSE= 4 bytes for a free space element

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. In the case of OSAM, the block 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 size specified for the primary data set group must be:

  size ≥ MAXSEG + FSE + RAP + OVERHEAD

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

  size ≥ MAXSEG + FSE + OVERHEAD

  where:

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

  FSE= 4 bytes for a free space element

  OVERHEAD= 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. The meaning of each of the operand's parameters depends on the type of data base being defined as shown in Figure 1.7. For a simple HISAM data base, the logical record length specified must be the same as the segment length specified. For both the VSAM KSDS and ESDS data set for HISAM and INDEX DBDs, the logical record length specified must be an even value. 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=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.

The operand is only valid for a GSAM data base.

SCAN=cyls
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=(fbff,fspf)
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.

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 cards must be included in the JCL for the batch processing region. To preserve the integrity of a data base, simultaneous use by multiple OS jobs is prohibited unless the processing intent of each is read-only or message or batch message processing is used.

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; one for the KSDS and one for the ESDS.

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 non-unique 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. The DDNAME for this DD statement is DFSVSAMP. The following example illustrates the defining, loading, processing, and printing of a HISAM VSAM data base. The circled numbers refer to explanatory notes that follow the example.

```
//SAMPLE JOB
//JOBLIB DD DISP=SHR,DSN=IMSVS10.PGMLIB
//       DD DISP=SHR,DSN=IMSVS10.RESLIB
① //JOBCAT DD DISP=SHR,DSN=IMSCAT1
② //BLDVSAM  EXEC  PGM=AMS,REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
 DELETE HISAM1.VSAM.KSDS
 DELETE HISAM1.VSAM.ESDS
 DEFINE CLUSTER (NAME(HISAM1.VSAM.KSDS) -
                 INDEXED KEYS (10,6) -
                 CYL (1,1) VOLUMES (VSIMSA) -
                 RECORDSIZE (678,678)) -
        DATA     (CONTROLINTERVALSIZE (2048))
 DEFINE CLUSTER (NAME(HISAM1.VSAM.ESDS) NONINDEXED -
                 CYL (1,1) VOLUMES (VSIMSA) -
                 RECORDSIZE (678,678) -
                 CONTROLINTERVALSIZE (2048))
/*
③ //LOAD EXEC PGM=DFSRRC00,REGION=320K,PARM='DLI,DFSDDLT0,HIBLSK41,,01'
//IMS DD DISP=SHR,DSN=IMSVS10.PSBLIB
// DD DISP=SHR,DSN=IMSVS10.DBDLIB
//DDI3I1 DD DISP=OLD,DSN=HISAM1.VSAM.KSDS
//DDI3O1 DD DISP=OLD,DSN=HISAM1.VSAM.ESDS
//PRINTDD DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//IEFRDER DD DUMMY
//SYSIN DD DISP=SHR,DSN=IMSVS10.TESTLIB(LOAD2F)
//DFSVSAMP DD *
2048,4
OPTIONS,DUMP=YES
/*
④ //GET  EXEC PGM=DFSRRC00,REGION=320K,PARM='DLI,DFSDDLT0,HIBASK41,,01'
//IMS DD DISP=SHR,DSN=IMSVS10.PSBLIB
// DD DISP=SHR,DSN=IMSVS10.DBDLIB
//DDI3I1 DD DISP=OLD,DSN=HISAM1.VSAM.KSDS
//DDI3O1 DD DISP=OLD,DSN=HISAM1.VSAM.ESDS
//PRINTDD DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//IEFRDER DD DUMMY
//DFSVSAMP DD *
678,6
2048,4
⑤ OPTIONS,DUMP=YES
/*
//SYSIN  DD  DSNAME=IMSVS10.TESTLIB(GN2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(CC2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(BOOL2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(GU2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(GNQ2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(NOTF2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(ISRTB),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(ISRT2),DISP=SHR
//       DD  DSNAME=IMSVS10.TESTLIB(GN2),DISP=SHR
⑥ //DELETE EXEC PGM=AMS,REGION=256K,COND=EVEN
//SYSPRINT DD SYSOUT=A
//DDI3I1 DD DISP=OLD,DSN=HISAM1.VSAM.KSDS
//DDI3O1 DD DISP=OLD,DSN=HISAM1.VSAM.ESDS
//SYSIN DD *
  PRINT  INFILE(DDI3I1)
  PRINT  INFILE(DDI3O1)
 DELETE HISAM1.VSAM.KSDS
 DELETE HISAM1.VSAM.ESDS
/*
```

Notes:

1. The JOBCAT DD statement defines the VSAM user catalog which will be used for all VSAM processing during the job.

2. Step BLDVSAM uses the VSAM Access Method Services utility program to define and allocate the data sets which make up the HISAM data base. HISAM1.VSAM.KSDS is the DSNAME of the KSDS, and HISAM1.VSAM.ESDS is the DSNAME of the ESDS.

3. Step LOAD loads the HISAM data base. DD card DDI3I1 defines the KSDS and DDI301 defines the ESDS. The DFSVSAMP data set contains parameters to define the IMS/VS VSAM buffer pool. When loading HISAM, the KSDS logical records are not processed in the buffer pool, so no subpool is defined for them.

4. Step GET processes the HISAM data base.

5. For information on control statement format and buffer pool structure see "Defining the IMS/VS VSAM Buffer Pool" in the IMS/VS Installation Guide.

6. Step DELETE uses Access Method Services to print the data sets and then deletes them from the VSAM data space and catalog.

ISAM/OSAM

Operating system procedures for execution of all IMS/VS region types are provided in the 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 OSAM DD statements:

```
//dd1      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 the 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, but recommended)

M = Master index creation (optional)

OPTCD=R indicates that incore 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 incore indexes are to be used. Incore 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 incore 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.

Since 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.

The user may specify the RECFM=FB, or it may be omitted. 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:

```
//dd1      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 that is placed in the data set group defined by the preceding DATASET statement in the input deck. The SEGM statement defines a segment type, the segments position in a data base hierarchy, the physical characteristics of the segment, and how the segment is to be related to other segments. At least one SEGM statement must immediately follow a DATASET statement, and a maximum of 255 SEGM statements are allowed in a DBD generation. SEGM statements must be placed in the input deck in hierarchic sequence. 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:

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
SEGM	NAME=	segname1		R	R	R	R	R	R			
	,PARENT=	0	1	O	O	O	O	O	O			
		or										
		((segname2	2	R	R	R	R		R			
		,[SNGL / DBLE])				O	O					
		,(lpsegname								X	X	X
		,[VIRTUAL / PHYSICAL]								X	X	X
		,dbname1))								X	X	X
	,BYTES=	(max bytes		R	R	R	R	R				
		,min bytes)	3		O	O	O					
	,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])	4	O	O	O	O	O				
	,SOURCE=	((segname							R	X		X
		,[KEY / DATA]							O	X		X
		,dbname)							R	X		X
		(,segname	5						R			
		,[KEY / DATA]							O			
		,dbname))	5						R			
	,COMPRTN=	(routinename	3		O	O	O					
		,[KEY / DATA]	3		O	O	O					
		,(INIT)	3		O	O	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. Required on SEGM statements for all dependent segment types.
3. VSAM is the prerequisite for variable length segments, segment edit/compression and secondary indexing. Invalid for a simple HISAM data base.
4. Required when a segment type does not have a unique sequence field.
5. Required to define a concatenated segment type. Not allowed for a simple HISAM data base.

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
specifies the name of the physical parent for all segment types except roots of the segment type being defined.

SNGL
DBLE
SNGL, when specified, causes a 4-byte physical child first pointer to be placed in occurrences of the physical parent of the segment type being defined. If the parent segment specifies PTR=HIER or PTR=HIERBWD, the SEGM statement for the child segment cannot specify SNGL or DBLE in the PARENT operand.

DBLE, when specified, causes a 4-byte physical child first and a 4-byte physical child last pointer to be placed in occurrences of the physical parent of the segment type being

defined. If the physical parent of the segment type being defined is not a root and it has heirarchic pointers specified, then this parameter is ignored. This parameter will default to SNGL if the physical parent uses twin pointers rather than hierarchic.

SNGL/DBLE may be specified only for HDAM or HIDAM data bases.

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
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.

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" specify 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. For tape, the maximum is 32K. 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.

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 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; the 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 four.

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

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 IFO110 ARITHMETIC OVERFLOW assembler error message may occur when the DBDGEN utility is attempting to calculate a recommended space allocation. If this should occur during a HISAM DBD generation, it would be necessary for the user to determine his own space allocation.

POINTER=
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 prefix of occurrences of the segment being defined.

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

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)
```

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

*** indicates this parameter 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 1-8 (Part 1 and Part 2) 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	HSAM SHSAM SHISAM	HISAM	HDAM	HIDAM	INDEX
Reserved							
Ptr to next segment in hierarchy	HIER	INVALID	INVALID	IGN	VALID (1)	VALID (1)	IGN
Ptr to next and previous segments in hierarchy	HIERBWD	INVALID	INVALID	IGN	VALID (1)	VALID (1)	IGN
Ptr to next occurrence of physical twins	TWIN	INVALID	INVALID	IGN	VALID (1)	VALID (1)	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

INVALID - This parameter cannot be specified.

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

VALID - This parameter is valid. If the segment type is being defined in an HDAM or HIDAM data base, HIER, HIERBWD, TWIN, or TWINBWD should be specified. The default value is TWIN except when the HIER or TWIN parameters are specified for the root segment type of a HIDAM data base that uses VSAM as the OS/VS access method. In this case the default is NOTWIN.

Figure 1-8 (Part 1 of 2). Use of POINTER= Operand Parameters (No Logical Relationship)

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

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.

Figure 1-8 (Part 2 of 2). Use of POINTER= Operand Parameters (Logical Relationship)

Notes:

1. 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.

2. 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.

3. 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).

4. 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.

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.

```
          B
RULES=(PPP,FIRST)
       LLL LAST
       VVV HERE

          B
       PPP
       LLL
       VVV
```

specifies the rules used for insertion, deletion, and replacement of occurrences of the segment type being defined.

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
LAST
HERE

specifies where new occurrences of the segment type defined by this SEGM statement are inserted into their physical data base. This value is only used when processing segments with no sequence field or with a non-unique sequence field. The value is ignored when specified for a segment type with a unique sequence field defined.

FIRST: For segments without a sequence field defined, a new occurrence is inserted before all existing physical twins. For segments with a non-unique 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 non-unique 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 physcial 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 non-unique 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.

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.

1. 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.

2. 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 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 explicitely 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.

Note: For the Interactive Query Facility (IQF), if a bidirectional logical relationship is implemented by using virtual pairing, and if the virtual logical child is defined, then the user must reference the virtual logical child (via the SOURCE operand) when: (1) a segment is being defined in a logical DBDGEN and (2) the segment consists of the concatenation of the virtual logical child and the physical parent of the real logical child.

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 HSAM, simple HSAM, simple HISAM, and logical data bases. In addition, the COMPRTN= operand is invalid for the primary INDEX of a HIDAM data base. When used for a HISAM data base, it must not change the sequence field offset for HISAM root segments. Segments specifying the COMPRTN parameter must reside in a VSAM data set. 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 one to eight 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. Since 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:

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	1/2		R	R	R	R		X	X	X
		,dbname)	1/2		R	R	R	R		X	X	X
	[,POINTER= ,PTR=]	SNGL	3					O		X	X	X
		DBLE								X	X	X
		NONE								X	X	X
		INDX	4			O	R					
		SYMB	5		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. VSAM is the prerequisite for variable length segments, segment edit/compression and secondary indexing. Invalid for a simple HISAM data base.

2. Required for HIDAM primary index, optional for a secondary index.

3. Required for primary index of HIDAM data base.

4. 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 ommitted or specified as PTR=SNGL on the LCHILD statement of the INDEX DBD.

5. 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.

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 segname1 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 segname1. dbname can be omitted when segname1 is defined in this DBD generation. Both segname1 and dbname must be 1- to 8-character alphameric values.

POINTER=

used to specify the pointers used in logical or secondary index relationships. When the POINTER= keyword is omitted from any index specification, POINTER=SNGL is the default. When the POINTER= keyword is omitted from a logical relationship specification, POINTER=NONE 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 this case, omit the PTR= operand or specify PTR=SNGL on the LCHILD statement. An LCHILD statement for a HIDAM primary index must precede the LCHILD statements for secondary indexes.

SYMB
can be used in the DBDGEN for the target data base of a secondary index to specify that the concatenated keys of index target segments are to be placed in 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 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 used during 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.

RULES=
is used for logical relationhips no sequence field or a non-unique sequence field has been defined for the 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: 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. States that the insert is dependent on the position established by the previous DL/I call.

When a new occurrence of a logical child is inserted from it's physical parent, no previous position exists for the logical child on it's 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 non-unique sequence field has been defined.

Note: 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 that can be referred to 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, and primary HIDAM INDEX data bases.

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:

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

KEY

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

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 fldname1 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.

U
M

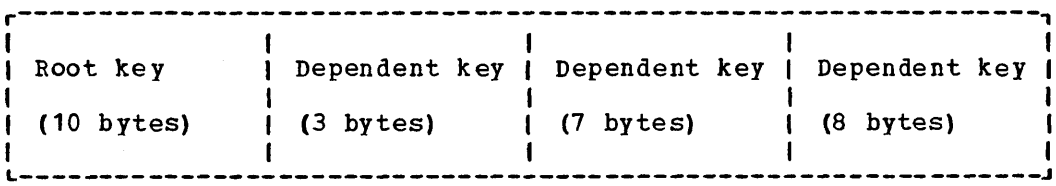
the keyword U indicates that only unique values are allowed in the sequence field of occurrences of the segment type. For a root segment type, the sequence field of each occurrence must contain a unique value. For a dependent segment type, the sequence field of each occurrence under a given physical parent segment must contain a unique value. The keyword M indicates that duplicate values are allowed in the sequence field of occurrences of the segment type.

When no sequence field or a non-unique 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.

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 eight 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:



If three system related fields were to consist of bytes 2-8 of the root key, byte 1 of the second key and bytes 5-6 of the 4th 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 insure 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 four 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.

START=

specifies the starting position 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. 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 parents 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

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.

FIELD Statement Considerations for the Interactive Query Facility (IQF)

1. The parent of the lowest segment involved in a query path must be identified by a unique concatenated key within the data base used to process the query. One method that can be used to ensure compliance with this rule is to define a unique sequence field for every segment involved in a query path, except for the lowest. Root key values must be unique.

2. If a logical child is involved in a query path, the concatenated key of its destination parent must be unique. One method that can be used to ensure compliance with this rule is to define a unique sequence field for every segment whose sequence field is a component of the destination of the parent's concatenated key.

3. A field that is indexed by IQF can be no greater than 250 bytes in length.

4. Scattered sequence fields are not supported by IQF.

5. All fields of a virtual logical child that are to be used in an IQF query must be defined by FIELD or *FIELD macro statements that refer to the data of the virtual logical child. (IQF does not automatically refer to field definitions provided for a real logical child and duplicate them under the virtual logical child at the appropriate offsets as does IMS).

6. When a virtual logical child is defined, and when the user provides the virtual logical child in the input data stream provided to the IQF utility before the corresponding definition of the real logical child, the user must provide a FIELD or *FIELD macro statement for the virtual logical child such that the last byte of the virtual logical child data is included within the range of data defined by FIELD or *FIELD macro statement.

7. A packed decimal field used as an index entry must have a C-zone for positive; F-zone does not qualify.

XDFLD -- Define an Indexed Field

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.

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 or HSAM 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 1,000 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. The index target segment, which is the segment defined by the preceding SEGM statement in the DBD generation input deck must not be a logical child segment type.

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 one to eight 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 one-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' since 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. 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

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

KEY

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

This statement must be included. It indicates the end of DBD generation control cards used to define the DBD.

FINISH STATEMENT

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

KEY

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

This statement must be included. It sets a nonzero condition code for link-edit if there are DBD generation errors.

END STATEMENT

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

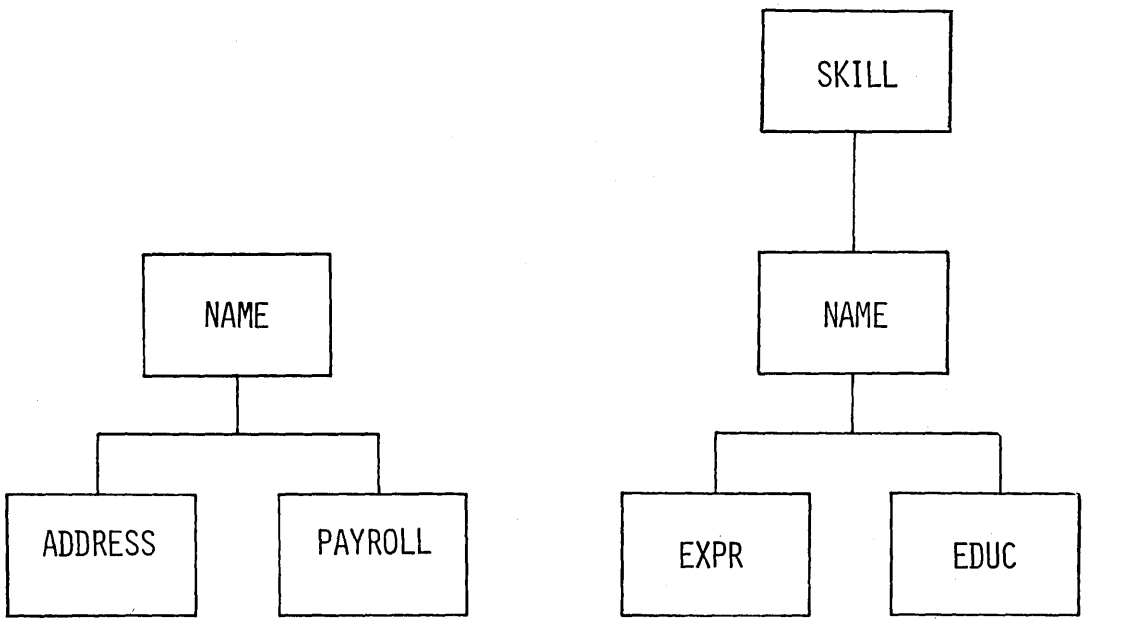
KEY

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

This statement must be entered. It indicates the end of input statements to the OS/VS assembler.

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 and primary HIDAM Index data bases without secondary indexes or logical relationships. All of the examples (Figure 1-10 through Figure 1-14) are based on either one or both of the data structures depicted below in Figure 1-9.



Payroll data structure

Skills inventory data structure

Figure 1-9. Payroll and Skills Inventory Data Structures

HSAM DBD GENERATION

The examples in Figure 1-10 (below) show the DBD generation control statements that define the skills inventory and payroll data stuctures 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 1-10. HSAM DBD Generations

HISAM DBD GENERATION

The examples in Figure 1-11 (below) 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,OVFIW=HISAMOVF,DEVICE=2314

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=2314

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 1-11. HISAM DBD Generations

HDAM DBD GENERATION

The examples in Figure 1-12 (below) 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 SKILLINV Data Base with Hierarchic Pointers

```
DBD     NAME=SKILLINV,ACCESS=HDAM,RMNAME=(RAMDMODL,1,500,824)
DATASET     DD1=SKILHDAM,DEVICE=2314,BLOCK=1648,SCAN=5

SEGM    NAME=SKILL,BYTES=31,FREQ=100,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,FREQ=500,PTR=H,PARENT=SKILL
FIELD   NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM    NAME=EXPR,BYTES=20,FREQ=10,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,FREQ=5,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=2314,BLOCK=1648,SCAN=5

SEGM    NAME=SKILL,BYTES=31,FREQ=100,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,FREQ=500,PTR=T,PARENT=((SKILL,SNGL))
FIELD   NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM    NAME=EXPR,BYTES=20,FREQ=10,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,FREQ=5,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 1-12. 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 1-13 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,ACCES=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 1-13. 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 1-14 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=2314,BLOCK=1648,SCAN=5

SEGM  NAME=SKILL,BYTES=31,FREQ=100,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,FREQ=500,PTR=H,PARENT=SKILL
FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM  NAME=EXPR,BYTES=20,FREQ=10,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,FREQ=5,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=2314,BLOCK=1648,SCAN=5

SEGM  NAME=SKILL,BYTES=31,FREQ=100,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,FREQ=500,PTR=T,PARENT=((SKILL,SNGL))
FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C

SEGM  NAME=EXPR,BYTES=20,FREQ=10,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,FREQ=5,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 1-14 (Part 1 of 2). HIDAM and Primary HIDAM Index DBD Generations

INDEX DBD Generation for HIDAM Data Base SKILLINV

```
DBD   NAME=INDEXDB,ACCESS=INDEX
DATASET DD1=INDXDB1,DEVICE=2314

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 1-14 (Part 2 of 2). HIDAM and Primary HIDAM Index DBD Generations

GSAM DBD GENERATION

The following example, Figure 1-15, 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 1-15. GSAM DBD Generations

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 1-16 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.

Defining the three logical relationships shown below in a physical data base or bases, allows the corresponding logical data base(s) to be defined.

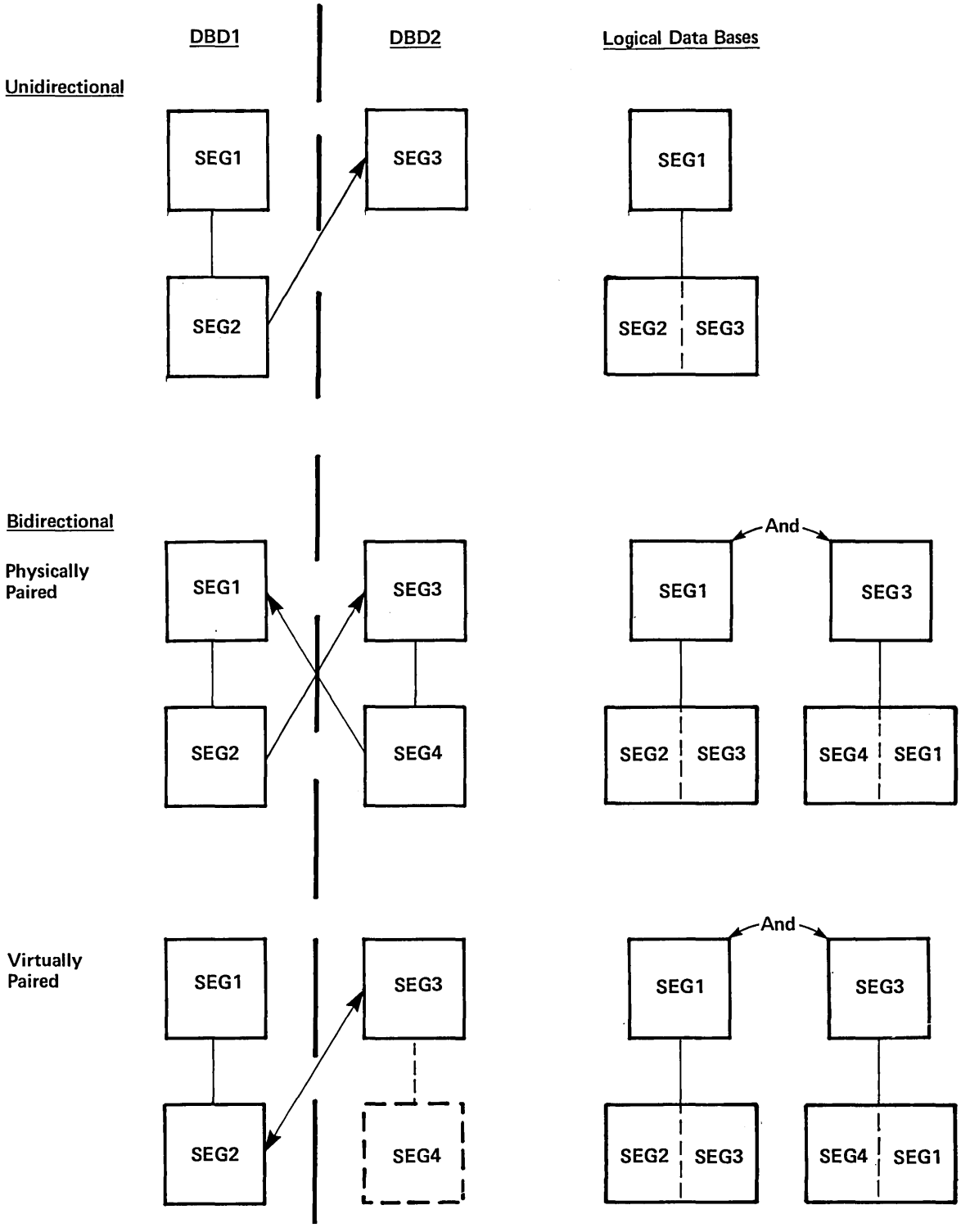
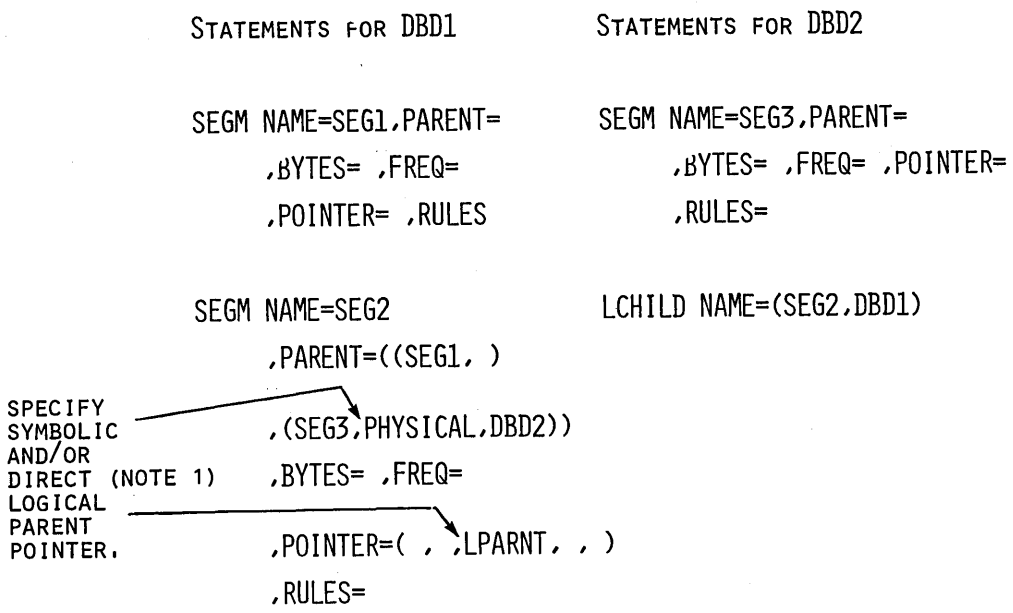


Figure 1-16 (Part 1 of 4). Summary of Logical Relationships

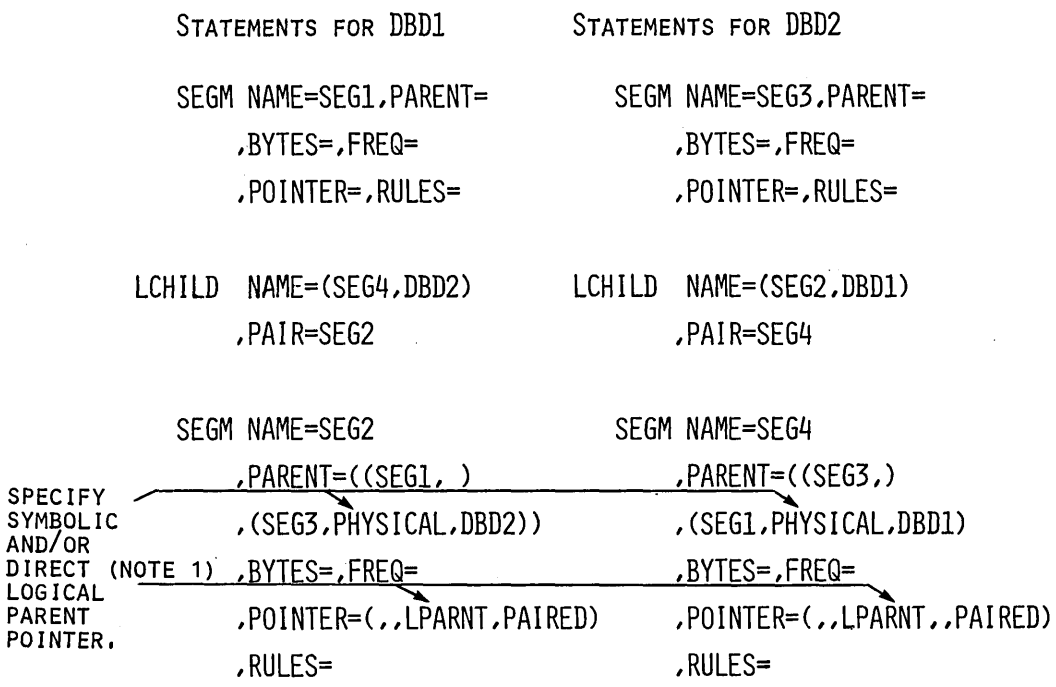
UNIDIRECTIONAL LOGICAL RELATIONSHIP



Note: The direct address pointer can be specified only when the logical parent is in an HDAM or HIDAM data base.

Figure 1-16 (Part 2 of 4). Summary of Logical Relationships

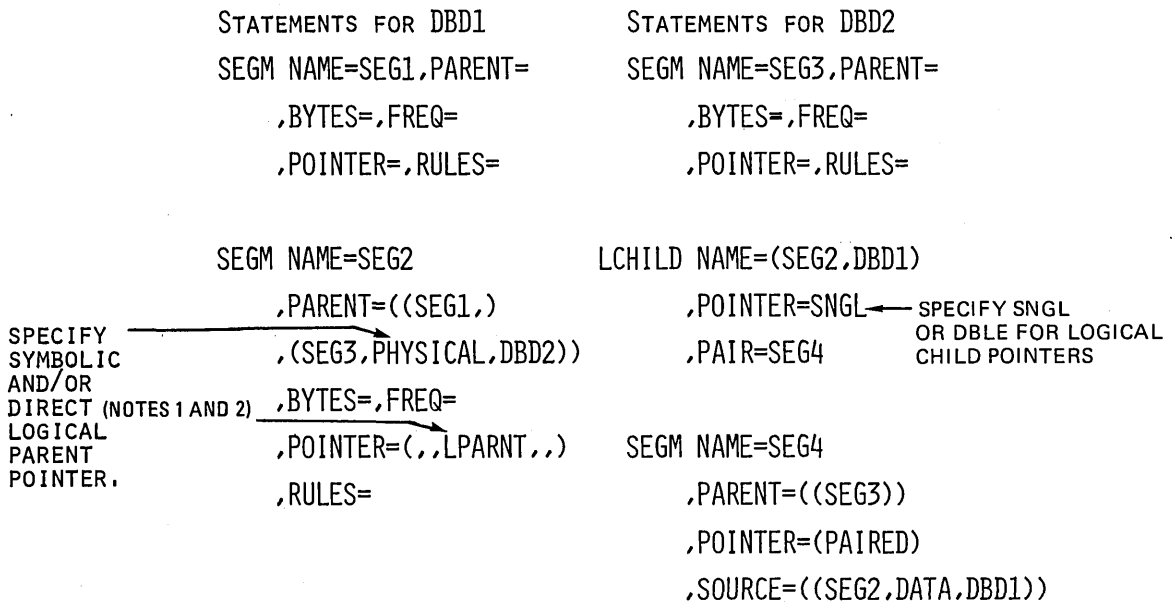
PHYSICAL PAIRED BIDIRECTIONAL LOGICAL RELATIONSHIP



Note: The direct address pointer can be specified only when the logical parent is in an HDAM or HIDAM data base.

Figure 1-16 (Part 3 of 4). Summary of Logical Relationships

VIRTUALLY PAIRED BIDIRECTIONAL LOGICAL RELATIONSHIP



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.

Figure 1-16 (Part 4 of 4). Summary of Logical Relationships

Figure 1-17 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 1-17.

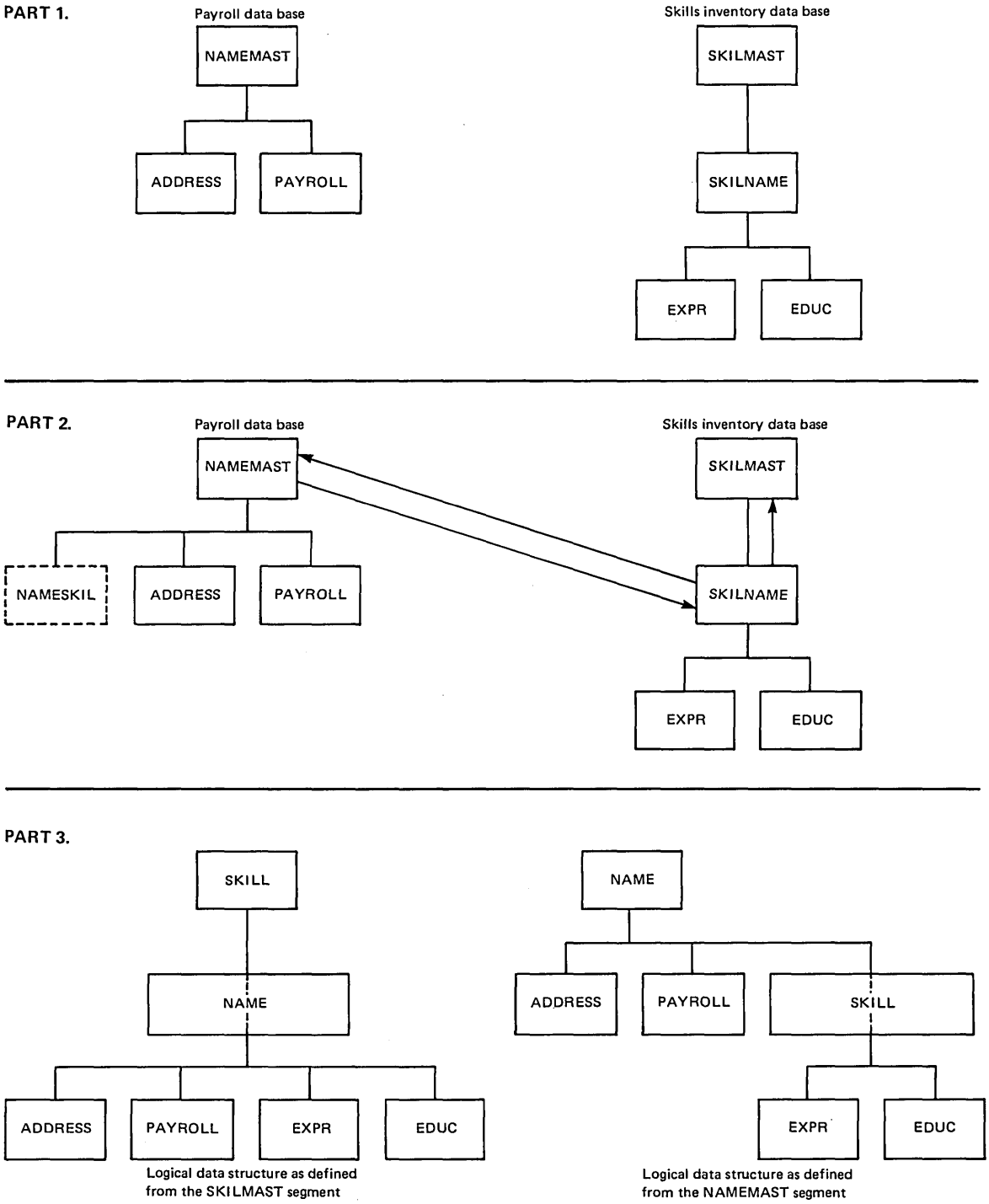


Figure 1-17. Logical Relationship between Physical Data Bases and the Resulting Logical Data Bases That Can Be Defined

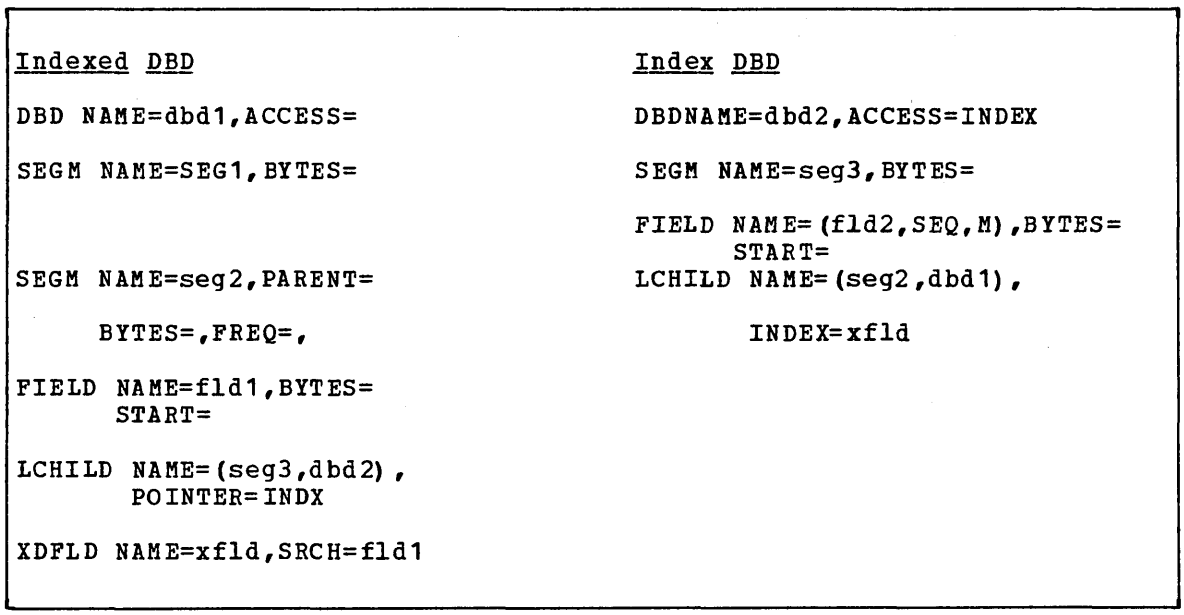
The following example (continued on the next page) shows the DBD generation statements necessary to define: 1) the payroll and skills inventory data structures depicted in Part 2 of Figure 1-17 as a HIDAM and HDAM data base with a virtually paired bidirectional logical relationship between the two data bases, and 2) the logical data structures depicted in Part 3 of Figure 1-17 as logical data bases.

```
DBD     NAME=PAYROLDB,ACCESS=HIDAM
DATASET DD1=PAYHIDAM,DEVICE=2314,BLOCK=1648,SCAN=3

SEGM    NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV),                    X
        BYTES=150,FREQ=1000
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,FREQ=2,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,FREQ=1,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=2314,BLOCK=1648,SCAN=5

SEGM  NAME=SKILMAST,BYTES=31,FREQ=1000,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,FREQ=500,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,FREQ=10,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,FREQ=5,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

Figures 1-18, 1-19, and 1-20 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 1-18 shows the statements required when the index target and index source segment types are the same. In Figure 1-19, the index target and index source segment types are different. Figure 1-20 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.



Indexed DBD	Index DBD
DBD NAME=dbd1,ACCESS=	DBDNAME=dbd2,ACCESS=INDEX
SEGM NAME=SEG1,BYTES=	SEGM NAME=seg3,BYTES=
	FIELD NAME=(fld2,SEQ,M),BYTES= START=
SEGM NAME=seg2,PARENT=	LCHILD NAME=(seg2,dbd1),
BYTES=,FREQ=,	INDEX=xfld
FIELD NAME=fld1,BYTES= START=	
LCHILD NAME=(seg3,dbd2), POINTER=INDX	
XDFLD NAME=xfld,SRCH=fld1	

Figure 1-18. Same Index Source and Target Segment Types

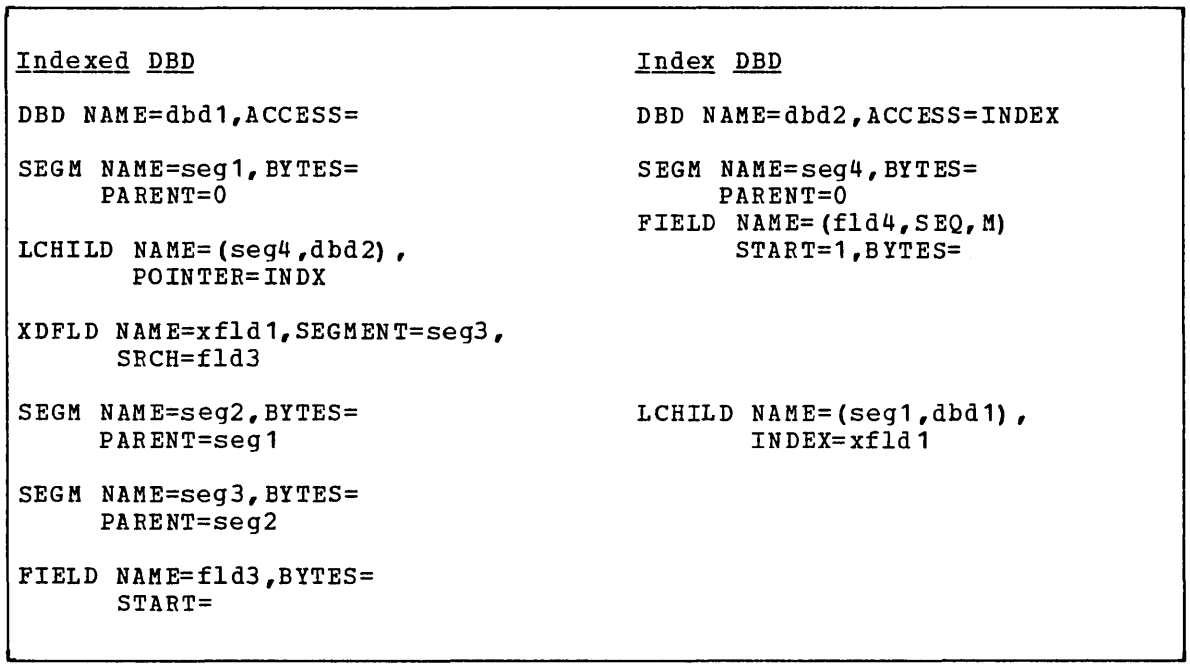


Figure 1-19. Different Index Source and Target Segment Types

```
Indexed DBD                              Index DBD

DBD NAME=dbd1,ACCESS=                    DBD NAME=(dbd2,dbd3),ACCESS=INDEX

SEGM NAME=seg1,BYTES=                    SEGM NAME=seg3,BYTES=
     PARENT=0                                 PARENT=0

FIELD NAME=fld1,BYTES=                   FIELD NAME=(fld3,SEQ,u),
      START=                                   START=1,BYTES=
FIELD NAME=fld2,BYTES=
      START=

LCHILD NAME=(seg3,dbd2),                 LCHILD NAME=(seg1,dbd1),
       POINTER=INDX                             INDEX=xfld1

XDFLD NAME=xfld1,SRCH=fld2,
      CONST=C'2',

SEGM NAME=seg2,BYTES=                    SEGM NAME=seg5,BYTES=
     PARENT=seg1                              PARENT=0

FIELD NAME=fld4,BYTES=                   FIELD NAME=(fld10,SEQ,u),
      START=                                   START=1,BYTES=

LCHILD NAME=(seg5,dbd3)                  LCHILD NAME=(seg2,dbd1),
       POINTER=INDX                             INDEX=xfld2

XDFLD NAME=xfld2,SRCH=fld4
      CONST=C'1',
```

Figure 1-20. Shared Secondary Index Data Base DBD Generation

DBDGEN FOR A SHARED SECONDARY INDEX DATA BASE

Figure 1-21 shows a data base indexed by three secondary indexes in a shared secondary index data base. Figure 1-22 shows the DBD generation statements that define the indexed data base, the primary index data base, and the secondary index data base.

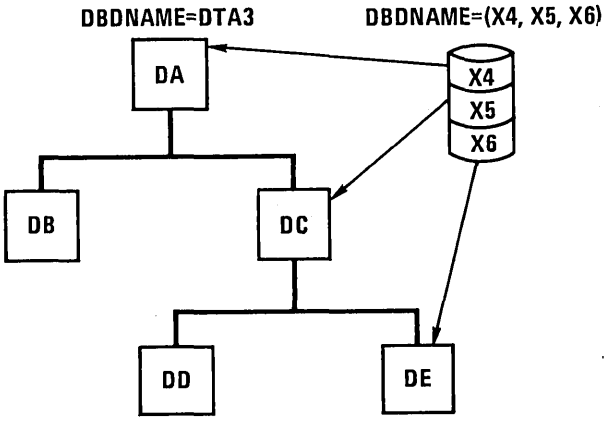


Figure 1-21. Data Indexed by Three Secondary Indexes in a Shared Secondary Index Data Base

DBDGEN for Indexed Data Base

```
DBD     NAME=DTA3,ACCESS=HIDAM
DATASET DD1=D1,DEVICE=2314
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=2314
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=2314,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 1-22. 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 PCBs (Program Communication Blocks) 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 or batch program. The name of the PSB and its associated application program must be the same in a teleprocessing system.

IMS/VS and IMS/360 Version 2 PSB generation statements are compatible. This allows the user to take an IMS/360 Version 2 PSB generation control card deck, input the deck to IMS/VS PSB generation, and create an IMS/VS PSB.

PSB generation places the created PSB in the PSB library. Each PSB is a member of the operating system partitioned data set IMSVS.PSBLIB. At execution time of IMS/VS, either in an online control or a batch processing region, the necessary PSBs associated with message or batch programs to be executed are loaded into main storage and used by DL/I.

PSB RULES

There are four 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 programs and in batch programs which interface with the IMS/VS message queues.
- PCB statements for DL/I data bases. These are used by message and batch 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 and batch processing programs to define logical data structures.
- 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. Therefore, 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.

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 excuting 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, the reader should refer to the IMS/VS Application Programming Reference Manual.

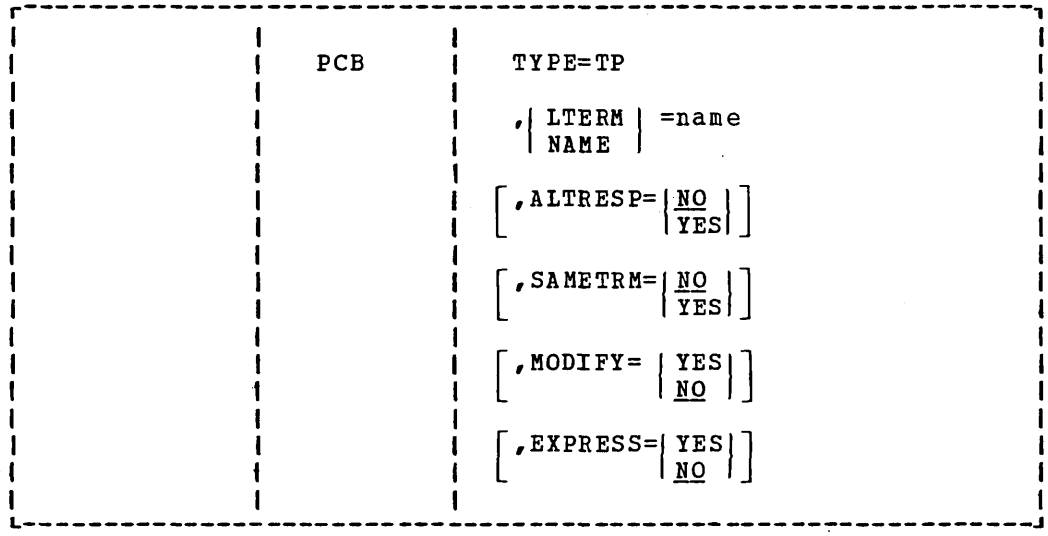
PSB CONTROL STATEMENT FORMAT

To reiterate, no PCB statement is needed in PSB generation for the I/O PCB: IMS/VS builds it automatically. This is true for both message processing application programs and 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. 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. Only message processing programs and batch message processing programs may normally have alternate PCB statements in their associated PSBs.

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.

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, 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 abend.

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 since the transaction that caused the message will be reprocessed.

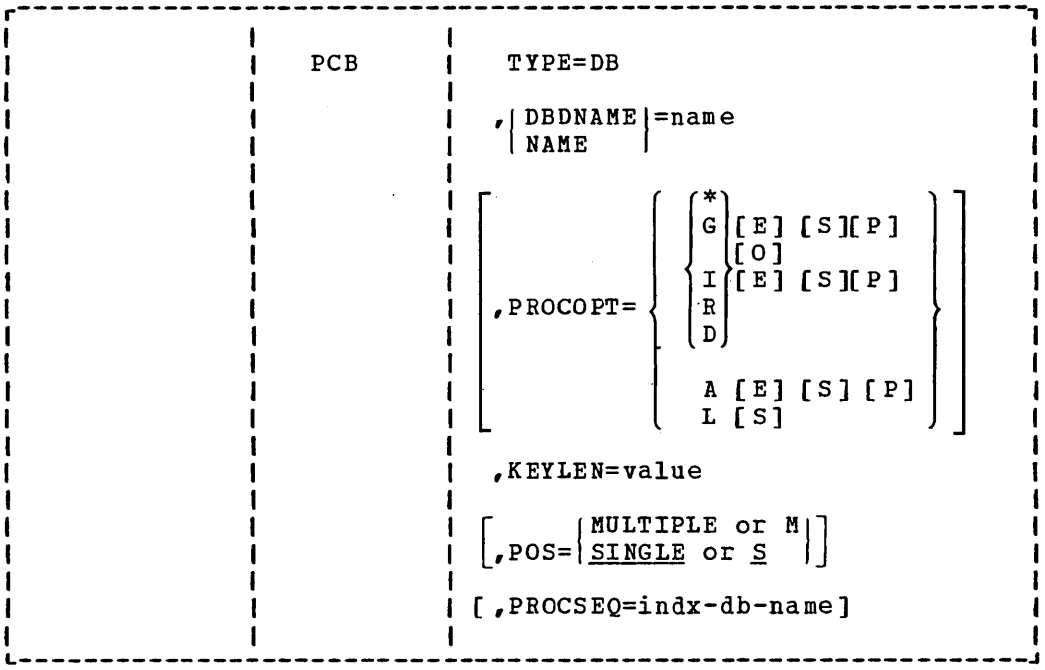
DL/I DATA BASE PCB STATEMENT

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 maximum value for application programs executing in all IMS region types (MSG, DL/I, etc.).

If an existing user PSB contains the maximum number of PCBs that can be defined in a PSBGEN, from two to four PCBs must be removed if the PSB is used as input to the IQF utility programs -- one PCB each for the System Data Base, Phrase Data Base, short Index Data Base (if defined), and long Index Data Base (if defined).

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



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

where:

TYPE=
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. The reader should refer to the SENSEG card description presented later in this chapter.

The keywords DBDNAME and NAME are synonomous. 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.

D - Delete function.

A - All, includes the above four functions.

P - Required if command code D is to be used on get type calls or insert calls. Determines maximum length of the I/O area.

O - Read only; do not enqueue to check availability. Must be specified as GO or GOP only.

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

An U801 ABEND can occur when you use the 'O' option for PCBs on dynamic backout by program isolation if the data base has had any inserts or deletes by another user.

An U801 ABEND can also 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.

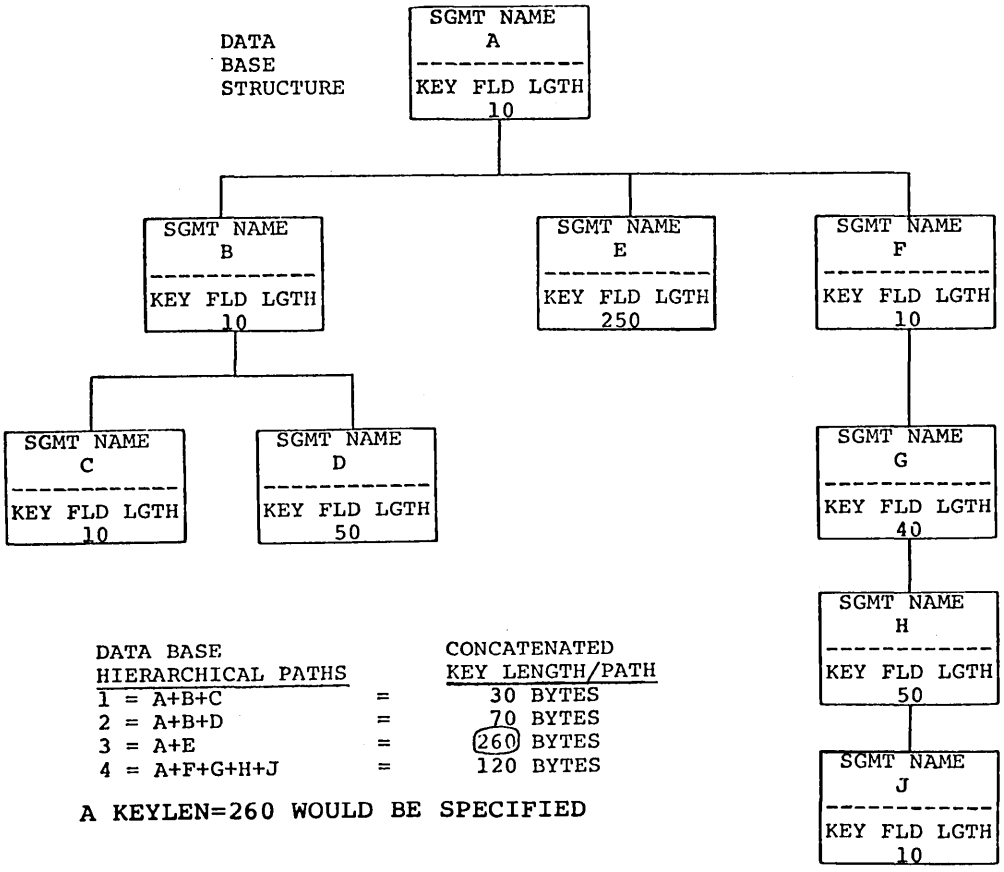
Notes:

1. If any PCBs in the PSB have PROCOPT of L or LS and reference HISAM, HIDAM, or INDEX data bases explicitly or implicitly, 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.

2. 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.

3. If the 'O' option is used for a PCB, and any SENSEG statement in the PCB has an update (I, R, D, or A) PROCOPT, the 'O' option is ignored and a warning message is produced.

KEYLEN=
is the value specified in bytes of the longest concatenated key for a hierarchical path of sensitive segments used by the application program in the logical data structure. The following example explains the definition of KEYLEN:



DATA BASE HIERARCHICAL PATHS		CONCATENATED KEY LENGTH/PATH
1 = A+B+C	=	30 BYTES
2 = A+B+D	=	70 BYTES
3 = A+E	=	260 BYTES
4 = A+F+G+H+J	=	120 BYTES

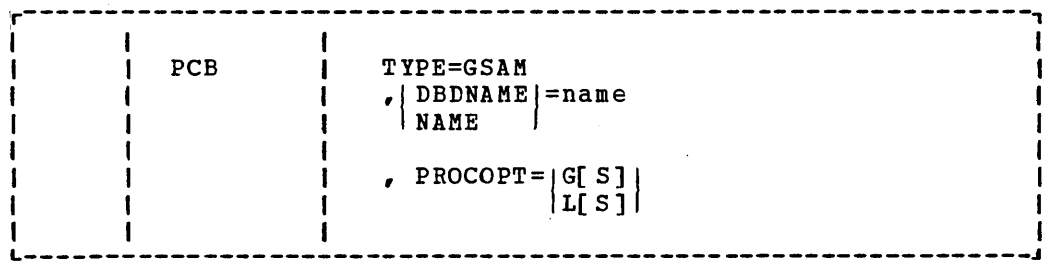
A KEYLEN=260 WOULD BE SPECIFIED

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 the IMS/VS Application Programming Reference Manual. The performance variation between single and multiple positioning is insignificant. Multiple positioning is not supported by HSAM.

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. The operand is invalid if PROCOPT is 'L' or 'LS'.

The format for the GSAM data base PCB statement is:



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 characters defined below:

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 convention is:

TP PCBs - first
DB PCBs - second
GSAM PCBs - last

SENSEG STATEMENT -- SENSITIVE SEGMENTS

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.

The format of the SENSEG statement is:

```
SENSEG   NAME=name
        ,PARENT=name
                0
                          *
                          G
                          I   [E][S][P]
        ,PROCOPT=         R
                          D

                          A [E][S][P]
                          K
        [,INDICES=list1]
```

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

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 zero. 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. One additional option can be used on the SENSEG statement, however, which does not apply to the PCB statement. A PROCOPT of K indicates key sensitivity only. The segment is not moved to the user's I/O area; only the key is placed in the concatenated 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, SENSEG PROCOPT

overrides the PCB PROCOPT. If PROCOPT= L or LS is specified on the preceding PCB statement, this operand must be omitted.

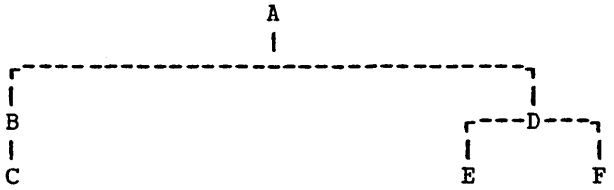
Do not specify a SENSEG statement for a virtual logical child segment type if PROCOPT= L or LS is specified. The Replace or Delete functions also imply the Get function.

If key sensitivity is specified for a segment, the sequence field of the segment must not be used as an IQF field.

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. For list1, up to 32 DBD names of secondary indexes can be specified. If two or more names are specified, names must be separated by commas and the list enclosed in parentheses.

The order in which SENSEG cards are sequenced after a PCB statement must follow the hierarchical order of segment types defined in DBDGEN, or for secondary indexing, the sequence of SENSEG statements reflects the secondary processing sequence of segment types in a data base.

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
```

PSBGEN STATEMENT

The fourth type of statement required for a PSB generation is one that specifies characteristics of the application program. The format for this statement is:

```
┌──────────┬──────────┬──────────────────────────────────────┐
│          │          │  PSBNAME=name                        │
│          │          │                                      │
│          │          │       {COBOL                         │
│          │ PSBGEN   │ ,LANG=  PL/I                         │
│          │          │         ASSEM}                       │
│          │          │        {0                            │
│          │          │ [,MAXQ= nr}]                         │
│          │          │                                      │
│          │          │ [,CMPAT= {YES                        │
│          │          │          NO} ]                       │
│          │          │                                      │
│          │          │  [,IOASIZE=value]                    │
│          │          │                                      │
│          │          │  [,SSASIZE=value]                    │
│          │          │                                      │
│          │          │  [,IOEROPN=n or                      │
│          │          │            (n,WTOR) ]                │
└──────────┴──────────┴──────────────────────────────────────┘
```

where:

PSBNAME=name
: is the parameter keyword for the alphameric name of this PSB. The name value for the PSBNAME must be eight characters or less 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 value specified is in bytes, with a maximum of 256,000.

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 256,000.

IOEROPN=
is 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 parantheses are required. If you code only IOEROPN=n, parentheses are not required.

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 four 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.

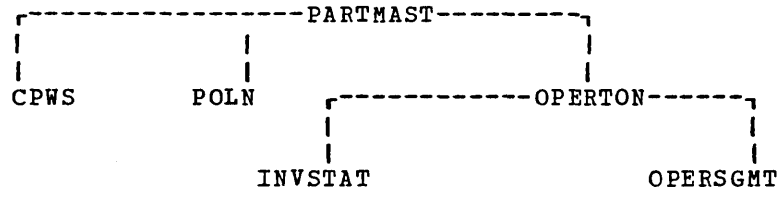
```
//PSBGEN   JOB  MSGLEVEL=1
//         EXEC    PSBGEN,MBR=
//C.SYSIN  DD   *


        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 a "BUILD PSB" operation.

PSB GENERATION EXAMPLES

A PSB generation is to be done for a message processing program to process the following hierarchical data structure. Output messages are to be transmitted to logical terminals OUTPUT1 and OUTPUT2 in addition to the terminal representing the source of input.



SAMPLE 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
```

Sample 2 shows these statements being used for a batch program, where programs using this PSB do not reference the teleprocessing PCBs in the batch environment.

SAMPLE 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
```

Sample 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.

SAMPLE 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
```

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.

ASSEMBLY LISTING

An operating system assembly language listing of the PSB created by PSB generation execution is provided, except when PRINT NOGEN is specified.

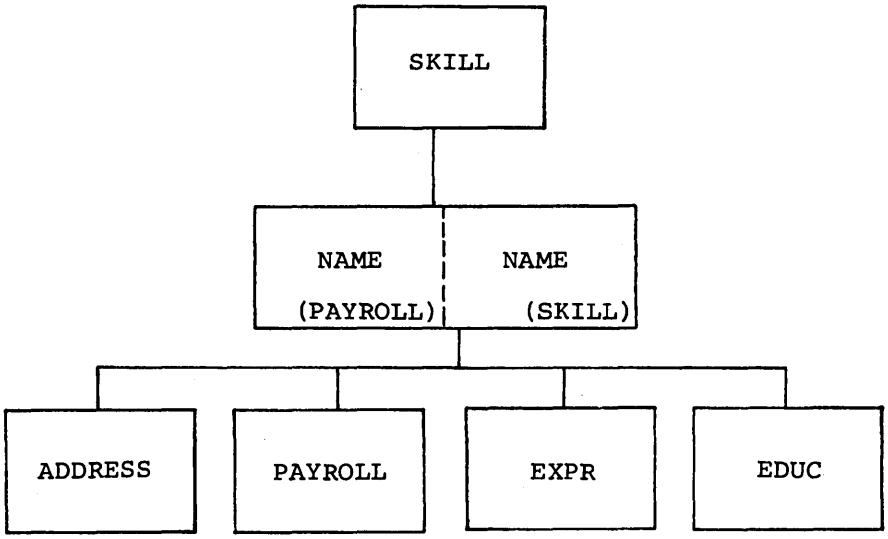
LOAD MODULE

PSB generation is a two-step operating system job. Step 1 is a macro assembly execution which 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.

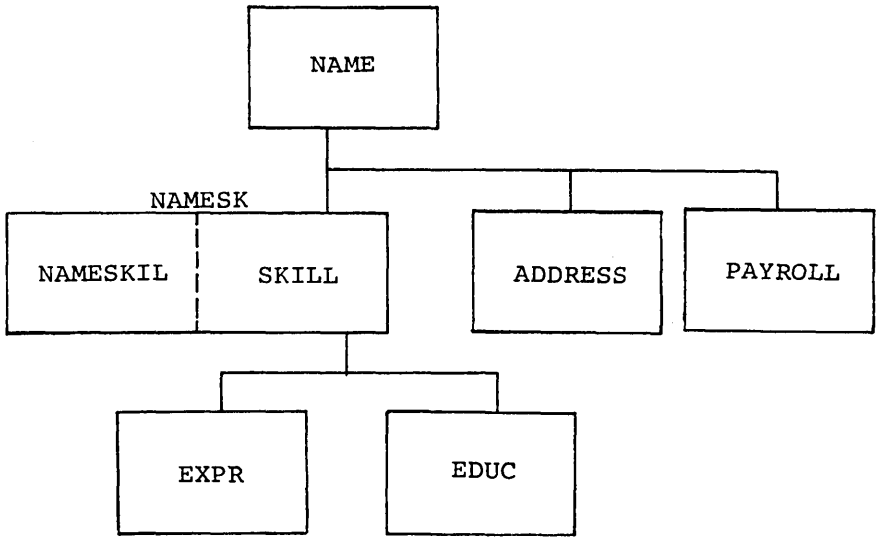
PSB GENERATION ERROR CONDITIONS

The PSB error conditions are contained in the IMS/VS Messages and Codes Reference Manual.

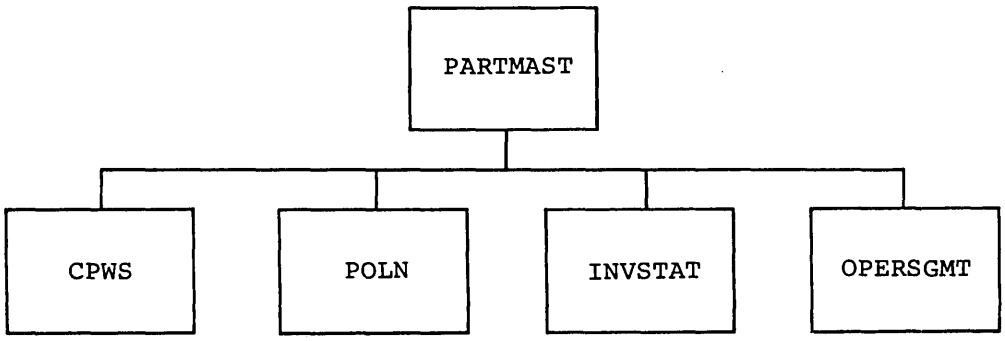
LOGICAL DATA BASE




```
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
```

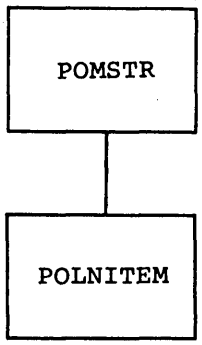



```
PCB         TYPE=DB,DBDNAME=LOGICDB,PROCOPT=A,KEYLEN=241,POS=M
SENSEG      NAME=NAME,PARENT=0,PROCOPT=G
SENSEG      NAME=NAMESK,PARENT=NAME,PROCOPT=G
SENSEG      NAME=EXPR,PARENT=NAMESK,PROCOPT=G
SENSEG      NAME=EDUC,PARENT=NAMESK,PROCOPT=G
SENSEG      NAME=ADDRESS,PARENT=NAME,PROCOPT=G
SENSEG      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
```

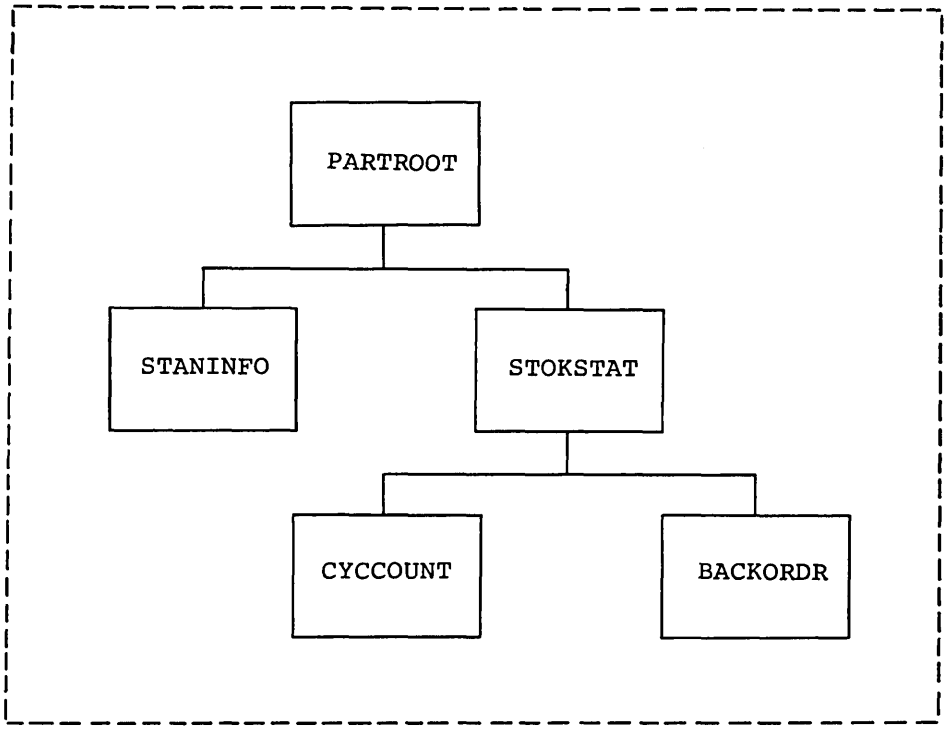
EXAMPLE: PODR (Purchase Order) DATA BASE




```
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

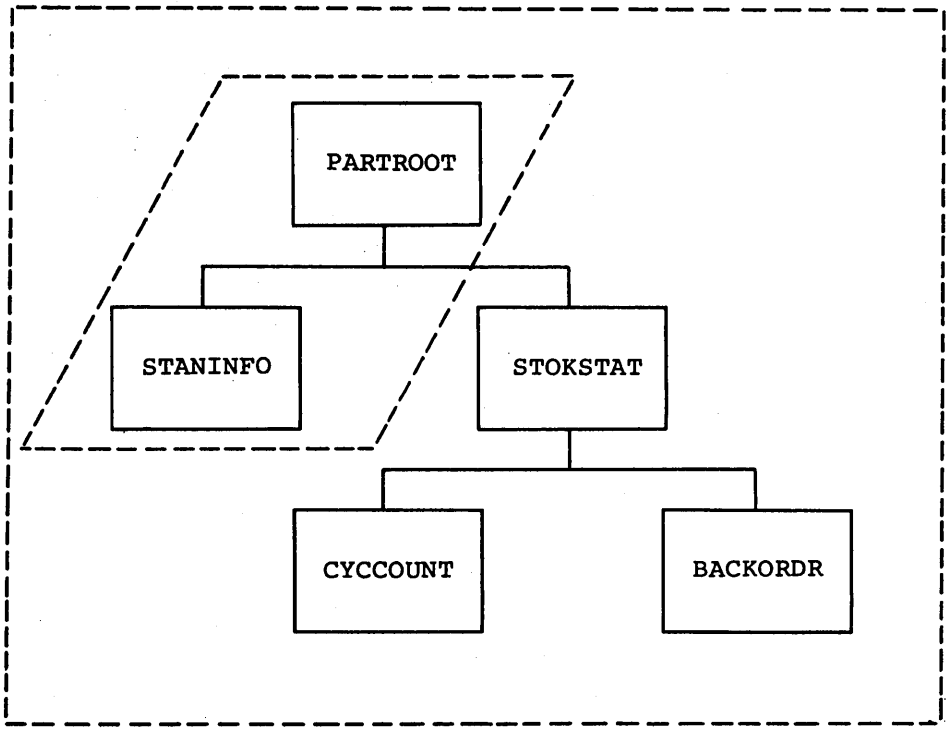
(DBDNAME=DI21PART)




```
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=L,KEYLEN=43
SENSEG      NAME=PARTROOT,PARENT=0
SENSEG      NAME=STANINFO,PARENT=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
SENSEG      NAME=CYCCOUNT,PARENT=STOKSTAT
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM01
END
```

EXAMPLE 2: SAMPLE PROBLEM - APPLICATION DATA BASE

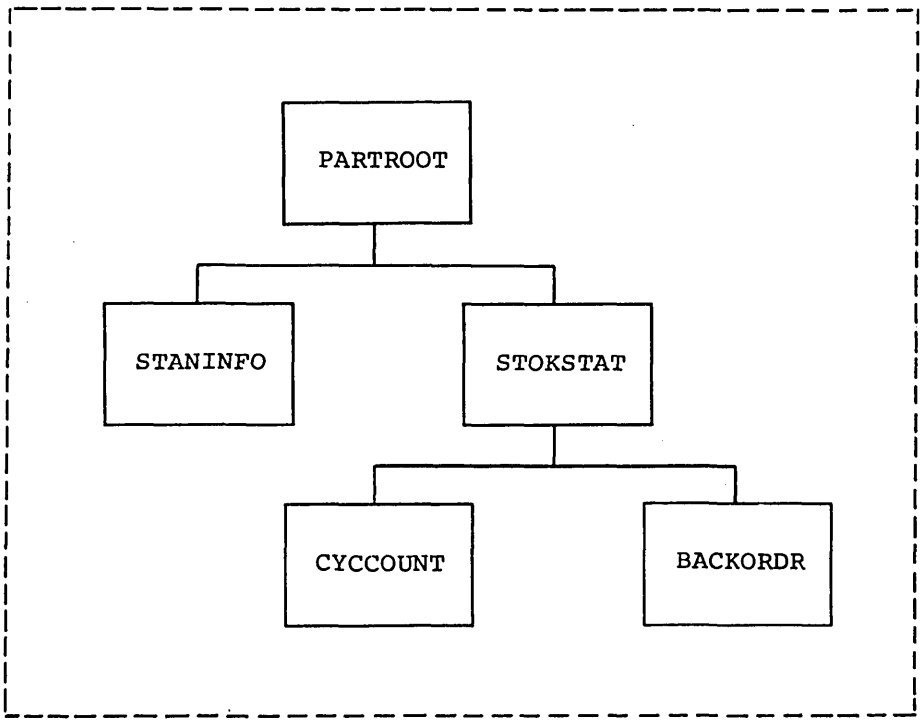
(DBDNAME=DI21PART)




```
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

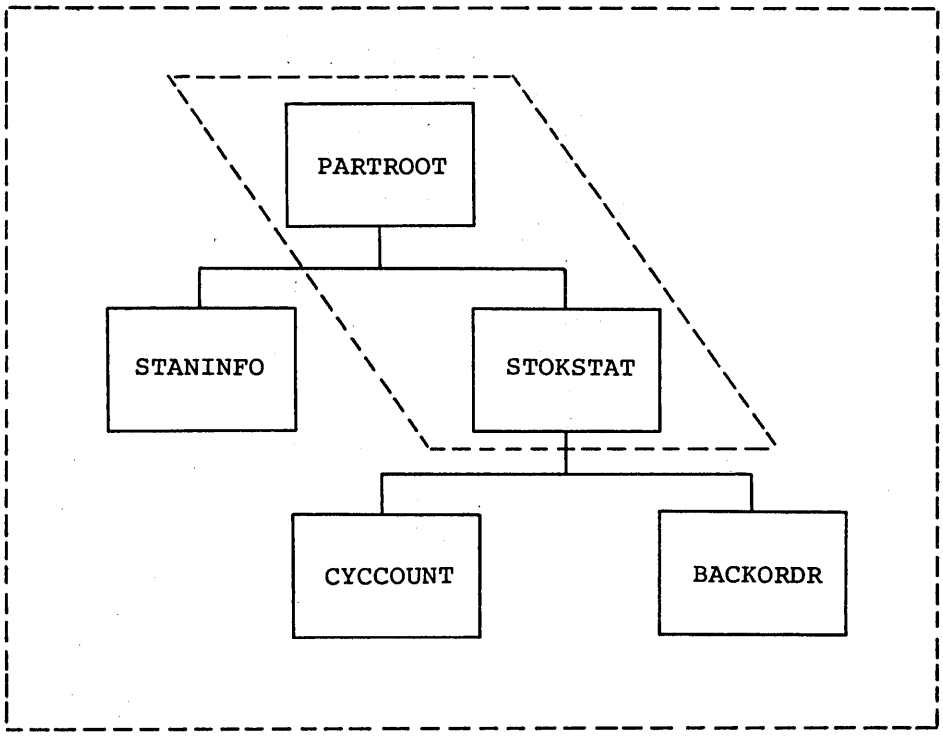
(DBDNAME=DI21PART)




```
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

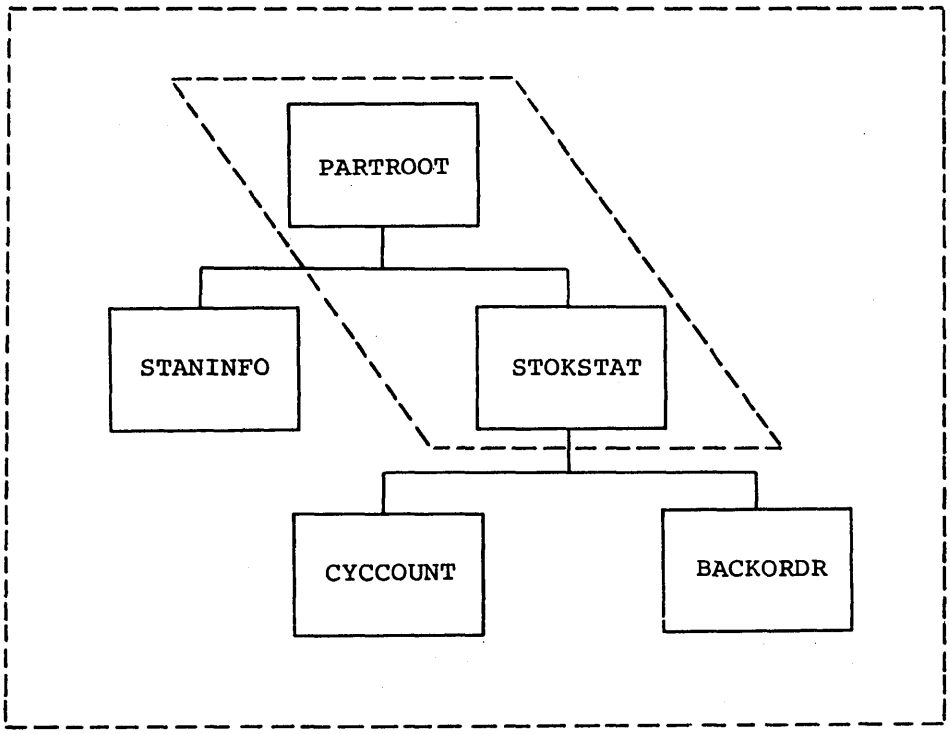
(DBDNAME=DI21PART)




```
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

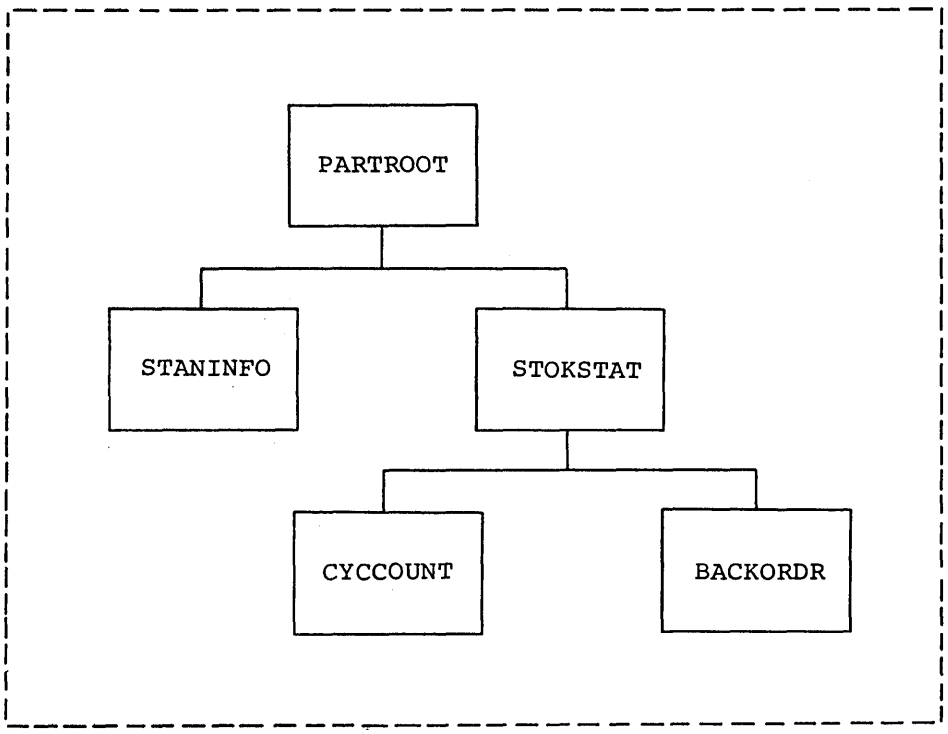
(DBDNAME=DI21PART)




```
   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

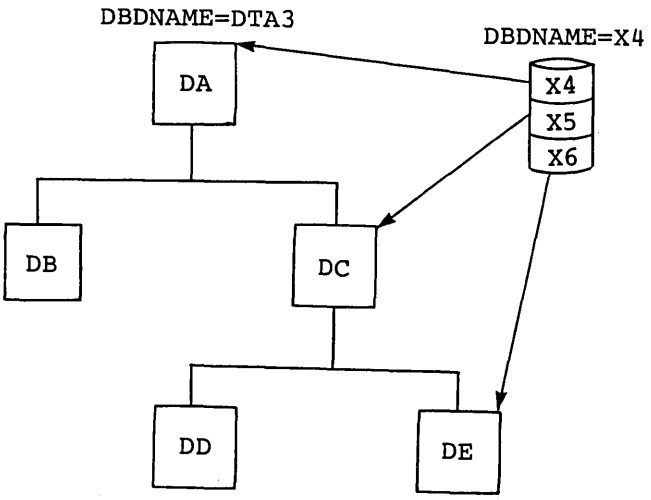
(DBDNAME=DI21PART)



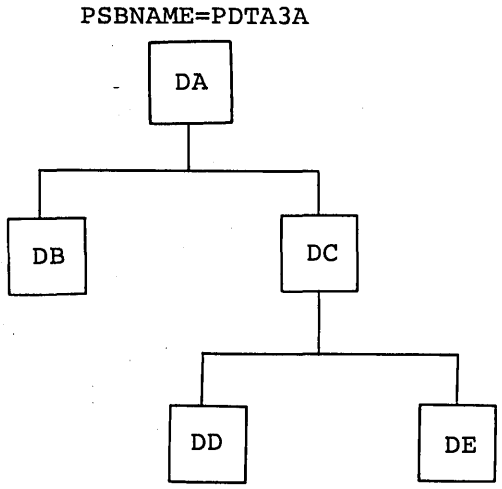

```
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=43
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
SENSEG      NAME=CYCCOUNT,PARENT=STOKSTAT
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM07
END
```

EXAMPLE 7: SHARED SECONDARY INDEX

A. Data base structure

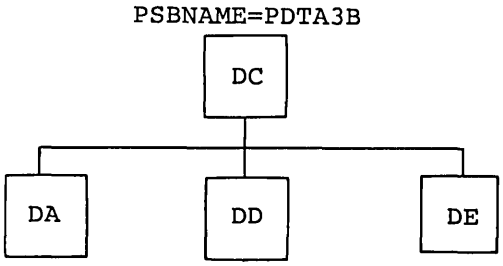


B. Data base structure for index through DA



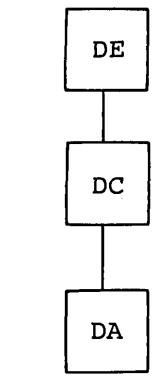

```
PCB TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X4
    SENSEG  NAME=DA,PARENT=0
    SENSEG  NAME=DB,PARENT=DA
    SENSEG  NAME=DC,PARENT=DA,INDICES=X5
    SENSEG  NAME=DD,PARENT=DC
    SENSEG  NAME=DE,PARENT=DC,INDICES=X6
PSBGEN LANG=COBOL,PSBNAME=PDTA3A
END
```

C. Data base structure through index for 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
```

D. Data base structure through index for 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
```

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 pre-building 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 main storage than building the ACBs dynamically from PSBLIB and DBDLIB.

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 cards, 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 which require that the associated PSB or DBD be changed, these changes must be made before the utility is run.

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 be executed using only 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.

The diagram in Figure 3-1 shows the functional relationship of the input/output data sets and their naming requirements.

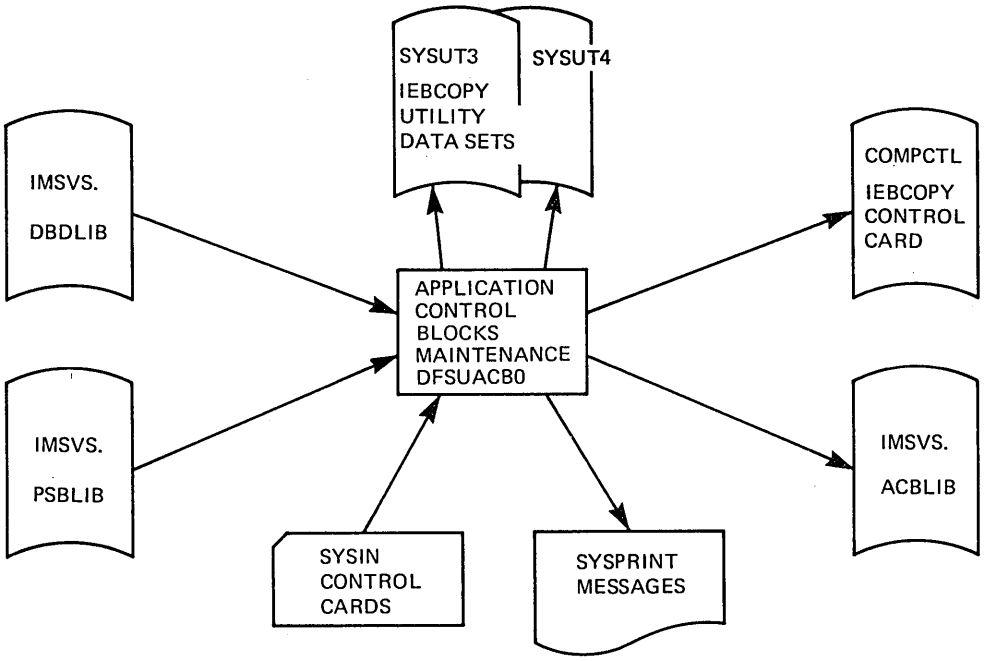


Figure 3-1. Application Control Blocks Maintenance Utility

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 100-150K.

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	Defines the partitioned data set named IMSVS.RESLIB. (See examples 1, 2, and 3 in this chapter.)

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 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 the OS/VS Utilities, GC35-0005, 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. Systems such as HASP convert input stream data to reflect a unit record type device to the user program. This prohibits reread of the data set. 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 card data set. 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 1-71. 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 1-15 of the continuing statement must be blank.

```
Col
 1  Operation            Operand

    BUILD        { [PSB= {ALL            } ] }
                 {       {(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 scratched and their space made available for reuse. (It should be noted that DBDs are <u>not</u> scratched. An OS utility must be run to scratch a DBD.) 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. 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. Since 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 card. 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 be referenced in a BUILD DBD statement. This is the only way the DBD can be replaced in the IMSVS.ACBLIB without doing a BUILD PSB=ALL into a newly allocated data set.

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.

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

Example 1: This example creates blocks for all PSBs which currently reside in IMSVS.PSBLIB. All blocks currently existing in IMSVS.ACBLIB are scratched and their 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 DISPosition of 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
//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
//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 which 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 not 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
//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
/*
```

PART 2. DATA BASE UTILITIES

Part 2 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 4, "Data Base Reorganization/Load Processing," describes the utilities used for 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 5, "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 6, "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 4. DATA BASE REORGANIZATION/LOAD PROCESSING

This chapter describes the eight IMS/VS utility programs that are used for reorganization/load processing of data bases. An overview is presented initially to provide a brief description of each utility. The following topics are also presented early in this chapter:

- 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.

OVERVIEW

The eight utility programs involved in data base reorganization/load processing are:

1. HISAM Reorganization Unload (DFSURUL0)
2. HISAM Reorganization Reload (DFSURRL0)
3. HD Reorganization Unload (DFSURGU0)
4. HD Reorganization Reload (DFSURGL0)
5. Data Base Prereorganization (DFSURPR0)
6. Data Base Scan (DFSURGS0)
7. Data Base Prefix Resolution (DFSURG10)
8. Data Base Prefix Update (DFSURGP0)

These utilities are of two types -- physical reorganization utilities and logical relationship resolution utilities.

PHYSICAL REORGANIZATION UTILITY PROGRAMS

There are four physical reorganization utility programs: (1) HISAM Reorganization Unload, (2) HISAM Reorganization Reload, (3) HD Reorganization Unload, and (4) HD Reorganization Reload. 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 listed 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 strucutral 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.

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: 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 Unload utility.

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 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 the user-provided program that performs 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. He must provide a program for that purpose and appropriate PSBs to indicate that the data base is being loaded. Refer to the PSBGEN 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.

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. See 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, there are two options available to the user regarding insertion of the paired segments into their data base(s). The paired segments may be inserted either at initial data base load or after initial load when operating in update mode. (Additional information on the use of these options is provided in the IMS/VS System/Application Design Guide.)

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 4-2 for the utilities that must be executed and the order in which they must be run.

3. Work data sets created during initial loads and reorganizations must be concatenated as input to the Prefix Resolution utility.

4. The use of non-unique sequence fields for one or more segments in a data base permits the occurrence of non-unique concatenated keys for one or more segment types. If the user defines non-unique sequence fields for segments in a data base, and if he permits occurrences of a given segment type 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 non-unique 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 non-unique 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 non-unique concatenated keys, he may insert the logical children at other than initial data base load time. In this manner, logical parents with non-unique concatenated keys can each own logical children. However, if logical children that have a logical parent with a non-unique 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

non-unique 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

Since 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 LISTVOC 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 since 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.

REORGANIZATION/LOAD FLOWCHART

The flowchart depicted in Figure 4-1 and the accompanying description explain how the user may determine the necessary utility programs required for reorganization and/or load processing of a data base.

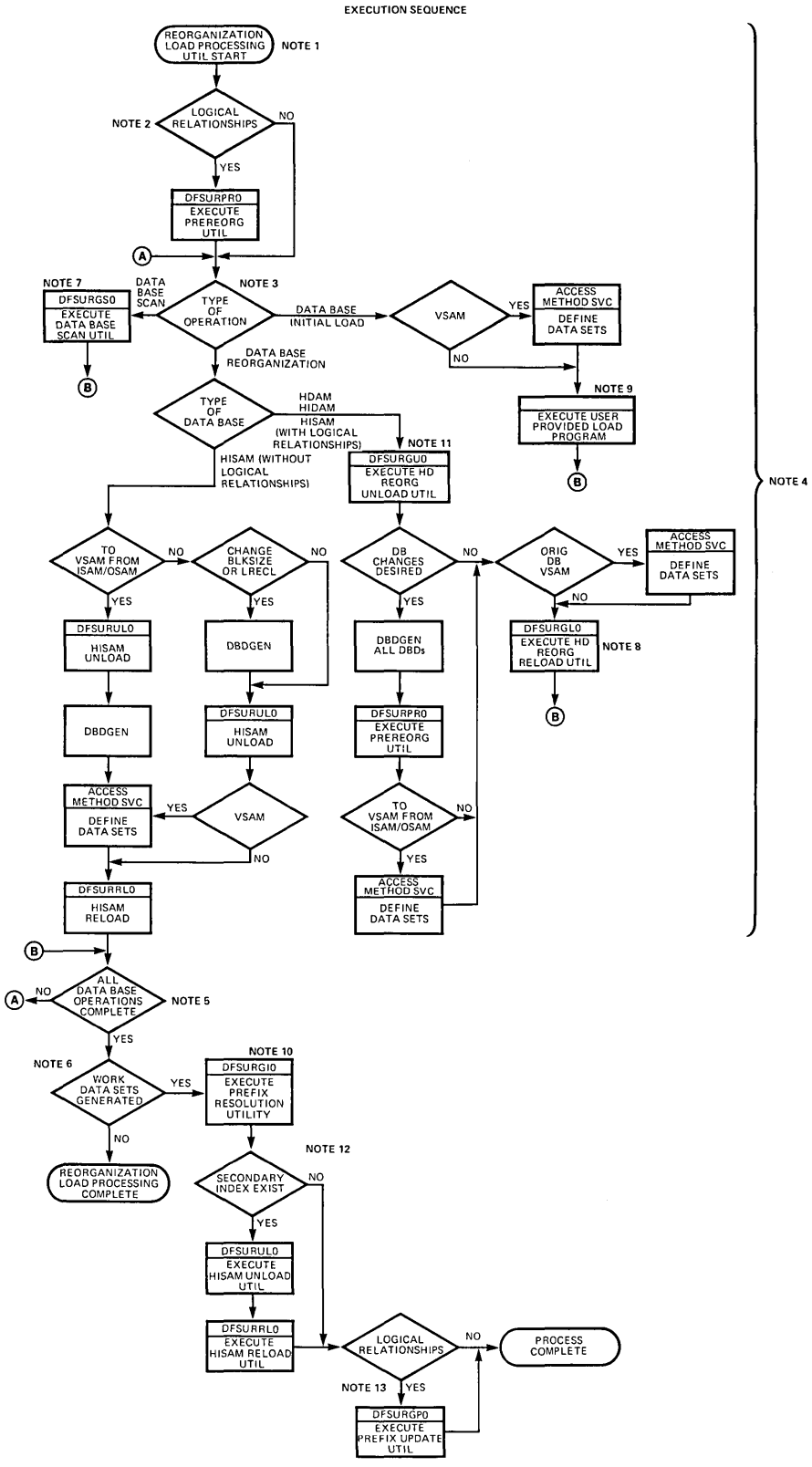


Figure 4-1. Data Base Reorganization/Load Flowchart

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 load reorganized, or scanned.

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 recommended that the DB Scan utility be run before a data base is 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 his 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 his 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 output from the Data Base Scan utility, the HD Reorganization Reload utility, and the user's initial data base load program 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.

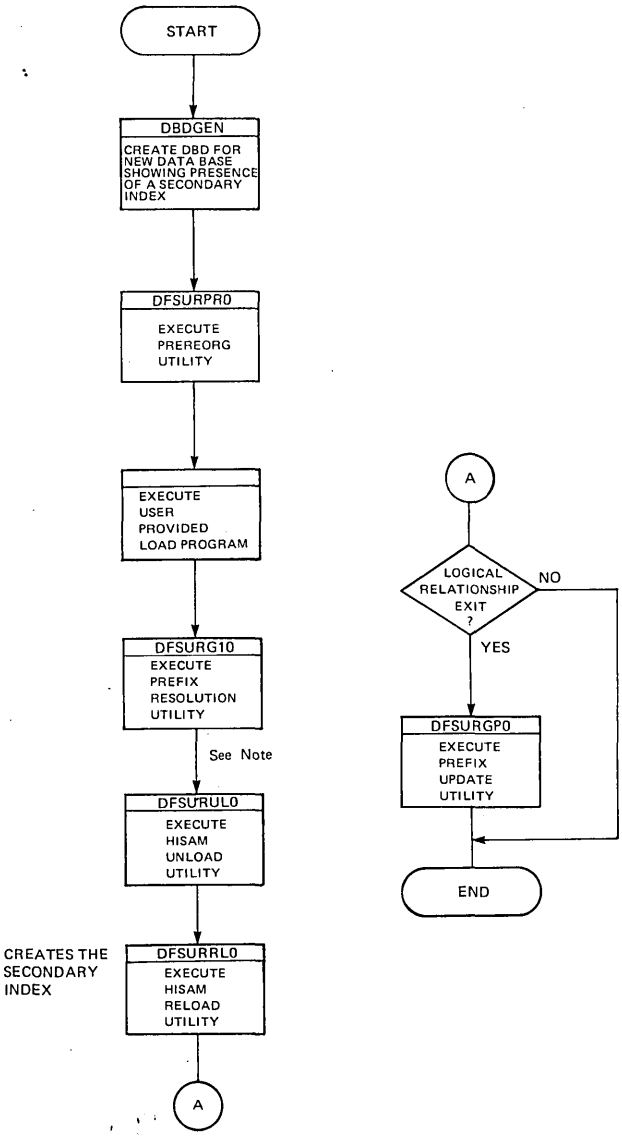
11. This path must be taken for HISAM data bases with logical relationships or if structural changes are required (for example, HISAM to HDAM, pointer changes, additional segments, or adding a secondary index).

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. The prefix update utility must be run if a data base contains physical pairing with symbolic pointers.

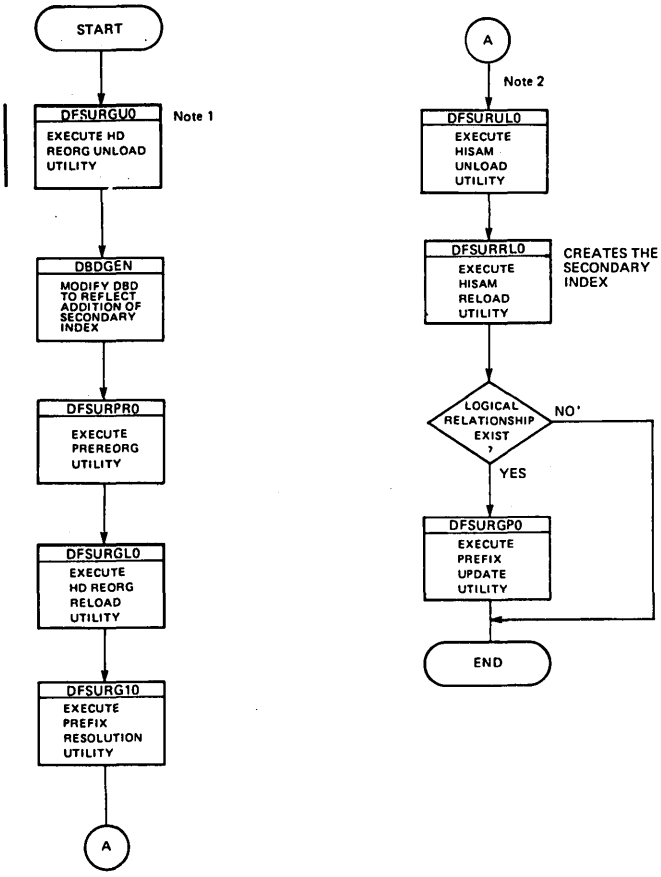
LOADING A SECONDARY INDEX

The flowcharts in Figures 4-2 and 4-3 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, or to merge the new information into an existing shared secondary index, or to replace a member in an existing shared secondary index.

Figure 4-2. 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 4-3. 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 which can be used as input to either the Data Base Recovery utility or the HISAM Reorganization Reload utility. In addition, this utility performs the function of formatting 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 blocksize of the output device, up to a maximum of 16K. Since 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. 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 log tapes created between unload and reload will refer to the old location and recovery is impossible.

2. 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.

3. 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.)

4. To change block or logical record lengths of VSAM data sets, a new DBD must 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.

5. 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.

6. 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 where 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 plus 1 byte to allow for the larger dependent pointers in VSAM.

7. 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.)

A flow diagram of the HISAM Reorganization Unload utility is shown in Figure 4-4.

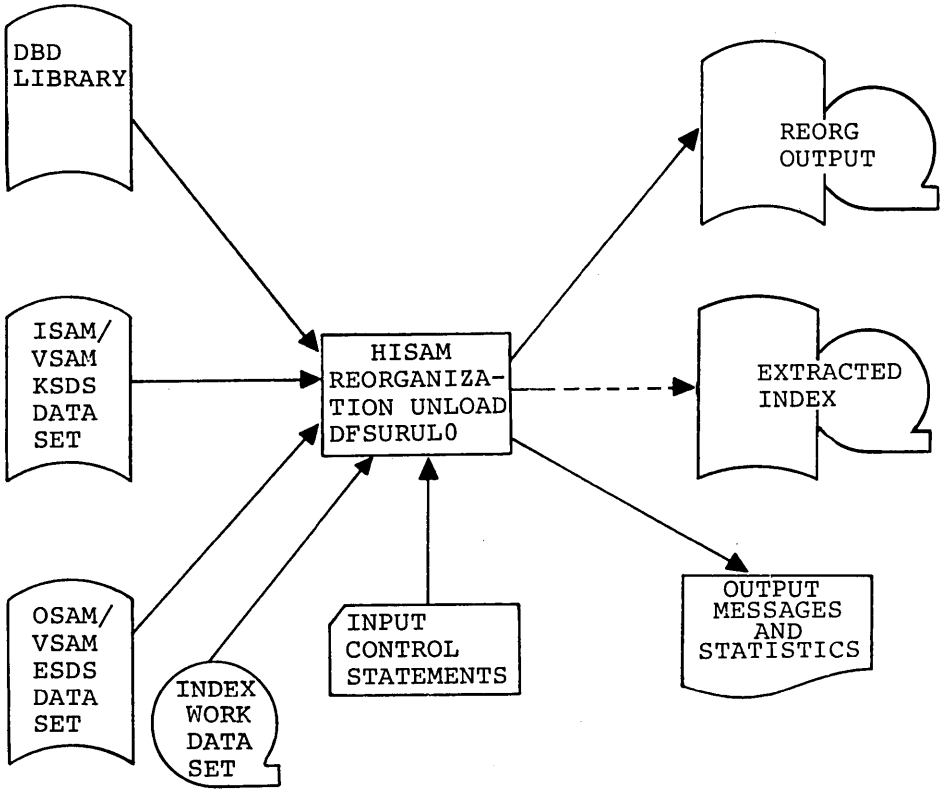


Figure 4-4. HISAM Reorganization Unload Utility

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' The normal IMS/VS positional parameters such as SPIE and BUF can follow the program name in the PARM field.
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 through the output stream.
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 ISAM/VSAM KSDS 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 ISAM/VSAM KSDS data set to be reorganized.
osamin DD	Defines the OSAM/VSAM ESDS 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 OSAM/VSAM ESDS 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, at least 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 which 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 if the data bases described by the isamin and osamin DD statements are VSAM data sets. (For additional information, see the discussion on "Defining the IMS/VS VSAM Buffer Pool" in the 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.

UTILITY CONTROL STATEMENTS

```
1  2 3 4          13          22       31       40       49  50          80
┌──────────────────────────────────────────────────────────────────────────┐
│ {R} { } {1} {M} dbdname    prime-     out1-    out2-    index-  [c][comments] │
│ {X} { } {2} {E}            ddname     ddname   ddname   ddname                │
│             {R}                                                          │
└──────────────────────────────────────────────────────────────────────────┘
```

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 2, depending on the number of output copies required. There is no default, and 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 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. E=EXTRACT 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 Prefix Resolution.

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. 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 functions are specified in position 3 of this statement.

50 Comments can be placed in positions 50 through 80.

```
 1         9                                                              80
┌──────────────────────────────────────────────────────────────────────────┐
│                                                                          │
│ [OPTIONS=([ABEND    ] [,STATS ])]                                        │
│          ([ABENDOFF]  [,NSTAT])                                          │
│                                                                          │
└──────────────────────────────────────────────────────────────────────────┘
```

This is an optional control statement.

ABEND Terminate with user 359 ABEND if any condition arises causing termination of the run.

ABENDOFF Turn 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 Provide 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 which more fully explains 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.)

The example follows:

```
//DBREORG  JOB  1,1,MSGLEVEL=1
//STEP1    EXEC PGM=DFSRRC00,PARM='ULU,DFSURUL0',REGION=250K
//IMS        DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//DBHI2A     DD  DSN=IMSVS.DBHI2A,DISP=SHR,DCB=BUFNO=10
//DBHO2A     DD  DSN=IMSVS.DBHO2A,DISP=SHR
//DBOUT1     DD  DSN=IMSVS.DBOUT1,UNIT=2400,LABEL=(,SL),
//   VOL=SER=UNLD2A,DISP=(NEW,KEEP)
//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.

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
//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=2400,LABEL=(,SL),
//    VOL=SER=DBOUT1,DISP=(NEW,KEEP)
//DBOUT2    DD  DSN=IMSVS.DBOUT1B,UNIT=2400,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=2400,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=2400,LABEL=(,SL),
//  VOL=SER=DBOUT4,DISP=(NEW,KEEP)
//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

/*
```

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.)

```
//INDREOR  JOB  1,1,MSGLEVEL=1
//STEP1   EXEC  PGM=DFSRRC00,PARM='ULU,DFSURUL0',REGION=250K
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//DBIDX1    DD  DSN=IMSVS.INDX1,DISP=SHR
//DXOUT1    DD  DSN=IMSVS.DBXOUT1,DISP=(MOD,KEEP),UNIT=2400,LABEL=(,SL),
//  VOL=SER=DXOUT1
//DBIDX2    DD  DSN=IMSVS.INDX2,DISP=SHR
//DXOUT2    DD  DSN=IMSVS.DBXOUT2,DISP=(,KEEP),UNIT=2400,LABEL=(,SL),
//  VOL=SER=DXOUT2
//NDXDS     DD  DSN=IMSVS.NDXWDS,DISP=(OLD,DELETE)
//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
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//DBIDX2    DD  DSN=IMSVS.INDX2,DISP=SHR
//DBIDX1    DD  DSN=IMSVS.INDX1,DISP=SHR
//DBXOUT1   DD  DSN=IMSVS.DBXOUT1,DISP=(MOD,KEEP),UNIT=2400,LABEL(,SL),
//  VOL=SER=DXOUT1
//DFSEXTDS  DD  DSN=IMSVS.EXTDS1,DISP=(MOD,KEEP),UNIT=2400,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
/*
```

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 Prefix Resolution 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 HISAM Reload 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
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//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
/*
```

Note: An "X" or "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.

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. An example of the output messages and statistics obtained from this utility is shown in Figure 4-5.

HIERARCHICAL INDEXED SEQUENTIAL

DATA BASE REORGANIZATION UNLOAD

DATA SET GROUP STATISTICS

DATA BASE - DI3DB02

ISAM DD - DDI3IA
OSAM DD - DDI3OA

ISAM ROOTS			OSAM ROOTS		OSAM DEPENDENTS	
IN	OUT	DELETED	IN	DELETED	IN	OUT
9	9	0	8	8	12	15

TOTAL NUMBER OF RECORDS OUT = 24
COPY 1 ON VOLUME(S)- 123456

ROOT OVERFLOW CHAINS (#)				DEPENDENT OVERFLOW CHAINS (#)				ROOTS WITHOUT OSAM CHAINS (BYTES)			
NO.	LONGEST	SHORTEST	AVERAGE	NO.	LONGEST	SHORTEST	AVERAGE	NO.	LONGEST	SHORTEST	AVERAGE
3	3	1	1	5	4	1	2	4	708	410	512

SEGMENT 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	10	4.00	A11NXXXX	1	9	1.00
4	1.30	5	1.15	AD2TGIJK	2	12	1.30
5	1.15	0	0.00	ADEPAFXX	3	14	1.50
3	1.20	0	0.00	AF2TADEX	2	16	1.20

TOTAL SEGMENTS IN DATA SET GROUP = 51 AVERAGE DATA SET GROUP RECORD LENGTH = 1,508

Figure 4-5. 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
- ISAM DD: ISAM/VSAM KSDS ddname of data set group
- OSAM DD: OSAM/VSAM ESDS ddname of data set group

- ISAM ROOTS (#)

  Statistics dealing with root records in ISAM/VSAM KSDS

  . IN Number of old ISAM/VSAM KSDS roots read
  . OUT Number of new ISAM/VSAM KSDS roots written
  . DELETED Number of old ISAM/VSAM KSDS 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.)

- OSAM ROOTS (#)

  Statistics dealing with old roots in OSAM/VSAM ESDS

  . IN Number of old OSAM/VSAM ESDS roots read
  . DELETED Number of old OSAM/VSAM ESDS roots deleted

- OSAM DEPENDENTS (#)

  Statistics dealing with dependent records in OSAM/VSAM ESDS

  . IN Number of old dependent records in OSAM/VSAM ESDS read
  . OUT Number of new dependent records in OSAM/VSAM ESDS written

- TOTAL NUMBER OF RECORDS OUT (#)

  Number of records, both ISAM/VSAM KSDS roots and OSAM/VSAM ESDS 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 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 ISAM/VSAM KSDS which had overflow chains to root records in OSAM/VSAM ESDS

  - NO. Number of ISAM roots with chains into OSAM/VSAM ESDS
  - LONGEST Largest number of OSAM/VSAM ESDS roots chained off one ISAM/VSAM KSDS root
  - SHORTEST Smallest nonzero number of OSAM/VSAM ESDS roots chained off one ISAM/VSAM KSDS root
  - AVERAGE Average number of OSAM/VSAM ESDS roots chained off one ISAM/VSAM KSDS root (of those with chains)

- DEPENDENT OVERFLOW CHAINS (#)

  Statistics dealing with roots (both ISAM/VSAM KSDS and OSAM) which had dependents in OSAM/VSAM ESDS

  - NO. Number of roots with OSAM/VSAM ESDS dependent records
  - LONGEST Largest number of OSAM/VSAM ESDS dependent records chained off one root
  - SHORTEST Smallest nonzero number of OSAM/VSAM ESDS dependent records chained off one root
  - AVERAGE Average number of OSAM/VSAM ESDS dependent records chained off one root (of those roots which had dependents)

- ROOTS WITHOUT OVERFLOW CHAINS (BYTES)

  Statistics dealing with roots (both ISAM/VSAM KSDS and OSAM) which had no dependents

  - NO. Number of roots without dependent chains
  - LONGEST Largest data base record (in bytes) with no OSAM/VSAM ESDS dependent records
  - SHORTEST Shortest data base record (in bytes) with no OSAM/VSAM ESDS dependent records
  - AVERAGE Average data base record length (in bytes) of those roots with no OSAM/VSAM ESDS 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.

The functions of this utility can be performed by the Utility Control Facility, if desired. Refer to the chapter entitled "Utility Control Facility" in this publication 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.
- OPTIONS=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 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.

A flow diagram of the HISAM Reorganization Reload utility is shown in Figure 4-6.

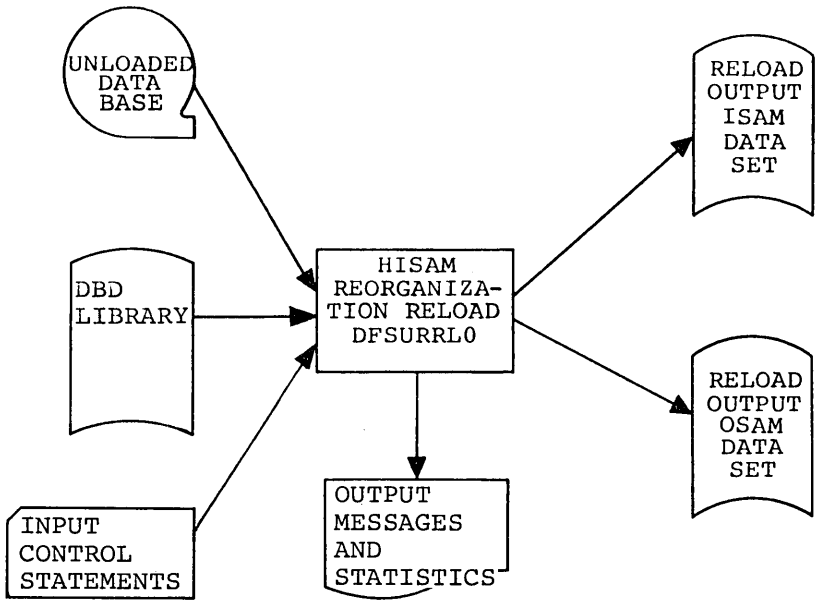


Figure 4-6. HISAM Reorganization Reload Utility

JCL REQUIREMENTS

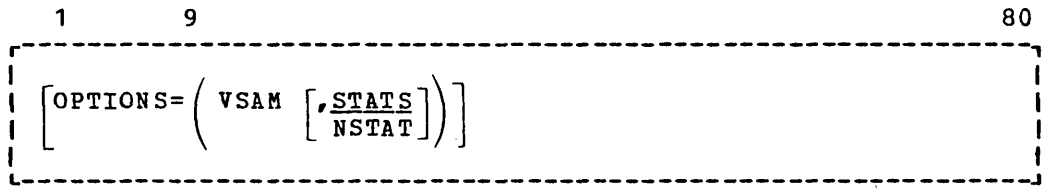
The HISAM Reorganization Reload utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

EXEC — This statement must be in the form: PGM=DFSRRC00, PARM='ULU,DFSURRL0'.

The normal IMS/VS positional parameters such as SPIE and BUF can follow the program name in the PARM field.

IMS DD	Defines the library containing the DBD which 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 ISAM/VSAM KSDS 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 OSAM/VSAM ESDS 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 if the data bases described by the isamout1 and osamout1 DD statements are VSAM data sets. (For additional information, see the discussion on "Defining the IMS/VS Buffer Pool" in the 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.

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. (See the IMS/VS Installation Guide.)

  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.

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',REGION=300K
//IMS         DD  DSN=IMSVS.DBDLIB,DISP=OLD
//SYSPRINT    DD  SYSOUT=A
//DFSUIN01    DD  DSN=IMSVS.DBOUT1A,UNIT=2400,LABEL=(,SL),
//  VOL=SER=DBOUT1,DISP=(OLD,KEEP)
//DBHI1A      DD  DSN=IMSVS.DBHI1A,UNIT=2314,VOL=SER=DBVOL1,
//    SPACE=(CYL,(2,,1)),DISP=(NEW,KEEP),DCB=(DSORG=IS,
//       BUFNO=10)
//DBHO1A      DD  DSN=IMSVS.DBHO1A,UNIT=2314,VOL=SER=DBVOL2,
//    SPACE=(CYL,(2,1)),DISP=(NEW,KEEP)
//DFSUIN02    DD  DSN=IMSVS.DBOUT2A,UNIT=2400,LABEL=(,SL),
//    VOL=SER=DBOUT3,DISP=(OLD,KEEP)
//DBHI2A      DD  DSN=IMSVS.DBHI2A,UNIT=2314,VOL=SER=DBVOL1,
//    SPACE=(CYL,(2,,1)),DISP=(NEW,KEEP),DCB=(DSORG=IS,
//      BUFNO=10)
//DBHO2A      DD  DSN=IMSVS.DBHO2A,UNIT=2314,VOL=SER=DBVOL2,
//  SPACE=(CYL,(2,1)),DISP=(NEW,KEEP)
//DFSUIN03    DD  DSN=IMSVS.DBOUT3A,UNIT=2400,LABEL=(,SL),
//  VOL=SER=DBOUT4,DISP=(OLD,KEEP)
//DBHI3A      DD  DSN=IMSVS.DBHI3A,UNIT=2314,VOL=SER=DBVOL1,
//  SPACE=(CYL,(2,,1)),DISP=(NEW,KEEP),DCB=(DSORG=IS,
//    BUFNO=10)
//DBHO3A      DD  DSN=IMSVS.DBHO3A,UNIT=2314,VOL=SER=DBVOL2,
//  SPACE=(CYL,(2,1)),DISP=(NEW,KEEP)
//SYSIN       DD  *
OPTIONS=STATS
/*
```

Note: 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), and a DFSVSAMP data set is required.

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, accomplished by explanatory information, is shown in Figure 4-7.

HIERARCHICAL INDEXED SEQUENTIAL

DATA BASE REORGANIZATION RELOAD

DATA SET GROUP STATISTICS

DATA BASE - DI3D302

ISAM DD - DDI3IA
OSAM DD - DDI3OA

		DEPENDENT OVERFLOW CHAINS (#)				ROOTS WITHOUT OSAM CHAINS (BYTES)			
ISAM ROOTS	OSAM DEPENDENTS	NO.	LONGEST	SHORTEST	AVERAGE	NO.	LONGEST	SHORTEST	AVERAGE
9	15	3	4	1	2	6	1,715	410	652

SEGMENT LEVEL STATISTICS

SEGMENT NAME	SEGMENT LEVEL	TOTAL SEGMENTS BY SEGMENT TYPE RELOADED	DIFFERENCE
A11NXXXX	1	9	
AD2TGIJK	2	12	
ADEPAFXX	3	14	
AF2TADEX	2	16	

TOTAL SEGMENTS IN DATA SET

UNLOADED	RELOADED	DIFFERENCE
51	51	0

Figure 4-7. 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 the IMS/VS Messages and Codes Reference Manual.)

Statistics are normally provided on every execution of the HISAM Reload utility. Since this increases reload time slightly, installations that are billed by CPU 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
- ISAM DD: ISAM/VSAM KSDS ddname of data set group
- OSAM DD: OSAM/VSAM ESDS ddname of data set group

- ISAM 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.)

- OSAM Dependents

  Number of dependent records loaded

- Dependent Overflow Chains (#)

  Statistics dealing with roots with dependents chained into OSAM/VSAM ESDS

  - NO. Number of ISAM/VSAM KSDS roots with OSAM/VSAM ESDS dependent records
  - LONGEST Largest number of OSAM/VSAM ESDS dependent records chained off one ISAM/VSAM KSDS root
  - SHORTEST Smallest nonzero member of OSAM/VSAM ESDS dependent records chained off one ISAM/VSAM KSDS root
  - AVERAGE Average number of OSAM/VSAM ESDS dependent records chained off ISAM/VSAM KSDS roots (of those with chains)

- Roots without OSAM/VSAM ESDS Chains (BYTES)

  Statistics dealing with ISAM/VSAM KSDS roots which have no OSAM/VSAM ESDS dependent records

  - NO. Number of roots with no OSAM/VSAM ESDS dependent records
  - LONGEST Largest root record (in bytes) with no OSAM/VSAM ESDS dependent records
  - SHORTEST Smallest root record (in bytes) with no OSAM/VSAM ESDS dependent records
  - AVERAGE Average length of root records (in bytes) with no OSAM/VSAM ESDS 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.

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. 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 the chapter entitled "Utility Control Facility" in this publication 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 which apply to making structural changes are discussed below.

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:

- 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.
- Logical record length and block size can be changed, or changes can be made from ISAM/OSAM format to VSAM format which require that the new logical record length equal the old logical record length plus 1 byte.

- 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.

- 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.

- Names of existing segment types must not be changed.

- 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.

- Existing segment lengths can be changed. IMS/VS cannot alter the data content, however, except to truncate data if the segment is made smaller.

  If the segment is made larger, it will be filled with the bit pattern that follows the original segment in storage. 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.

- 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.

- Segment pointer options for either HDAM or HIDAM can be changed. If, however, the data base contains logical relationships and if either 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.

- 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.

A flow diagram of the HD Reorganization Unload utility is shown in Figure 4-8.

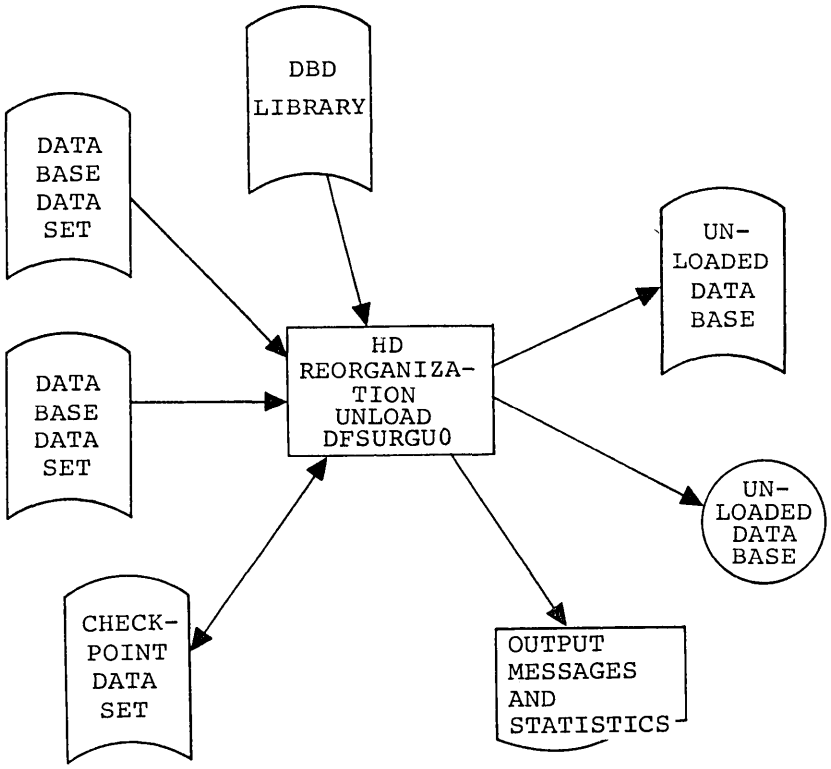


Figure 4-8. HD Reorganization Unload Utility

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. Since output is blocked to the maximum size the output device can handle, up to 16K, 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 and BUF can follow dbdname.

HD Unload (DFSURGU0) 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 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.

IMS DD	Defines the library containing the DBD which describes the data base to be reorganized. 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.
DFSUCKPT DD	Defines the data set to be used to take checkpoints. If checkpoints are not desired, 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 base 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 if the data base DD statement describes a VSAM data set. (For additional information, see the discussion on "Defining the IMS/VS VSAM Buffer Pool" in the 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.

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.

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=250K
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//DFSUCKPT   DD  DSN=IMSVS.CHKPT,UNIT=SYSDA,
//   SPACE=(TRK,(50)),VOL=SER=222222,DISP=(NEW,KEEP)
//DFSURGU1   DD  DSN=IMSVS.UNLOAD1,UNIT=2400,
//   VOL=SER=TAPE11,LABEL=(,SL),DISP=(NEW,KEEP)
//DFSURGU2   DD  DSN=IMSVS.UNLOAD2,UNIT=2400,
//   VOL=SER=TAPE21,LABEL=(,SL),DISP=(NEW,KEEP)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=2314,
//   VOL=SER=DB0001,DISP=OLD
//HDINDEXI   DD  DSN=DATABASE.INDEXI,UNIT=2314,DCB=DSORG=IS,
//   VOL=SER=DB0002,DISP=OLD
//HDINDEXO   DD  DSNAME=DATABASE.INDEXO,UNIT=2314,
//         VOLUME=SER=DB0003,DISP=OLD
```

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.

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
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//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=2400,    [Note 2]
//   LABEL=(,SL),VOL=(,,,2,SER=(TAPE11,TAPE12)),
//   DISP=(MOD,KEEP)
//DFSURGU2   DD  DSN=IMSVS.UNLOAD2,UNIT=2400,    [Note 2]
//   LABEL=(,SL),VOL=(,,,2,SER=(TAPE21,TAPE22)),
//   DISP=(MOD,KEEP)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=2314,
//   VOL=SER=DB0001,DISP=OLD
//HDINDEXI   DD  DSN=DATABASE.INDEX1,UNIT=2314,DCB=DSORG=IS,
//   VOL=SER=DB0002,DISP=OLD
//HDINDEXO   DD  DSNAME=DATABASE.INDEXO,UNIT=2314,
//         VOLUME=SER=DB0003,DISP=OLD
```

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.

   Since 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 4-9.

HIERARCHICAL DIRECT DATA BASE REORGANIZATION UNLOAD

DFS3353I WARNING - NO CHK PNT DATA SET SUPPLIED. NO CHECK POINTS TAKEN
DFS3350I NO CHECK POINT INPUT. NORMAL UNLOAD REQUESTED
DFS3344I DDNAME - DFSURGU2 - NOT SUPPLIED. 1 COPY CREATED
DATA BASE - DH41DB02 HAS BEEN UNLOADED
COPY 1 ON VOLUME(S) - STORAGE
DFS3352I 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 4-9. 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 the 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 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. It should be noted 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. It should be noted that the segment descriptions are mapped from top to bottom, in the same order they were described in the DBD.

Under Record Level Statistics, the fields, from left to right, are 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. It should be noted 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. It should be noted 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. Since 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.

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.

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.

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.

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.

A flow diagram of the HD Reorganization Reload utility is shown in Figure 4-10.

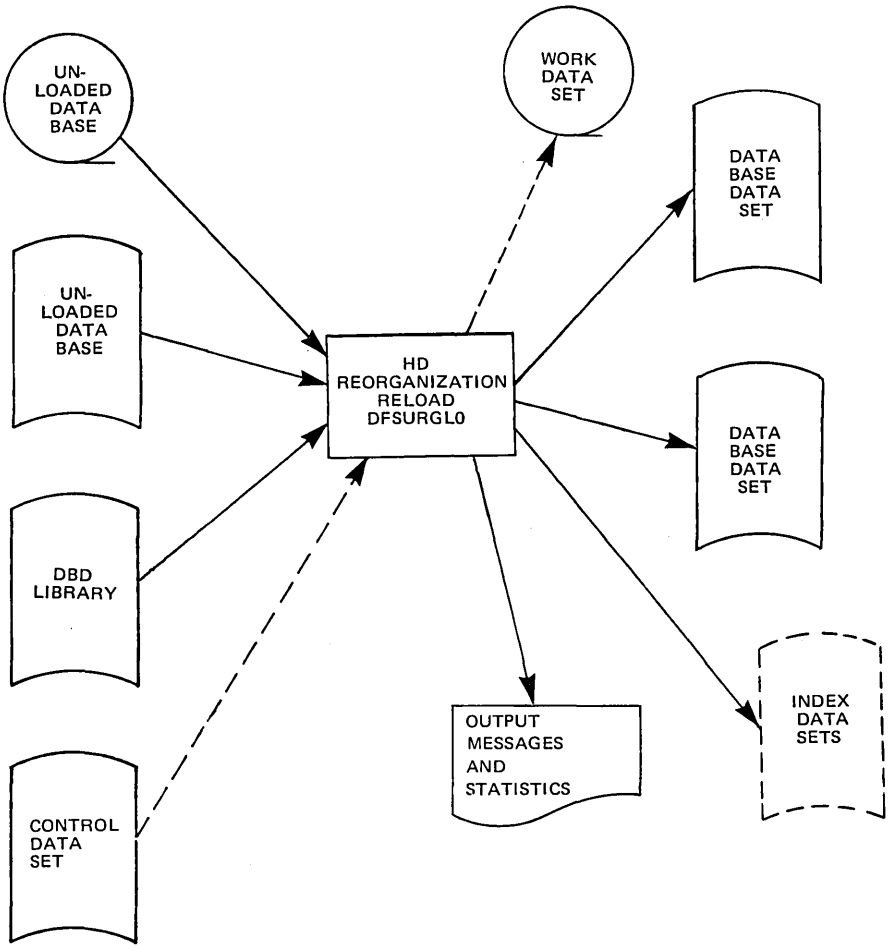


Figure 4-10. HD Reorganization Reload Utility

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 and BUF can follow dbdname.

HD Reload 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 was required to have two blocks in storage.

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. This DD statement must be specified.

The DCB parameters for the DD statement must include LRECL=300, 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).

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 if the data base DD statement describes a VSAM data set. (For additional information, see the discussion on "Defining the IMS/VS VSAM Buffer Pool" in the 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.

RETURN CODES

The following return codes are provided at program termination.

Code	Meaning
0	Data base reload successful.
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
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=OLD
//SYSPRINT   DD  SYSOUT=A
//DFSUINPT   DD  DSN=IMSVS.UNLOAD1,UNIT=2400,
//   VOL=SER=TAPE11,LABEL=(,SL),DISP=OLD
//DFSURWF1   DD  DSN=IMSVS.WRKTAPE1,UNIT=2400,
//   VOL=SER=WKTAPE,LABEL=(,SL),DISP=(NEW,KEEP),
//   DCB=(BLKSIZE=1008,LRECL=300,RECFM=VB)
//HDSKILLS   DD  DSN=DATABASE.SKILLS,UNIT=2314,
//   VOL=SER=DB0002,DISP=(NEW,KEEP),SPACE=(CYL,(10,10))
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=2314,DISP=(OLD,KEEP),
//   VOL=SER=IMSMSC
```

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
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=OLD
//SYSPRINT   DD  SYSOUT=A
//DFSUINPT   DD  DSN=IMSVS.UNLOAD1,UNIT=2400,LABEL=(,SL),
//   VOL=SER=TAPE11,DISP=OLD
//DFSURWF1   DD  DSN=IMSVS.WRKTAPE1,UNIT=2400,
//   VOL=SER=WKTAPE,LABEL=(,SL),DISP=(NEW,KEEP),
//   DCB=(BLKSIZE=1008,LRECL=300,RECFM=VB)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=2314,
//   VOL=SER=DB0001,DISP=OLD
//HDINDEXI   DD  DSN=DATABASE.INDEXI,UNIT=2314,
//   VOL=SER=DB0003,DCB=DSORG=IS,DISP=(NEW,KEEP),
//   SPACE=(CYL,(5))
//HDINDEXO   DD  DSN=DATABASE.INDEXO,UNIT=2314,
//   VOL=SER=DB0003,DISP=(NEW,KEEP),SPACE=(CYL,(10,10))
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=2314,DISP=(OLD,KEEP),
//   VOL=SER=IMSMSC
```

OUTPUT MESSAGES AND STATISTICS

The HD Reorganization Reload 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 4-11.

```
        HIERARCHICAL DIRECT DATA BASE REORGANIZATION RELOAD

                        SEGMENT LEVEL STATISTICS

                                    TOTAL SEGMENTS BY SEGMENT TYPE
  SEGMENT          SEGMENT
   NAME            LEVEL          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

DFS3541I  NO ERRORS DETECTED. DATA BASE RELOAD SUCCESSFUL
```

Figure 4-11. 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 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 PREREORGANIZATION UTILITY (DFSURPRO)

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 which 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 4-12.

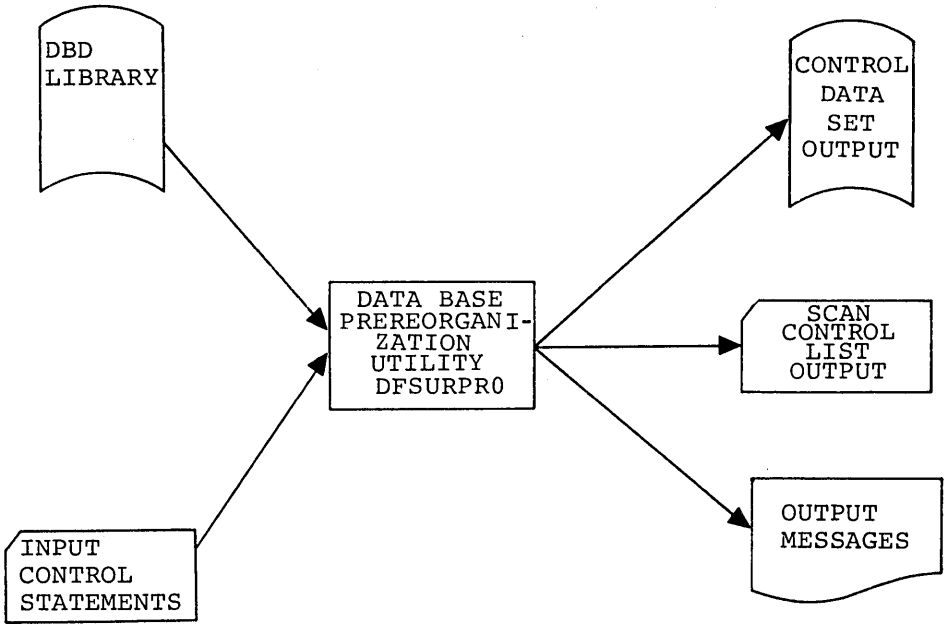


Figure 4-12. Data Base Prereorganization Utility

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.
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 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.

UTILITY CONTROL STATEMENTS

```
 1                                                              80
┌──────────────────────────────────────────────────────────────┐
│                                                              │
│ [DBIL=database name1,database name2,.....] [comments]        │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

This utility control statement names data bases that are being initially loaded. 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.)

```
 1                                                              80
┌──────────────────────────────────────────────────────────────┐
│                                                              │
│ [DBR=database name1,database name2,.....] [comments]         │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

This utility control statement names data bases that are 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.

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.)

```
 1                                                              80
┌──────────────────────────────────────────────────────────────┐
│                                                              │
│ [OPTIONS=( [{NOPUNCH}] [,STAT] [,SUMM])]                    │
│            [{PUNCH  }]                                       │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

This utility control statement indicates whether any optional information is to be provided during the reorganization of the data base. 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 bypass writing out the same message (such as DFS878I) more than once and to accumulate and then print merely the number of times the message was issued.

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 DFSURPRO 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'
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=2314,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 nonreorganized data bases 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 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 the chapter entitled "Utility Control Facility" in this publication for a description of its operation.

A flow diagram of the Data Base Scan utility is shown in Figure 4-13.

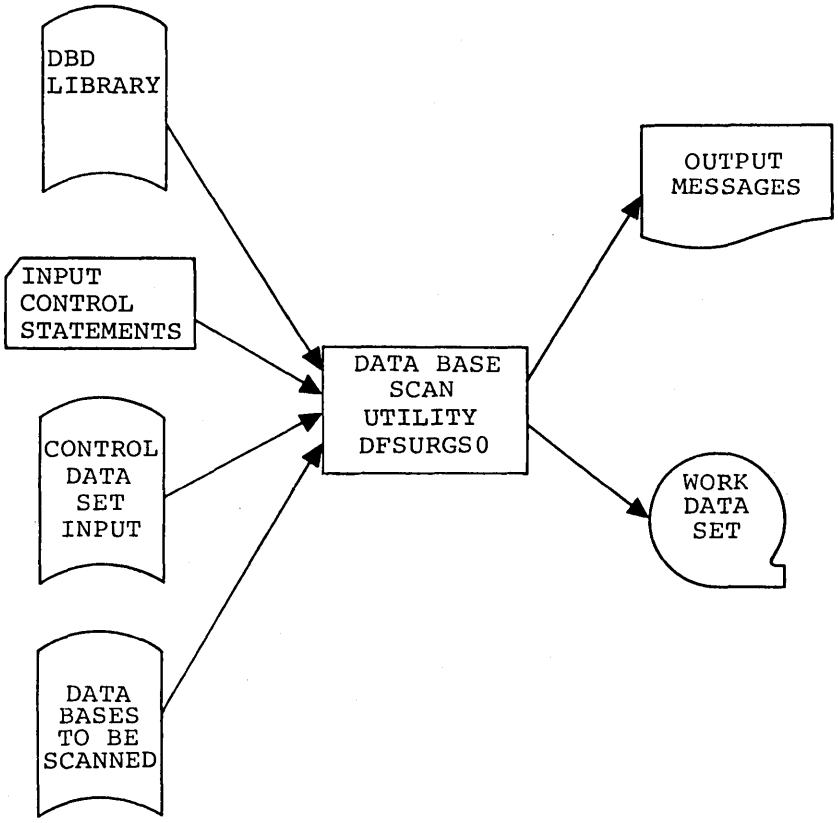


Figure 4-13. Data Base Scan Utility

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. Data Base Scan (DFSURGS0) 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 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.
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.
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 Initial Load and Data Base Scan. The data set can reside on tape or a direct access device. DCB parameters specifed 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 if the data bases described by the database DD statement are VSAM data sets. (For additional information, see the discussion on "Defining The IMS/VS VSAM Buffer Pool" in the IMS/VS Installation Guide.) The data set can reside on a tape, direct access device, card reader, or be routed through the input stream.

UTILITY CONTROL STATEMENTS

```
 1                                                                  80
┌──────────────────────────────────────────────────────────────────┐
│                                                                  │
│ [DBS=database-name,segment-name[,SEQ]]                           │
│                                [,SEG]                            │
└──────────────────────────────────────────────────────────────────┘
```

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 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.

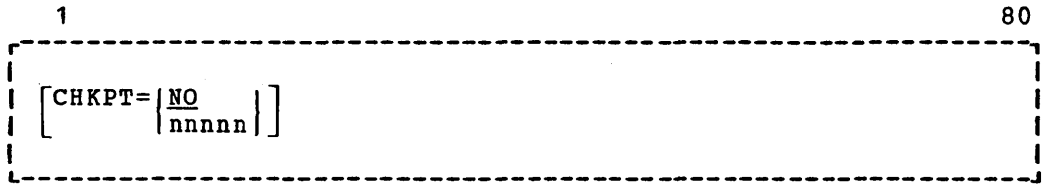
It should be noted that even if a scan list is provided through DBS control statements, the control data set must still be provided to this utility since it contains other information that is required by the Scan utility.

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.

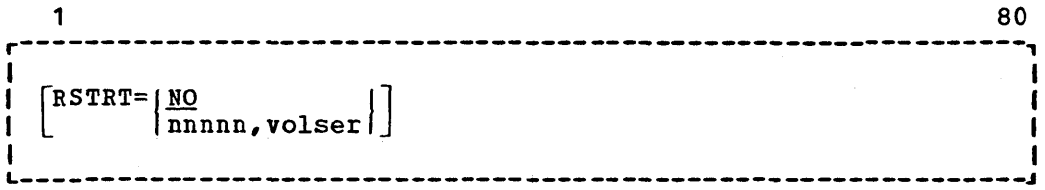
The SEQ and SEG options specify the method to be used to scan a data base. The SEQ option indicates that a data base is to be scanned sequentially by using unqualified GN calls. The SEG option indicates that a data base is to be scanned by using GN (Get Next) calls qualified by segment name. The scan 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.



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.



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 (DFS875I) is issued to the OS/VS system console indicating this program name, the checkpoint number, and the volume serial. Since 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).

This optional utility control statement should be included when a storage dump is required for diagnostic purposes. If included in the input stream, any return code greater than zero causes user abend 955 to be issued if any abnormal condition arises during execution of this utility. 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
//IMS    DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSURWF1   DD  DSN=IMSVS.URWF1,UNIT=2400,VOL=SER=TAPE11,
// LABEL=(,SL),DISP=(NEW,KEEP),DCB=(LRECL=300,BLKSIZE=1008,
//   RECFM=VB)
//HDRELTD    DD  DSN=DATABASE.DBRELATD,UNIT=2314,
//   VOL=SER=DB0003,DISP=OLD
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=2314,
//   VOL=SER=IMSMSC,DISP=OLD
```

Note: DD statements must be included for all logically related data bases.

OUTPUT MESSAGES

The output messages issued by this utility include DFS870I, DFS862I, and DFS871I. The DFS870I message indicates when the scan processing started for the data base named on the DBS utility control statement. Message DFS862I indicates which segments with the data base are being scanned and whether the search is sequential or by segment. Message DFS871I indicates that scan processing is completed.

Other output messages issued by this utility denote error conditions which are fully explained in the 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 the chapter entitled "Utility Control Facility" in this publication for a description of its operation.

RESTRICTIONS

The Data Base Prefix Resolution utility uses the OS/VS Sort/Merge programs. Since 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. It may be omitted otherwise.

      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 the IMS/VS Messages and Codes Reference Manual for an explanation of the message.

Note: IMS/VS makes no assumption of sequence for unkeyed or non-unique 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.

A flow diagram of the Data Base Prefix Resolution utility is shown in Figure 4-14.

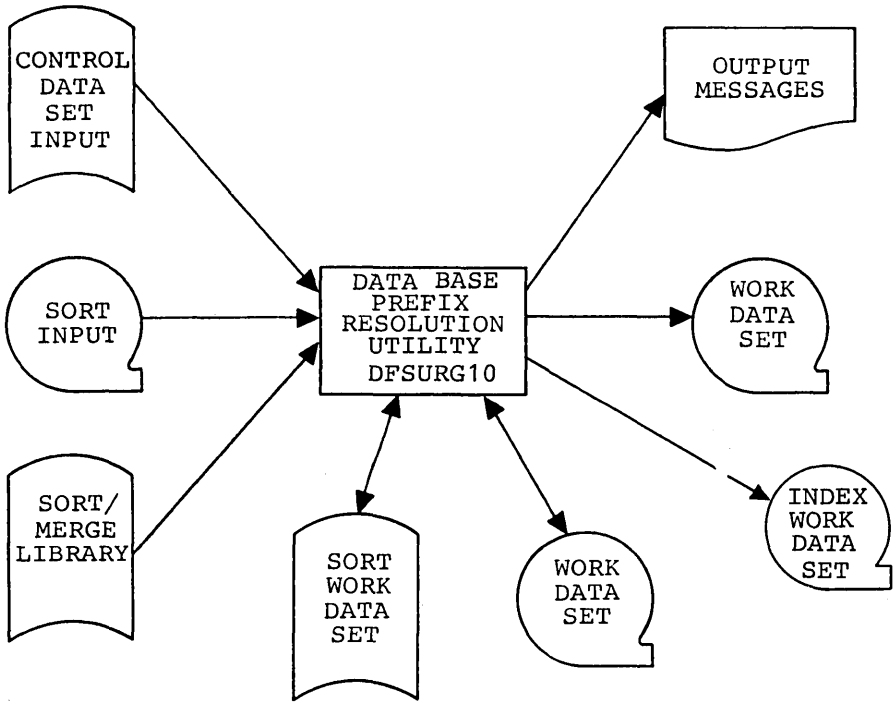


Figure 4-14. Data Base Prefix Resolution Utility

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. Since this program invokes the operating system Sort, program efficiency can be improved by increasing region size.

The PARM field options and defaults are:

```
PARM='[[BALN|OSCL|POLY]][,CKPT],[FILSZ=n|SIZE=n,|FILSZ=En|SIZE=En] CORE={nnnnnn|061440|MAX} [,MSG={NO|CC|CP|AC|AP}] [,ABEND]'
```

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 those keywords shown above for the standard operating system Sort/Merge will be passed to the attached Sort/Merge program.

If the positional parameter CKPT 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 six-digit figure.

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 since 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
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 you may add or delete at exit E15.

  For example, if you estimate your total data set size to be 5000 records, specify FILSZ=E5000. The maximum allowable size is E+8(Ennnnnnnn).

If you use the balanced disk technique, the Sort/Merge program prints message IGH070I, which states either (1) that you have not specified the size of the file or (2) the decimal number of records that you have specified.

If you omit this operand, 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.

You should give at least an estimated file size if possible since this greatly improves SM1's optimization and hence performance.

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=120. 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.

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 specifed 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=300. 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. All notes on DFSURWF3, above, apply to this data set also. 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.

RETURN CODES

The following return codes are provided at program termination:

Code	Meaning
0	No errors detected.
4	Returned when either one or both of the following messages have been issued during program execution: DFS878, DFS885
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 no data 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 OS/VS Sort/Merge program. This return code takes precedence over the above return codes.

Note: For return codes larger than 16, the same 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 since the input work data set required by DFSURGP0 would not have been 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 the 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,MSGIEVEL=1
//STEP1  EXEC    PGM=DFSURG10,REGION=100K
//SYSOUT DD      SYSOUT=A
//SYSPRINT DD    SYSOUT=A,DCB=BLKSIZE=1200
//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=2400,VOL=SER=TAPE11,
|   //  LABEL=(,SL),DISP=(OLD,KEEP),DCB=(LRECL=900,BLKSIZE=1008,
//    RECFM=VB)
//DFSURWF2   DD  DSN=IMSVS.URWF2,UNIT=2400,VOL=SER=TAPE21,
|   //  LABEL=(,SL),DISP=(NEW,DELETE),DCB=(LRECL=900,
//     BLKSIZE=1008,RECFM=VB)
//DFSURWF3   DD  DSN=IMSVS.URWF3,UNIT=2400,VOL=SER=TAPE31
//  LABEL=(,SL),DISP=(NEW,KEEP),DCB=(LRECL=300,BLKSIZE=1008,
//   RECFM=VB)
//DFSURCDS   DD  DSN=IMSVS.RLCDS,UNIT=2314,DISP=OLD,
//          VOL=SER=IMSMSC
//DFSURIDX   DD  DSN=IMSVS.URIDX,DISP=(,KEEP),
//          UNIT=2400,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.

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 desired. Refer to the chapter entitled "Utility Control Facility" in this publication for a description of its operation.

A flow diagram of the Data Base Prefix Update utility is shown in Figure 4-15.

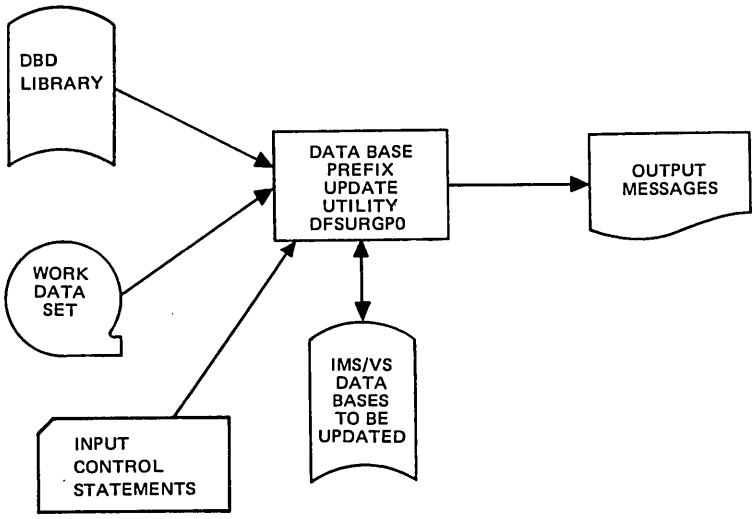


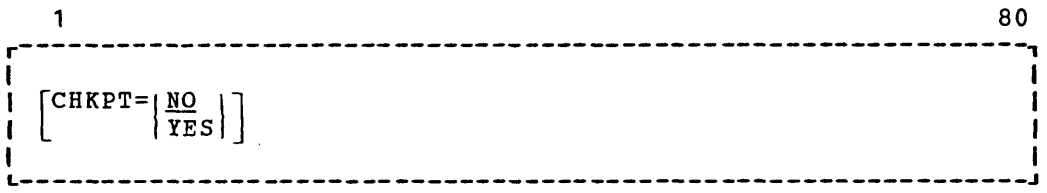
Figure 4-15. Data Base Prefix Update Utility

JCL REQUIREMENTS

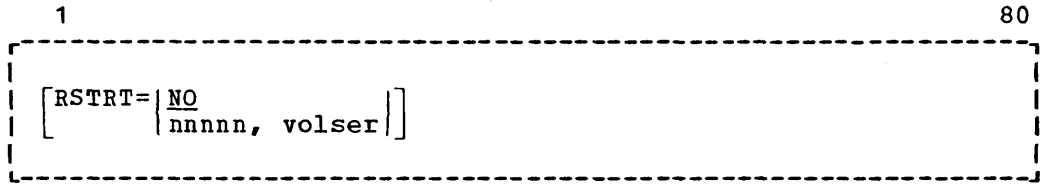
The Data Base Prefix Update utility is executed as a standard OS/VS job. A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define inputs and outputs are required.

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.
IMS DD	Defines the library containing the DBDs which 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.
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.
DFSVSAMP DD	Describes the data set that contains the buffer information required by the DL/I Buffer Handler. This DD statement is required if the data bases described by the database DD statement are VSAM data sets. (For additional information, see the discussion on "Defining The IMS/VS VSAM Buffer Pool" in the 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.

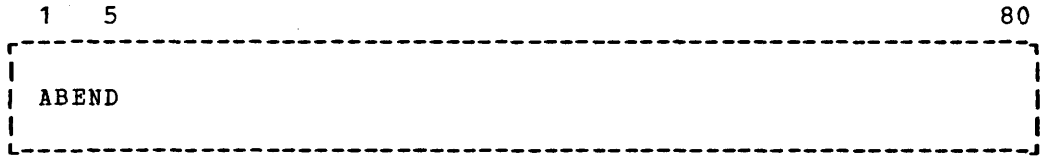


This utility control statement indicates that checkpoint operations are to be performed during operation of this program. The Prefix Resolution utility automatically writes records on the DFSURWF3 data set as a service for the Prefix Update utility, and these records are used as checkpoints. As each 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.



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 (DFS875) is issued indicating the checkpoint number, volume serial, and program name. Processing continues from the restart point.



This optional utility control statement should be included when a storage dump is required for diagnostic purposes. If included in the input stream, any return code greater than zero causes user abend 955 to be issued if any abnormal condition arises during execution of this utility. 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
//IMS    DD      DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSURWF3   DD  DSN=IMSVS.URWF3,UNIT=2400,VOL=SER=TAPE31,
// LABEL=(,SL),DISP=(OLD,KEEP),DCB=(LRECL=300,BLKSIZE=1008,
//   RECFM=VB)
//HDPAYROL   DD  DSN=DATABASE.PAYROLL,UNIT=2314,
//   VOL=SER=DB0001,DISP=OLD
//HDSKILLS   DD  DSN=DATABASE.SKILLS,UNIT=2314,
//   VOL=SER=DB0002,DISP=OLD
```

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.

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 PROCESSING UTILITIES

This section provides descriptions of several sample procedures that will enable the user to perform commonly used data base operations with the minimum JCL. 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, and they are not supported as part of IMS/VS program products. 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 Name	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)

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 of 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 meaning:

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=.

DFSLRLOD PROCEDURE

```
//          PROC    PGML=,PSB=,SOUT=3,BUF=8,SPIE=0,TEST=0
//P         EXEC    PGM=DFSRRC00,PARM='ULU,DFSURPR0',REGION=150K
//*
//*         THIS STEP EXECUTES THE PREREORGANIZATION UTILITY (DFSURPR0)
//*
//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
//L         EXEC    PGM=DFSRRC00,PARM='DLI,&PGML,&PSB,&BUF,&SPIE&TEST',
//          REGION=250K
//*
//*         THIS STEP EXECUTES THE INITIAL DATABASE LOAD PROGRAM.
//*
//IMS       DD      DSN=IMSVS.PSBLIB,DISP=SHR
//          DD      DSN=IMSVS.DBDLIB,DISP=SHR
//DFSURWF1  DD      DSN=&&WF1,UNIT=SYSDA,DISP=(NEW,PASS),
//          SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=300,BLKSIZE=1008)
//DFSURCDS  DD      DSN=&&CDS,DISP=(OLD,PASS),UNIT=SYSDA
//DFSVSAMP  DD      *
XXXX,YY
//R         EXEC    PGM=DFSURG10,REGION=200K
//*
//*         THIS STEP EXECUTES THE PREFIX RESOLUTION UTILITY (DFSURG10).
//*
//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,PASS),
//          DCB=(LRECL=300,BLKSIZE=1008,RECFM=VB)
//DFSURWF2  DD      UNIT=SYSDA,SPACE=(CYL,(1),,CONTIG),DSN=&&WF2,
//          DISP=(,PASS),DCB=(LRECL=300,BLKSIZE=1008,
//          RECFM=VB)
//DFSURWF3  DD      UNIT=SYSDA,SPACE=(CYL,(1),,CONTIG),DSN=&&WF3,
//          DISP=(,PASS),DCB=(LRECL=300,BLKSIZE=1008,
//          RECFM=VB)
//DFSURCDS  DD      DSN=&&CDS,DISP=(OLD,PASS),UNIT=SYSDA
//DFSURDIX  DD      DSN=&&IDX,UNIT=SYSDA,DISP=(NEW,PASS),
//          SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=300,BLKSIZE=1008)
//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)
```

Note: The DFSURDIX DD statement is used only for secondary indexes.

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=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//L.X      DD      DSN=ICS.DDB.INDEXI,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=(DSORG=IS,
//         OPTCD=WM)
//L.XO     DD      DSN=ICS.DDB.INDEXO,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1,1))
//L.DHSK0301 DD    DSN=IMS.DDB.DHSK0301,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//L.X2     DD      DSN=ICS.DDB.INDEXI2,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=(DSORG=IS,
//         OPTCD=WM)
//L.XO2    DD      DSN=ICS.DDB.INDEXO2,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(,KEEP),SPACE=(CYL,(1,1))
//L.DISK0201 DD    DSN=IMS.DDB.DISK0201,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=DSORG=IS
//L.DISK0202 DD    DSN=IMS.DDB.DISK0202,UNIT=2314,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=2314
//U.X       DD     DSN=*.LOAD.L.X,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//          UNIT=2314
//U.XO      DD     DSN=*.LOAD.L.XO,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//          UNIT=2314
//U.DHSK0301 DD    DSN=*.LOAD.L.DHSK0301,DISP=(OLD,KEEP),
//          VOL=SER=IMSDBS,UNIT=2314
//U.X2      DD     DSN=*.LOAD.L.X2,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//          UNIT=2314
//U.XO2     DD     DSN=*.LOAD.L.X02,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//          UNIT=2314
//U.DISK0201 DD    DSN=*.LOAD.L.DISK0201,DISP=(OLD,KEEP),
//          VOL=SER=IMSDBS,UNIT=2314
//U.DISK0202 DD    DSN=*.LOAD.L.DISK0202,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//          UNIT=2314
```

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 logcial 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 most likely 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).
//*
//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.
//*
//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=300,
//          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).
//*
//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=300,
//          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).
//*
//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=300,
//          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).
//*
//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=300,BLKSIZE=1009,RECFM=VB)
//DFSURWF2 DD       UNIT=SYSDA,SPACE=(CYL,(1),,CONTIG),DSN=&&WF2,
//          DISP=(,PASS),DCB=(LRECL=300,BLKSIZE=1008,RECFM=VB)
//DFSURWF3 DD       UNIT=SYSDA,SPACE=(CYL,(1),,CONTIG),DSN=&&WF3,
//          DISP=(,PASS),DCB=(LRECL=300,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 DATABASE TO
//*         BE INITIALLY LOADED AND TWO OTHER LOGICALLY RELATED DATABASES
//*         TO BE REORGANIZED.  THE TWO DATABASES TO BE REORGANIZED ARE
//*         HIDAM.  THE DATABASE 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.
//*
//L.SYSIN  DD       DSN=ICS.CSOR(HPHI02),DISP=SHR
//*
//*         THE FOLLOWING JCL DEFINES THE DATABASE TO BE LOADED.
//*
//L.DISK0201 DD     DSN=IMS.DDB.DISK0201,UNIT=2314,VOL=SER=IMSDBS,
//          DISP=(NEW,KEEP),SPACE=(CYL,(2,,1)),DCB=DSORG=IS
//L.DISK0202 DD     DSN=IMS.DDB.DISK0202,UNIT=2314,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 DATABASES
//*        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=2314,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//UL.X     DD      DSN=ICS.DDB.INDEXI,UNIT=2314,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),DISP=(OLD,DELETE)
//UL.XO    DD      DSN=ICS.DDB.INDEXO,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//RL.DHSK0201 DD   DSN=IMS.DDB.DHSK0201,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//RL.X     DD      DSN=ICS.DDB.INDEXI,UNIT=2314,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),SPACE=(CYL,(2,,1)),
//         DISP=(NEW,KEEP)
//RL.XO    DD      DSN=ICS.DDB.INDEXO,UNIT=2314,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 DATABASES
//*        TO BE REORGANIZED.  NOTE THAT TWO SETS OF DD CARDS ARE
//*        REQUIRED, AS DESCRIBED ABOVE.
//*
//UL.DHSK0301 DD   DSN=IMS.DDB.DHSK0301,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//UL.X2    DD      DSN=ICS.DDB.INDEXI2,UNIT=2314,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),DISP=(OLD,DELETE)
//UL.XO2   DD      DSN=ICS.DDB.INDEXO2,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(OLD,DELETE)
//RL.DHSK0301 DD   DSN=IMS.DDB.DHSK0301,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(2,1))
//RL.X2    DD      DSN=ICS.DDB.INDEXI2,UNIT=2314,VOL=SER=IMSDBS,
//         DCB=(DSORG=IS,OPTCD=WM),SPACE=(CYL,(2,,1)),
//                    DISP=(NEW,KEEP)
//RL.XO2   DD      DSN=ICS.DDB.INDEXO2,UNIT=2314,VOL=SER=IMSDBS,
//         DISP=(NEW,KEEP),SPACE=(CYL,(1.1))
//RESOLVE  EXEC    PROC=DFSLRRES
//*
//*        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=2314
//U.X      DD      DSN=*.LOAD.L.X,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=2314
//U.XO     DD      DSN=*.LOAD.L.XO,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=2314
//U.DHSK0301 DD    DSN=*.LOAD.L.DHSK0301,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=2314
//U.X2     DD      DSN=*.LOAD.L.X2,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=2314
//U.XO2    DD      DSN=*.LOAD.L.XO2,DISP=(OLD,KEEP),VOL=SER=IMSDBS,
//         UNIT=2314
//U.DISK0201 DD    DSN=*.LOAD.L.DISK0201,DISP=(OLD,KEEP),
//         VOL=SER=IMSDBS,UNIT=2314
```

CHAPTER 5. DATA BASE RECOVERY UTILITIES

The Data Base Recovery System has four utility programs for recovering a data base. These utilities are discussed separately in this chapter to describe how they can be used in an installation. Appropriate JCL and examples are provided for each utility. This chapter assumes that the reader is familiar with the "Data Base Design Considerations" chapter in the IMS/VS System/Application Design Guide.

OVERVIEW

IMS/VS utilizes a very simple concept for data base recovery. It involves creating a physical image copy of a data base at a specified point in time and keeping a log of all physical changes made to the data after that time. Then, if the data base that was copied becomes physically damaged, it can be fully restored by: (1) restoring the image copy to DASD, and (2) applying the logged changes to the new copy.

The Data Base Recovery utilities and their functions are as follows:

1. Data Base Image Copy utility for creation of dump images of data bases;

2. Data Base Change Accumulation utility for accumulation of data base changes from IMS/VS log tapes since the last complete image dump;

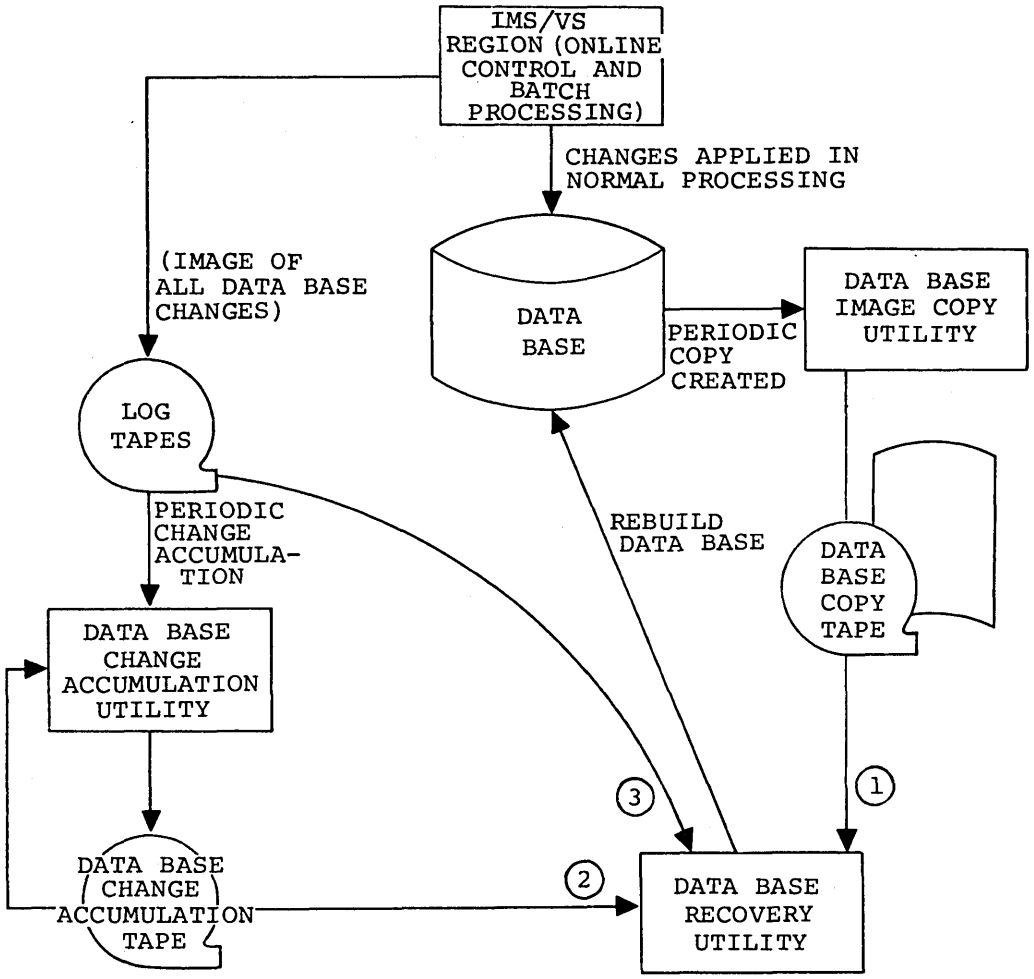
3. Data Base Recovery utility for restoration of the data base, using a prior data base image copy and the accumulated changes from IMS/VS log tapes;

4. Data Base Backout utility for removal of changes made to data bases by selected application programs.

A fifth utility program, the IMS/VS System Log Terminator (DFSFLOT0), is used to terminate the IMS/VS System Log in the event of an operating system or hardware failure, thus enabling use of the log by the four principle programs of the recovery system. (The System Log Terminator utility is described in the "Log Maintenance Utilities" chapter of this manual.)

The Data Base Recovery System operates as a set of IMS/VS utility programs in an IMS/VS batch processing region. For those data bases which consist of multiple data sets, recovery is done by individual data set. To recover a complete data base composed of multiple data sets, data base recovery must be performed for each of its component data sets.

The diagram in Figure 5-1 illustrates the process of data base recovery.



1. Data base (data set) copy tape created when the data base was last dumped, through the Data Base Image Copy utility. (Note that a direct access device can also be used.)

2. Data base (data set) change accumulation tape created from log tapes available since the last data base copy creation by the Data Base Change Accumulation utility.

3. Log tapes created since the last data base copy creation but not incorporated into the change accumulation tape. This would include at least the log tape in use at the time problems were encountered with the data bases.

Figure 5-1. Data Base Recovery System

The Data Base Backout utility is not depicted in Figure 5-1. It should be noted, however, that one or more of the log tapes represented in the figure could have been created by the Backout utility. (A description of this utility is provided later in this chapter.)

When the status of a data base is not known because the program that was updating the data base terminated abnormally, the 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 tape 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 checkpoint record. It also creates a log tape that must be used as input to any future data base recovery operation; it may also be used as input to the Data Base Change Accumulation utility.

To expand on the discussion of data base recovery, the different steps involved in rebuilding the data sets of a data base are summarized below.

1. Logging the changed data for a segment replace, insert, or delete, including the identification of the "updated" segment.

   This function is performed in the IMS/VS control program region for all data bases used in online, or concurrent online, and batch message processing. For data bases used exclusively for batch processing or batch processing not concurrent with online processing, the logging function is performed in the IMS/VS batch processing region.

2. Selecting the changed data base log records from the log data set and sorting them in order by data set within data base.

   Selecting and sorting are performed as part of the change accumulation tape creation by the Data Base Change Accumulation utility.

3. Merging the sorted change records with the prior cumulative changes, keeping only the most recent data. Merging is performed as part of the change accumulation tape creation by the Data Base Change Accumulation utility.

4. Dumping the data sets of data bases periodically to provide a backup copy at that point in time using the Data Base Image Copy utility.

5. Reading the prior copy of the data set to be restored and merging the cumulative changes, thereby creating a partially restored data set.

6. Reading log tapes not included in the most recent cumulative changes and updating the data set to the point at which the error occurred.

   Functions 5 and 6 are performed using the Data Base Recovery utility.

Note: All updates to the data set at recovery time can be applied from the log tapes (step 1). In some installations, however, it would be highly desirable to include steps 2 and 3; frequent data set image copying reduces recovery time and allows all cumulative changes which preceded the copy to be dropped from the sorted change accumulation tape. Each installation will have to determine which approach best suits their needs. Two important considerations are the frequency of recovery requirements and time-savings factors (based on using one approach versus another).

USE OF THE SYSTEM LOG TAPE FOR DATA BASE RECOVERY

The IMS/VS System logs all information associated with the modification of data within an existing data base. The information placed on an IMS/VS system log tape whenever a data base modification is made includes the following:

- identification of the data base;
- identification of the data set within a data base by providing 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 does not support updates to data bases, the data base recovery utilities do not support the HSAM data base organization.

For an HSAM 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 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.

DATA BASE IMAGE COPY UTILITY (DFSUDMP0)

The Data Base Image Copy utility creates an output copy of the data sets within a data base. As illustrated previously (see Figure 5-1), this output is used as input to the Data Base Recovery utility.

Multiple data sets can be copied on mixed device types with one execution of the Image Copy utility. For operations convenience, all data sets of a data base should be copied at the same time.

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.

A flow diagram of the Data Base Image Copy utility is shown in Figure 5-2.

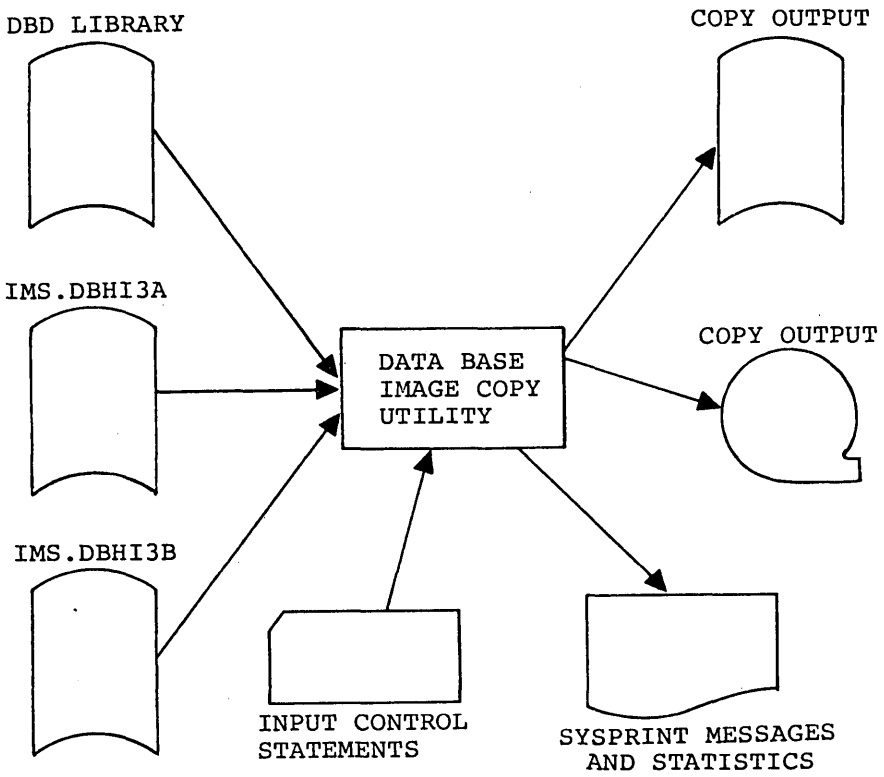


Figure 5-2. Data Base Image Copy Utility

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.

Since 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 if logical relationships exist and the Prefix Update utility has been run. Since data base recovery is done on a physical replacement basis, a reloaded data set is not physically the same as it was before unload.

Note: The output from the HISAM Reorganization Unload utility can also be used as input to the Data Base Recovery utility. The following chapter of this manual describes how this is accomplished.

The functions of this utility can be performed under control of the Utility Control Facility, if desired. Refer to the chapter entitled "Utility Control Facility" for a description of its operation.

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 catalogued procedure containing the required statements or be in the form: PGM=DFSRRC00,PARM='UDR,DFSUDMP0,dbdname' The normal IMS/VS positional parameters such as BUF and SPIE can follow dbdname. (See the IMS/VS Installation Guide for additional parameters that can be specified in executing a batch processing region.) where dbdname is the name of the DBD that includes the data set to be copied/dumped. A default region size of 52K is normally adequate for execution.
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.
YSPRINT 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.
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.
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.

DFSVSAMP DD	Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required if the data bases described by the datain DD statements are VSAM data sets. (For additional information, see the discussion on "Defining the IMS/VS VSAM Buffer Pool" in the 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.

UTILITY CONTROL STATEMENT

```
 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 Set 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 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. If position 3 is blank, an image copy of the KSDS is obtained if position 13 specifies the ddname for the KSDS.
4	This must be the name of the 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	Comments may be placed in positions 40-80.

RETURN CODES

The Data Base Image Copy utility provides the following return codes:

Code	Meaning
0	Successful completion of all operations.
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=DFSRRC00,PARM='UDR,DFSUDMP0,DI32DB01'
//IMS       DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//DBHI3A    DD  DSN=IMS.DBHI3A,DISP=SHR,DCB=BUFNO=10
//DBAOUT1   DD  DSN=IMS.DBAOUT1,UNIT=2400,
//  VOL=SER=DBDMP1,LABEL=(,SL),DISP=(NEW,KEEP),
//  DCB=BUFNO=10
//SYSIN     DD  *

12 4         13        22        31        40                              80
D1 DI32DB01 DBHI3A    DBAOUT1             DUMP SINGLE DATA SET
/*
```

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='UDR,DFSUDMP0,DI32DB01'
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//DBHI3A   DD  DSN=IMS.DBHI3A,DISP=SHR,DCB=BUFNO=10
//DBHI3B   DD  DSN=IMS.DBHI3B,DISP=SHR,DCB=BUFNO=10
//DBAOUT1  DD  DSN=IMS.DBAOUT1,UNIT=2400,VOL=SER=DBDMP1,
//  LABEL=(,SL),DISP=(NEW,KEEP),DCB=BUFNO=10
//DBAOUT2  DD  DSN=IMS.DBAOUT2,UNIT=2400,VOL=SER=DBDMP2,
//  LABEL=(,SL),DISP=(NEW,KEEP),DCB=BUFNO=10
//DBBOUT1  DD  DSN=IMS.DBBOUT1,UNIT=2400,VOL=SER=DBDMP3,
//  LABEL=(,SL),DISP=(NEW,KEEP),DCB=BUFNO=10
//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: In both examples, 10 buffers were requested by the DCB parameter of the DD statement. This is not a requirement; performance can be enhanced, however, by providing adequate buffers.

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 tapes which is necessary for recovery. This is done by eliminating all non-data 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 tapes 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 tapes that may be required for data base recovery. The new log output data set created by Change Accumulation is used by the Data Base Recovery utility.

The input to the Data Base Change Accumulation utility consists of:

1. All log tapes 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 data set. This would be the output from the last execution of this utility.

3. DBD library which 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 5-3 and the discussion under "Utility Control Statements" later in this chapter.)

Output from the Data Base Change Accumulation utility consists of:

1. A new Change Accumulation 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.

Note: The functions of this utility can be performed by the Utility Control Facility, if desired. Refer to the chapter entitled "Utility Control Facility" in this publication for a discussion of its operation.

Figure 5-3 depicts the sources of input to the Data Base Change Accumulation utility and the output created by this utility.

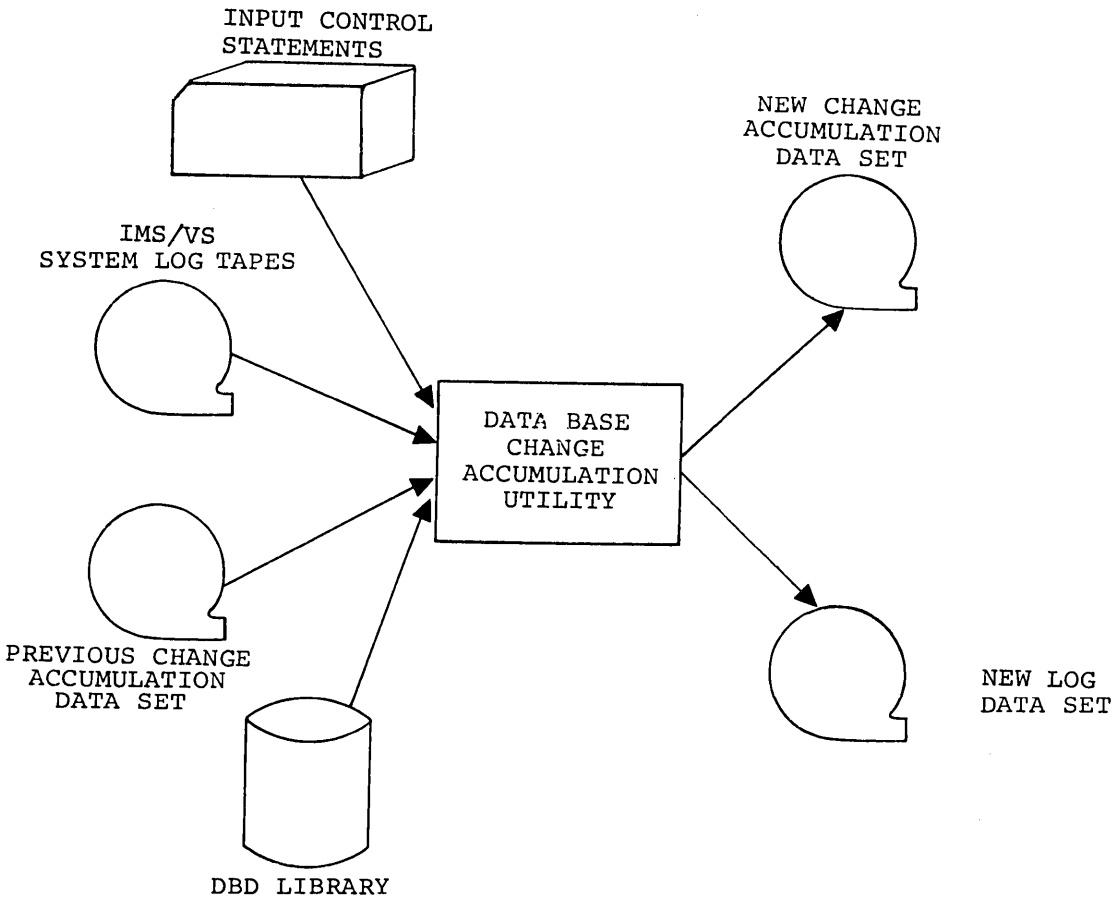


Figure 5-3. Data Base Change Accumulation Utility

USE OF PURGE DATE AND TIME

It is not necessary to specify a purge date and time if there are no data base records in the log tape for the data set being accumulated. If change accumulation records or a log tape span an Image Copy time or a reorganization time, however, a purge date and time are mandatory. Since, in this instance, the tape 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 must be specified. This ensures that data base changes are not lost and that those change records that are not needed are eliminated.

The user may specify one purge date and time for all data base log records that he is processing. 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.

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 of 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 DB1
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 DB2
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,INCR=iiiii',REGION=xxx where ssssss is the amount of storage in bytes that Sort/Merge can use for this application. The program defaults to 100,000 bytes if no value is specified. The region size specified should be the ssssss value plus 100K bytes, or 200K 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. iiiii is the size of the increment to be added to the internal sort table when it overflows. If the value iiiii is not specified, the internal sort table is defaulted to 20K as an incrementing value.
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 listed in the Preface of this publication.)
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, up to a maximum of 8K. Standard labels must be used.

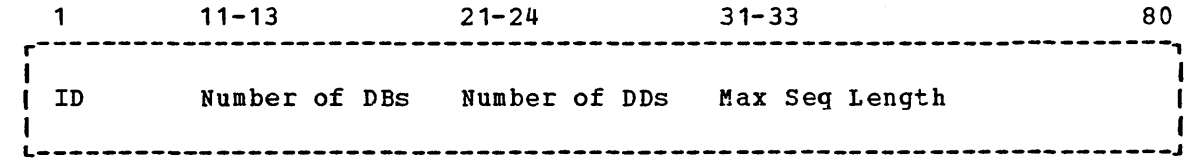
DFSUCUMO DD	Defines the old accumulated change input data set that is to be merged with the log input data in order to create the new accumulated change data set. If no old accumulated changes are to be merged, the following DD statement must be used: //DFSUCUMO DD DUMMY,DCB=BLKSIZE=100 This data set 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, up to 8K. 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. Multiple log tapes can be used as input by concatenating the data sets. This requires that the DD statements be in date and time sequence.
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.

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.
11	Positions 11-13 contain the maximum number of data base names supplied by succeeding DB0 or DB1 control statements. This value must be in the range 1-999 and must be left-justified or supplied with leading zeros. The default value for this entry is 16.

21 Positions 21-24 contain the maximum number of ddnames supplied by succeeding DB0 or DB1 control statements. This value must be in the range 1-9,999 and must be left-justified or supplied with leading zeros. The default value for this entry is 80.

31 Positions 31-33 contain the maximum length root sequence field contained within the log records to be processed as a result of a DB0 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 1-236 and must be left-justified or supplied with leading zeros. The default value for this entry is 10.

- DB0 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 options that are available are described in the following section:

Note: Each combination of data base name/ddname may appear on one control statement only.

```
1   4          12          21      31      41      51      61        80
┌──────────────────────────────────────────────────────────────────────┐
│                                                                      │
│ DB0 dbname   yydddhhmm   dd1     dd2     dd3     dd4     dd5         │
│     ␣*ALL                                                            │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘
```

Position	Description
1	Positions 1-3 must contain the characters DB0.
4	Positions 4-11 can contain a data base name or a ␣*ALL where position 4 is blank and positions 5-8 contain the characters *ALL. If *ALL is specified, all records will be accumulated, and the purge date and time in positions 12-20 will be applied to all records. 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 12-20 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.

21	Positions 21-28, 31-38, 41-48, 51-58, and 61-68 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 DB0 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 4-11, 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
┌──────────────────────────────────────────────────────────────────────┐
│      dbname                                                          │
│ DB1  ⌀*ALL    yydddhhmm   dd1     dd2     dd3     dd4     dd5       │
│      ⌀*OTHER                                                         │
└──────────────────────────────────────────────────────────────────────┘
```

Position	Description
1	Positions 1-3 must contain the characters DB1.
4	Positions 4-11 can contain a data base name, ⌀*ALL or ⌀*OTHER. If *ALL is specified, all records are written to the new log data set, and no DB0 control statements may be included. If *OTHER is specified, all records not described by a DB0 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 DB0 statement, it cannot also be specified on a DB1 statement. Only one DB1 statement specifying *OTHER can be supplied.
12	Positions 12-20 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 21-28, 31-38, 41-48, 51-58, and 61-68 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 4-11, 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, the number of data bases = 16, the number of ddnames = 80, and a maximum prime key length of 10 are assumed.
- If no DB0 or DB1 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 DB0 statement is used, the user may not also use a DB1 *ALL statement.

RETURN CODES

The Data Base Change Accumulation utility provides the following return codes:

Code	Meaning
0	Successful completion.
16	Unsuccessful completion.

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, 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
//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=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//DFSUCUM0  DD  DUMMY,DCB=BLKSIZE=100
//DFSUCUMN  DD  DSN=IMSVS.CUM1,UNIT=2400,VOL=SER=CUMTAP,DISP=(,KEEP)
//DFSUDD1  DD  DSN=IMSVS.NEWLOG,UNIT=2400,VOL=SER=LOGTAP,DISP=(,KEEP)
//DFSULOG  DD  DSN=IMSVS.LOG1,UNIT=2400,VOL=SER=LTAPE1,DISP=OLD
//         DD  DSN=IMSVS.LOG2,UNIT=2400,VOL=SER=LTAPE2,DISP=OLD
//SYSIN  DD  *
ID       002       002        030
DB0DI32DB01731751200
DB0DI32DB02731731500DDI3IA
DB1DI32DB02731731500DDI3OA
DB1 *OTHER
/*
```

Example 2

In this example, all data base change records are to be accumulated. The ID statement in this example specifies that no DBD name or ddnames are to be supplied. 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. The sort table is to be incremented by 10K instead of the default size of 20K. 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,INCR=10000',REGION=150K
//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=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD  UNIT=2314,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD  DSN=IMSVS.CUM1,UNIT=2400,VOL=SER=CUMTAP,DISP=OLD
//DFSUCUMN  DD  DSN=IMSVS.CUM2,UNIT=2400,VOL=SER=CUMTP2,DISP=(,KEEP)
//DFSUDD1  DD  DUMMY
//DFSULOG  DD  DSN=IMSVS.LOG1,UNIT=2400,VOL=SER=LTAPE3,DISP=OLD
//         DD  DSN=IMSVS.LOG2,UNIT=2400,VOL=SER=LTAPE4,DISP=OLD
//SYSIN  DD  *
ID        000      000        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.

The Data Base Recovery utility achieves a high rate of throughput by manipulating data sets individually, on a physical data replacement basis in a physical sequential manner. The basic input consists of an image copy data set, an accumulated change data set, and a log tape. (The accumulated change data set and log tape are optional.)

The recovery operation basically consists of two phases -- merging and updating. In the first phase, an image copy data set is merged with an Accumulated Change data set to create a restored or partially restored data set. In the second phase, the partially restored data set is randomly updated with log input.

It should be noted that the procedure described above is not the only approach for data base recovery. The Data Base Recovery utility accepts the following input:

- Image copy only input.
- Image copy and log input.
- Image copy and change accumulation input.
- Image copy and change accumulation and log input.
- Log only input.

  Note: Log only input might be used after a complete volume has been restored (using OS/VS dump/restore).

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.

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.

A flow diagram of the Data Base Recovery utility is shown in Figure 5-4.

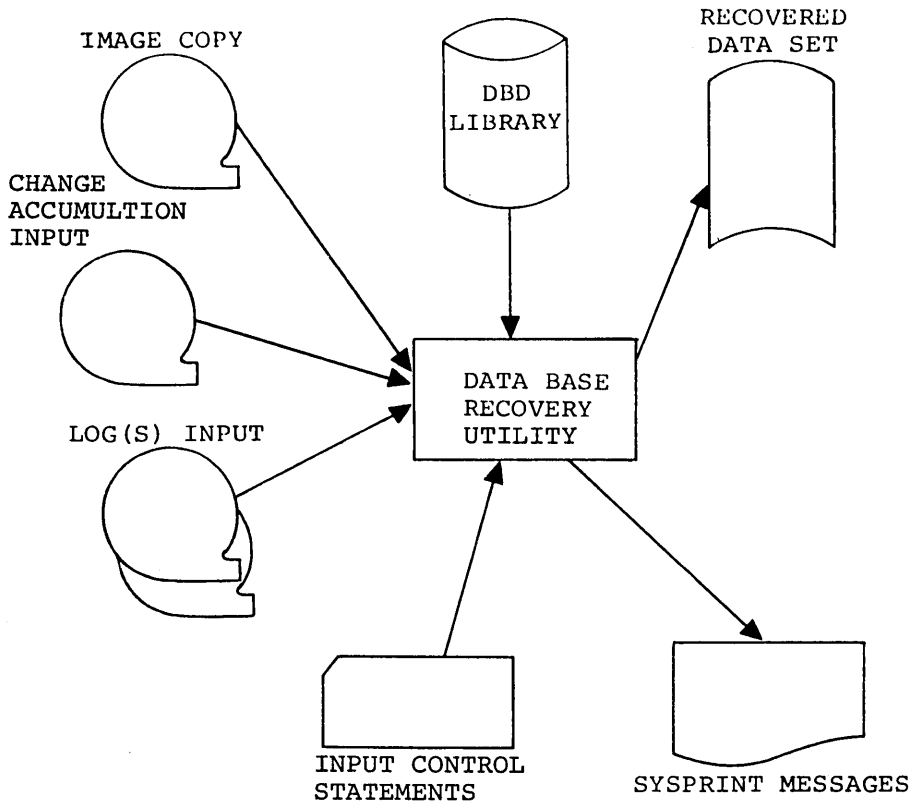


Figure 5-4. Data Base Recovery Utility

Notes:

1. The image copy can be a dump created by the Data Base Image Copy utility or, in the case of HISAM, a reorganization dump created by the HISAM Reorganization Unload utility.

2. The Change Accumulation input can be supplied only if the image copy input is supplied.

3. 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.

4. Only one data set recovery is allowed for each execution.

5. SYSPRINT can be blocked but must be a multiple of 121.

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 prior to 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.

TRACK RECOVERY OPTION

Data sets that are stored using VSAM can be recovered with the track recovery option (TRCV). 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.

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 since log tapes are not used with the track recovery option. 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 the IMS/VS Installation Guide for additional parameters that can be specified in executing a batch processing region.)

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 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. 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. Standard labels must be used.

DFSUCUM 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 may be supplied. This data set can reside on tape or a direct access volume. Standard labels must be used.

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. Standard labels must be used. This data set is not used with the track recovery option.

Multiple log tapes 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.)

ALTDD DD — Defines a work data set to be used by the TRCV option. This statement is necessary only if one or more control cards specify the TRCV option by means of a 'T' in the second column. This data set may reside on tape or a direct access device and uses DCB=LRECL=80. BLKSIZE may be specified on the DD statement. DISP=MOD must not be specified for this data set.

dataset1 DD — Defines the data set to be recovered. The ddname must be the same as the ddname in the DBD that describes this data set. It must also appear on the utility control statement. This data set must reside on a direct access volume.

DFSVSAMP DD — Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required if the data bases described by the dataset1 DD statements are VSAM data sets. (For additional information, see the discussion on "Defining the IMS/VS VSAM Buffer Pool" in the 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.

ALTPRINT DD — Defines the output message data set used by IEHATLAS. The data set can reside on a tape, direct access volume, or printer, or be routed through the output stream. ALTPRINT can be blocked but must be a multiple of 121.

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 the utility control statement contains a "T" in position 2 (see the discussion that follows). The DSNAME= operand must specify the name of the data set as found in a Format 1 DSCB in the VTOC. This name may be different from that of the "cluster name"; check your DEFINEs and VTOC listings.

UTILITY CONTROL STATEMENT

```
 1  2  3  4       12 13       22         31                              80
┌──────────────────────────────────────────────────────────────────────────┐
│                                                                          │
│ S  T  M  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 "M". If it is blank, an alternate track is assigned only if a write error occurs on the error track. If "M" is specified, the recovery program unconditionally gives control to the IEHATLAS program to determine whether an alternate track is to be assigned for each error track referenced in the data set.
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 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	Positions 31-80 can optionally contain comments.

RETURN CODES

The Data Base Recovery utility provides the following returns codes:

Code	Meaning
0	Requested recovery successful.
16	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'
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP DD  DSN=IMS.DBAOUT1,UNIT=2400,LABEL=(,SL),
//    VOL=SER=DBDMP1,DISP=(OLD,KEEP),DCB=BUFNO=10
//DFSUCUM DD  DSN=IMS.CUM1,UNIT=2400,LABEL=(,SL),
//    VOL=SER=DBCUM1,DISP=(OLD,KEEP),DCB=BUFNO=10
//DFSULOG  DD  DUMMY
//DBHI3A   DD  DSN=IMS.DBHI3A,UNIT=2314,
//    VOL=SER=DBASE1,DISP=(NEW,KEEP),
//    SPACE=(CYL,(20,,2)),DCB=(DSORG=IS,BUFNO=10)
//SYSIN  DD    *
S  DI32DB01 DBHI3A        RECOVERY FOR ISAM DATA SET
/*
```

Note: DUMP and accumulation change DD statements specify 10 buffers. This is not a requirement, but performance can be enhanced by specifying additional buffers. 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.

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 tapes. 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'
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A
//DUMPDS DD  DSN=IMS.DBBOUT1,UNIT=2400,LABEL=(,SL),
//   VOL=SER=DBDMP3,DISP=(OLD,KEEP),DCB=BUFNO=10
//DFSUCUM DD  DUMMY
//DFSULOG  DD  DSN=IMSLOG.MONDAY,UNIT=2400,LABEL=(,SL),
//   VOL=SER=LOG1,DISP=(OLD,KEEP),DCB=BUFNO=10
//         DD  DSN=IMSLOG.TUESDAY,UNIT=2400,LABEL=(,SL),
//   VOL=SER=LOG2,DISP=(OLD,KEEP),DCB=BUFNO=10
//DBHD3B   DD  DSN=IMS.DBHD3B,UNIT=2314,VOL=SER=DBASE2,
//   SPACE=(CYL,(20,10)),DISP=(NEW,KEEP),DCB=BUFNO=10
//SYSIN  DD  *
S  DD32DB01 DBHD3B  DUMPDS   RECOVERY FOR HDAM OSAM DATA SET
/*
```

Note: The log tape input can be concatenated but must be in date and time sequence.

Example 3

This example shows the JCL for recovering a VSAM data set with a ddname of DBVI3A in a data base named DI32DB01. Input is provided from the image copy and change accumulation change data sets.

```
//RECOVERY  JOB  1,1,MSGLEVEL=1
//STEP1  EXEC  PGM=DFSRRC00,PARM='UDR,DFSURDB0,DI32DB01'
//IMS      DD  DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT   DD  SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP DD  DSN=IMS.DBCOUT1,UNIT=2400,LABEL=(,SL),
//   VOL=SER=DBDMP1,DISP=(OLD,KEEP),DCB=BUFNO=10
//DFSUCUM DD  DSN=IMS.CUM1,UNIT=2400,LABEL=(,SL),
//   VOL=SER=DBCUM1,DISP=(OLD,KEEP),DCB=BUFNO=10
//DFSULOG  DD  DUMMY
//DBVI3A   DD  DSN=IMS.DBVI3A,DISP=OLD
//DFSVSAMP   DD  *
1024,10
/*
```

DATA BASE BACKOUT UTILITY (DFSBB000)

The Data Base Backout utility program facilitates the restoration of data bases from a point when updates were applied to a point before the program was initiated, or to a checkpoint. 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 bases used by the PSB must be online and available to the Backout utility.

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 the "IMS/VS System Execution" chapter of the IMS/VS Operator's Reference Manual.

A log tape is created during any backout process. This log tape must be included in any subsequent data base recovery attempt made against any of the data bases involved in the backout. The log tape 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 tape must not be used as input to any subsequent backout attempt.

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.

It should be noted that with multiple concatenated data sets using the data base backout utility, a LIFO (last in, first out) sequence is used. For a multiple volume data set, the VOL parameter of the JCL statement specifies the volumes in ascending sequence.

Figure 5-5 presents a summary of conditions that terminate the processing of the Data Base Backout utility.

Summary of Conditions Terminating the Processing of the Data Base Backout Utility

Region Being Backed Out	CHKP-ID specified	CHKP-ID not specified
MSG/BMP	1. If in PI, it is the first CHKP encountered; otherwise, it is the CHKP requested.	1. First CHKP record encountered.*
	2. Scheduling or Termination record for this PSB. 3. Message GU record (single mode).	
DL/I/DBB	1. CHKP record requested.	1. First CHKP record encountered.*
	2. Scheduling record for this PSB. 3. The Accounting record for opening the Log.	

* Note: The order of occurrence is referenced as from the end of the Log tape toward the start of the Log tape. (Read-backward processing.)

Figure 5-5. Conditions That Terminate the Data Base Backout Utility

Figure 5-6 depicts the data set requirements for the Data Base Backout utility.

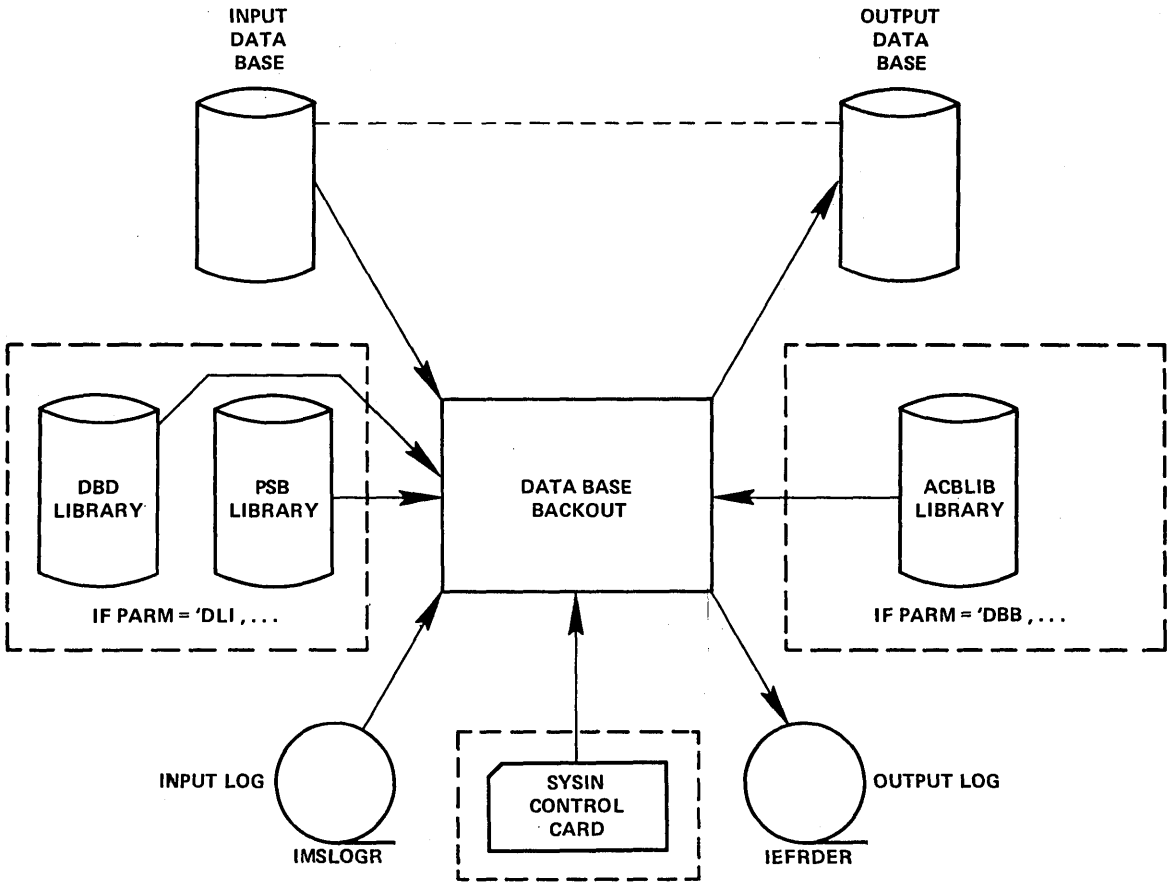


Figure 5-6. Data Set Requirements for the Data Base Backout Utility

JCL REQUIREMENTS

The Data Base 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 blocks required for the data base buffer pool.

See the IMS/VS Installation Guide for additional information on executing a batch processing region.

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 must be a tape file; read-backward is used. If the System Log Terminator was used to close the log tape, the Data Base Backout Utility must use the following parameters: LABEL=(2,BLP),DCB=(BLKSIZE=nnnn,LRECL=nnn,DSORG=PS, RECFM=U)
IEFRDER DD	Describes the system log tape created during backout. The data set usually resides on tape; however, a direct access volume can be used.
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.
SYSIN DD	Required only if a CHKP 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.

UTILITY CONTROL STATEMENT

Backout of a batch region normally terminates at the first checkpoint encountered on the input log tape (processed backwards). If it is necessary to backout to an earlier checkpoint, the CHKPT control statement identifies this earlier checkpoint.

```
1     7          16                                              80
┌──────────────────────────────────────────────────────────────────┐
│                                                                  │
│ CHKPT AAAAAAAA [COMMENTS]                                        │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

Position	Description
1	Positions 1-5 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 'CHKP' call. The ID is displayed as part of message DFS681I at the time the 'CHKP' call is made.
16	Positions 16-80 can optionally contain comments.

RETURN CODES

The Data Base 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)
28	No data base record in log. (DFS888I)
32	No blksize for IMSLOGR DD statement. (DFS890I)
36	Invalid record on input log. (DFS894I)
40	Incomplete spanned record on input log. (DFS896I)
44	No CHKPT ID on control statement. (DFS957I)
48	Specified CHKPT not found. (DFS958I)
52	Specified CHKPT not within last schedule. (DFS959I)

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 the program which uses PSB BIMALM01.

```
//LMBTCHB0 JOB 8,L,MSGCLASS=3,MSGLEVEL=(1,1)
//EXAMPLE   EXEC   PGM=DFSRRC00,PARM='DBB,DFSBB000,BIMALM01,004'
//IMSLOGR  DD  DSN=IMS.LOG,DISP=(OLD,KEEP),UNIT=(2400,,DEFER),
// VOL=SER=(FERRO,KAPLIK)
//IEFRDER  DD  DSN=IMS.LOG,UNIT=(2400,,DEFER),DCB=DSORG=PS,
// DISP=(NEW,KEEP)
//IMSACB   DD  DSN=IMSVS.ACBLIB,DISP=SHR
//CASE21   DD  DSN=ICS.CASE13(PRIME),DISP=SHR
//CASE21OF DD  DSN=ICS.CASE13OF,DISP=SHR
//LEV22    DD  DSN=ICS.LEVL22(PRIME),DISP=SHR
//LEV22OF  DD  DSN=ICS.LEVL22OF,DISP=SHR
//SYSIN    DD  *
CHKPT C0000001 CHECKPOINT NO. 1
/*
```

Note: The first parameter (DBB) can be replaced with DLI if dynamic block building is desired. If this replacement is made, the IMSACB DD statement should be replaced with the following:

```
//IMS  DD  DSN=IMSVS.PSBLIB,DISP=SHR
//     DD  DSN=IMSVS.DBDLIB,DISP=SHR
```

CHAPTER 6. 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 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 a root segment.
   - 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:

   - If the root sequence field is non-unique, 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 unique 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 since they will be reinserted by subsequent user/reload insert. This will occur only until a non-duplicate root is found. Once a segment with a new key or with different data is encountered, LB status codes will be returned for any subsequent duplicates.

3. The user must explicitly close any data sets he 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 (DB1 before DB2, 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.

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.

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:

Change Accumulation
Recovery
HISAM Reorganization Unload
HISAM Reorganization Reload
Data Base Scan
HDAM Reorganization Unload
HDAM Reorganization Reload
Initial Load
Prefix Resolution
Prefix Update
Reorganization Unload for Secondary Indexes
Reorganization Reload for Secondary Indexes
Image Copy
Data Base Zap

Processing continues 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 zeroes, restart processing begins from the point of termination.
- If the information is zeroes, restart processing is from the beginning of the program.

Since the user's program should respond with the next root or dependent to be loaded, the user I/O files may have to be adjusted by the application program.

The actual segment data is also provided since further inspection might be necessary to determine the restart point on the input files. (This would be important, for example, if there are non-unique 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	Meaning is 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 his 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 hex zeroes 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

Error-checking 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. 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 2,000 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.

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

User exits in all the utilities allow the user to examine records or to compile statistics. The user exits are specified (named) on utility control statements associated with a given function. No IMS/VS control data can be altered through these user exits. Only user data can be changed in an existing segment; length requirements must not be violated. With the HD Reorganization Unload utility (DFSURGU0) and the HD Reorganization Reload utility (DFSURGL0), exits can be used to delete or insert segments or to view blocks after loading. All IMS/VS data must be maintained by the user during user exit routine processing.

Since the user exit program is loaded before the utility is attached, it must reside in the LINKLIB or be defined in a STEPLIB or JOBLIB data set. The interface for the exit routines is a pointer in register 1 to the parameter list. The parameter list itself contains a pointer to the segment or record or to the value "0", a pointer to the DFSPRINT DCB, and a pointer to the Partition Specification Table (PST). If the first entry in the parameter list is "0", it indicates that processing is completing normally or for any reason other than an abend and that this is the final entry to the exit routine.

When running the HD Reorganization Reload utility, for example, control is passed to a single exit routine at one of two locations:

- at offset 0 when the reload record has been read
- at offset 4 when the record has been loaded

The user may write on the DFSPRINT data set by issuing a PUT macro statement for a "MOVE MODE" operation. This data set is opened prior to entry into the user exit routine.

For user exit processing, return codes are recognized in the HD Reorganization utilities to allow the user to inform the utility of segment disposition. These return codes 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.

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 under his 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 the OS/VS2 System Programming Library: Service Aids, GC28-0674.)

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 SU RU DU	SR DR RR 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.

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".

```
@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	Identifiers
FUNCTION xx END	FUNCTION= SEQ=
STATUS	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 is being processed.

- 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 JCL DD Statement Summary Table at the end of this section and 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 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 the IMS/VS System Programming Reference Manual.)

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=1008, and LRECL=300. DISP=KEEP must be specified by the user.

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=300, and BLKSIZE=1008. 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 since 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 data sets. (For additional information, see the discussion on "Defining the IMS/VS VSAM Buffer Pool" in the 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 Preface 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, prefix resolution, or prefix update 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=300, and BLKSIZE on this DD statement. 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).

The Prefix Resolution utility defaults the ddname of this data set to SORTIN. The SORTIN default can be overridden to DFSURWF1 by using the INDDS parameter on a FUNCTION=PR statement. This allows Prefix Resolution to use the same DD cards as the other utilities.

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=300. 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=300. 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. If this DD statement is included, the BLKSIZE parameter must be specified.

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 change accumulation.

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.

DFSUCUMO (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 data base recovery.

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 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.

ALTDD
DD

Defines the work data set to be used by the TRCV option. This DD statement is necessary only if one or more control statements request the track recovery option by specifying REQUEST=T. This data set contains control statements passed to the IEHATLAS program to assign alternate tracks. It can reside on a tape or direct access device. The DCB is defined with LRECL=80 and RECFM=FB. BLKSIZE can be specified on this DD statement and is usually BLKSIZE=800. DISP=MOD must not be specified.

ALTPRINT
DD

Defines the output message data set used by IEHATLAS. The data set can reside on a tape, direct access volume, or printer, or be routed through the output stream. This data set is specified in the program with the DCB operand LRECL=121. Blocksize can be specified on this DD statement.

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).

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 Image Copy). Input files used by the Data Base Recovery and Reorganization Reload utilities must also be defined.

JCL STATEMENTS SUMMARY TABLE

	*UCF	FUNCTION=Keywords Used on Utility Control Statements																	
		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										
SORTLIB			R						R										
SORTWKnn			R						R										
DFSURWF1				R		R	R		O					R					
DFSURWF2									R										
DFSURWF3									R	R									
DFSURIDX												R							
DFSEXTDS												E							
DFSULOG			O										R						
DFSUDD1			O																
DFSUCUMN			O																
DFSUCUMO			O																
DFSUDUMP													O						
DFSUCUM													O						
ALTDD													T						
ALTPRINT													T						
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.

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.

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 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

This control statement establishes certain UCF run specifications. The format of the statement is:

FUNCTION=OP

```
[ [,COND={RESTART}]]
  [      {CDS    }]
  [      {NORM   }]

[,REQUEST=(({[STATS  ]}[,SUMM] [,MSGALL]))]
           {[NOSTATS]}

[,CKPNT={0      }]
        {1-999999}
        {2000   }

[,MSGNUM=(ccc,...) ]
```

where:

COND={RESTART / CDS / NORM}
can only be used once per 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 / 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.

REQUEST=SUMM
specifies that a summary of the statistics is to be printed after the functional utility completes processing.

REQUEST=MSGALL
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={0 / 1-999999 / 2000}
specifies the checkpoint interval to be used as the default specification during the UCF run. 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=(ccc,...)
is used to set the error point abend flags. The value "ccc" is the message number extracted from the message identifier "DFScccI" 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 log tapes 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
                  [,SEQ=nnn]
                  [,REQUEST={STATS                                  }]
                            {NOSTATS                                }
                            {specified-on-option-statement          }
                  [,EXEC={STOP}]
                         {YES }
                         {NO  }
                  [,IDLEN={1-236}]                                    Note 1
                          {10    }
                  [,DBNUM={1-999}]                                    Note 1
                          {16    }
                  [,DDNUM={1-999}]                                    Note 1
                          {80    }
                  [                                                   Note 2
                    ,DBNAME={(dbname,...),RECID={0}}                  ]
                            {*ALL               {1}                   |
                            {*OTHER,RECID=1                           |
                    [,PURGDT={yyddd}][,PURGTM={hhmm}]                 |
                             {00000}          {0000}                  |
                    [,DBDDDS=(ddname-1[,...,ddname-5])]               ]
                  [,LOGIN={DFSULOG}]
                          {ddname }
                  [,LOGOUT={DFSUDD1}]
                           {ddname }
                  [,CUMIN={DFSUCUMO}]
                          {ddname  }
```

```
[,CUMOUT={DFSUCUMN}]
         {ddname  }

 [,OUTDDS=(sort-work-file,...)]

 [,EXITRTN=exit-module-name]
```

Note 1: This keyword can be specified only once per UCF job step.

Note 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=nnn
links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS   }
        {NOSTATS   }
        {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.

EXEC={STOP}
     {YES }
     {NO }
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.

CUMIN={DFSUCUMO}
      {ddname   }
defines the ddname on the Change Accumulation data set used as input to the referenced utilities.

CUMOUT={DFSUCUMN}
       {ddname   }
defines the ddname of the new Change Accumulation data set.

IDLEN={1-236}
      {10   }
specifies the maximum length of the root-sequence field within log records to be processed. The value must be in the range 1-236. 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 per UCF job step.

DBNUM=|1-999|
|16 |

specifies the maximum number of data base names which may be specified for the DBNAME keyword for FUNCTION=CA during the execution of UCF. This keyword corresponds to the "number of DBs" field on the "ID" utility control statement in the Data Base Change Accumulation utility. The default value is 16. This keyword can be specified only once per UCF job step.

DDNUM=|1-999|
|80 |

specifies the maximum number of ddnames that may be specified for all FUNCTION=CA control statements. This keyword corresponds to the "number of DDs" field on the "ID" utility control statement in the Data Base Change Accumulation utility. The default value is 80. This keyword can be specified only once per UCF job step.

DBNAME={dbname}
{*ALL }
{*OTHER}

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. The number of DBNAME=dbname entries for the FUNCTION=CA control statement cannot exceed the value specified for the DBNUM keyword. 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 per 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=|0|
|1|

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 which 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=|yyddd|
|<u>00000</u>|
specifies a purge date in the form "yyddd" where "yy" (00-99) is the year and "ddd" (000-365) 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=|hhmm|
|<u>0000</u>|
specifies a purge time in the form "hhmm" where "hh" (00-23) is the hour of the day and "mm" (00-59) 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.

DBDDDS=(ddname-1[,...,ddname-5])
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=|<u>DFSULOG</u>|
|ddname|
defines the ddname of the IMS/VS system log input for Change Accumulation.

LOGOUT=|<u>DFSUDD1</u>|
|ddname|
defines the ddname of the new data base log data set to be created on a Change Accumulation run.

OUTDDS=(sort-work-file,...)
allows the user to verify that all sort-work-file data sets are defined before executing this fucntion. 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=exit-module-name
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]

                   [,REQUEST=(([[STATS                                ]  [,SUMM]])
                              ([[NOSTATS                              ]       ]))
                              ([[specified-on-option-statement        ]       ])

                   [,EXEC=|STOP|]
                          |YES |
                          |NO  |

                   [,CKPNT=|0                                 |]
                           |1-999999                          |
                           |specified-on-option-statement      |
                           |2000                               |

                   [,EXITRTN=exit-module-name]

                   [,DBDDS=(ddname[,...])]

                   [,WF1DDS=|DFSURWF1|]
                            |ddname  |
```

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.

INDDS=|DFSUINPT|
|reload-file-ddname|
: 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.

SEQ=nnn
: links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS | NOSTATS | 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=SUMM

specifies that a summary of statistics is to be printed upon completion of this functional utility.

EXEC={STOP | YES | NO}

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={0 | 1-999999 | specified-on-option-statement | 2000}

specifies a user-supplied checkpoint interval used during the execution of this function. This value is equal to the number of root segments. 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=exit-module-name

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=(ddname,...)

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={DFSURWF1 | ddname}

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=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.

SEQ=nnn

links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS|NOSTATS|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.

EXEC={STOP|YES|NO}

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={0|1-999999|specified-on-option-statement|2000}

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 checkpoints are not to be taken for this function.

EXITRTN=exit-module-name

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.

DBDDS=(ddname,...)

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=(([DFSURGU1|ddname-1])[ DFSURGU2|,ddname-2])]
                   [,INDDS=[DFSURGU1|ddname]]
                   [,SEQ=nnn]
                   [,REQUEST=(([STATS|NOSTATS|specified-on-option-statement][,SUMM]))]
                   [,EXEC=[STOP|YES|NO]]
                   [,CKPNT=[0|1-999999|specified-on-option-statement|2000]]
                   [,EXITRTN=exit-module-name]
                   [,EXITRLD=reload-exit-module-name]
                   [,DBDDDS=(ddname,...)]
                   [,WF1DDS=[DFSURWF1|ddname]]
```

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.

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 DX functions are included in one execution of UCF, each must specify a unique output data set.

INDDS=reload-file-ddname
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.

SEQ=nnn
links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS / NOSTATS / 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=SUMM
specifies that a summary of statistics is to be printed upon completion of this functional utility.

EXEC={STOP / YES / NO}
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={0 / 1-999999 / specified-on-option-statement / 2000}
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 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=unload-exit-module-name
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.

WF1DDS=|DFSURWF1|
|ddname |
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 consistant across all references to this data set within an execution of the UCF.

EXITRLD=reload-exit-module-name
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=(ddname,...)
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=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=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=loadpgm
: 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 since it is located by BLDL and ATTACH commands.

ILPSBNAM=psbname
: 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 per control statement.

SEQ=nnn
: links the ZAP functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

EXEC={STOP|YES|NO}
: 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={0|1-999999|specified-on-option-statement|2000}
: 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. 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=(dsname,...)
: 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=|DFSURWF1|
       |ddname |
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.

DBDDS=(ddname,...)
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=exit-module-name
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 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
                    ,DBNAME=dbname

                    ,OUTDDS=(ddname-1[,ddname-2])

                    ,DBDDS=image-copy-input-ddname

                    [,SEQ=nnn]

                    [,REQUEST=((|[STATS                                ]|[,I])|))]
                               |[NOSTATS                                ]|
                               |[specified-on-option-statement           ]|

                    [,EXEC=|STOP|]
                           |YES |
                           |NO  |

                    [,CKPNT=|0                                  |]
                            |1-999999                           |
                            |specified-on-option-statement       |
                            |2000                               |

                    [,EXITRTN=exit-module-name]
```

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.

OUTDDS=(ddname-1[,ddname-2])
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=image-copy-input-ddname
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=nnn
links the zap functions (ZB and ZM) to this functional control statement by using a matching-sequence identifier.

REQUEST={STATS / NOSTATS / 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=I specifies an image copy of an index of a KSDS. If not specified, the DBDDDS keyword value must refer to a KSDS.

EXEC={STOP / YES / NO}
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={0 / 1-999999 / specified-on-option-statement / 2000}
specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records. 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=exit-module-name
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.

This statement is optional since 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={SORTIN|ddname}]

                  [,SEQ=nnn]

                  [,REQUEST=({[STATS|NOSTATS|specified-on-option-statement] [,SUMM]})]

                  [,EXEC={STOP|YES|NO}]

                  [,DBNAME=dbname]

                  [,EXITRTN=exit-module-name]

                  [,OUTDDS=(sort-work-file,...)]
```

where:

INDDS={SORTIN|ddname}
: specifies that the SORTIN 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=nnn
: links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS | NOSTATS | 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=SUMM

specifies that a summary of statistics is to be punched upon completion of this functional utility.

EXEC={STOP | YES | NO}

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=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.

EXITRTN=exit-module-name

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=(sort-work-file,...)

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 (DFSURGPO). 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 since 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]

             [,REQUEST=(({[STATS                          ]}[,SUMM]))]
                          [NOSTATS                         ]
                          [specified-on-option-statement   ]

             [,EXEC={STOP}]
                    {YES }
                    {NO  }

             [,CKPNT={0                             }]
                     {1-999999                      }
                     {specified-on-option-statement }
                     {2000                          }

             [,DBNAME=dbname]

             [,EXITRTN=exit-module-name]

             [,INDDS={DFSURWF3                          }]
                     {ddname-from-Prefix-Resolution-run }

             [,DBDDDS=(ddname,...)]
```

(Defaults underlined: STATS, YES, 2000, DFSURWF3.)

where:

SEQ=nnn
: links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS | NOSTATS | 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=SUMM
: specifies that a summary of statistics is to be printed upon completion of this functional utility.

DBNAME=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.

EXEC={STOP / YES / NO}
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={0 / 1-999999 / specified-on-option-statement / 2000}
specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of input work records. 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=exit-module-name
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={DFSURWF3 / ddname-from-Prefix-Resolution-run}
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).

DBDDDS=(ddname,...)
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 statment 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=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=ddname
: 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=|DFSUIN01 |
: |(reload-file-ddname,...)|
: 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 must reside on a direct access device.

DBDDDS=ddname
defines the ddname of the overflow (ESDS) data base data set required by this function. Only one DBDDDS can be specified per control statement.

SEQ=nnn
links the zap functions (ZB and ZM) to this functional control statement by using a matching-sequence identifer.

REQUEST={STATS / NOSTATS / specified-on-option-statement}
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={STOP / YES / NO}
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.

EXITRTN=exit-module-name
specifies an exit routine to be given contrcl 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.

CKPNT={0 / 1-999999 / specified-on-option-statement / 2000}
specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records. 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.

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:

```
FUNCTION=RU
                ,DBNAME=dbname

                ,KDSDD=ddname

                ,OUTDDS=(ddname-1[,ddname-2])

               [,DBDDDS=(ddname,...)]

               [,REQUEST=(([{EXTRACT}])[{ STATS                          }])]
                          [{MERGE  }]  [{,NOSTATS                        }]
                          [{REPLACE}]  [{specified-on-option-statement   }]

                [,SICON=character]                          Note 1

                [,SEQ=nnn]

                [,EXEC={STOP}]
                       {YES }
                       {NO  }

                [,CKPNT={0                                 }]
                        {1-999999                          }
                        {specified-on-option-statement     }
                        {2000                              }

                [,EXITRTN=exit-module-name]

                [,COPY={1}]
                       {2}

                [,EXTRACT={DFSEXTDS}]                       Note 2
                          {ddname  }

                [,IDXIN={DFSURIDX                }]
                        {secondary-index-ddname  }
```

Notes:

1. Required if either REQUEST=EXTRACT or REQUEST=REPLACE is specified.

2. Can only be specified with REQUEST=EXTRACT.

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.

KDSDD=ddname
defines the ISAM/VSAM KSDS 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=(ddname-1[,ddname-2])
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=(ddname,...)
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=nnn
links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS / NOSTATS / 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={EXTRACT / MERGE / 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.

EXEC={STOP / YES / NO}
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={0 | 1-999999 | specified-on-option function | 2000}

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=exit-module-name

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={1 | 2}

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={DFSEXTDS | ddname}

defines the ddname of the EXTRACT data base data set and must be specified if REQUEST=EXTRACT is used.

IDXIN={DFSURIDX | secondary-index-ddname}

describes the output data set (DFSURIDX) from the Prefix Resolution utility which contains secondary index information. This keyword is required and can be specified only once per control statement. This keyword corresponds to the index DD statement in the HISAM Reorganization Unload utility.

SICON=character

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.

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

               [,INDDS=|DFSUDUMP                                    |]
                       |image-copy-ddname                           |

               [,LOGIN=|DFSULOG|]
                       |ddname |

                [,SEQ=nnn]

               [,REQUEST= (|[|STATS                                 |]|[ {,T[ ,M ][ ,I ]} ]|)]
                           |[|NOSTATS                                |]|                    |
                           |[|specified-on-option-statement          |]|                    |

                [,EXEC=|STOP|]
                       |YES |
                       |NO  |

                [,CKPNT=|0                                          |]
                        |1-999999                                   |
                        |specified-on-option-statement               |
                        |2000                                        |

                [,CUMIN=|ddname |]
                        |DFSUCUM|

               [,EXITRTN=exit-module-name]
```

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
: 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=recovered-data-set-ddname
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=|DFSUDUMP|
|image-copy-ddname|
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.

LOGIN=|DFSULOG|
|ddname|
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.

SEQ=nnn
links the zap functions (ZB and ZM) to this functional control statement by using a matching-sequence identifier.

REQUEST={STATS | NOSTATS | 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[,M][,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.

"M" specifies that alternate tracks are to be assigned unconditionally for both read and write operations. If "M" is not specified and "T" is specified, alternate tracks are assigned only if a write error occurs on the error track. This parameter is specified with the "T" parameter and is invalid if specified alone.

"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={STOP | YES | NO}
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={0 | 1-999999 | specified-on-option-statement | 2000}

specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of log records. 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={ddname | DFSUCUM}

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.

EXITRTN=exit-module-name

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 since it is automatically generated by the UCF for normal processing. The format of the FUNCTION=SN control statement is:

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.

DBDDS=(data-base-data-sets,...)
: 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=[DFSURWF1 / ddname]

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=segment-name

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 per complete control statement.

SCANTYP=[SEQ / SEG]

defines the order of scan to be performed on the DBNAME associated with this same control statement. This keyword can be specified only once per 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=nnn

links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST=[STATS / NOSTATS / 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, STATS is the default.

EXEC=[STOP / YES / NO]

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=[0 / 1-999999 / specified-on-option-statement / 2000]

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. 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=exit-module-name

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 (DFSURRLO). 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

               [,INDDS={DFSUIN01          }]
                       {(reload-file-ddname,...)}

               [,DBDDS=(ddname,...) ]

               [,KDSDD=ddname]

               [,SEQ=nnn]

               [,REQUEST=(({[{STATS                                }] [,VSAM]}))]
                          {[{NOSTATS                               }]       }
                          {[{specified-on-option-statement         }]       }

               [,EXEC={STOP}]
                      {YES }
                      {NO  }

               [,CKPNT={0                                  }]
                       {1-999999                           }
                       {specified-on-option-statement      }
                       {2000                               }

               [,EXITRTN=exit-module-name]
```

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.

DBDDS=ddname
used to verify the existence of a data base data set that is to be reloaded.

KDSDD=ddname
defines the ISAM/VSAM KSDS 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).

INDDS={DFSUIN01 / reload-file-ddname}
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.

SEQ=nnn
links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS / NOSTATS / 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=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={STOP / YES / NO}
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.

```
        ⎧0
        ⎪1-999999
CKPNT=  ⎨specified-on-option-statement
        ⎩2000
```

specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records. 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=exit-module-name

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 which 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  ⎭

                    [,CKPNT=⎧0                              ⎫]
                            ⎪1-999999                       ⎪
                            ⎪specified-on-option-statement  ⎪
                            ⎩2000                           ⎭

                    [,EXITRTN=exit-module-name]

                    [,COPY=⎧1⎫]
                           ⎩2⎭
```

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.

KDSDD=ddname
defines the ISAM/VSAM KSDS 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=(ddname-1[,ddname-2])
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=(ddname,...)
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=nnn
links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS | NOSTATS | 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.

EXEC={STOP | YES | NO}
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={0 | 1-999999 | specified-on-option-statement | 2000}
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 checkpoints are not to be taken for this function.

EXITRTN=exit-module-name
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=|1|
|2|
provides for multiple copies of output data sets. COPY can be specified only once per 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:

```
FUNCTION=SX
          ,DBNAME=dbname
          ,KDSDD=ddname
          ,OUTDDS=(ddname-1[,ddname-2])
          [,INDDS=|DFSUIN01                |]
                  |(reload-file-ddname,...)|
          [,DBDDDS=(ddname,...)]
          [,SEQ=nnn]
          [,REQUEST=(|[|[|STATS                          |]][,VSAM]|)]
                     |[|[|NOSTATS                        |]]      |
                     |[|[|specified-on-option-statement|]]      |
          [,EXEC=|STOP|]
                 |YES |
                 |NO  |
          [,CKPNT=|0                                |]
                  |1-999999                         |
                  |specified-on-option-statement     |
                  |2000                              |
          [,EXITRTN=unload-exit-module-name]
          [,EXITRLD=reload-exit-module-name]
          [,COPY=|1|]
                 |2|
```

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.

KDSDD=ddname
defines the ISAM/VSAM KSDS 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.

DBDDDS=(ddname,...)
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.

OUTDDS=(unload-file-ddname-1[,unload-file-ddname-2])
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={DFSUIN01 | (reload-file-ddname)}
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.

SEQ=nnn
links the zap functions (ZB and ZM) to this control statement by using a matching-sequence identifier.

REQUEST={STATS | NOSTATS | 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=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={STOP / YES / NO}

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={0 / 1-999999 / specified-on-option-statement / 2000}

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. 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=unload-exit-module-name

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=reload-exit-module-name

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={1 / 2}

provides for multiple copies of output data sets. COPY can be specified only once per 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 his 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=8-hex-numbers|        Note 1
                    |,REP=8-hex-numbers   |

                    ,VALUE=compare/replace value

                    ,RBNID=dbblock

                    [,EXEC={STOP}]
                           {YES }
                           {NO  }

                    [                                    ]
                    [,RELATE={DU}                        ]
                    [        {DR}                        ]
                    [        {IM}                        ]
                    [        {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; make sure that the VERIFY precedes the REP since statements are not automatically sorted by the UCF.

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=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=ddname
: defines any ddname of a data base data set required by this function. Only one DBDDDS can be specified per control statement.

VERIFY=8-hex-numbers
: defines the hex offset (relative to zero) that the associated VALUE= field is to be compared against. If any verify fails, the entire zap terminates.

REP=8-hex-numbers
: defines the hex offset (relative to zero) that is to be replaced by the associated VALUE= data.

VALUE=compare/replace value
: defines the data (in hex 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 hex characters must be specified. This means that 4 bytes of data must be changed or compared per control statement. This keyword must be specified once per control statement.

RBNID=dbblock
: defines the data base block (in 2-byte hex representation) that is to be operated on by the associated VERIFY= or REP= data. 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 an 8-byte field. This value will be packed into a 4-byte field.

EXEC={STOP / YES / NO}
: 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=xx

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)		

SEQ=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=exit-module-name

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 his direction.

The format of the FUNCTION=ZM control statement is:

```
FUNCTION=ZM
             ⎧           ⎧CA⎫                  ⎫
             ⎪           ⎪DU⎪                  ⎪
             ⎪           ⎪DR⎪                  ⎪
             ⎪           ⎪IM⎪                  ⎪
             ⎪           ⎪PR⎪                  ⎪
             ⎪ ,MODULE=  ⎨PU⎬                  ⎪
             ⎨           ⎪RV⎪                  ⎬
             ⎪           ⎪SN⎪                  ⎪
             ⎪           ⎪SU⎪                  ⎪
             ⎪           ⎪SR⎪                  ⎪
             ⎪           ⎪RU⎪                  ⎪
             ⎪           ⎩RR⎭                  ⎪
             ⎩ ,MODULE=CF,CSECT=csectname      ⎭

             { ,VERIFY=8-hex-numbers }
             { ,REP=8-hex-numbers    }              Note 1

              ,VALUE=compare/replace value
              ⎡          ⎧CA⎫              ⎤
              ⎢          ⎪DU⎪              ⎥
              ⎢          ⎪DR⎪              ⎥
              ⎢          ⎪IM⎪              ⎥
              ⎢          ⎪PR⎪              ⎥
              ⎢ ,RELATE= ⎨PU⎬ [,SEQ=nnn]   ⎥    Note 2
              ⎢          ⎪RV⎪              ⎥
              ⎢          ⎪SN⎪              ⎥
              ⎢          ⎪SU⎪              ⎥
              ⎢          ⎪SR⎪              ⎥
              ⎢          ⎪CF⎪              ⎥
              ⎢          ⎪RU⎪              ⎥
              ⎣          ⎩RR⎭              ⎦

              [,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; make sure that the VERIFY precedes the REP since statements are not automatically sorted by the UCF.

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=xx
defines the load module to be zapped. This keyword can be specified only once per 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=csectname
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=8-hex-numbers
defines the hex offset (relative to zero) that the associated VALUE= field is to be compared with. If any verify fails, the entire zap terminates.

REP=8-hex-numbers
defines the hex offset (relative to zero) that is to be replaced by the associated VALUE= data.

VALUE=compare/replace value
defines the data (in hex 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 hex characters must be specified. This means that 4 bytes of data must be changed or compared per control statement. The VALUE keyword must be specified once per control statement.

RELATE=xx
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=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=exit-module-name
allow the user to obtain control at the time the zap has been executed.

KEYWORDS SUMMARY TABLES

The following table 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	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
DBNUM	1	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
DDNUM	1	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RECID	n	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
DBNAME	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	*	*	*	*	*
DBDDDS	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 <u>one</u> time and only if no other keyword with an "O" designation in this chart has <u>not</u> 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.

The following table 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	IL	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	
ILPSBNAM						r												
MODULE																		r
VERIFY																	r	r
REP																	r	r
VALUE																	r	r
RBNID																	r	

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.	Verify that warnings are not errors; if no errors exist, take same action as for return code 0. Otherwise, the DFSNJRNL and DFSNCDS msut have been saved.

8	Serious errors have occurred. UCF terminated.	Correct errors and begin restart processing. The DFSJRNL and DFSNCDS data sets must be available.
12	UCF failed and terminated with an unrecoverable error.	Correct the error and start the UCF from the beginning. The DFSJRNL and DFSNCDS data set must be available. Do not attempt restart processing.

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'
//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),DCB=BLKSIZE=1600
//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'
//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)),DCB=(RECFM=VB,BLKSIZE=1008,
//             LRECL=300)
//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'
//DFSPRINT    DD      SYSOUT=A
//IMS         DD      DSN=IMSVS.PSBLIB,DISP=SHR
//            DD      DSN=IMSVS.PSBLIB,DISP=SHR
//DFSNJRNL    DD      DSN=&&NJRNL1,UNIT=SYSDA,DISP=(,KEEP),
//             SPACE=(CYL,(2,2)),DCB=(RECFM=VB,BLKSIZE=1008,
//             LRECL=300))
//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'
//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),DCB=(RECFM=VB,LRECL=300,
//             BLKSIZE=1008)
//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=DFSUCUMO,LOGIN=DFSULOG,   *
      LOGOUT=DFSUDD1,DBDDDS=(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)
//DFSUCUMO    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 3. IMS/VS SYSTEM LOG UTILITIES

Part 3 has two chapters that describe the utilities used for restart, recovery, and analysis of the system log data.

Chapter 7, "Log Maintenance Utilities," describes the utilities used to produce a new and usable system log data set from one that contains read errors and the utility used to recover log data that was lost as a result of a system failure. Control statements and examples are provided.

Chapter 8, "Log Data Formatting Utilities," describes the utilities used for IMS/VS system log tape analysis and the types of reports that are produced. Control statements and examples and report examples are included.

CHAPTER 7. LOG MAINTENANCE UTILITIES

OVERVIEW

One of the basic components of the IMS/VS control program is the IMS/VS system log data set.

The information placed on the IMS/VS system log is used for many purposes. Some of these purposes are statistics, accounting, restart, and data base recovery. All input messages received and all output messages sent are recorded on the system 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. Accounting information about computer usage by application program can be derived from the application accounting record.

IMS/VS SYSTEM LOG

The IMS/VS control program includes a common service routine, the system recorder. This routine is designed to facilitate the placing of data on the system log. This data is used primarily for restart, recovery and offline statistical analysis (accounting, etc.). The following information is written:

a. For restart and recovery:

	Data	Log Code	When Written
1)	Checkpoint data. Batch checkpoint data.	40 41	When a checkpoint is taken.
2)	Record indicating an OS/VS data set OPEN or CLOSE.	20 21	When an IMS/VS data set, used for message processing, is opened or closed.
3)	Record indicating changes to a data base.	50 51 52	When a data base insert, delete, or replace is made.

b. For both restart and statistics:

1)	Message received from a terminal.	01	When a complete input message is received or when a queue buffer is full. Cancelled input messages are not logged.
2)	Message sent to a terminal or another program.	03	When a complete output message is received or when a queue buffer is full.
3)	Message queue control blocks.	3X	When the message queue blocks change.

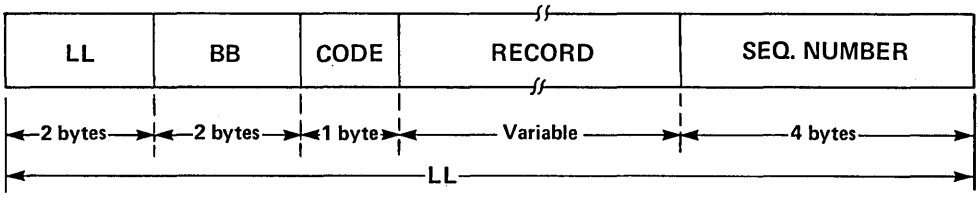
4) Application accounting record.	07	When an application program terminates.
5) IMS/VS accounting record.	06	When the IMS/VS system is started or stopped.

LOG FORMAT

Records are written on the log using spanned BSAM variable length blocked records. Since different types of records are written on the log for different purposes, some method must be used to identify each logical record.

The fifth byte of each logical record is called the log code and can be used to identify that logical record. You can then look at the fifth byte of each logical record, process those records with which you are concerned, and bypass any record with which you are not concerned. Additional information about log codes is supplied in the IMS/VS Program Logic Manual.

Each logical record written on the log is in the following format:



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 one-byte log code; and the record is of variable length.

Control information, in the form of the message prefix, is included with each message received or sent. In this prefix are the destination or source, date and time, and an input or output sequence number. The prefix format is found in the IMS/VS Program Logic Manual.

LOG DATA SET ALLOCATION

The IMS/VS procedure includes DD statements for old and new log data set allocations. (See the IMS/VS Installation Guide.) The old log DD statement label is IMSLOGR. The new log DD statement label is IEFRDER.

STATISTICAL REPORTS

Statistical reports provide a means of evaluating line and terminal loading, traffic volumes, response times, and accounting (billing) information. Samples of statistical reports produced by the IMS/VS Statistical Analysis utility program are shown at the end of this chapter.

SYSTEM LOG RECOVERY UTILITY PROGRAM (DFSULTR0)

The purpose of DFSULTR0 is to produce a new and usable system log data set from a system log data set that contains read errors. Either a single or dual system log data set is used for input; the complete execution of DFSULTR0 varies on which input type is used.

DFSULTR0 has two modes of operation, duplicate (DUP) mode and replace (REP) mode. Both modes are required when a single system log data set is used. REP mode may be required when a dual system log data set is used depending on the success of DUP mode functions.

The following discussion of DFSULTR0 is divided according to input type.

SINGLE SYSTEM LOG INPUT

Figure 7-1 shows an overview of DFSULTR0 activities when a single system log is used for input. Both modes of operation must be executed.

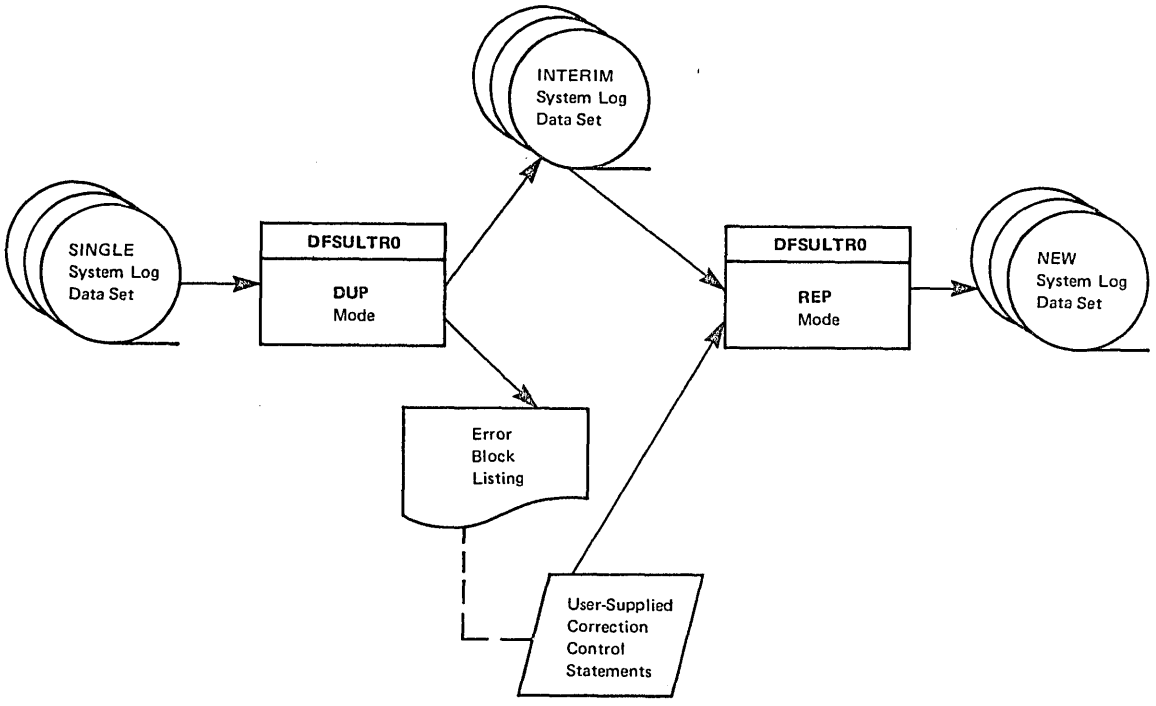


Figure 7-1. DFSULTR0 with Single System Log Input

DUP Mode

Using a single system log data set for input, DUP mode:

- reads the input data set;
- creates an interim log on which each error encountered on the input data set is uniquely identified;
- produces a character and hexadecimal overprint listing of error blocks.

The listing produced by DUP mode is used as a guide for creating the correction control statements required by REP mode.

REP Mode

REP mode produces a new and usable system log data set. This is accomplished by:

- copying the error-free records from the interim log produced by DUP mode onto a new system log data set;
- merging user-supplied correction control statements with the error-free records from the interim log.

DUAL SYSTEM LOG INPUT

When dual system log input is used, the complete operation of DFSULTR0 depends on the success of DUP mode functions. REP mode is usually unnecessary. In the following discussion, the terms 'primary' and 'secondary' are used to identify the two logs of a dual log data set.

DUP Mode

Using dual log data sets as input, DUP mode:

- reads the primary log and copies the log contents onto a new system log until an error block is encountered on the primary log;
- positions a read operation on the secondary log at the location where the primary log error was encountered;
- reads the secondary log and copies the log contents onto the new system log until an error is encountered on the secondary log;
- positions a read operation on the primary log at the location where the secondary log error was encountered;
- repeats the above steps until a complete new system log is produced.

If errors at the same position on both input logs are not encountered, the system log produced by DUP mode is correct, and REP mode is not required. Figure 7-2 illustrates the above DFSULTR0 activities.

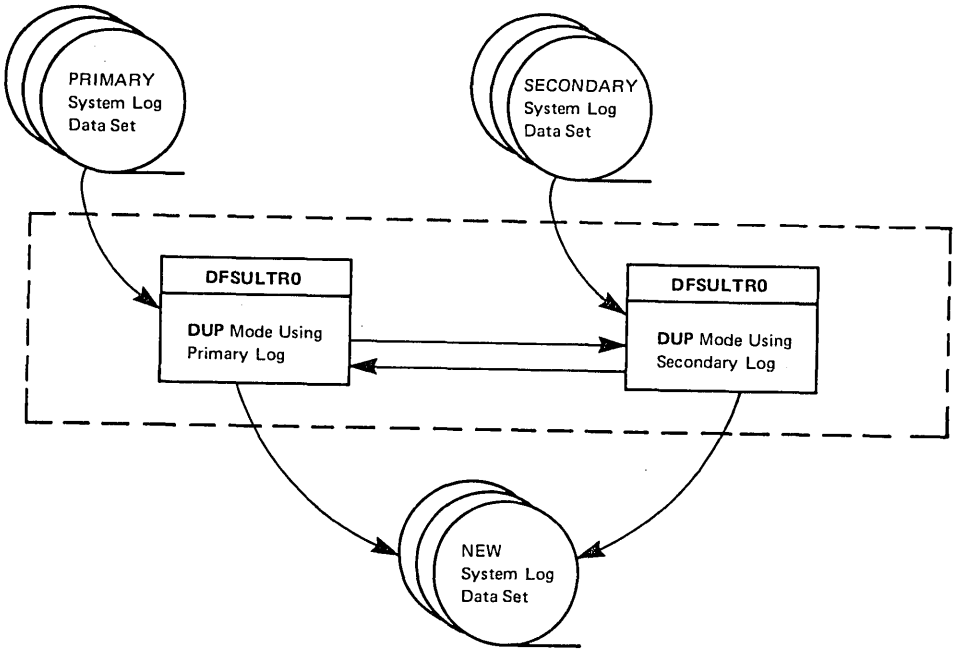


Figure 7-2. DFSULTR0 with Dual System Log Input and all Errors Reconcilable

If DUP mode encounters an error on one log record and then finds the same log record to be in error on the other log, it:

- copies both error blocks onto the new system log, which must, in this instance, be considered an interim log; error blocks are uniquely identified on the interim log (primary log errors are identified by AXXXXX, secondary log errors by BXXXXX).
- produces a character and hexadecimal overprint listing of the error blocks; the errors from both log data sets are printed (primary log errors are identified by AXXXXX, secondary log errors by BXXXXX).

The listing of error blocks produced by DUP mode is used as a guide for creating the correction control statements required by REP mode. Errors from both logs should be examined, although corrections can only be supplied relative to one log (reference AXXXXX or BXXXXX) during REP mode.

REP Mode

REP mode produces a new and usable system log data set. This is accomplished by:

- copying the error-free records from the interim log produced by DUP mode onto a new system log data set;
- merging user-supplied correction control statements with the error-free records from the interim log.

Figure 7-3 illustrates DFSULTR0 operation when dual log input is used and DUP mode was unable to reconcile all error blocks.

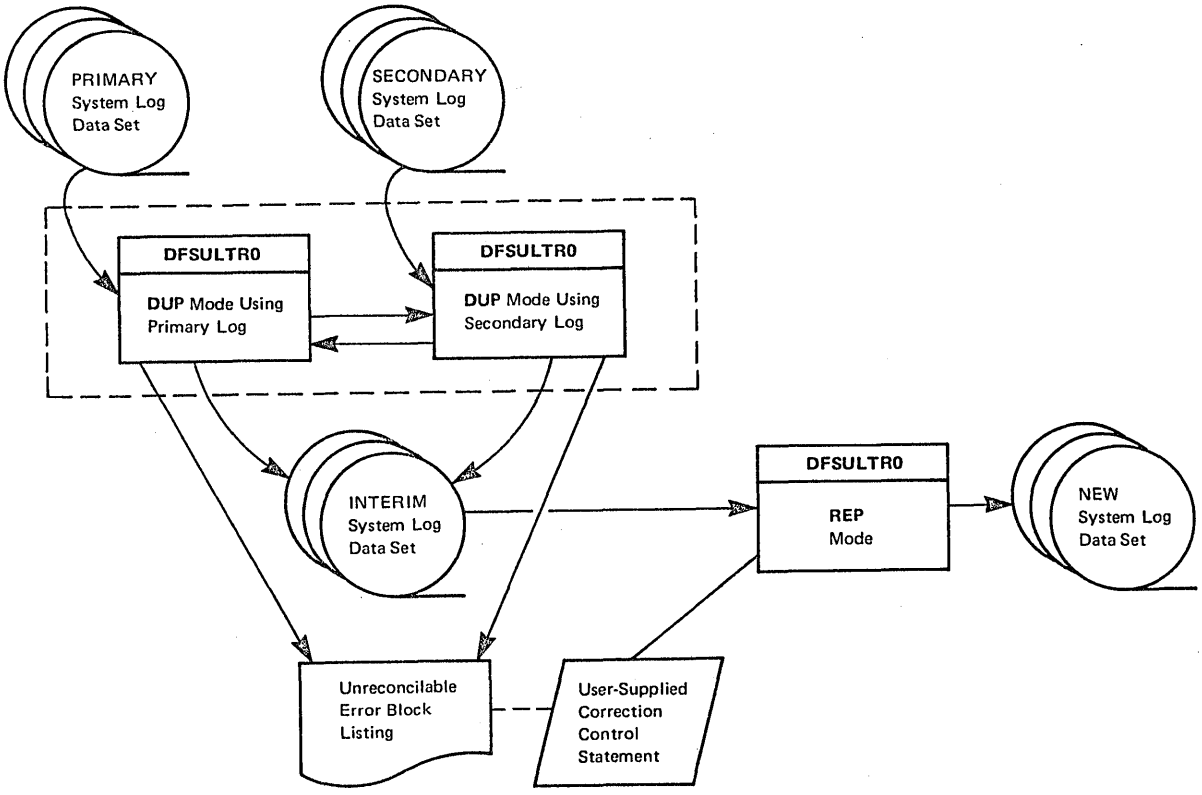


Figure 7-3. DFSULTR0 with Dual System Log Input and One or More Errors Not Reconcilable

JCL REQUIREMENTS

```
//RECOVER  JOB (ACCTGINFORMATION)
//STEP EXEC PGM=DFSULTR0
//SYSPRINT DD  SYSOUT=A,DCB=[RECFM=FBA,LRECL=133,BLKSIZE=133]
//IEFRDER  DD  UNIT=2400,VOL=SER=LOG01,DSN=IMSLOG,DISP=OLD
//IEFRDER2 DD  UNIT=2400,VOL=SER=LOG002,DSN=IMSLOG2,DISP=OLD [1]
//NEWRDER  DD  UNIT=2400,VOL=SER=NEWLOG,DSN=IMSLOG,                  *
//            DISP=(NEW,KEEP),DCB=DSORG=PS [2]
//NEWRDER2 DD  UNIT=2400,VOL=SER=NEWLOG2,DSN=IMSLOG2,                *
//            DISP=(NEW,KEEP),DCB=DSORG=PS
//SYSIN    DD  *
     .
     .
  CORRECTION CONTROL STATEMENTS
     .
     .
```

[1] Include this statement only when dual log input is used.

[2] If this statement is not included, only a listing of unreconcilable errors is produced.

Correction Control Statement Format

DUP Mode

One control statement is required with DUP in columns 1-3.

REP Mode

At least one control statement is required. Any number can be included. There are three subtypes depending on column 16 for REP.

Beginning in Column	Format	Meaning
1	REP	Indicates REP mode.
5	SEQ={A|B}XXXXX	Indicates the identification number of the block to be changed. The number is provided in the listing output by DUP mode.
16	POS=YYYYYY	Indicates the starting position, relative to 1, of the data to be replaced.
	or	
16	SKIP	Indicates the output tape will not contain this block of data.
	or	
16	CLOSE	Indicates the output tape will be closed right before this error block.
27	DAT=XXX...XXX	2 to 50 hexadecimal characters (0-9,A-F) representing the replacement data.

When multiple REP statements are provided, the identification numbers (SEQ=) must be in ascending block number sequence.

SYSTEM LOG TERMINATOR UTILITY PROGRAM (DFSFLOT0)

At the time of a system failure, the System Log Terminator utility program can be used to recover the log data that was lost as a result of the failure. A storage dump, taken at the time of the failure, is required. IMS/VS system definition places the load member DFSFLOT0 into the IMSVS.RESLIB data set. This program is executed under operating system control using either SYS1DUMP or the standalone dump output tape.

The expediency with which the System Log Terminator program terminates the system log largely depends on the degree of the system failure. The program:

- locates the log work area, buffers, DCB, DEB, and UCB
- positions the log tape to write the remaining buffers
- writes the remaining log buffers
- closes the log data set

The messages issued by DFSFLOT0 for unsuccessful attempts to terminate the system log are listed below. More definitive information is provided in the IMS/VS Messages and Codes Reference Manual.

- DFS942I - SYSTEM LOG NOT CLOSED - aa

  where:

  aa =

  08 - Unable to find proper storage location in the dump.

  16 - Log DCB was already closed at the time the dump was taken; the log tape can be used for restart/recovery.

  20 - Buffer control block not located -- invalid location.

  24 - Unable to position system log tape for termination.

- DFS943I - ERROR ON LOG TAPE - WRITE ERROR
  READ ERROR

- DFS645A - ENTER BUFFER ADDRESS(ES) IN HEX

JCL REQUIREMENTS

EXEC	This statement must be in the form PGM=DFSFLOT0.
SYSPRINT DD	This statement is for the SYSOUT system writer.
LOGTAPE DD	Defines the log tape to be terminated. DISP must have been (NEW,KEEP) at execution time. DISP=MOD is not supported by IMS/VS for system log tapes. The System Log Terminator program does not properly position a MOD tape for termination; it does not fill in all fields in the standard trailer label and thus does not open for MOD output.
DUMP DD	Defines the SYS1.DUMP data set.

JCL EXAMPLE

```
//STEPNAME EXEC    PGM=DFSFLOT0
//SYSPRINT DD      SYSOUT=A
//LOGTAPE  DD      DSN=IMSLOG,VOL=SER=XXXXXX,DISP=(NEW,KEEP),          *
// UNIT=2400,LABEL=(,SL)
//DUMP     DD      DSN=SYS1.DUMP,VOL=SER=NNNNNN,DISP=OLD,UNIT=2400,    *
// DCB=(RECFM=U,BLKSIZE=2056),LABEL=(,NL)
```

CHAPTER 8. LOG DATA FORMATTING UTILITIES

This chapter describes the (1) IMS/VS Statistical Analysis utility, (2) File Select and Formatting Print utility, (3) IMS/VS Log Transaction Analysis utility, and (4) IMS/VS Log Tape Merge utility.

OVERVIEW

The IMS/VS Statistical Analysis utility program modules reside in IMSVS.RESLIB and can be used for analyzing the information on the IMS/VS system log tapes. 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 tape, a new log tape, tailored to your own specifications, can be obtained by using the IMS/VS Log Transaction Analysis utility.

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 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.

IMS/VS STATISTICAL ANALYSIS UTILITY

The flow of the IMS/VS Statistical Analysis utility is shown in Figure 8-1.

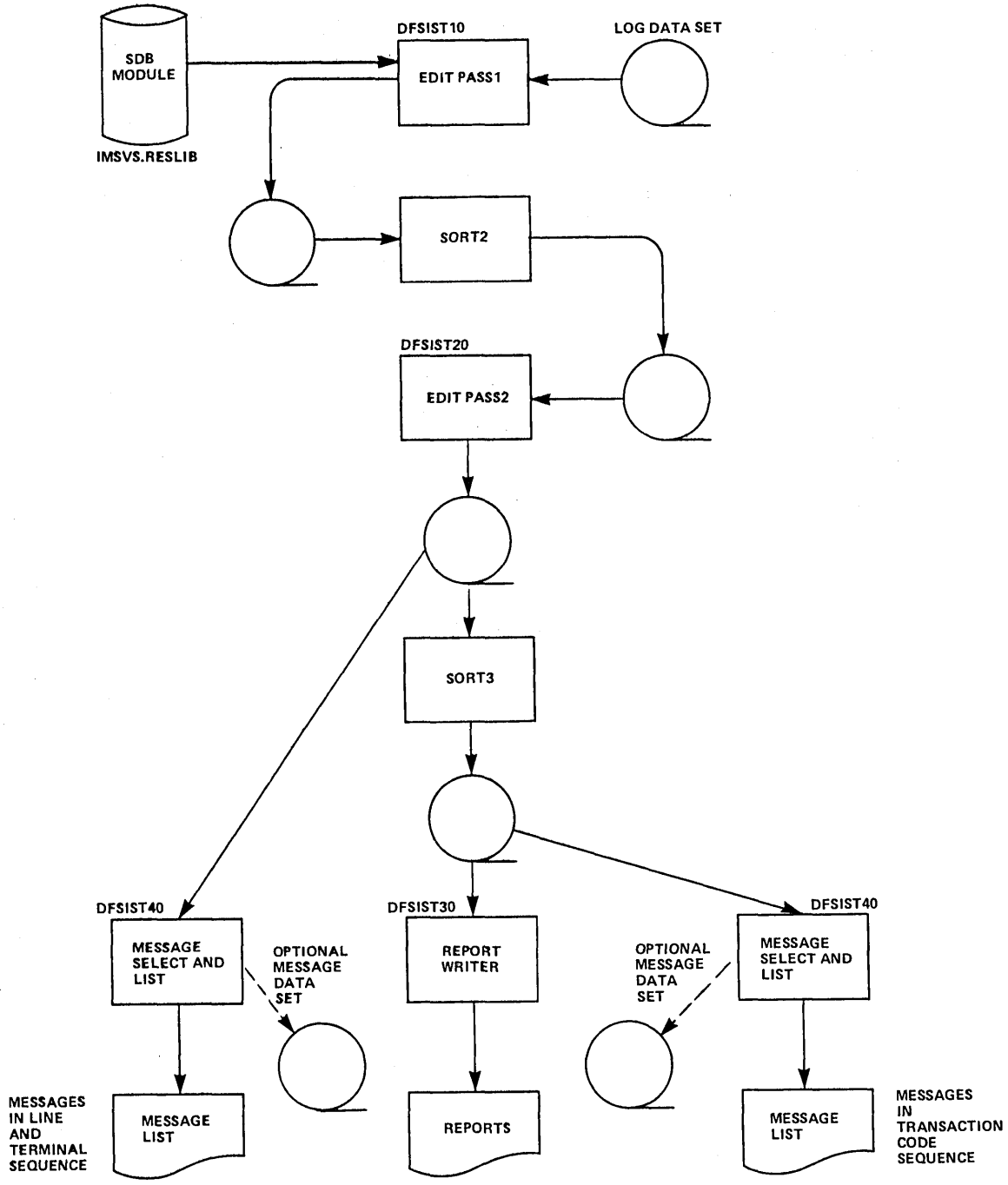


Figure 8-1. Statistical Analysis Utility Program Flow

SORT AND EDIT PASS1 (DFSISTS0)

The functions of SORT and EDIT PASS1 are to select from the system log tapes those records used by the statistics programs; to 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; and to edit the records so that, when sorted, the input message, and all output sent as a result of that input, are contiguous.

The format of the message record prefix, as it exists on the log tape is such that the Security Directory Block module (DFSISDBX) must be loaded. DFSISDBX correlates table offsets with the transaction code and logical terminal names. DFSISTS0 accepts a 1-digit parameter field identifying the suffix for DFSISDBX. The absence of this parameter field causes the program to load the Security Directory Block module named DFSISDB0; the default suffix is 0. (See the IMSGEN statement, SUFFIX operand, in the IMS/VS Installation Guide.)

The JCL for SORT and EDIT PASS1 must contain a JOBLIB or STEPLIB statement for the library containing the DFSISDBX module (IMSVS.RESLIB). Care must be taken to ensure that the DFSISDBX module that is referenced represents the system configuration at the time the logs being processed were created. Failure to do so may cause invalid data in the reports.

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.

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 jobstream, approximately 40% of the output of DFSIST20 by coding NOTXT on DFSISTS0.

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', from each LTERM, and output messages (S), X'03', to each LTERM. 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 tape 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. Time for input messages (R) and time for output messages (S) is time of dequeue. The report counts the same messages as the "Line and Terminal Report" described in 3 above. Input is sorted by transaction code.

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').

   - Percentile report shows shortest response, longest response, and 25th, 50th, 75th, and 95th percentile response.

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 CPU 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 per 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.)

     Average CPU time is the total message CPU time, divided by the number of messages. It is not rounded. The final average CPU 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, you should process all the log data sets 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)
```

- 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 8.2. Also see the appropriate operating system sort/merge program manual.

Note: The LRECL and BLKSIZE for the log tape 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 IEFRDER DD statement in the "IMS" cataloged procedure.

//JOBLIB DD	Describes the program library containing the ISDB module used while the log being processed is created, and 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 the ISDB suffix is not '0', a parameter for this suffix is required. If DFSIST40 will not be run, 'NOTXT' improves performance. Its format is: //ST1 EXEC PGM=DFSISTS0[PARM='Z,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=2400 where XXXXXX is the volume serial number of the log tape 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.

// 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 for sort. 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,13,24,CH,A),SIZE=XXXX	
// EXEC	Invokes the second statistics edit module, DFSIST20. If DFSIST40 (Message Select & 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 DFSSIST20. 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 tahn SORTOUT in the previous step. BLKSIZE must be increased proportionately.

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 which 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 reprots 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	This invokes DFSIST30 (Report Writer). Its format is: //ST5 EXEC PROGRAM=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, since I/O is performed using QSAM, with QSAM acquiring the buffers. Its format is:
	
	//PRINTDCB DD SYSOUT=A,DCB=(BLKSIZE=133,
	LRECL=133,REDFM=FA)

// EXEC	Invokes Message Select and Edit (DFSIST40). This step is optional, and a number of options apply. (See preceding section entitled "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 (see Figure 8-1). 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)

//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 8-2 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 tape.

```
//STATS     JOB 1,NAME,MSGCLASS=A,MSGLEVEL=1,PRTY=8
//JOBLIB    DD DSN=IMSVS.RESLIB,DISP=SHR
//ST1     EXEC PGM=DFSISTS0,PARM='Z'
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//LOGIN     DD UNIT=2400,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,13,24,CH,A),SIZE=E200
/*
//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)
/*
//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)
/*
//ST6     EXEC PGM=DFSIST40
//IMSLOGP   DD SYSOUT=A,DCB=(BLKSIZE=133,LRECL=133,RECFM=FA)
//IMSLOGI   DD DSN=&&ED34IN,DISP=(OLD,DELETE)
//SYSIN     DD *
TRANS CODE=(ALL,I,O)
NON PRINT=HEX
/*
//
```

Figure 8-2. 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.

- 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 03/06/76

                 TOTAL
DESTINATION      MESSAGES

ELEANOR              1
SW1050               1
T2741N1              1
T2742N3              1
```

```
     MESSAGES -- PROGRAM TO PROGRAM           DATE 03/06/76

                 TOTAL
DESTINATION      MESSAGES

PA10                 107
```

RELATIVE TERMINAL NUMBER

LINE AND TERMINAL REPORT DATE 03/06/76 PAGE 1

LINE	RTA	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																		
	PRINTER1	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 03/06/76

TRANSACTION      TOTAL
CODE             MESSAGES

(IMSSYS)             5
```

```
    MESSAGES -- PROGRAM TO PROGRAM           DATE 03/06/76

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 TOTALS	S	19	2,415	127	0	0	0	19	0	0	0	0	0	0	0	0	0	0
	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 TRANSACTIONS = 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	COUNTS GU	GN	ISRT*	DATA BASE GU	GN	GNP	GHU	GHN	GHNP	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 03/15/76 P A G E 1

I M S DAY 03/15/76

START TIME 09:01:75

STOP TIME 09:04:30

*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)

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         PPEFIX    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         PPEFIX    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 and display of data from the IMS/VS log data set. The program can:

- Print an entire log data set;
- Print 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;
- Select and print log records in a READ BACKWARD processing mode. This processing applies only to physical blocks, not logical records;
- 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 options and selection ranges as well as to specify 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 users installation. Additionally, an EXEC and appropiate 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.
SYSPRINT	This statement describes the output data set to contain the formatted print records and control messages. It will usually be defined as SYSOUT=A.
SYSIN	This statement describes the input control data set. This file must be in card image format.

input data DD	These statements define 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. If this file is to be processed in a READ BACKWARD method, it must be a tape file and it is processed using BSAM.
SYSUT1 DD	This statement defines the default ddname used for data input if explicit reference is not used.
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.

INPUT AND OUTPUT

All data input except for READ BACKWARD is processed using QSAM and can reside on either tape or direct access storage devices. READ BACKWARD uses BSAM and resides on tape. Data set organization should be physical sequential, while the record format can be fixed or variable in length, blocked or unblocked, or of undefined length. Since multiple log data sets can be used as input, multiple ddnames can be defined or, in the case of only one input log data set, the default ddname of SYSUT1 can be used. The data set containing control information must be in card image form.

Parameter statements are produced on the output print data set in the same format and sequence as they are processed. Data output is displayed in both hexadecimal and EBCDIC form, 32 bytes per line, with the hexadecimal relative offset value preceding each line. When error conditions are encountered, error messages are produced following the parameter to which they apply.

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 print processing, where the input data is read, analyzed, and compared with the selection parameters to determine the applicability of the record for printing.

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. Operands on these statements can be extended to additional statements, to a maximum of nine, by placing a nonblank character in position 72 and continuing the operand in position 16 of the next statement.

The CONTROL statement defines the beginning and ending limits of the data set to be scanned and defines the tape processing direction. It also provides the ddname of the data input if the default name of SYSUT1 is not used. Inclusion of this statement is optional if the default operands are satisfactory.

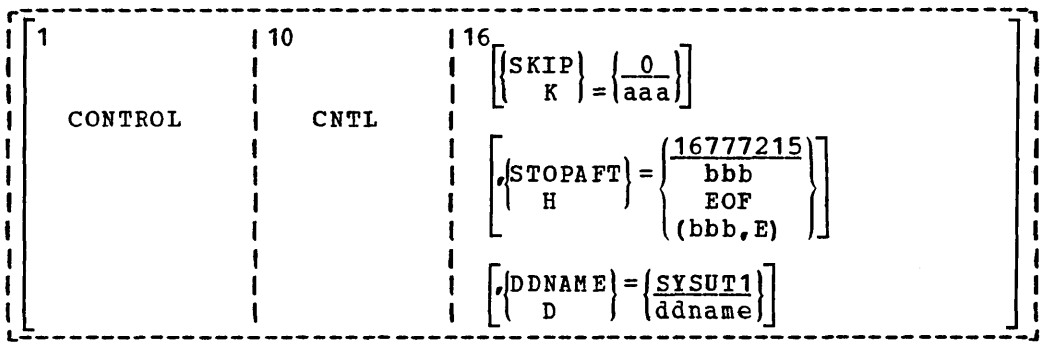
The OPTION statement defines the test or series of tests to be performed upon the data of the candidate record to determine its qualification for selection. One or more tests can be executed on each logical record by the appropriate number of OPTION statements, creating the logical "OR" function. Records 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. The maximum length of selection parameters that may be specified by the VALUE= operand is 510 bytes.

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 abbreviated keywords cause the same functions as their associated full keywords, and their parameters are coded the same as their full keyword's parameters.



SKIP=
K=

This keyword is used to define 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

The value specified must be in the range of zero to 999999, and cannot have embedded commas.

STOPAFT=
H=

This keyword is used to define 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

The value specified must be in the range of 1 to 99999999 with no embedded commas. If the value zero is specified, one record is processed. EOF, which denotes end-of-file condition, allows record processing beyond the stated maximum of 99999999 records.

E

This parameter 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=
D=

This keyword is used to identify 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

This name must be the DDNAME of the input file if the default of SYSUT1 is not used.

OPTION Statement

The OPTION statement causes one set of tests to be constructed. One or more OPTION statements can be defined for one or more sets of tests to be performed against each input record. When all operands of this statement are omitted, all records processed by phase two of the program are printed. The abbreviated keywords cause the same functions as their associated full keywords and their parameters are coded the same as their keywords.

1	10	16
OPTION	PRINT	[{OFFSET | O}={1 | aaa}][,{FLDTYP | T}={X | C}] [,{VALUE | V}=bbb] [,{FLDLEN | L}={1 | ddd}] [,{COND | C}={E | M | T[Y | N] | MT[Y | N] | ET[Y | N]}] [,{EXITR | E}={IEFBR14 | name}]

PRINT

This parameter is used to cause records satisfying this test, and prior tests in this series if this is a multifield test, to be displayed on the SYSPRINT data set.

EXITR=
E=

This keyword is used to define the exit routine to be given control when a candidate record satisfies this OPTION test. Satisfying this card implies a COND=E condition, with any remaining tests (prior to the actual COND=E) being ignored. The exit routine is passed a parameter list, pointed to by register one. The format of the list is two fullwords -- the first is a pointer to the selected LOG record, the second is a pointer

to the SYSPRINT DCB. On return via register fourteen, a test is made on register fifteen. If it is zero, the selected log record is formatted and put out to the SYSPRINT data set. A nonzero value bypasses this formatting. Record selection continues with the next record.

name

This value is the member name of the exit routine to be given control. This member is loaded into storage and therefore must reside in a data set having a STEPLIB or JOBLIB. If not in either of these data sets, it must reside in LINKLIB. This member is loaded into storage at the first exit call to the module and is deleted when selection ends. If multiple OPTION statements specify the same member, multiple copies of this member can be loaded. To avoid this, assure that this member is reenterable.

If the EXITR parameter is omitted, an OS/VS dummy module, IEFBR14, is loaded and control given to it under the same circumstances as mentioned above. This routine clears register 15 and returns immediately to the LOG PRINT.

Note: On entry to the exit routine, registers are setup as follows:

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.

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.

On return from the exit routine, register 15 is used to decide whether or not to dump the record before retrieving the next record. If register 15 contains a nonzero value, the record is not dumped.

OFFSET=
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, the user 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 one.

FLDTYP=
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 -- two 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, then 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=
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 characters of blank or comma are 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 "non-null" characters in this field.

COND=
C=

This keyword defines the type of test and its relationship to other tests in the group.

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 the record is printed.

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.

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 one 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. Since the T parameter assumes a default value of one, 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 a "Test Under Mask" condition.

FLDLEN=
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 one to and including 255. The default is 1.

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

This example shows the JCL and control statements required to print all records from a data base Image Copy data set

```
//EPRT  JOB
//IMAGEX1  EXEC PGM=DFSERA10
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSN=IMAGE,VOL=SER=123456,UNIT=2400,LABEL=(,SL),
//              DISP=(OLD,KEEP)
//SYSIN    DD   *
* THE ABSENCE OF A CONTROL STATEMENT WILL ASSUME SYSUT1 INPUT
* THIS OPTION STATEMENT WILL CAUSE THE ENTIRE FILE TO BE PRINTED
OPTION     PRINT
END
/*
```

This example shows the JCL and control statements required to print the first 30 records from an IMS/VS log tape

```
//RINT  JOB
//LOGEX1   EXEC PGM=DFSERA10
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//LOGTAP1  DD   DSN=IMSLOG,VOL=SER=111111,UNIT=2400,LABEL=(,SL),
//              DISP=(OLD,KEEP)
//SYSIN    DD   *
* THIS CONTROL STATEMENT DEFINES THE INPUT TO BE LOGTAP1
* AND ONLY THE FIRST 30 RECORDS FROM THE FILE WILL BE USED
CONTROL CNTL DDNAME=LOGTAP1,STOPAFT=30
OPTION PRINT
END
/*
```

This example shows the JCL and control statements required to extract log records of type X'50'. Only those records after record No. 1000 are to be examined, all selected records are passed to a user exit routine for processing.

```
//RINT JOB
//LOGEX2   EXEC PGM=DFSERA10
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//LOGTAP1  DD   DSN=IMSLOG,UNIT=2400,VOL=SER=111111,LABEL=(,SL),
//              DISP=(OLD,KEEP)
//SYSIN    DD  *
CONTROL CNTL SKIP=1000,DDNAME=LOGTAP1
* THIS OPTION STATEMENT CAUSES ONLY THOSE RECORDS WITH A HEX '50' IN
* RECORD BYTE 5 TO BE PRINTED ON THE SYSPRINT DATA SET.
OPTION PRINT OFFSET=5,FLDTYP=X,FLDLEN=1,COND=E,VALUE=50,EXITR=USREXRT
END
/*
```

This example shows the JCL and control statements required to print record No. 158 of an OSAM Image Copy data set and all type X'50' records on a log tape that references this block number (assuming unblocked OSAM).

```
//PRNT  JOB
//COMBEX1  EXEC PGM=DFSERA10
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//IMAGFILE DD   DSN=OSAMIMAG,UNIT=SYSDA,DISP=SHR,VOL=SER=DA0002
//SYSUT1   DD   DSN=IMSLOG,UNIT=SYSDA,VOL=SER=DA0003,DISP=SHR
//SYSIN    DD   *
* THIS CONTROL STATEMENT CAUSES THE INPUT FILE IMAGFILE TO CLOSE
* AFTER THE FIRST RECORD OF THE SELECTED GROUP IS PRINTED
CONTROL  CNTL   STOPAFT=(1,E),DDNAME=IMAGFILE
OPTION   PRINT FLDEN=4,OFFSET=1,FLDTYP=X,COND=E,VALUE=0000009E
* THIS END STATEMENT CAUSES THE SELECTION OF THE FILE TO BEGIN.
END
* THIS CONTROL DEFAULTS TO THE STANDARD INPUT FILE, SYSUT1
CONTROL CNTL
* THIS STATEMENT LIMITS THE SELECTION TO ONLY THOSE RECORDS
* THAT CONTAIN A HEX '50' IN RECORD BYTE 5
OPTION   PRINT  FLDLEN=1,OFFSET=5,FLDTYP=X,COND=M,VALUE=50
* THIS STATEMENT FURTHER LIMITS SELECTION TO ONLY THOSE RECORDS THAT
* CONTAIN THE DATA BASE NAME OF DATABAS1 IN RECORD BYTES 25 THRU 32
OPTION PRINT FLDTYP=FLDTYP=C,FLDLEN=8,OFFSET=25,COND=M,VALUE=DATABAS1
* THIS STATEMENT FURTHER LIMITS SELECTION TO ONLY THOSE RECORDS THAT
* CONTAIN THE FLAG MARKING THIS RECORD AS AN OSAM RECORD
OPTION   PRINT  FLDTYP=X,OFFSET=7,COND=MTN,VALUE=04
* THIS STATEMENT FURTHER LIMITS SELECTION TO ONLY THOSE RECORDS THAT
* CONTAIN THE RBN OF 0000009E
OPTION   PRINT  FLDLEN=4,OFFSET=43,COND=E,VALUE=0000009E,FLDTYP=X
* THIS STATEMENT CAUSES THE SELECTION OF RECORD TO BEGIN
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).

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 invoke this exit routine follow.

```
Col. 1      10       16

CONTROL     CNTL
OPTION      PRINT    OFFSET=5,FLDLEN=2,
                     VALUE=67aa,COND=E,
                     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

Program Isolation Trace Record Format and Print Module (DFSERA40)

The Program Isolation (PI) Trace Format and Print module accepts log records of Type X'67' as an exit routine from the File Select and Formatting Print program (DFSERA10) and formats them onto the SYSPRINT output data set. These records are trace records produced by calls to the ENQ/DEQ routine from the online region and are generated in the PI Trace module DFSDPIT0.

The PI Format and Print module (DFSERA40) is dynamically loaded during the execution of DFSERA10 and must therefore reside in the LINKLIB or in a JOBLIB or STEPLIB data set.

The control statements required to invoke this exit routine are:

```
Col. 1      10       16

CONTROL     CNTL
OPTION      PRINT    OFFSET=5,FLDLEN=2,
                     VALUE=67aa,COND=E,
                     EXITR=DFSERA40
END
```

The following is sample output.

```
WHO    OFST   PST    TIME(*=ET)          FUNC  LVI RC        ID
DLR    77CC   00     052781320F0EE000    01    01  00  00001004 0004 01 00
DLA    033E   00     052781321BD14000    GU              00000022 (See note below)
DLR    769A   00     05278132225DF000    12    02  00  00001004 0004 01 00
VSM    0FDE   00     052781322288D000    48    03  00  002CF03C 0000 00 00
VSM    1192   00     052781322305F00C    50    03  00  0CD9AD18 0000 00 00
VBH    1936   00     0527813223302000    48    03  00  0000000E 0005 01 00
```

Note: DL/I function, count of function since last schedule.

Explanation of Column Headings

SPL — Reserved for future use.

WHO — Specifies the DL/I modules that issued the call to DFSFXC10. The three characters selected come from the xx portion of the full module name DFSxxx00. *** indicates that IMS cannot identify the module issuing the call.

OFST — Specifies the offset in the module to where the call was issued. If *** prints for WHO, then the offset (OFST value) is the contents of Register 14 and is the return address of the calling module.

PST — Specifies the PST that was used for the call. It is either the value obtained from the schedule record (code '08'), or if WHO contains ***, it is the PST address in memory.

TIME — This field contains the 8 bytes obtained by the STCK instruction issued at the start of the call to DFSFXC10. If an * follows the value, a wait occurred. If TIME was requested in the /TRACE command, the value is the number of timer units waited until the request could be satisfied.

FUNC — Specifies the one byte function code passed to DFSFXC10. See the PIPARM DSECT discussion in the IMS/VS System Manual for a full explanation.

LVL — Specifies the level of control for call. 01 = test; 02 = update; 03 = exclusive.

RC — Specifies the return code from DFSFXC10. 00 = OK to proceed; 04 = wait resource controlled by another PST; 08 = deadlock, this PST should reset and start over; 0C = invalid call.

FB — Reserved.

ID — 8 byte identification of resource for call. The format is a 4-byte RBA, a 2-byte DMB, a 1-byte DCB number, and a 1-byte code field.

shift fields — The next five fields are used to trace a CA/CI split in a VSAM DSDS data set. They show the DMB and DCB number of the affected data set, the old REA, the new RBA, how many bytes were shifted, and by how much they were shifted.

DL/I call — A record is added to the trace to show when a DL/I call was issued. This record comes from the DL/I analyzer, and shows the 4-byte function and the decimal value of the appropriate PST accounting field.

IMS/VS LOG TRANSACTION ANALYSIS UTILITY (DFSILTA0)

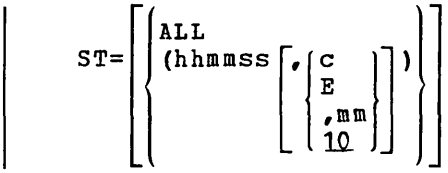
The IMS/VS Log Transaction Analysis utility (DFSILTA0) collects information about individual occurrences of IMS/VS transactions, based on records in the IMS/VS system log data set. The information collected includes transaction identification, source, message processing program, protect key, priority, and the class of the transaction. 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, the time the output message starts to the terminal, and the time that the output message is completed. 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 tape, DFSILTA0 can provide a new log tape tailored to an installation's specifications. DFSILTA0 is put into IMSVS.RESLIB by IMS/VS system definition.

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 tape.

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 tape from the specified start time.

The Log Transaction Analysis utility scans records between checkpoints. Records before the first checkpoint on an intermediate log tape would only be analyzed by reference to a checkpoint on a previous tape.

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 tape and a report from the tape.

NOLOG
specifies that a new IMS/VS log tape is not to be produced.

NOREPORT
specifies that no report is to be produced.

PROGRAM OUTPUTS

DFSILTA0 produces:

- A new IMS/VS log tape, if specified.
- A report, on disk, that can be sorted to produce a sequenced report.
- A detailed report in input sequence, if specified.
- A summary report.

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 Tape Analysis Report

Figure 8-3 is an example of an IMS/VS Log Tape Analysis Report.

Detailed Report Format

Identification	Starting Position	Length
Sequence Number	2	5
Transaction Code	8	8
Priority of Transaction	17	1
Class of Transaction	19	3
Input Relative Line	23	2
Input Relative Terminal	25	2
Output Relative line	28	2
Output Relative Terminal	30	2
Processing Type	33	1 *
Program Name	35	8
Protect Key of Region	44	1
Time of SMB Enqueue	46	10 **
Time of Message GU	57	10 **
Time of CNT Enqueue	68	10 **
Time of Program End or Next Message GU	79	10 **
Time of CNT GU	90	10 **
System IDs	102	3
Time in Input Queue	105	6 ***
Time Processing	112	6 ***
Time in Output Queue	119	6 ***
Total Time	126	6 *** #

* Processing Types:

- A - Program Abend or Unconnected Transaction
- S - Send/Receive Processing
- T - Transmit Only Processing
- C - Conversational S/R Processing
- D - Transmit Only Conversational Processing
- P - Program Switch S/R 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 SET command was in use for the transaction code.)
- R - Program was running at time of IMS abend.

** Time HH.MM.SS.T

***Time SSSS.T

\# Total Time is from SMB enqueue to CNT dequeue.

Figure 8-3. IMS/VS Log Tape Analysis Report

```
*****************************************************************************************************************************

          * * *          LOG TAPE ANALYSIS          * * *                                                        PAGE 00001
 *** IMS LOG TAPE FIRST ACCOUNTING RECORD DATA
 IMS STARTED    AT TIME =  7.45.29.0 , DATE = 76.078


 SPECIFIED START TIME IS  7.54.20.6


TO END OF TAPE

 PROCESSED 00012 TRANSACTIONS THIS RUN
 REPORT DATA SET CONTAINS 00015 COMPLETE RECORD SETS


     JOB       STEP    P      CLASSES
     NAME      NAME    K **************
   SAMPMSG   SAMPREG   1   2   3   4   5


PRELOG NORMAL END OF JOB


          * * *          LOG TAPE ANALYSIS          * * *                                                        PAGE 00002
 SEQ    TRANS    P  C    IN   OUT  P    PGM     P *SMB**ENQ* MSG*SCHEDULE *CNT**ENQ* **MSG**END** *CNT**GU**  IN Q   PROC  OUT Q  TOTAL
 NBR     CODE    R  L   RLRT RLRT T    NAME    K HH.MM.SS.T HH.MM.SS.T C HH.MM.SS.T HH.MM.SS.T C HH.MM.SS.T SSSS.T SSSS.T SSSS.T SSSS.T

00001 PART     7   4 0201 0201 S DFSSAM02 1  7.55.09.9  7.55.10.0  7.55.13.7  7.55.13.7  7.55.13.7      .1    3.7           3.8
00002 DSPALLI  7   2 0201 0201 S DFSSAM07 1  7.55.13.7  7.55.13.8  7.55.14.2  7.55.14.2  7.55.14.2      .1     .4           .5
00003 DSPINV   7   4 0201 0201 S DFSSAM03 1  7.55.14.2  7.55.14.3  7.55.14.6  7.55.14.6  7.55.14.6      .1     .3           .4
00004  CHKPT   0002**********************   7.55.14.7  7.55.14.7  7.55.14.7  7.55.14.7  7.55.14.7
00005 ADDPART  7   4 0201 0201 S DFSSAM04 1  7.55.15.3  7.55.15.3  7.55.15.8  7.55.15.8  7.55.15.8      .0     .5           .5
00006 ADDINV   7   5 0201 0201 S DFSSAM04 1  7.55.15.8  7.55.15.8  7.55.16.0  7.55.16.0  7.55.16.0      .0     .2           .2
00007 DSPINV   7   4 0201 0201 S DFSSAM03 1  7.55.16.0  7.55.16.0  7.55.16.2  7.55.16.2  7.55.16.2      .0     .2           .2
00008 DLETINV  7   5 0201 0201 S DFSSAM04 1  7.55.16.3  7.55.16.3  7.55.16.5  7.55.16.5  7.55.16.5      .0     .2           .2
00009 DLETPART 7   4 0201 0201 S DFSSAM04 1  7.55.16.5  7.55.16.5  7.55.16.7  7.55.16.7  7.55.16.7      .0     .2           .2
00010 CLOSE    7   3 0201 0201 S DFSSAM05 1  7.55.16.8  7.55.16.8  7.55.17.1  7.55.17.1  7.55.17.1      .0     .3           .3
00011 DSPINV   7   4 0201 0201 S DFSSAM03 1  7.55.17.2  7.55.17.2  7.55.17.3  7.55.17.3  7.55.17.3      .0     .1           .1
00012 DISBURSE 7   2 0201 0201 S DFSSAM06 1  7.55.17.3  7.55.17.4  7.55.17.6  7.55.17.6  7.55.17.6      .1     .2           .3
00013 DSPINV   7   4 0201 0201 S DFSSAM03 1  7.55.17.6  7.55.17.7  7.55.17.8  7.55.17.8  7.55.17.8      .1     .1           .2
00014  CHKPT   0003**********************   7.55.17.9  7.55.17.9  7.55.17.9  7.55.17.9  7.55.17.9
00015  CHKPT   0004**********************   7.55.19.9  7.55.19.9  7.55.19.9  7.55.19.9  7.55.19.9
```

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 tape.

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 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=2400`

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=2400,DCB=(RECFM=VBS,LRECL=6000,`
	`// 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 program schedule time within region:
	
	`SORT FIELDS=(44,1,CH,A,57,10,CH,a),SIZE=E500`

PART 4. PERFORMANCE REPORTING AND SERVICE UTILITIES

Part 4 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 9, "Performance Reporting Utilities," describes the utilities that organize, format, and print performance-related reports. Control statements and examples and report examples are included.

Chapter 10, "System Service Utility," describes the print utility that copies messages produced by the online control program from the online data sets to a system output device.

CHAPTER 9. 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 information on operating the DB Monitor, see the IMS/VS System Programming Reference Manual and the IMS/VS Operator's Reference Manual.

The reports summarize and categorize the traced IMS/VS activities at various levels of detail. Each monitor trace provides enough information for DFSUTR30 to produce a number of different reports. The values shown in these reports are not intended to reflect actual values that are received by a user's execution of this utility. A VSAM statistics report, unique to the DFSUTR30 program, is also printed. These reports contain totals, averages, and user-defined or default distribution displays for the activities reported. Values accumulated include elapsed time, IWAIT time, CPU time, scheduled time to first DL/I call, elapsed execution time, and queue statistics. The meaning of these values is explained below under "Definitions of Terms Used in Reports." An understanding of the basic units of information is essential for effective use of the reports.

The following reports are produced by DFSUTR30:

- Statistics from Data Base Buffer Pools and VSAM Buffer Subpools
- Program I/O
- DL/I Call Summary
- VSAM Statistics
- Monitor Overhead
- Distribution Appendix

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.

DEFINITIONS OF TERMS USED IN REPORTS

The following are explanations of key terms used in the reports to describe activities or subtasks in the IMS/VS control region.

ELAPSED TIME: Time recorded by the time of day clock, from the start of the activity or subtask until the end of the activity or subtask.

IWAIT TIME: Elapsed time, during which an IMS/VS control region subtask was inactive, waiting for a resource or the completion of an event. An exception to this definition occurs when the IWAIT time is related to VSAM activity. In this case, the IWAIT time is defined as the elapsed time between the entry to and the exit from the VSAM routines. During this VSAM time, an I/O access and wait may or may not have occurred.

NOT IWAIT (ELAPSED TIME - IWAIT TIME): Elapsed time minus all IWAIT time identified for the subtask or activity. It includes any time spent by OS/VS, or by any other higher priority tasks running in the systems when the IMS/VS control region was interrupted and dispatchable, and when the subtask to which the CPU time refers had been executing at the time of the interrupt. Note that this may approximate total CPU time if the IMS/VS control region is the high priority task, and if no low priority tasks are causing interrupts.

CPU TIME: Actual CPU time used by an application program.

SCHEDULE TO 1ST DL/I CALL: Elapsed time accumulated for the following actions to occur: the region to gain control after being scheduled; the program either to be located in the region by contents supervision of VS, or to be brought in by program load; the program to issue the first DL/I call that requires dispatching of the Call Analyzer Module (DFSDLA00).

ELAPSED EXECUTION TIME: Elapsed time from IMS/VS dispatch of the first DL/I call by a program until the IMS termination of that program.

MAXIMUM TIME: Longest single time duration noted for an event.

TOTAL TIME: Sum of all the time durations noted for a group of events.

MEAN TIME: Quotient of the total time (above) divided by the number of occurrences of a certain event.

STATISTICS FROM DATA BASE BUFFER POOLS AND VSAM BUFFER SUBPOOLS REPORTS

These two summary reports are formatted displays of the contents of selected statistics of the data base buffer pool and VSAM buffer subpool that were collected for batch activity over the entire run of the Data Base Monitor (DFSMNTB0).

The pool ending values and the difference between starting and ending values cannot be computed for these summary reports unless there are pool ending statistics written on the trace tape. These ending values are recorded only if the Monitor is ended before the IMS/VS batch job is terminated.

The following message is printed if the summary reports are not printed:

```
NO DATA BASE BUFFER DATA AT END TIME ON MONITOR LOG TAPE;
****DATA BASE BUFFER POOL CANCELLED****
```

The VSAM Buffer Subpool Summary report is not produced if the VSAM facility is not activated through the IMS/VS system definition or if the ending values are not written on the trace tape. In either instance, the following information message is printed:

```
NO VSAM BUFFER POOL TRACES ON MONITOR LOG TAPE;
****VSAM BUFFER POOL REPORT CANCELLED****
```

Quotient formulas are not relevant to these reports (as they are to statistics reports produced by the DC Monitor Report Print program) since no transactions are involved.

An example of each summary report follows (Figures 9-1 and 9-2).

Figure 9-1. Statistics from Data Base Buffer Pools Report

```
IMS MONITOR    ***BUFFER POOL STATISTICS***      TRACE START 1976 108  13:24:10     TRACE STOP  1976 108  13:30:16  PAGE  0001


                                    D A T A   B A S E   B U F F E R   P O O L


                                                          13.24.10                13.30.16
                                                        START TRACE               END   TRACE          DIFFERENCE

NUMBER OF BLOCK REQUESTS RECEIVED                             1                       11                 10
NUMBER OF REQUESTS SATISFIED FROM POOL                        2                       12                 10
NUMBER OF READ REQUESTS ISSUED                                3                       13                 10
NUMBER OF BUFFER ALTERATIONS RECEIVED                         4                       14                 10
NUMBER OF OSAM WRITES ISSUED                                  5                       15                 10
NUMBER OF BLOCKS WRITTEN                                      6                       16                 10
NUMBER OF NEW BLOCKS CREATED IN POOL                          7                       17                 10
NUMBER OF CHAINED WRITES ISSUED                               8                       18                 10
NUMBER OF BLOCKS WRITTEN ON WRITE CHAINS                      9                       19                 10
NUMBER OF POOL COMPACTION PERFORMED                          10                       15                  5
NUMBER OF BUFFERS COMBINED                                   11                       16                  5
NUMBER OF BUFFERS MOVED                                      12                       17                  5
NUMBER OF RETRIEVE BY KEY CALLS                              13                       18                  5
NUMBER OF GN CALLS RECEIVED                                  14                       19                  5
NUMBER OF BISAM READS OR OISAM SETLS                         15                       20                  5
NUMBER OF WRITE ERROR BUFFERS CURRENTLY IN THE POOL          16                       21                  5
NUMBER OF WRITE ERROR BUFFERS DURING THIS EXECUTION          17                       22                  5
```

IMS MONITOR ***BUFFER POOL STATISTICS*** TRACE START 1976 164 6:58:19 TRACE STOP 1976 164 6:58:38 PAGE 0002

V S A M B U F F E R P O O L

SUBPOOL ID : 1
SUBPOOL BUFFER SIZE : 2048
TOTAL BUFFERS IN SUBPOOL : 20

	6.58.19 START TRACE	6.58.38 END TRACE	DIFFERENCE
NUMBER OF RETRIEVE BY RBA CALLS	0	503	503
NUMBER OF RETRIEVE BY KEY CALLS	0	73	73
NUMBER OF LOGICAL RECORDS INSERTED INTO ESDS	0	0	0
NUMBER OF LOGICAL RECORDS INSERTED INTO KSDS	0	13	13
NUMBER OF LOGICAL RECORDS ALTERED IN THIS SUBPOOL	0	179	179
NUMBER OF TIMES BACKGROUND WRITE FUNCTION INVOKED	0	0	0
NUMBER OF SYNCHRONIZATION CALLS RECEIVED	0	0	0
NUMBER OF VSAM GET CALLS ISSUED	0	249	249
NUMBER OF VSAM SCHBFR CALLS ISSUED	0	21	21
NUMBER OF TIMES CONT INT REQUESTED ALREADY IN POOL	0	357	357
NUMBER OF CONTR INT READ FROM EXTERNAL STORAGE	0	9	9
NUMBER OF VSAM WRITES INITIATED BY IMS/VS	0	0	0
NUMBER OF VSAM WRITES TO MAKE SPACE IN THE POOL	0	0	0
NUMBER OF PERM WRT ERROR BUFFS NOW IN THE SUBPOOL	0	0	0
LARGEST NUMB OF PERM ERR BUFFS EVER IN THE SUBPOOL	0	0	0

Figure 9-2. Statistics from VSAM Buffer Subpools Report

PROGRAM I/O REPORT

This is a summary report of the total and mean IWAIT intervals recorded for I/O IWAITs caused by DL/I calls by each program scheduled during the trace.

The data is arranged, under each program, by PCB name, ddname, and module identification of the module that issued the IWAIT.

The data under the column heading "IWAITS" indicates the number of times that DL/I calls for the associated PCB were required to wait for I/O activity to complete. The data set for which the I/O took place is indicated by the ddname. Entries under the heading "Module" are abbreviated identifications for IMS/VS modules. The cross-reference is:

Abbreviated ID	Module Name
DBH	DFSDBHR0
DLE	DFSDDLE0
VBH	DFSDVSM0

PCB subtotals and a batch total are provided.

Figure 9-3 is an example of the Program I/O report.

Figure 9-3. Program I/O Report

```
IMS MONITOR    ****PROGRAM I/O****          TRACE START 1976 164   6:58:19     TRACE STOP  1976 164   6:58:38  PAGE  0003

                                        ----IWAIT TIME---
PCB NAME        IWAITS      TOTAL       MEAN      MAXIMUM    DDNAME    MODULE   DISTR. NO.
DHVBTZ03           243    1713301       7050       429487    DHSK0901  VBH          55
                   115    1032590       8979       512137    DXSK0101  VBH          56
                   102     979931       9607       260054    DHSK0903  VBH          57
                     2      55334      27667        53252    DHSK0904  VBH          58
PCB TOTAL          462    3781156       8184

BASIC TOTAL        462    3781156       8184
```

DL/I CALL SUMMARY REPORT

This is a compact DL/I call summary report for the trace. All DL/I calls issued by every program scheduled during the trace are arranged as follows:

- PCB name.
- For each PCB, the call function employed.
- For each call function, the segment accessed and its level number.
- For each segment, the return code obtained.

For each line in this report, the number of DL/I calls recorded, the IWAITs per call, and both the average and maximum elapsed time and not-IWAIT times are given. A batch total of DL/I calls is provided at the end of the report.

Figure 9-4 is an example of the DL/I Call Summary report. The column entries, from left to right, are:

- PCB NAME — The 8-character PCB name.
- CALL FUNC. — The 4-character DL/I call function.
- LEV NO. — The data base level number reached in this call, or blank.
- SEGMENT — The 8-character segment name accessed by this call.
- STAT CODE — The status code returned by this call.
- DL/I CALLS — The number of calls noted having the unique combintation of the above five attributes.
- IWAITS — The number of I/O WAITs observed for the calls.
- IWAITS/CALL — Quotient of the above two items.
- ELAPSED TIME — For an explanation of ELAPSED TIME and NOT IWAIT TIME, see "Definition of Terms Used in Reports" earlier in this chapter.
- NOT IWAIT TIME

IMS MONITOR ****DL/I CALL SUMMARY**** TRACE START 1976 164 6:58:19 TRACE STOP 1976 164 6:58:38 PAGE 0004

PCB NAME	CALL FUNC	LEV NO. SEGMENT	STAT CODE	DL/I CALLS	IWAITS	(C) IWAITS/ CALL	(A) ..ELAPSED TIME... MEAN	MAXIMUM	(B) .NOT IWAIT TIME.. MEAN	MAXIMUM	DISTRIB. NUMBER	
DHVBT203	ISRT	(01)A1111111		13	123	9.46	108725	257129	65723	226091	3	A,B,C
	GN	(01)A1111111		17	11	0.64	11110	48373	9394	39996	4	A,B,C
	GN	(00)	GB	2	0	0.00	2616	2621	2616	2621	7	A,B,C
	GU	(01)A1111111		25	35	1.40	16999	72658	11467	68101	8	A,B,C
	DLET	(01)A1111111		11	95	8.63	122870	562700	101070	550433	11	A,B,C
	ISRT	(00)	GE	1	2	2.00	14991	14991	6525	6525	14	A,B,C
	ISRT	(03)ACB33333	GE	1	4	4.00	297337	297337	26789	26789	15	A,B,C
	GN	(02)AA222222		4	1	0.25	5084	12256	4567	10188	17	A,B,C
	GU	(02)AC222222		6	11	1.83	17056	24889	11748	18068	13	A,B,C
	GN	(01)A1111111	GA	1	1	1.00	10903	10903	8846	8846	20	A,B,C
	GU	(02)AD222222		2	6	3.00	56002	71814	18077	25025	21	A,B,C
	DLET	(02)AA222222		4	16	4.00	117125	299341	78366	152431	23	A,B,C
	GU	(02)AA222222		3	2	0.66	9384	17670	6373	8636	24	A,B,C
	ISRT	(02)AA222222		4	10	2.50	29166	37864	17785	21928	25	A,B,C
	DLET	(00)	DJ	2	0	0.00	743	746	743	746	26	A,B,C
	GU	(00)	GE	8	20	2.50	32284	116194	10096	15584	27	A,B,C
	ISRT	(02)AC222222		4	22	5.50	65728	99122	39496	56004	28	A,B,C
	GN	(02)AC222222	GK	2	2	1.00	8529	14035	4622	6220	29	A,B,C
	GU	(02)AB222222		1	2	2.00	33831	33831	25963	25963	31	A,B,C
	DLET	(02)AC222222		2	25	12.50	177610	303510	139335	233629	32	A,B,C
	GN	(02)AC222222		4	3	0.75	10263	15346	4777	5993	34	A,B,C
	REPL	(02)AC222222		2	2	1.00	40852	69948	37615	68563	35	A,B,C
	GU	(01)A1111111	GE	4	4	1.00	9580	16110	7371	11761	37	A,B,C
	DLET	(03)ACA33333		1	3	3.00	33056	33056	21171	21171	38	A,B,C
	GU	(03)ACA33333		1	1	1.00	41278	41278	39163	39163	39	A,B,C
	GN	(00)AC222222	GB	1	1	1.00	9992	9992	4112	4112	40	A,B,C
	GN	(02)AC222222	GA	1	0	0.00	3503	3503	3503	3503	41	A,B,C
	GN	(03)ACB33333		2	0	0.00	9317	15611	9317	15611	42	A,B,C
	GN	(03)ACB33333	GK	1	0	0.00	2996	2996	2996	2996	43	A,B,C
	GN	(03)ACA33333		3	0	0.00	3943	6421	3943	6421	44	A,B,C
	GN	(02)AB222222		2	0	0.00	13883	23064	13883	23064	45	A,B,C
	GN	(02)AB222222	GK	1	1	1.00	11926	11926	6115	6115	46	A,B,C
	GN	(02)AA222222	GA	1	0	0.00	3513	3513	3513	3513	47	A,B,C
	GN	(03)AAA33333		3	0	0.00	3158	3402	3158	3402	48	A,B,C
	ISRT	(03)ACA33333		3	8	2.66	41948	59973	35659	55831	49	A,B,C
	ISRT	(03)ACB33333		3	15	5.00	69929	101899	51312	67678	50	A,B,C
	ISRT	(02)AB222222		3	19	6.33	210897	357225	37391	39769	51	A,B,C
	ISRT	(03)AAA33333		3	7	2.33	33177	62173	26812	52213	52	A,B,C
	GU	(03)ACB33333		1	3	3.00	8011118	8011118	7323207	7323207	53	A,B,C
	BATCH TOTAL			153	455	2.97	97373		75761			

Figure 9-4. DL/I Call Summary Report

VSAM STATISTICS REPORT

This report provides statistics on a per call basis for changes in 13 selected subpool statistics between the start and end of VSAM activity. The statistics reported are:

RET RBA	Number of retrieves by RBA calls received by the buffer handler.
RET KEY	Number of retrieves by key calls received by the buffer handler.
ISRT ES	Number of logical records inserted into ESDSs.
ISRT KS	Number of logical records inserted into KSDSs.
BFR ALT	Number of logical records altered.
BKG WTS	Number of times background write function invoked.
SYN PTS	Number of synchronization calls received by the buffer handler.
GETS	Number of VSAM GET calls issued by the buffer handler.
SCHBFR	Number of VSAM SCHBFR calls issued by the buffer handler.
FOUND	Number of times VSAM found the control interval requested was already in the subpool.
READS	Number of times VSAM read a control interval from external storage.
USR WTS	Number of VSAM writes initiated by IMS/VS.
NUR WTS	Number of VSAM writes initiated in the subpool.

The report contains a set of the above statistics for each combination of PCB, call function, and ddname detected in the trace. An occurrence count is printed. Each set of statistics is a summation of the changes in all subpools divided by the number of occurrences. Summary lines show totals for each PCB, for each ddname under the PCB, and for the complete trace.

Figure 9-5 is an example of the VSAM Statistics report.

IMS MONITOR ****VSAM STATISTICS**** TRACE START 1976 164 6:58:19 TRACE STOP 1976 164 6:58:38 PAGE 0005

PCBNAME	DDNAME	DL/I FUNC	VSAM CALLS	GET RBA	GET KEY	ISRT ESDS	ISRT KSDS	BFR ALT	BKG WTS	SYN PTS	GETS	SCHBFR	FOUND	READS	ISR WTS	NUR WTS
DHVBTZ03	DHSKJ901	ISRT	114	1.71	0.00	0.00	0.00	0.98	0.00	0.00	0.75	0.26	0.92	0.00	0.00	0.00
		GN	10	9.40	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	2.00	0.00	0.00	0.00
		GU	39	2.10	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	1.89	0.10	0.00	0.00
		DLET	80	2.67	0.00	0.00	0.00	1.30	0.00	0.00	0.70	0.00	0.70	0.00	0.00	0.00
	DD TOTAL		243	2.41	0.00	0.00	0.00	0.88	0.00	0.00	0.98	0.12	1.05	0.01	0.00	0.00
	DXSKJ101	ISRT	54	0.11	0.85	0.00	0.48	0.18	0.00	0.00	1.33	0.00	3.33	0.00	0.00	0.00
		GN	5	0.80	2.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	4.00	0.00	0.00	0.00
		GU	34	0.47	2.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	3.88	0.11	0.00	0.00
		DLET	22	4.27	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00
	DD TOTAL		115	1.04	1.26	0.00	0.22	0.27	0.00	0.00	1.49	0.00	3.26	0.03	0.00	0.00
	DHSKJ903	ISRT	49	2.69	0.00	0.00	0.00	1.10	0.00	0.00	0.65	0.24	0.65	0.12	0.00	0.00
		GU	9	2.66	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	1.77	0.22	0.00	0.00
		GN	5	4.40	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	2.00	0.00	0.00	0.00
		DLET	37	3.02	0.00	0.00	0.00	1.40	0.00	0.00	0.59	0.00	0.59	0.00	0.00	0.00
		REPL	2	3.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DD TOTAL		102	2.90	0.00	0.00	0.00	1.07	0.00	0.00	0.80	0.11	0.78	0.07	0.00	0.00
	DHSKJ904	GU	2	2.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	1.00	1.00	0.00	0.00
	DD TOTAL		2	2.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	1.00	1.00	0.00	0.00
PCB TOTAL			462	2.17	0.31	0.00	0.05	0.77	0.00	0.00	1.07	0.09	1.54	0.03	0.00	0.00
FAIGT TOTAL			462	2.17	0.31	0.00	0.05	0.77	0.00	0.00	1.07	0.09	1.54	0.03	0.00	0.00

Figure 9-5. VSAM Statistics Report

MONITOR OVERHEAD REPORT

This report provides the total elapsed time during which the Monitor was active, and the total of the time intervals between entry to and exit from the Monitor module. The report also includes the number of Monitor records that were written and the average Monitor time per entry.

Figure 9-6 is an example of the Monitor Overhead report.

```
IMS MONITOR    ****MONITOR OVERHEAD****        TRACE START 1976 164   6:58:19     TRACE STOP  1976 164   6:58:38  PAGE  0006

MONITOR OVERHEAD DATA
      17795 MILLISECONDS, TRACE INTERVAL
       2825 MILLISECONDS, MONITOR MODULE TIME
       1232 MONITOR RECORDS WERE PRODUCED
       2291 MICROSECONDS PER MONITOR ENTRY
```

Figure 9-6. Monitor Overhead Report

DISTRIBUTION APPENDIX REPORT

This report is useful in determining how the values of any of the parameters in the Program I/O and DL/I Call Summary reports were distributed. By referring, for instance, to the example of the DL/I Call Summary report (Figure 9-19), you will note that it shows the overall total, mean and maximum values of IWAIT time intervals. Note also the numbers that appear under the heading "Distribution Number"; these numbers are unique distribtion identifiers that point to entries in the Distribution Appendix report which contain histogram data. Distribution identifiers 3A, 3B, and 3C from the last column of the DL/I Call Summary report can be traced to the Distribution Appendix report and located in the first column of that report. Distribution 3A is for Elapsed Time, distribution 3B for Not-IWAIT time, and distribution 3C for IWAITs per call.

There are two lines of print for each distribution entry in the Distribution Appendix report. The first line indicates the unique distribution identifier (as described above) and the class intervals. The second line of each entry shows the count for each "bucket" in the distribution. Notice that there are ten "buckets" in each distribution -- the lower limit of the first bucket is zero; the upper limit of the tenth "bucket" is "INF", infinity. The class interval for each "bucket" has an assigned default value. This value can be redefined. For a description of how this can be done, see "Redefining Distribution Intervals" later in this chapter.

Figure 9-7 is an example of the Distribution Appendix report.

Figure 9-7. Distribution Appendix Report

```
IMS MONITOR   ****DISTRIBUTION APPENDIX****      TRACE START 1976 164   6:58:19     TRACE STOP  1976 164   6:58:38  PAGE  0007

#  3A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          0          0          0          0          1          9          2          1

   3B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          0          0          0          0          0          9          3          1          0

   3C.........0.........0.........1.........2.........3.........4.........5.........6.........7.........8....INF
                0          0          0          0          0          0          1          1          2          9

#  4A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          3          8          4          0          2          0          0          0

   4B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          3          8          4          1          1          0          0          0

   4C.........0.........0.........1.........2.........3.........4.........5.........6.........7.........8....INF
               11          4          1          0          0          1          0          0          0          0

#  7A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          2          0          0          0          0          0          0          0

   7B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          2          0          0          0          0          0          0          0

   7C.........0.........0.........1.........2.........3.........4.........5.........6.........7.........8....INF
                2          0          0          0          0          0          0          0          0          0

#  8A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          3          2          9         10          0          1          0          0

   8B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          3          8         12          1          0          1          0          0

   8C.........0.........0.........1.........2.........3.........4.........5.........6.........7.........8....INF
                5          5         15          0          0          0          0          0          0          0

#  11A........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          0          0          0          0          5          4          1          1

   11B........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          0          0          0          1          8          0          1          1

   11C........0.........0.........1.........2.........3.........4.........5.........6.........7.........8....INF
                0          0          0          0          0          2          4          1          3          1

#  14A........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                0          0          0          0          1          0          0          0          0          0
```

Redefining Distribution Intervals

The basic units of information presented in the DL/I Call Summary report and the Program I/O report contain distribution information. The Not-IWAIT Time, the Elapsed Time unit, and the I/O IWAITS per DL/I call unit associated with each report are distinct units and have a unique distribution identifier.

The distributions are set up to give ten intervals. Zero is assumed to be the lower boundary and infinity the upper boundary of the distribution interval range. Between zero and infinity, nine finite, non-negative numbers are needed to complete the distribution definition. This is depicted in the following example.

```
0 1 2 3 10 15 20 40 80 160 INF
*                          *
*                          *
* assumed                  * assumed
  lower boundary             upper boundary
```

Each of the distributions for a report has a default, or predefined definition. The user can override these defaults by specifying redefinitions of these distributions in his control statements.

Control Statement Format: The general format for specifying redefinition of a distribution is:

```
Dn n1,n2, . . .
```

where:

Dn
: starts in column 1 and is the distribution identifier (ID); for example, D11 or D12.

n1 through n9
: are each 8 digits or less, and each is a positive number between 0 and 99999999.

Each redefinition can occupy more than one control statement. The format for continuation statements follows the usual OS/VS rules:

- The last value on the first statement must be followed by a comma and at least one blank.
- The first value on the continuation statement cannot start before column 2 and not after column 10.
- Comments can be punched in the statements; they must be preceded by at least one blank.

When the user redefines distributions, the control statements must identify one of the six possible distributions -- D11, D12, D13, D23, D24, or D34. The redefinitions are temporary for the JOB step.

The distribution identifiers are a subset of those of DFSUTR20 and have the same ID codes. They are:

- In DL/I Call Summary Report

  Elapsed Time per call - D11
  Not-IWAIT Time per call - D12
  IWAITs per DL/I - D13

- In Program I/O Report

  IWAIT Time for ISAM/OSAS/I/O - D23
  IWAIT Time for VSAM - D24
  IWAIT Time for HSAM I/O - D34

The default values of distribution intervals are identical to those of DFSUTR20:

D11, D12	(0,1000,2000,4000,8000,16000,32000,64000,128000, 256000, INF)
D13	(0,0,1,2,3,4,5,6,7,8,INF)
D23, D24	(0,2000,8000,24000,50000,100000,150000,200000,250000, 300000,INF)
D34	(0,2000,4000,8000,16000,32000,64000,96000,128000, 160000,INF)

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).
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 log tape (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.

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
//SYSPRINT DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=A
//SYSUT1   DD   DSN=IMSMON,VOL=SER=IMSDA1,UNIT=2400,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 DFSUTR30. The values shown in these reports are not intended to reflect actual values that are received by a user's execution of this utility. For information on operating the DC Monitor, see the IMS/VS System Programming Reference Manual and the IMS/VS Operator's Reference Manual.

The following types of reports are produced by DFSUTR20:

- System Configuration (data about OS/VS and IMS/VS systems used)
- Statistics from Buffer Pools (data collected at beginning and end of trace)
- Region (data on timing, IWAITs, and DL/I calls presented by region)
  - Region Summary
  - Region IWAIT
- Program (data on timing, IWAITs, DL/I calls, scheduling and dequeueing that is presented by an application program)
  - Programs by Region
  - Program Summary
  - Program I/O
- Communication (data on communication subtask timing, IWAITs, transmitted and received blocksizes, intersystem traffic and queueing)
  - Communication Summary
  - Line Functions
  - Communication IWAIT
- Transaction Queueing (data on queue lengths and scheduling occurrences presented by transaction type)
- DL/I Call Summary (statistics on all DL/I calls issued by every program)
- Distribution Appendix (statistical distribution of above data)

DEFINITION OF TERMS USED IN REPORTS

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). For an explanation of those terms that apply to DFSUTR20, see "Definition of Terms Used in Reports" earlier in this chapter.

OUTPUT FROM DFSUTR20

The reports previously listed represent the output of the Monitor Report Print program. Samples of each report are included in this chapter.

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.

REPORT SELECTION

By means of the Analysis control data set (described later in this chapter under "Input to DFSUTR20"), you 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 DL/I call summary report is desired.

ONLY DLI if only a DL/I call summary report is desired.

DIS or DISTRIBUTION if a distribution appendix is desired for the reports requested.

If none of the above options is selected, all of the reports will be printed except the DL/I Call Summary report and the Distribution Appendix report.

STATISTICS FROM BUFFER POOLS REPORT

These summary reports are a formatted display of the contents of selected items of the message queue, data base buffer pool, VSAM buffer pool and message format buffer pool that were collected at the beginning and end of trace. These reports are preceded by a System Configuration report which gives information about VS and IMS systems used. (An example of a System Configuration report is shown in Figure 9-8.)

S Y S T E M C O N F I G U R A T I O N

SYSTEM CONFIGURATION	:	OS/VS1
RELEASE LEVEL	:	1
IMS/VS VERSION	:	1
MODIFICATION NUMBER	:	2

Figure 9-8. System Configuration Report

Since the pool ending values and the difference between starting and ending values cannot be computed if there is no pool ending statistics record written on the trace tape, this summary is produced only if the Monitor is ended before termination of the IMS/VS control region.

In the case where only the ending values of one buffer pool are not written on the trace tape, the corresponding summary report is not computed and the following information message is printed:

```
"NO QUEUE BUFFER POOL TRACES AT END TIME ON MONITOR LOG TAPE
****QUEUE BUFFER POOL REPORT CANCELLED****"
```

The VSAM buffer pool summary report and the message format buffer pool report, or only one of them, will not be produced if the corresponding facility is not invoked through the IMS/VS system definition. In this case, the following information message is printed:

```
"NO VSAM BUFFER POOL TRACES ON MONITOR LOG TAPE
****VSAM BUFFER POOL REPORT CANCELLED****"
```

At the end of the message queue pool summary, the data base buffer pool summary, and the message format buffer pool summary, a quotient shows the total number of WAITS per transaction. This is one of the various performance ratios that can be calculated with the help of each summary report. It may not be significant in itself, but its variation during a tuning operation is important.

An example of the Statistics from Buffer Pools report, showing the pool parameters and the quotients produced by this report, is shown in Figure 9-9.

M E S S A G E Q U E U E P O O L

	7.36.08 START TRACE	8.01.13 END TRACE	DIFFERENCE
NUMBER OF LOCATE CALLS FROM QMGR	49	3776	3727
NUMBER OF RECORD RELEASE CALLS FROM QMGR	30	1728	1698
NUMBER OF LOCATE AND ALTER CALLS FROM QMGR	131	9364	9233
NUMBER OF REQUESTS TO PURGE THE Q POOL	2	2	0
NUMBER OF ADDRESS TO DRRN TRANSLATION REQUESTS	77	26414	26337
NUMBER OF REQUESTS TO WAIT FROM QMGR	0	0	0
NUMBER OF READ REQUESTS	7	4523	4516
NUMBER OF WRITE REQUESTS(TOTAL)	0	2892	2892
NUMBER OF WRITES DONE BY PURGE	0	0	0
NUMBER OF WAITS FOR PURGE COMPLETION	0	0	0
NUMBER OF WAITS BECAUSE NO BUFFER AVAILABLE	0	0	0
NUMBER OF WAITS FOR OTHER DECB TO READ THIS BUFFER	0	17	17
NUMBER OF WAITS FOR OTHER DECB TO WRITE THIS BUFFER	0	112	112
NUMBER OF WAITS FOR CONFLICTING END DEQ BUFFER REQ	0	0	0
NUMBER OF PSBS UNCHAINED FROM BUFFERS	0	70	70
NUMBER OF CALLS TO QMGR.(TOTAL)	114	28053	27939
NUMBER OF CALLS TO REPOSITION A LOST BUFFER	0	0	0
NUMBER OF CALLS TO ENQ A MESSAGE	10	1268	1258
NUMBER OF CALLS TO DEQ ONE OR MORE MESSAGE	5	102	97
NUMBER OF CALLS TO CANCEL INPUT OR OUTPUT	12	41	29

QUOTIENT : $\frac{\text{TOTAL NUMBER OF OSAM READS + OSAM WRITES + ALL IWAITS}}{\text{TOTAL NUMBER OF TRANSACTIONS}}$ = 6.48

Figure 9-9 (Part 1 of 5). Statistics from Buffer Pools

Figure 9-9 (Part 2 of 5). Statistics from Buffer Pools

```
***I M S  M O N I T O R***  BUFFER POOL STATISTICS     TRACE START 1976 049  7:36:08     TRACE STOP  1976 049   8:01:13  PAGE 0003

                    D A T A  B A S E  B U F F E R  P O O L

                                                    7.36.08                 8.01.13
                                                  START TRACE               END   TRACE          DIFFERENCE

NUMBER OF BLOCK REQUESTS RECEIVED                       3                    23687               23684
NUMBER OF REQUESTS SATISFIED FROM POOL                  1                    21108               21107
NUMBER OF READ REQUESTS ISSUED                          1                     2371                2370
NUMBER OF BUFFER ALTERATIONS RECEIVED                   0                     2335                2335
NUMBER OF OSAM WRITES ISSUED                            0                      103                 103
NUMBER OF BLOCKS WRITTEN                                0                      104                 104
NUMBER OF NEW BLOCKS CREATED IN POOL                    0                        7                   7
NUMBER OF CHAINED WRITES ISSUED                         0                        6                   6
NUMBER OF BLOCKS WRITTEN ON WRITE CHAINS                0                        7                   7
NUMBER OF POOL COMPACTION PERFORMED                     0                      304                 304
NUMBER OF BUFFERS COMBINED                              0                      821                 821
NUMBER OF BUFFERS MOVED                                 0                     1928                1928
NUMBER OF RETRIEVE-BY-KEY CALLS                         2                     1064                1067
NUMBER OF GN CALLS RECEIVED                             0                      661                 661
NUMBER OF BISAM READS OR QISAM SETLS                    1                      201                 200
NUMBER OF WRITE ERROR BUFFERS CURRENTLY IN THE POOL     0                        0                   0
NUMBER OF WRITE ERROR BUFFERS DURING THIS EXECUTION     0                        0                   0


QUOTIENT :  TOTAL NUMBER OF OSAM READS + OSAM WRITES + BISAM READS / TOTAL NUMBER OF TRANSACTIONS =  2.29
```

V S A M B U F F E R P O O L

SUBPOOL ID	1
SUBPOOL BUFFER SIZE	128
TOTAL PSTS MODIFIED IT	5
TOTAL BUFFERS IN SUBPOOL	10

	12.04.30 START TRACE	12.58.36 END TRACE	DIFFERENCE
PERM WRT ERROR BUFFS IN SUBPOOL COUNT	10	30	20
LARGEST NUMBER OF PRM ERROR BUFFS EVER IN SUBPOOL		30	
TOTAL NUMBER OF PRM ERROR BUFFERS	10	40	30
TOTAL NUMBER OF IWAIT ISSUED	127	249	122
BLOCK REQUESTS RECEIVED COUNT	96	127	31
REQUEST SATISFIED FROM SUBPOOL.NOT NEW BLOCK NEEDED	127	248	121
READ REQUESTS ISSUED COUNT	127	241	114
BUFFER ALTERATIONS RECEIVED COUNT	0	10	10
NUMBER OF ESDS WRITES ISSUED COUNT	0	0	0
NEW BLOCKS CREATED IN SUBPOOL COUNT	10	40	30
RETRIEVE BY KEY CALLS	35	64	29
GN CALLS	50	123	73
VSAM GETS BY KEY ISSUED COUNT	0	64	64
COUNT OF BLOCKS WRITTEN BY BUFFER HANDLER	80	240	160
COUNT OF CHAINED WRITES BY BUFFER HANDLER	80	128	48
COUNT OF BLOCKS WRITTEN ON CHAINS BY BUFFER HANDLER	80	49	0
COUNT OF BLOCKS WRITTEN BY BACKGROUND WRITE	49	80	31
COUNT OF CHAINED WRITES BY BACKGROUND WRITE	0	145	145
COUNT OF BLOCKS WRITTEN ON CHAINS BY BACKGROUND WRITE	40	245	205
NUMBER OF TIMES BACKGROUND WRITE INVOKED	0	97	97

QUOTIENT : $\frac{\text{TOTAL NUMBER OF IWAITS ISSUED}}{\text{TOTAL NUMBER OF TRANSACTIONS}} = 2.49$

Figure 9-9 (Part 3 of 5). Statistics from Buffer Pools

Figure 9-9 (Part 4 of 5). Statistics from Buffer Pools

```
***I M S  M O N I T O R***  BUFFER POOL STATISTICS    TRACE START 1976 203  21:56:53   TRACE STOP 1976 203  22:37:19  PAGE 0004

                    V S A M  B U F F E R  P O O L

                                                    SUBPOOL ID                 :        1
                                                    SUBPOOL BUFFER SIZE        :     2048
                                                    TOTAL BUFFERS IN SUBPOOL   :       20

                                                  21.56.53        22.37.19
                                                  START TRACE     END   TRACE      DIFFERENCE

NUMBER OF RETRIEVE BY RBA CALLS                         5              19              14
NUMBER OF RETRIEVE BY KEY CALLS                         8              19              11
NUMBER OF LOGICAL RECORDS INSERTED INTO ESDS            0               0               0
NUMBER OF LOGICAL RECORDS INSERTED INTO KSDS            2               7               5
NUMBER OF LOGICAL RECORDS ALTERED IN THIS SUBPOOL       6              17              11
NUMBER OF TIMES BACKGROUND WRITE FUNCTION INVOKED       0               0               0
NUMBER OF SYNCHRONIZATION CALLS RECEIVED                6              26              20
NUMBER OF VSAM GET CALLS ISSUED                        14              41              27
NUMBER OF VSAM SCHBFR CALLS ISSUED                      0               0               0
NUMBER OF TIMES CONT INT REQUESTED ALREADY IN POOL     21              67              46
NUMBER OF CONTR INT READ FROM EXTERNAL STORAGE         11              13               2
NUMBER OF VSAM WRITES INITIATED BY IMS/VS               3              11               8
NUMBER OF VSAM WRITES TO MAKE SPACE IN THE POOL         0               0               0
NUMBER OF PERM WRT ERROR BUFFS NOW IN THE SUBPOOL       0               0               0
LARGEST NUMB OF PERM ERR BUFFS EVER IN THE SUBPOOL      0               0               0
```

Figure 9-9 (Part 5 of 5). Statistics from Buffer Pools

I M S M O N I T O R BUFFER POOL STATISTICS TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0005

M E S S A G E F O R M A T B U F F E R P O O L

	7.36.08 START TRACE	8.01.13 END TRACE	DIFFERENCE
NUMBER OF P/F REQUESTS	0	0	0
NUMBER OF I/F REQUESTS	0	0	0
NUMBER OF P/F I/O'S	0	0	0
NUMBER OF I/F I/O'S	0	0	0
NUMBER OF I/F AND BLOCK STILL ON P/F QUEUE	0	0	0
NUMBER OF I/F WAITS FOR P/F LOAD TO COMPLETE	0	0	0
NUMBER OF TIMES POOL COMPRESS WOULD BE SUCCESSFUL	0	0	0
NUMBER OF DIRECTORY I/O OPERATIONS	44	44	0
NUMBER OF TIMES BLOCK WASHED FOR FREE	0	0	0
NUMBER OF TIMES P/F REQUEST IGNORED	0	0	0
NUMBER OF F/B REQUESTS	0	0	0
NUMBER OF TIMES F/B REQUEST IGNORED	0	0	0
NUMBER OF TIMES I/F ON F/B QUEUE	0	0	0
NUMBER OF TIMES I/F ON I/F QUEUE	0	0	0
NUMBER OF TIMES F/B ON I/F QUEUE	0	0	0
NUMBER OF TIMES P/F ON P/F QUEUE	0	0	0
NUMBER OF TIMES P/F ON I/F QUEUE	0	0	0
NUMBER OF TIMES P/F ON F/B QUEUE	0	0	0
NUMBER OF TIMES THERE WAS NO DIR ENTR FOR A BLOCK	0	0	0
NUMBER OF TIMES I/O ERRORS POINT OP READ MACRO	0	0	0

QUOTIENT : $\frac{\text{TOTAL NUMBER OF PREFETCH I/O'S + IMMEDIATE FETCH I/O'S + DIRECTORY I/O'S OPERATIONS}}{\text{TOTAL NUMBER OF TRANSACTIONS}} = 0.00$

REGION REPORTS

The following reports provide data on timing, IWAITs, and DL/I calls presented by region.

Region Summary Report

Region timing information is printed for every message region (both MPP and BMP) active during the trace. This summary report categorizes regions as follows:

- Scheduling and termination
- Schedule to first DL/I call
- Elapsed execution
- DL/I call
- Idle for intent
- Checkpoint
- Region Occupancy

It should be noted that some of the values shown for region timing overlap in the timeframe of the trace period. Elapsed time for scheduling and termination are included in idle-for-intent time. The elapsed time for execution includes the elapsed time for DL/I calls. In general, the trace time period is slightly greater than the sum of scheduling and termination, schedule to first DL/I call, elapsed execution time, and idle time because of intent. Differences between this sum and trace time can be attributed to transactions active at the startup and shutdown of the tracing, or idle regions at the start or end of a trace.

In DFSUTR20, a program name of PGMREGnn is created for each region in the IMS/VS system that is active at the time the trace is started. The "nn" refers to the IMS/VS region ID. Available data for these partially executed programs is accumulated under these assigned names in the reports. This data for PGMREGnn programs is significant if the programs were executed in a batch message processing region. Programs initiated after tracing was started are identified by their actual names. These assigned names, PGMREGnn, propagate throughout other reports.

Figure 9-10 is an example of the Region Summary report.

		OCCURRENCES	(A) ELAPSED TIME TOTAL	ELAPSED TIME MEAN	ELAPSED TIME MAXIMUM	(B) NOT IWAIT TIME (ELAPSED-IWAIT) TOTAL	NOT IWAIT TIME (ELAPSED-IWAIT) MEAN	NOT IWAIT TIME (ELAPSED-IWAIT) MAXIMUM		DISTRIBUTION NUMBER
SCHEDULING AND TERMINATION										
**REGION	1	17	31726632	1866272	17877808	2528959	148762	1463876		78A,B
**REGION	2	3	66959304	22329768	55125343	8750140	2916716	6130273		48A,B
**REGION	3	3	25506878	8502292	14750637	7340523	2446841	3680510		27A,B
**REGION	4	1	6117151	6117151	6117151	5617424	5617424	5617424		1A,B
**TOTALS		24	130339965	5430831		24237055	1009877			
SCHEDULE TO FIRST DLZI CALL										
**REGION	1	18	17512898	972938	4987555					77
**REGION	2	3	9634074	3211358	7791042					47
**REGION	3	3	6213237	2071079	3083621					26
**REGION	4	1	1022103	1022103	1022103					10
**TOTALS		25	34382312	1375292						
ELAPSED EXECUTION										
**REGION	1	18	134599345	7477741	55088590					71
**REGION	2	3	730467392	243495797	459033334					37
**REGION	3	3	859800354	286533451	622004398					19
**REGION	4	1	233932780	233932780	233932780					2
**TOTALS		25	1958619371	78344794						
DLZI CALLS									IWAITCALL(C)	
**REGION	1	4103	118681715	28925	9317785	67660933	16495	9237120	0.18	72A,B,C
**REGION	2	5264	671803157	127622	45537798	306904507	58302	28768273	0.53	38A,B,C
**REGION	3	5330	754396455	129399	46295357	622476641	106771	46295357	0.22	19A,B,C
**REGION	4	2686	179270985	66742	23603995	166066841	61926	23603995	0.11	3A,B,C
**TOTALS		17833	1724152312	96412		1163128922	65041		0.29	

IDLE FOR INTENT NONE

CHECKPOINT NONE

Region Occupancy

**REGION	1	12.2%
**REGION	2	53.6%
**REGION	3	27.2%
**REGION	4	16.0%

Figure 9-10. Region Summary Report

Scheduling and Termination: Lines under this heading, for each region, show the number of occurrences of scheduling and termination in the region, and both the elapsed execution time and not IWAIT time associated with scheduling and termination. The total of all intervals, the maximum single interval, and the mean interval values are shown.

The elapsed time during which the scheduler is internally waiting is not included in the elapsed time shown for scheduling and termination.

This line of the report does not include data for a message region that was not scheduled but was executing at the start of the trace.

Schedule to First DL/I Call: The lines under this subheading show the elapsed time for the following to occur: the region to gain control after being scheduled; the program either to be located in the region or to be brought into the region; or the program to issue the first DL/I call requiring dispatching of the IMS/VS Call Analyzer Module.

This section does not appear for a message region or a batch message region that was not scheduled during the trace; it does not appear for one that was scheduled but did not issue a DL/I call following the scheduling. The number of program loads is one less than the number of schedulings, if the trace was ended after the scheduling but before the first DL/I call of the last scheduling.

The transaction class scheduling mechanism of IMS/VS can be used to control what programs are scheduled in a message region. Another report, the Programs by Region Summary, can be used to determine the names and the frequency of usage of the programs executing in a message region.

Elapsed Execution: Lines under this subheading give the number of executions of programs in each region and the elapsed time for each execution.

The number of executions is one less than the number of schedulings and schedule to first DL/I calls in the case of a program that was scheduled, issued a DL/I call, but was still executing when the trace was ended.

A list of the programs executing in this region can be obtained from the Programs by Region Summary.

DL/I Call: Lines uder this subheading give the total number of DL/I calls from each region during the trace, the total, mean and maximum elapsed execution time intervals to complete those calls, and the total, mean and maximum non-IWAIT intervals used to complete those calls. The number of IWAITs per call is computed and displayed for each region under the heading "IWT/CALL".

Idle for Intent: Lines under this subheading give the number of times a region was in the idle state because of intent, the total elapsed time it was in this state, the mean time in this state, and the maximum of those times.

A region that is idle because of intent can again fail to be scheduled because of intent conflicts. In this case, the number of times the region was idle because of intent does not equal the number of intent failures.

The second failure would not be added to the count of the times the region was in the "idle because of intent state", but the elapsed time would be added to total elapsed time. These failures are reflected, by program and data base, in the Intent Fail report.

Idle for intent occurrences begin when the scheduler waits for intent; the occurrences end when a transaction is scheduled in the region.

The idle because of intent information is useful in determining the effect of intent-checking during the scheduling process. This could prove useful, in combination with the Intent Failure report, in determining the benefits of IMS/VS program isolation for existing operational applications. This information could also be useful in checking the effect of the scheduling of a program with intent conflict into the same region.

Checkpoint: This line is produced if a checkpoint occurs during the trace.

The line gives the number of times checkpoint was dispatched, the total elapsed time of the checkpoints, the mean elapsed time for a checkpoint, and the maximum of those elapsed times.

The line also gives the total non-IWAIT time for the checkpoints, the average non-IWAIT time for a checkpoint, and the maximum of those non-IWAIT times.

Region Occupancy

Lines under this sub-heading indicate the percentage of time that the region was occupied. This value is determined from the formula:

$$\text{Region Occupancy} = \frac{\text{scheduling + termination + schedule to 1st DL/I call + program elapsed + idle for intent}}{\text{trace elapsed}}$$

The value of trace elapsed time is the difference between the time recorded for the first traced IMS/VS event and the last traced IMS/VS event.

Region IWAIT Report

This summary report furnishes IWAIT information for each region for the following events:

- Scheduling and Termination
- DL/I Calls
- Checkpoint

The checkpoint lines appear on the region 0 summary page. There is a separate page produced for each region that was active during the trace.

For the events listed above, the summary complements the information given in the Region Summary report. Figure 9-11 is an example of the Region IWAIT report.

Figure 9-11. Region IWAIT Report

```
IMS MONITOR   ****REGION IWAIT****          TRACE START 1976 049  7:36:08   TRACE STOP  1976 049  8:01:13  PAGE 0007

                                        ....IWAIT TIME...                    DISTRIBUTION
**REGION  1   OCCURRENCES      TOTAL        MEAN      MAXIMUM    FUNCTION     MODULE   NUMBER
SCHEDULING + TERMINATION
                        13   144007210   11077477   85993120   NO MESSAGES  MSC          60
...SUB=TOTAL...         13   144007210   11077477

                       482    27678069      57423     994480   DD=LGMSG     QMG          64
                         7      428934      61276     130135   DD=QBLKS     QMG          65
                        11      919855      83623     135654   DD=SHMSG     QMG          68
                         1      135357     135357     135357   PSB=DFSDDLT8 BLR          84
                         1      936886     936886     936886   DMB=DD41M803 BLR          85
                         1      894213     894213     894213   PSB=DFSDDLT4 BLR          88
                         1     1767878    1767878    1767878   DMB=DX41M401 BLR          89
                         1      974486     974486     974486   DMB=DH41M403 BLR          90
...TOTAL...            518   177742888     343132

DL/I CALLS
                        25     4546411     181856    1949197   DD=SHMSG     QMG          73
                       654     9704679      61325    2835270   DD=LGMSG     QMG          74
                         8      594032      74254     164653   DD=QBLKS     QMG          75
                        64     2765752      43214     514074   DD=M8SK0301  DBH          76
                        15     1863925     124261     387428   DD=DXM40301  DBH          86
                        24     1339492      55812     266457   DD=DHM40301  DBH          87
                        11     1065042      96822     170394   DD=M1I3I1    DBH          91
                        17     1785449     105026     400308   DD=M1I301    DBH          92
...TOTAL...            768    21000782      66407

CHECKPOINT
...TOTAL...
```

Scheduling and Termination: Lines under this subheading give the number of occurrences of IWAITs on the block loader during the scheduling of programs into each region. The ddname field is not used here. Instead, an indication is given that the IWAITs are by module BLR for ACBLIB. The BLR function is indicated by the notations "DMB=", "PSB=", and "INT=" for DMB load, PSB load, and intent list, respectively. A subtotal of the BLR activity is produced.

A line is produced under this sub-heading for each reason for which the scheduler does an internal IWAIT. The possible reasons include (1) checkpoint, (2) no messages, (3) BLR busy, and (4) intent. The IWAIT times given for internal scheduler IWAITs are not included in the scheduler elapsed time shown in the Region Summary.

A line is produced under this sub-heading for each combination of DDNAME DD= and module ID that caused I/O IWAITs during region termination. The number of occurrences, and the maximum, mean, and total of these IWAIT times are given.

DL/I Calls: A line is produced, under this sub-heading, for each combination of ddname and module ID which caused I/O IWAITs during a DL/I call in each region during the trace. The lines give, for each combination, the number of occurrences of an IWAIT, and the maximum, mean and total IWAIT times for these occurrences. IWAITs for storage are indicated by "STG.=", and the pool identification is shown.

This section of the summary can be used to spot-check any abnormal IWAIT times by data set which, if they occurred, might indicate I/O path (channels, etc.) contention problems or data set contention problems (arm contention). The user should be aware of the placement and use of those data sets or packs that are accessed during the time that the online IMS/VS system is running. This region report section can also indicate OSAM overflow chains and ISAM index search time because ddname and IWAIT times identify each accessed data set.

Checkpoint: This subheading appears only on first page of this summary. This subheading appears only on first page of this summary. A line is produced for each combination of ddname and module ID that caused I/O IWAITs during checkpoints. The number of occurrences of these IWAITs, and the maximum, mean and total times are given. IWAITs for storage are indicated by "STG.=", and the pool identification is shown.

PROGRAM REPORTS

The following reports provide data on timing, IWAITs, DL/I calls, scheduling and dequeueing that are presented by an application program.

Programs by Region Report

This summary report displays by region, and for each program scheduled within the region, the number of occurrences and time intervals related to program elapsed execution time and program load time (schedule to first DL/I call). This information is useful to calculate the ratios of program load time to program execution time. Figure 9-12 is an example of the Programs by Region report.

IMS MONITOR ****PROGRAMS BY REGION**** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0011

	OCCURRENCES	(A) ELAPSED EXECUTION TIME TOTAL	MEAN	MAXIMUM	(B) SCHED. END TO FIRST DL/I CALL TOTAL	MEAN	MAXIMUM	DISTRIBUTION NUMBER
**REGION 1								
DFSDDLT8	2	55387643	27693821	42060281	5561454	2780727	4987555	933A,B
DFSDDLT4	1	55088590	55088590	55088590	1050126	1050126	1050126	943A,B
DFSDDLT1	15	24123112	1608207	5481014	10901318	726754	1715183	952A,B
REGION TOTALS	18	134599345	7477741		17512898	972938		
**REGION 2								
DFSDDLT9	1	180119479	180119479	180119479	1150476	1150476	1150476	892A,B
DFSDDLT7	1	459033384	459033384	459033384	692556	692556	692556	904A,B
DFSDDLT2	1	91334529	91334529	91334529	7791042	7791042	7791042	923A,B
REGION TOTALS	3	730487392	243495797		9634074	3211358		
**REGION 3								
DFSDDLT6	1	139243615	139243615	139243615	450325	450325	450325	872A,B
DFSDDLT3	1	622004398	622004398	622004398	2679291	2679291	2679291	882A,B
DFSDDLT5	1	98352341	98352341	98352341	3083621	3083621	3083621	911A,B
REGION TOTALS	3	859600354	286533451		6213237	2071079		
**REGION 4								
HHTASK41	1	233932780	233932780	233932780	1022103	1022103	1022103	862A,B
REGION TOTALS	1	233932780	233932780		1022103	1022103		

Figure 9-12. Programs by Region Report

Time intervals are expressed in terms of maximum observed, mean, and total of all occurrences. Totals and overall mean values are shown for each region. The value given under the heading "Occurrences" is the number of times the program was scheduled into the region. The values given under the headings "Elapsed Execution Time" and "Sched. End to First DL/I Call" are defined in the section "Definition of Terms Used in Reports."

The count of schedule to first DL/I calls is one greater than the program elapsed execution count if the program was scheduled during the trace, issued a DL/I call, and had not terminated execution at the time the trace was stopped.

Program Summary Report

This report provides the following information for each program scheduled into any region during the trace:

- PSB name associated with the program.
- Number of times it was scheduled.
- Number of transactions dequeued.
- Total DL/I calls made.
- Ratio of DL/I calls per transaction.
- Number of I/O IWAITs sustained.
- Ratio of IWAITs to DL/I calls.
- Ratio of transactions dequeued per scheduling.
- CPU time used per scheduling.
- Elapsed time per scheduling.
- Average schedule to first DL/I call interval.
- Average elapsed time per transaction.
- Totals for all programs.

Figure 9-13 is an example of the Program Summary report.

PSBNAME	NO. SCHEDS.	TRANS. DEQ.	DL/I CALLS	DL/I CALLS /TRAN	I/O IWAITS	I/O IWAITS /CALL	(A) TRAN: DEQD. /SCH.	(B) CPU TIME /SCHED.	(B) DISTR. NO.	(A) ELAPSED TIME /SCHED.	(B) SCHED.TO 1ST DLI /SCHED.	(B) DISTR. NO.	ELAPSED TIME /TRANS.
DFHTASK	41	0	2686	-	312	0.1	0.0	10210538	861A,B	233932780	1022103	863A,B	-
DFSDDLT6	1	116	2577	22.2	336	0.1	116.0	3973658	871A,B	139243615	450325	873A,B	1200375
DFSDDLT3	1	117	2580	22.0	625	0.2	117.0	4115982	881A,B	622004398	2679291	883A,B	5316276
DFSDDLT9	1	116	2577	22.2	363	0.1	116.0	4293328	891A,B	180119479	1150476	893A,B	1552754
DFSDDLT7	1	181	1342	7.4	1996	1.4	181.0	6964256	903A,B	459033384	692556	905A,B	2536096
DFSDDLT5	1	91	673	7.3	340	0.5	91.0	1112020	910A,B	98352341	3083621	912A,B	1080794
DFSDDLT2	1	182	1345	7.3	474	0.3	182.0	2253212	922A,B	91334529	7791042	924A,B	501838
DFSDDLT8	2	91	2048	22.5	324	0.1	45.5	1766661	932A,B	27603921	2780727	934A,B	608655
DFSDDLT4	1	181	1342	7.4	416	0.3	181.0	2272686	942A,B	55088590	1050126	944A,B	304356
DFSDDLT1	15	88	713	8.1	28	0.0	5.8	86202	951A,B	1608207	726754	953A,B	274126
**TOTALS	25	1163	17883	15.3	5214	0.2	46.5	1600881		78344794	1375292		1684109

Figure 9-13. Program Summary Report

Program I/O Report

This is a report of the total, maximum, and mean IWAIT intervals recorded for I/O IWAITs caused by DL/I calls by each program scheduled during the trace.

The data is arranged, under each program, by PCB name and by ddname, and the module ID of the module that IWAITed. IWAITs for storage are indicated by "STG.=", and the pool identification is shown.

Program I/O caused by DL/I calls for the I/O PCB are grouped together as the first data for each PSB in the report. The data given under the column heading "IWAITS" indicates the number of times that DL/I calls for the associated PCB were required to wait for I/O activity to complete. The data set for which the I/O took place is indicated by the ddname. The entries under the heading "Module" are abbreviated module identifications for IMS/VS modules. The following list is a cross-reference between the abbreviated identification and the actual IMS/VS module name.

Abbreviated ID	Module Name
BLR	DFSDBLR0
DBH	DFSDBHR0
DLE	DFSDDLE0
MSC	DFSSMSC0
PMM	DFSFPMM0
PRF	DFSFPRF0
QMG	DFSQMGR0
SMN	DFSISMN0
VBH	DFSDVSM0

Subtotals are given for each PCB under a PSB, and for all PCBs under each PSB. Also provided is a total for all programs. Figure 9-14 is an example of the Program I/O report.

IMS MONITOR ****PROGRAM I/O**** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0013

PSBNAME	PCB NAME	IWAITS	....IWAIT TIME.... TOTAL	MEAN	MAXIMUM	DDNAME	MODULE	DISTR. NO.
HHTASK41	DH41SK01	43	1280851	29787	59410	DHSK0101	DBH	864
		84	3701608	44069	1043804	DHSK0102	DBH	865
		80	2149923	26874	148936	DHSK0103	DBH	866
		47	2462519	52394	644552	DHSK0104	DBH	867
		12	358664	29888	51945	DXSK0102	DBH	868
		46	3250379	70660	519082	DXSK0101	DBH	869
	PCB TOTAL	312	13204144	42320				
PSB TOTAL		312	13204144	42320				
DFSDDLT6	I/O PCB	289	10060815	34812	203613	LGMSG	QMG	874
		4	308544	77136	190499	QBLKS	QMG	875
		10	422585	42258	95736	SHMSG	QMG	879
	PCB TOTAL	303	10791944	35616				
	DH414602	11	261953	23813	47141	DHM60201	DBH	876
		13	386547	29734	61692	DHM60202	DBH	877
		9	607942	67549	220147	M6SKI201	DBH	878
	PCB TOTAL	33	1256442	38074				
PSB TOTAL		336	12048386	35858				
DFSDDLT3	I/O PCB	302	24942107	82589	1240412	LGMSG	QMG	884
		9	1405743	156193	902019	QBLKS	QMG	885
		15	1622663	108177	544727	SHMSG	QMG	887
	PCB TOTAL	326	27970513	85799				
	DI424301	189	37376447	197758	23907210	M3I30B	DBH	886
		34	2391815	70347	290155	M3I30C	DBH	888
		11	445752	40522	77794	M3I30A	DBH	889
		25	4536361	181454	559021	M3I31B	DBH	899
		13	3111696	239361	968493	M3I31C	DBH	900
		27	4845355	179457	436190	M3I31A	DBH	901
	PCB TOTAL	299	52707426	176279				
PSB TOTAL		625	80677939	129084				
DFSDDLT9	I/O PCB	284	27048641	95241	16089986	LGMSG	QMG	894
		4	854168	213542	763510	QBLKS	QMG	895
		14	578986	41356	159412	SHMSG	QMG	898
	PCB TOTAL	302	28481795	94310				
	DD414901	53	24370624	459823	22147472	M9SK0101	DBH	896
		8	306067	38258	57177	M9SK0102	DBH	897
	PCB TOTAL	61	24676691	404535				
PSB TOTAL		363	53158486	146442				
DFSDDLT7	I/O PCB	389	54245257	139447	27127637	LGMSG	QMG	906

Figure 9-14. Program I/O Report

COMMUNICATION REPORTS

The following reports provide data on communication subtask timing, IWAITs, transmitted and received blocksizes, intersystem traffic, and queueing.

Communication Summary Report

This report contains a line for each teleprocessing line or VTAM node that was active during the trace.

For each teleprocessing line or VTAM node, the number of communication subtasks dispatched is given by the number shown under the heading "Occurrences." Both elapsed and non-IWAIT intervals of time are given for those subtasks in terms of maximum interval, mean interval, and the total of intervals.

The column heading designations "Elapsed Time" and "Not IWAIT Time" are defined under "Definition of Terms Used in Reports" earlier in this chapter.

The last line on this summary is labeled "Prefetch" and is a total of all MFS format block prefetch events.

Figure 9-15 is an example of the Communication Summary report.

IMS MONITOR ****COMMUNICATION SUMMARY**** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0016

LINE NUMBER	OCCURRENCES	(A) ELAPSED TIME TOTAL	ELAPSED TIME MEAN	ELAPSED TIME MAXIMUM	(B) NOT IWAIT TIME TOTAL	(ELAPSED-IWAIT) MEAN	(ELAPSED-IWAIT) MAXIMUM	DISTRIBUTION NUMBER	
2	146	9317033	63815	2182147	8304458	56879	2182147	957	A,B
1	7	13614395	1944913	3070687	11658462	1665494	3070687	959	A,B
18	15	464065	30937	242098	267279	17818	179353	961	A,B
13	16	576310	36019	191700	217089	13568	157047	968	A,B
19	16	712902	44556	461050	614444	38402	431818	974	A,B
30	2209	131069761	59334	39016347	46985709	21270	20518222	979	A,B
33	2208	81198479	36774	7599042	26785461	12131	1663861	985	A,B
36	2208	116897734	52942	35979604	58478913	26485	35979604	992	A,B
17	13	710336	54679	361584	177097	13622	81035	997	A,B
10	11	1369554	124504	541085	161771	14706	88998	1002	A,B
12	14	626236	44731	162946	70276	5019	28712	1009	A,B
18	15	366190	24412	226231	54660	3644	24024	1014	A,B
35	1752	48458539	27658	2905525	13385312	7640	903176	1018	A,B
29	1269	65877424	51107	9251980	12922504	10025	1011988	1024	A,B
32	646	29930424	46331	1579276	4564404	7065	650230	1031	A,B
14	13	1196559	92043	430236	92424	7109	36547	1036	A,B
31	1268	30317047	23538	2778946	11260378	8742	1903904	1042	A,B
34	1268	30740321	23866	2885485	8609457	6684	493338	1044	A,B
11	35	484579	13845	101300	167046	4772	73508	1053	A,B
28	620	7434545	11991	345790	4574033	7377	165135	1056	A,B
TOTAL	13809	571362988	41376		209351177	15160			

Figure 9-15. Communication Summary Report

Line Functions

This report provides the following information after the trace:

- Identification for each line or VTAM node.
- Device type.
- Count of received blocks.
- Count of transmitted blocks.
- Mean and maximum blocksize.
- Number of line-inactive intervals for line and device (inactive because of IMS/VS processing of an I/O interrupt).
- Turnaround interval mean and maximum.
- Number of requests from an MFS-supported terminal for next page of a multipage message.

Figure 9-16 is an example of the Line Functions report.

IMS MONITOR ****LINE FUNCTIONS*** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0020

NODE OR LINE NUMBER	DEVICE TYPE	(A) RECEIVE BLOCKS	MEAN RECEIVE BLKSIZE	MAX. RECEIVE BLKSIZE	(B) TRANS. BLOCKS	MEAN TRANS. BLKSIZE	MAX. TRANS. BLKSIZE	DIST. NUMBER	TURN AROUND INTERVALS	MEAN INTERVAL	MAX. INTERVAL	DIST. NUMB.	PAGING REQUESTS
2	2740-1	33	13864	13864	102	133	133	958A,B	146	63815	2182147	956	0
16	LOC SYS	0		0	11	129	129	962A,B	15	30937	242098	960	0
13	LOC SYS	0		0	12	129	129	969A,B	16	36019	191700	967	0
19	LOC SYS	0		0	11	129	129	975A,B	16	44556	461050	973	0
30	LOC SYS	2208	80	80	0		0	983A,B	2209	59334	39016347	978	0
33	LOC SYS	2207	80	80	0		0	990A,B	2208	36774	7599042	984	0
36	LOC SYS	2207	80	80	0		0	995A,B	2208	52942	35979604	991	0
17	LOC SYS	0		0	9	129	129	998A,B	13	54679	361584	996	0
15	LOC SYS	0		0	7	129	129	1003A,B	11	124504	541085	1001	0
12	LOC SYS	0		0	10	129	129	1010A,B	14	44731	162946	1008	0
18	LOC SYS	0		0	10	129	129	1016A,B	15	24412	226231	1013	0
35	LOC SYS	1751	80	80	0		0	1022A,B	1752	27658	2905525	1017	0
29	LOC SYS	1289	80	80	0		0	1028A,B	1289	51107	9251980	1023	0
32	LOC SYS	645	80	80	0		0	1033A,B	646	46331	1579276	1030	0
14	LOC SYS	0		0	9	129	129	1037A,B	13	92043	430236	1035	0
31	LOC SYS	1287	80	80	0		0	1048A,B	1288	23538	2778946	1041	0
34	LOC SYS	1287	80	80	0		0	1049A,B	1288	23866	2885485	1043	0
11	LOC SYS	0		0	19	128	128	1054A,B	35	13845	101300	1052	0
28	LOC SYS	620	80	80	0		0	1057A,B	620	11991	345790	1055	0
ALL LINES		13533	113		200	130			13802	40410			0

Figure 9-16. Line Functions Report

Communication IWAIT Report

This report is complementary to the Communication Summary report. I/O IWAITs that occur during the dispatches of the communication subtask are displayed for each teleprocessing line or node active during the trace.

A separate line is given for each combination of function and module identification that caused the IWAIT.

The IWAITs related to MFS services are indicated as follows:

```
I/O=DIR=(format name):  I/O IWAIT for directory
I/O=BLK=(format name):  I/O IWAIT for block
DCB=DIR=(format name):  IWAIT for busy DCB for directory
DCB=BLK=(format name):  IWAIT for busy DCB for block
```

The block size involved in each of the above MFS functions is shown under the column heading "BLKSIZE." IWAITs for storage are indicated by "STG.=", and the pool identification is given.

Other IWAITs are indicated as "DDNAME=" functions, and the ddname causing the IWAIT is provided.

IWAIT interval durations are expressed as the maximum observed, the mean, and the total of all occurrences in each case. For a definition of IWAIT time, see the section "Definition of Terms Used in Reports."

This report can be used to check I/O activity on the message queue data sets, and MFS data set for high frequency of occurrence or high elapsed I/O times. Any changes in placement of the data sets, or in the size of the queue pool or format pool, can be monitored for these effects by using this report.

The last group of lines of this summary displays the total I/O IWAIT occurrences caused by MFS format block prefetches.

Figure 9-17 is an example of the Communication IWAIT report.

Figure 9-17. Communication IWAIT Report

IMS MONITOR ****COMMUNICATION IWAIT***** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0017

NODE OR LINE NUMBER	OCCURRENCES	IWAIT TIME TOTAL	IWAIT TIME MEAN	IWAIT TIME MAXIMUM	FUNCTION	BLKSIZE	MODULE	DIST. NO.
10	1	134041	134041	134041	DDNAME=LGMSG		QMG	963
	1	62745	62745	62745	DDNAME=SHMSG		QMG	964
LINE TOTAL..	2	196786	98393					
1	4	345320	86330	137450	DDNAME=QBLKS		QMG	965
	13	818287	62945	133863	DDNAME=LGMSG		QMG	966
	5	792326	158465	316485	DDNAME=SHMSG		QMG	972
LINE TOTAL..	22	1955933	88906					
13	4	276002	69000	81126	DDNAME=LGMSG		QMG	970
	3	83219	27739	38098	DDNAME=SHMSG		QMG	971
LINE TOTAL..	7	359221	51317					
19	1	69226	69226	69226	DDNAME=LGMSG		QMG	976
	1	29232	29232	29232	DDNAME=SHMSG		QMG	977
LINE TOTAL..	2	98458	49229					
30	3	1128231	376077	968088	DDNAME=QBLKS		QMG	980
	438	82307569	187916	38458257	DDNAME=LGMSG		QMG	981
	8	648252	81031	171127	DDNAME=SHMSG		QMG	982
LINE TOTAL..	449	84084052	187269					
33	2	115063	57531	59784	DDNAME=QBLKS		QMG	986
	421	53706846	127574	7563718	DDNAME=LGMSG		QMG	987
	8	589109	73638	162682	DDNAME=SHMSG		QMG	989
LINE TOTAL..	431	54413018	126248					
2	5	429579	85915	128077	DDNAME=LGMSG		QMG	988
	2	583046	291523	367794	DDNAME=SHMSG		QMG	1007
LINE TOTAL..	7	1012625	144660					
36	5	222747	44549	48262	DDNAME=SHMSG		QMG	993
	403	58153174	144300	17431123	DDNAME=LGMSG		QMG	994
	1	42900	42900	42900	DDNAME=QBLKS		QMG	1006
LINE TOTAL..	409	58418821	142833					
17	4	442900	110725	226041	DDNAME=LGMSG		QMG	999
	1	90839	90839	90839	DDNAME=SHMSG		QMG	1000
LINE TOTAL..	5	533739	106747					

TRANSACTION QUEUEING REPORT

A line is printed for each transaction type encountered during the trace. Column headings are:

- Transaction -- the 8-character transaction name.
- Number dequeued -- the total number of this transaction type processed during the trace.
- Number Scheds. -- the total number of times that a PSB was scheduled for this transaction.
- Minimum, mean, and maximum number of messages on the queue when the program was scheduled for this transaction.
- Dequeued Mean -- the mean of the number of these transactions dequeued for a single scheduling of the program.

This report is useful in monitoring the effective process limit count for each transaction. It can also be useful in indicating transaction queue depths at each scheduling. Figure 9-18 is an example of the Transaction Queueing report.

Figure 9-18. Transaction Queueing Report

IMS MONITOR ****TRANSACTION QUEUEING**** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0024

TRANSACTION	NUMBER DEQUED	NUMBER SCHEDS.	(B) ON QUEUE WHEN SCHEDULED..... MINIMUM	MEAN	MAXIMUM	(A) DEQUED MEAN	DISTRIBUTION NUMBER
SKS6	116	1	115	115.00	115	116.00	870 A,B
SKS3	117	1	101	101.00	101	117.00	880 A,B
SKS9	116	1	115	115.00	115	116.00	890 A,B
SKS7	181	1	180	180.00	180	181.00	902 A,B
SKS5	91	1	90	90.00	90	91.00	909 A,B
SKS2	182	1	101	101.00	101	182.00	921 A,B
SKS8	91	2	0	44.50	89	45.50	931 A,B
SKS4	181	1	180	180.00	180	181.00	941 A,B
SKS1	88	15	0	0.13	1	5.86	950 A,B

DL/I CALL SUMMARY REPORT

This is a compact DL/I call report for the trace. All DL/I calls issued by every program scheduled during the trace are arranged as follows:

- Program name.
- For each program, every PCB used.
- For each PCB, the call function employed.
- For each call function, the segment accessed and its level number.
- For each segment, the return code obtained.

The DL/I calls made with the I/O PCB are grouped together for each PSB, and a subtotal of I/O PCB calls is given. For each line in this report, the number of DL/I calls recorded, the IWAITs per each call, and both the average and maximum elapsed, and non-IWAIT times are given. Figure 9-19 is an example of the DL/I Call Summary report.

Figure 9-19. DL/I Call Summary Report

IMS MONITOR ****DL/I CALL SUMMARY**** TRACE START 1976 049 7:36:08 TRACE STOP 1976 049 8:01:13 PAGE 0026

PSB NAME	PCB NAME	CALL FUNC	LEV NO.	SEGMENT	STAT CODE	DL/I CALLS	IWAITS	(C) IWAITS/ CALL	(A) ELAPSED TIME MEAN	ELAPSED TIME MAXIMUM	(B) IWAIT TIME MEAN	IWAIT TIME MAXIMUM	DISTRIB. NUMBER	
HHTASK41	DH41SK01	STAT	(00)		GE	1	0	0.00	637	637	637	637	93	A,B,C
		STAT	(00)			2	0	0.00	834	834	834	834	94	A,B,C
		PURG	(00)	AD222222		1	1	1.00	27218	27218	4667	4667	95	A,B,C
		GN	(00)	AD222222	GB	12	0	0.00	1581	2657	1581	2657	96	A,B,C
		GN	(02)	AD222222		91	3	0.03	3521	75178	2417	17724	97	A,B,C
		GN	(02)	AD222222	GK	13	3	0.23	9398	65935	3177	14870	98	A,B,C
		GN	(02)	AB222222		58	21	0.36	27681	694913	7419	279655	99	A,B,C
		GN	(01)	A1111111	GA	25	1	0.04	13077	214030	12093	214030	100	A,B,C
		GN	(02)	AC222222		73	4	0.05	4334	94008	2683	14658	101	A,B,C
		GN	(01)	A1111111		111	6	0.05	150442	14316862	148714	14237993	102	A,B,C
		GN	(08)	ADAAAAAA		32	0	0.00	2102	7901	2102	7901	103	A,B,C
		GN	(07)	ADAAAAA7		26	1	0.03	587564	15144812	586427	15144812	104	A,B,C
		GN	(06)	ADAAAA66		24	0	0.00	1368	2754	1368	2754	105	A,B,C
		GN	(05)	ADAAA555		24	0	0.00	1306	2879	1306	2879	106	A,B,C
		GN	(04)	ADAA4444		24	0	0.00	1236	1628	1236	1628	107	A,B,C
		GN	(03)	ADA33333		32	0	0.00	2168	8044	2168	8044	108	A,B,C
		GN	(02)	AD222222	GA	15	1	0.06	4192	41412	1684	3784	109	A,B,C
		GN	(04)	ACBB4444		23	0	0.00	17712	377079	17712	377079	110	A,B,C
		GN	(04)	ACBB4444	GA	8	0	0.00	1476	1921	1476	1921	111	A,B,C
		GN	(08)	ACBABAAB		17	0	0.00	9673	132436	9673	132436	112	A,B,C
		GN	(08)	ACBABAAB	GK	11	0	0.00	1752	5283	1752	5283	113	A,B,C
		GN	(08)	ACBABAAA		27	2	0.07	3030	45718	2020	18437	114	A,B,C
		GN	(07)	ACBABAA7		26	2	0.07	3436	32408	1848	6969	115	A,B,C
		GN	(06)	ACBABA66		28	1	0.03	3190	20950	2571	12530	116	A,B,C
		GN	(05)	ACBAB555		12	0	0.00	1684	4150	1684	4150	117	A,B,C
		GN	(05)	ACBAB555	GK	13	0	0.00	1852	6443	1852	5443	118	A,B,C
		GN	(05)	ACBAA555		34	1	0.02	2338	32313	1505	4003	119	A,B,C
		GN	(04)	ACBA4444		30	0	0.00	1558	4318	1558	4318	120	A,B,C
		GN	(03)	ACB33333		33	2	0.06	3592	33303	2578	8735	121	A,B,C
		GN	(03)	ACB33333	GK	13	1	0.07	3004	20675	1697	4462	122	A,B,C
		GN	(03)	ACA33333		39	2	0.05	3296	68149	1714	6426	123	A,B,C
		GN	(02)	AC222222	GK	17	3	0.17	17469	215195	6352	66259	124	A,B,C
		GN	(02)	AB222222	GA	10	3	0.30	8720	52084	2407	7150	125	A,B,C
		GN	(05)	AAAA5555		44	4	0.09	7631	150639	4827	120433	126	A,B,C
		GN	(04)	AAAA4444		20	1	0.05	4478	51772	2287	7966	127	A,B,C
		GN	(04)	AAAB4444	GK	13	0	0.00	16670	201947	16670	201947	128	A,B,C
		GN	(04)	AAAA4444		32	0	0.00	1309	3593	1309	3593	129	A,B,C
		GN	(03)	AAA33333		51	0	0.00	1813	5554	1813	5554	130	A,B,C
		GN	(02)	AA222222		59	0	0.00	10946	538218	10946	538218	131	A,B,C
		GN	(03)	ADA33333	GA	8	0	0.00	1490	2870	1490	2870	132	A,B,C
		GN	(04)	ADAA4444	GA	8	0	0.00	1328	1646	1328	1646	133	A,B,C
		GN	(05)	ADAAA555	GA	8	0	0.00	1358	1755	1358	1755	134	A,B,C
		GN	(06)	ADAAAA66	GA	8	0	0.00	1510	3036	1510	3036	135	A,B,C

The column entries, from left to right, are:

PSB NAME - The 8-character program name.

PCB NAME - The 8-character PCB name, or "I/O PCB" if the call was to the message queue.

CALL FUNC - The 4-character DLI call function.

LEV NO. - The data base level number reached in this call, or blank

SEGMENT - The 8-character segment name accessed by this call.

STAT CODE - The status code returned by this call.

DL/I CALLS - The number of calls noted having the unique combination of the above six attributes.

IWAITS - The number of I/O WAITs observed for the calls.

IWAITS/CALL - Quotient of the above two items.

ELAPSED TIME or NOT IWAIT TIME - See the section "Definition of Terms Used in Reports" for an explanation of these terms.

INTENT FAILURE SUMMARY REPORT

This report summarizes the number of times a given combination of PSB and DMB names caused a failure to schedule because of intent conflicts. An example is included in Figure 9-20.

POOL SPACE FAILURE SUMMARY REPORT

This report summarizes the number of times in each region a given amount of storage was unavailable in the given pool ID. The value given under the heading "OCCURRENCES" is the number of times the specified amount of storage was unavailable in the specified pool. An example is included in Figure 9-20.

DEADLOCK EVENT SUMMARY REPORT

This report summarizes the number of times a given pair of programs reached a deadlock over a segment in the given DMB. An example is included in Figure 9-20.

Data given is:

REQ-ING PSB - The program requesting access.

LOSING PSB - The program caused to back-out.

DMB NAME - The DMB involved in this deadlock.

OCCURRENCE - The number of times a deadlock was detected having the unique combination of the above three attributes.

IMS/VS MONITOR TRACE DATA REPORT

This report gives the total elapsed time during which the monitor was active, and the number of monitor records written. Figure 9-20 includes an example of this report.

INTENT FAILURE SUMMARY

PSBNAME	DMBNAME	OCCURRENCES
SSTPSBNM	SSTDMBNM	1
TOTAL		1

POOL SPACE FAILURE SUMMARY

POOL ID	BYTES REQ.	OCCURRENCES
PDMP	8888	1
PDMX	7777	1
TOTAL		2

DEADLOCK EVENT SUMMARY

REQ-ING PSB	LOSING PSB	DMB NAME	OCCURRENCES
PSBNAME1	TPPSBRE3	DBASEBAL	1
TOTAL			1

MONITOR TRACE DATA

1504872 MILLISECONDS, TRACE INTERVAL
120839 MONITOR RECORDS WERE PRODUCED

Figure 9-20. IMS/VS Monitor Reports

DISTRIBUTION APPENDIX REPORT

This report is a numbered display of distribution information. Previously described reports summarize the totals obtained for events in various categories. The distributions show how the individual measurements are distributed in magnitude. The distributions are, in effect, histograms. Each distribution consists of two lines of print. The first line indicates the index number of the distribution and shows the values assigned to each "bucket" in the distribution. There are ten "buckets" in each distribution; the lower limit of the first bucket is zero, and the upper limit of the tenth bucket is infinity. The values for each bucket have been assigned a default value. The values may be redefined. (See the section entitled "Specifying Distribution Redefinition" for additional information.)

The distribution number associated with each distribution is a key that connects the distribution with a line of a previously described summary report. The columns of each summary report headed "Means" (the per unit quantity columns) have counterparts in the Distribution Appendix report. By noting the index number associated with summary data lines, the related distribution detail information is easily located in the Distribution Appendix report. The units and the meaning of the numbers displayed in this report are defined by the related summary report entries.

As an example of its use, suppose it is desired to determine how the values of IWAIT time intervals were distributed in magnitude. By referring to the example of the Region IWAIT report (Figure 9-11), you can see that it shows the overall total, mean and maximum values of these intervals. Note also the numbers that appear under the heading "Distribution Number." Each number points to an entry in the Distribution Appendix report which contains the desired histogram data.

Figure 9-21 is an example of the Distribution Appendix report.

		0	1000	2000	4000	8000	16000	32000	64000	128000	256000	INF
#	1A	0	0	0	0	0	0	0	0	0	0	2
	1B	0	0	0	0	0	0	0	0	0	0	2
#	2	0	200000	400000	600000	800000	1000000	1200000	1400000	1600000	1800000	1
#	3A	61	1245	405	494	163	85	95	61	36	41	
	3B	61	1245	418	528	221	88	49	20	21	35	
	3C	2521	0	84	1	61	2	5	3	8	3	
#	4	0	2000	8000	24000	50000	100000	150000	200000	250000	300000	INF
		0	1	11	27	4	0	0	0	0	0	0
#	5	0	2000	8000	24000	50000	100000	150000	200000	250000	300000	INF
		2	7	25	42	3	2	2	0	0	1	
#	6	0	2000	8000	24000	50000	100000	150000	200000	250000	300000	INF
		1	5	41	29	2	2	0	0	0	0	
#	7	0	2000	8000	24000	50000	100000	150000	200000	250000	300000	INF
		0	0	3	36	5	2	0	0	0	1	
#	8	0	2000	8000	24000	50000	100000	150000	200000	250000	300000	INF
		0	0	5	6	1	0	0	0	0	0	
#	9	0	2000	8000	24000	50000	100000	150000	200000	250000	300000	INF
		0	0	10	21	11	1	0	0	0	3	
#	10	0	50000	100000	150000	200000	250000	300000	350000	400000	450000	INF
		0	0	0	0	0	0	0	0	0	1	
#	11	0	1000	2000	4000	8000	16000	32000	64000	128000	256000	INF
		0	0	0	0	0	0	0	0	1	0	
#	12	0	1000	2000	4000	8000	16000	32000	64000	128000	256000	INF
		0	0	0	0	0	0	1	0	0	0	
#	13	0	1000	2000	4000	8000	16000	32000	64000	128000	256000	INF
		0	0	0	0	0	0	0	0	1	0	

Figure 9-21. Distribution Appendix Report

DISTRIBUTION REDEFINITION

Each of the basic units of information defined in the section "Definition of Terms Used in Reports" contains distribution information. In order to allow users to define their own distribution intervals, and thus better suit the analysis to their particular efforts, the CPU time unit, the elapsed time unit, the I/O IWAITs per DL/I call unit, and the queue statistics unit distributions associated with each report are considered distinct, and have a unique distribution identifier (ID). These identifiers are specified in Figures 9-22 and 9-23.

The distributions are set up to give ten intervals. Zero is assumed to be the lower boundary, and infinity is assumed to be the upper boundary of the distribution interval range. Between zero and infinity, nine finite, non-negative numbers are needed to complete the distribution definition. This is shown in the following example.

```
0 1 2 3 10 15 20 40 80 160 INF
*                          *
*                          *
* assumed                  * assumed
  lower boundary             upper boundary
```

Each of the CPU time and elapsed time distributions for a report has a default, or predefined definition. The default values are presented in the section "Default Values of Distributions." Thus, the user can run the Monitor Report Print program and specify either no distribution redefinitions in his Analysis Control data set, or redefinitions for selected reports. See "Specifying Distribution Redefinition" later in this chapter.

Name of Report	ID	Description
Region Summary		
Scheduling and Termination	D1	Elapsed time
	D2	Not IWAIT time
Schedule end to 1st DL/I call	D3	
Elapsed execution time	D4	
DL/I calls	D5	Elapsed time
	D6	Not IWAIT time
IWAITs per DL/I call	D7	
Idle for intent	D8	
Checkpoint	D20	Elapsed time
	D21	Not IWAIT time
Programs by Region		
Elapsed execution time	D30	
Schedule end to 1st DL/I call	D31	
Program Summary		
CPU time per schedule	D15	
Transactions dequeued per schedule	D14	
Elapsed time per schedule	D9	
Schedule end to 1st DL/I call	D10	
Communication Summary		
Line elapsed time	D18	
Line not IWAIT time	D19	
Line Functions		
Received block length	D36	
Transmitted block length	D37	
Inactive intervals	D38	
Transaction Queueing		
Transactions on queue at schedule	D17	
Transactions dequeued per schedule	D16	
Prefetch format blocks	D28	Elapsed time
	D29	Not IWAIT time
DL/I Call Summary		
PSB IWAITs per DL/I call	D13	
PSB elapsed time per call	D11	
PSB not IWAIT time per call	D12	

Figure 9-22. Distribution Identifiers

Function	ID	Module Key
Storage	D22	SMN
ISAM/OSAM I/O	D23	DBH
VSAM I/O	D24	VBH
Scheduler internal	D25	MSC
Queue manager I/O	D26	QMG
Block loader I/O	D27	BLR
MFS block I/O	D32	PRF
MFS directory I/O	D33	PRF
HSAM I/O	D34	DLE
Format Buffer Pool Space	D35	PMM

Figure 9-23. IWAIT Time Distribution Identifiers

Default Values of Distribution Definitions

- D1, D2 D5, D6, D9, D10, D11, D12, D15, D18, D19, D20, D21, D22, D25, D27, D28, D29, D30, and D31:
  (0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF)

- D3:
  (0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, INF)

- D4:
  (0, 200000, 400000, 600000, 800000, 1000000, 1200000, 1400000, 1600000, 1800000, INF)

- D7, D13:
  (0, 0, 1, 2, 3, 4, 5, 6, 7, 8, INF)

- D8:
  (0, 100000, 200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000, INF)

- D14, D16, D17:
  (0, 1, 2, 3, 4, 5, 10, 15, 30, 90, INF)

- D23, D24, D26, D32:
  (0, 2000, 8000, 24000, 50000, 100000, 150000, 200000, 250000, 300000, INF)

- D33, D34, D35:
  (0, 2000, 4000, 8000, 16000, 32000, 64000, 96000, 128000, 160000, INF)

- D36, D37:
  (0, 10, 20, 40, 60, 100, 200, 400, 800, 1000, INF)

- D38:
  (0, 1000, 10000, 100000, 200000, 500000, 800000, 1000000, 1500000, 2000000, INF)

INPUT TO DFSUTR20

The Monitor Report Print program runs as a batch program, with a tape sequential data set as input. This tape was created by the IMS/VS Monitor module (DFSMNTR0) during IMS/VS executions.

JCL Requirements

// JOB
Initiates the job.

// EXEC
Specifies the program name (PGM=DFSUTR20, REGION=512K).

//SYSPRINT DD
Specifies the output data set that is to contain the reports and control messages. It is usually coded as SYSOUT=A.

//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 the section "Specifying Distribution Redefinitions" for information on how to respecify the distributions.

- If you want only the DL/I 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.

  ONLY DLI

- To generate the DL/I 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 DL/I Call Summary report are printed. The statement starts in card image column 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.

  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); that is, D2 or D45 (see Figure 9-22 and 9-23 for identifiers).

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 usual 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.

The redefined distributions input statements must identify the distributions as shown in Figure 9-22 and 9-23. The redefinitions are temporary for the JOB step.

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
```

JCL Example (DFSUTR20)

The following JCL produces a complete set of reports, including the DL/I 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
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1 DD DSN=IMSMON,UNIT=2400,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 Report Utility Program lists all transactions that had to wait while trying to enqueue on a data base record. The waiting transaction and the holding transaction are printed for each transaction that was held in a wait status. The elapsed time of the wait is shown if the /TRACE ALL option is used during execution.

The report is generated from X'67FE' records from the IMS/VS system log tape. The function of DFSPIRP0 is to select from the system log tape all X'67FE' transaction records within a start/stop time specified by the user. These records are passed to OS/VS Sort via an E15 user exit. The records are sorted in ascending order by the sort key comprised of DMB number, DCB number, RBA number and sequence number.

Once the log tape has been read, the sorted records are returned to DFSPIRP0 via an E35 user exit from OS/VS Sort. Each record is checked to determine if any record had to wait for an enqueue. If a record had to wait, a line of the report is generated. Totals are maintained by DMB name, DCB number and ID (RBA) number.

JCL REQUIREMENTS

//JOBLIB DD	Describes the program library containing the utility program. Its format is: DSN=IMSVS10.RESLIB,DISP=SHR
// EXEC	Invokes the utility program. Its format is: //PITRACE EXEC PGM=DFSPIRP0,REGION=150K

//LOGTAPE DD	Describes the log tape input data set. Multiple volumes are concatenated if the log tape extends over more than one volume. Its format is:
	
	//LOGTAPE DD DSN=IMSLOG,DISP=OLD,VOL=SER=xxxxxx,
	// UNIT=2400
	
	where xxxxxx is the serial of the log tape being processed. If an unlabeled tape is used, the DCB parameters used to generate the tape must be specified.

//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 since 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.

UTILITY CONTROL STATEMENT

An optional utility control statement can be used to indicate the starting and stopping time desired for the report. The statement can be coded in free format but should not extend past position 71. The format of the statement is:

```
PRINT START=hhmmss,STOP=hhmmss
```

where: hh = hours
mm = minutes
ss = seconds

- If only PRINT is specified, the entire log tape is scanned. A start-time-only causes the log tape to be scanned from the specified start time to the end of the tape. Conversely, a stop-time-only causes the tape to be scanned from the beginning to the indicated stop time. The default times are START=000000 and STOP=235959.

JCL EXAMPLE

```
//PITSTATS  JOB  MSGLEVEL=1
//JOB       DD   DSN=IMSVS10.RESLIB,DISP=SHR
//          EXEC PGM=DFSPIRP0,REGION=150K
//LOGTAPE   DD   DSN=IMSLOG,UNIT=2400,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)
//SORTOUT   DD   SYSOUT=A
//SORTWK06 DD    UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SYSPRINT DD    SYSOUT=A
//SORTLIB   DD   DSN=SYS1.SORTLIB,DISP=SHR
//SYSIN     DD   *
          PRINT START=090000,STOP=235959
/*
```

CHAPTER 10. SYSTEM SERVICE UTILITIES

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.

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 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 VBA. 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.
//SPOOLnn DD	These (where nn is any valid alphameric identifier) describe the spool data set to be printed. This is normally DSNAME=IMSVS.SYSnn, where 'nn' is assigned by system definition.

A sample print procedure is contained in the IMS/VS Installation Guide.

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.

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.

INDEX

IMS/VS Version 1 Utilities Reference Manual Printed in U.S.A. SH20-9029-3

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

IMS/VS Version 1
Utilities Reference Manual
SH20-9029-3

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Fold and Staple

First Class Permit
Number 6090
San Jose, California

Business Reply Mail

No postage necessary if mailed in the U.S.A.

Postage will be paid by:

IBM Corporation
P. O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Staple

IMS/VS Version 1 Utilities Reference Manual Printed in U.S.A. SH20-9029-3

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)