IMS/ESA

IBM

Utilities Reference: Database Manager

Version 6

SC26-8769-05

IMS/ESA

IBM

Utilities Reference: Database Manager

Version 6

SC26-8769-05

Note

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

Fifth Edition (July 2000) (Softcopy Only)

This edition replaces and makes obsolete the previous edition, SC26-8769-04. This edition is available in softcopy only. The technical changes for this edition are summarized under “Summary of Changes” on page xxiii. THe technical changes are indicated by a vertical bar to the left of a change.

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

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

Contents

Notices

Programming Interface Information

This book is intended to help database administrators and system programmers run the IMS utility programs.

This book also documents General-use Programming Interface and Associated Guidance Information provided by IMS.

General-use programming interfaces allow the customer to write programs that obtain the services of IMS.

General-use programming interface

General-use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking: General-use Programming Interface and Associated Guidance Information.

End of General-use programming interface

Trademarks

Product Names

In this book, the licensed program DB2 for MVS/ESA is referred to as “DB2.”

Preface

This book is a reference manual for database administrators and system programmers who use the IMS Version 6 utilities for the IMS Database Manager (IMS DB) to administer the IMS system.

For DBCTL users, all utilities, commands, and parameters that are valid for IMS DB are valid for DBCTL, unless otherwise noted.

This book is one of three utilities reference manuals in the IMS library. The scope of the three books is as follows:

- *IMS/ESA Utilities Reference: System* describes utilities that apply to IMS at a system level or that affect both database and data communications operations.
- *IMS/ESA Utilities Reference: Database Manager* (this book) describes utilities that affect database operations.
- *IMS/ESA Utilities Reference: Transaction Manager* describes utilities for data communications.

Organization of This Book

This book has six parts and an appendix.

- "Part 1. Reorganization Utilities" on page 1 describes the utilities you use to scan and reorganize databases.
- "Part 2. Backup Utilities" on page 143 describes the utilities you use to make backup copies of a database.
- "Part 3. Recovery Utilities" on page 169 explains the utilities you use to recover databases.
- "Part 4. Conversion Utilities" on page 223 explains the MSDB-to-DEDB conversion utility.
- "Part 5. Report and Test Utilities" on page 235 contains information on generating, interpreting, and using the DB reports, as well as a utility for testing sequential buffering results.
- "Part 6. Utility Control Facility" on page 275 describes how to use the Utility Control Facility as a controller for the execution of the other utilities.
- "Appendix A. Summary of DEDB Utility Commands" on page 329 summarizes the utility commands you use with the DEDB utilities.
- "Appendix B. Database Utilities in an RSR Environment" on page 337 explains considerations about database utilities in a Remote Site Recovery (RSR) environment.

For a complete list of all books this manual cites, see the "Bibliography" on page 341.

Prerequisite Knowledge

IBM offers a wide variety of classroom and self-study courses to help you learn IMS. For a complete list of courses, visit the IMS Web site at: http://www.software.ibm.com/data/ims

The reader should be familiar with MVS, and with IMS concepts, facilities, and access methods. The prerequisite publications are:

- *IMS/ESA Release Planning Guide*
- *IMS/ESA Administration Guide: System*
- *IMS/ESA Administration Guide: Database Manager*
- *IMS/ESA Administration Guide: Transaction Manager*

Organization of Utility Descriptions

So that you can find information easily, most utilities are consistently described in this way:

- Overview of the utility's functions
- Restrictions that apply to the utility, such as processing that cannot be done concurrently with the utility
- Input and output
- Job control statements that are needed to run the job
- Utility control statements used to specify various processing options

When applicable, the descriptions also include:

- Output messages and statistics reports the utility produces
- Error processing, with return codes and their meaning
- Examples of how to use the utility

Change Indicators

Technical changes are indicated in this publication by a vertical bar (|) to the left of the changed text.

Syntax Diagrams

The following rules apply to the syntax diagrams used in this book:

Arrow symbols

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

►►───	Indicates the beginning of a statement.
───►	Indicates that the statement syntax is continued on the next line.
►───	Indicates that a statement is continued from the previous line.
───►◄	Indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the ►─── symbol and end with the ───► symbol.

Conventions

- Keywords, their allowable synonyms, and reserved parameters, appear in uppercase for MVS and OS/2 operating systems, and lowercase for UNIX operating systems. These items must be entered exactly as shown.
- Variables appear in lowercase italics (for example, *column-name*). They represent user-defined parameters or suboptions.
- When entering commands, separate parameters and keywords by at least one blank if there is no intervening punctuation.
- Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs) and numbers exactly as given.

- Footnotes are shown by a number in parentheses, for example, (1).
- A ƀ symbol indicates one blank position.

Required items

Required items appear on the horizontal line (the main path).

```
►►──REQUIRED_ITEM────────────────────────────────────────────────►◄
```

Optional Items

Optional items appear below the main path.

```
►►──REQUIRED_ITEM──┬───────────────┬─────────────────────────────►◄
                   └─optional_item─┘
```

If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

```
                   ┌─optional_item─┐
►►──REQUIRED_ITEM──┴───────────────┴─────────────────────────────►◄
```

Multiple required or optional items

If you can choose from two or more items, they appear vertically in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.

```
►►──REQUIRED_ITEM──┬─required_choice1─┬──────────────────────────►◄
                   └─required_choice2─┘
```

If choosing one of the items is optional, the entire stack appears below the main path.

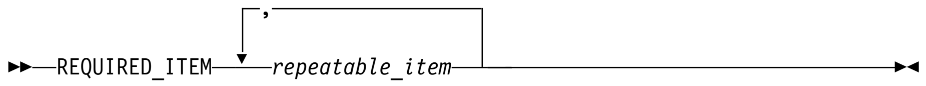
```
►►──REQUIRED_ITEM──┬──────────────────┬──────────────────────────►◄
                   ├─optional_choice1─┤
                   └─optional_choice2─┘
```

Repeatable items

An arrow returning to the left above the main line indicates that an item can be repeated.

```
                   ┌─────────────────┐
                   ▼                 │
►►──REQUIRED_ITEM────repeatable_item─┴───────────────────────────►◄
```

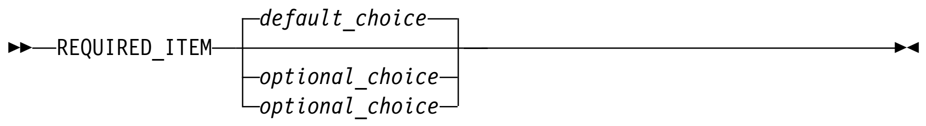
If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can specify more than one of the choices in the stack.

Default keywords

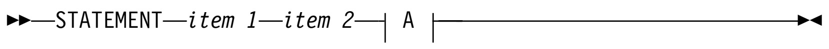
IBM-supplied default keywords appear above the main path, and the remaining choices are shown below the main path. In the parameter list following the syntax diagram, the default choices are underlined.



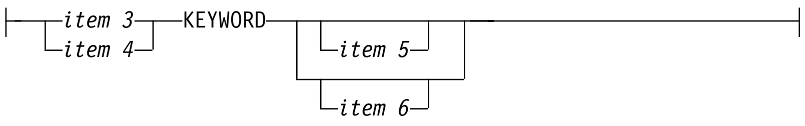
IMS-specific syntax information

Fragments

Sometimes a diagram must be split into fragments. The fragments are represented by a letter or fragment name, set off like this: | A |. The fragment follows the end of the main diagram. The following example shows the use of a fragment.

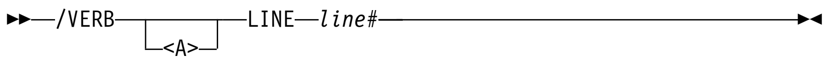


A:

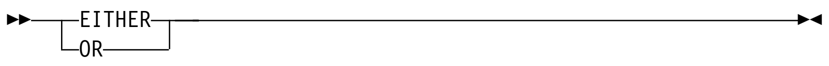


Substitution-block

Sometimes a set of several parameters is represented by a substitution-block such as **<A>**. For example, in the imaginary /VERB command you could enter `/VERB LINE 1`, `/VERB EITHER LINE 1`, or `/VERB OR LINE 1`.



where <A> is:



Parameter endings

Parameters with number values end with the symbol '#', parameters that are names end with 'name', and parameters that can be generic end with '*'.

```
►►──/MSVERIFY──┬─MSNAME──msname──┬──────────────────────────────────►◄
               └─SYSID──sysid#───┘
```

The MSNAME keyword in the example supports a name value and the SYSID keyword supports a number value.

Supported Environments for Various Utilities

Table 1, Table 2 on page xx, and Table 3 on page xxi provide a comprehensive listing of all the IMS/ESA utilities contained in all three of the Utilities Reference books. The figures also indicate whether the utility supports either DBCTL or DCCTL or both.

Table 1 lists the database utilities described in the *Utilities Reference: Database Manager*.

Table 1. Listing of Database Utilities and Supported Environments

Utility or Report Name	Module Name	Supports DBCTL	Supports DCCTL
Reorganization Utilities:			
HISAM Reorganization Unload utility	DFSURUL0	X	
HISAM Reorganization Reload utility	DFSURRL0	X	
HD Reorganization Unload utility	DFSURGU0	X	
HD Reorganization Reload utility	DFSURGL0	X	
Database Surveyor utility	DFSPRSUR	X	
Partial Database Reorganization utility	DFSPRCT1 and DFSPRCT2	X	
Database Prereorganization utility	DFSURPR0	X	
Database Scan utility	DFSURGS0	X	
Database Prefix Resolution utility	DFSURG10	X	
Database Prefix Update utility	DFSURGP0	X	
MSDB Maintenance utility	DBFDBMA0		
DEDB Initialization utility	DBFUMIN0	X	
DEDB Sequential Dependent Scan utility	DBFUMSC0	X	
DEDB Sequential Dependent Delete utility	DBFUMDL0	X	
High Speed DEDB Direct Reorganization utility	DBFUHDR0	X	
Backup Utilities:			

Table 1. Listing of Database Utilities and Supported Environments (continued)

Utility or Report Name	Module Name	Supports DBCTL	Supports DCCTL
Database Image Copy utility	DFSUDMP0	X	
Online Database Image Copy utility	DFSUICP0	X	
Recovery Utilities:			
Database Change Accumulation utility	DFSUCUM0	X	
Database Recover utility	DFSURDB0	X	
Batch Backout utility	DFSBBO00	X	
MSDB Dump Recovery utility	DBFDBDR0		
DEDB Area Data Set Create utility	DBFUMRI0	X	
DEDB Area Data Set Compare utility	DBFUMMH0		X
Conversion Utilities:			
MSDB-to-DEDB Conversion utility	DBFUCDB0		
Utility Control:			
Utility Control Facility	DFSUCF00	X	
Report and Test Utilities:			
Program-Isolation-Trace Report utility	DFSPIRP0	X	
Database-Monitor Report Print utility	DFSUTR30		
Sequential Buffer Test utility	DFSSBHD0		
Report Interpretation:			
Interpreting Database Monitor Reports			

Table 2 lists the data communications utilities described in the *Utilities Reference: Transaction Manager*.

Table 2. Listing of Data Communications Utilities and Supported Environments

Utility or Report Name	Module Name	Supports DBCTL	Supports DCCTL
Generation Utilities:			
MFS Language utility	DFSUPAA0		X
Reorganization Utilities:			
MFS Device Characteristics Table utility	DFSUTB00		X
Service Utilities:			
Spool SYSOUT Print utility	DFSUPRT0		X
Multiple Systems Verification utility	DFSUMSV0		X
MFS Service utility	DFSUTSA0		X

Table 2. Listing of Data Communications Utilities and Supported Environments (continued)

Utility or Report Name	Module Name	Supports DBCTL	Supports DCCTL
Time-Controlled Operations Verification utility	DFSTVER0		X

Table 3 lists the system utilities described in the *Utilities Reference: System*.

Table 3. Listing of System Utilities and Supported Environments

Utility or Report Name	Module Name	Supports DBCTL	Supports DCCTL
Generation Utilities:			
Database Description (DBD) Generation utility	DBDGEN	X	X
Program Specification Block (PSB) Generation utility	PSBGEN	X	X
Application Control Block (ACB) Maintenance utility	ACBGEN	X	X
Service Utilities:			
Dynamic Allocation Macro utility	DFSMDA	X	X
Online Change utility	DFSUOCU0	X	X
Security Maintenance utility	DFSISMP0	X	X
Dynamic SVC utility	DFSUSVC0	X	X
Log Utilities:			
Log Recovery utility	DFSULTR0	X	X
Log Archive utility	DFSUARC0	X	X
Log Merge utility	DFSLTMG0		X
Analysis and Report Utilities:			
File Select and Formatting Print utility	DFSERA10	X	X
Offline Dump Formatter utility	DFSOFMD0	X	X
Statistical Analysis utility	DFSISTS0		X
Log Transaction Analysis utility	DFSILTA0		X
Fast Path Log Analysis utility	DBFULTA0	X	X
IMS-Monitor Report Print utility	DFSUR20	X	X
Report Interpretation:			
Interpreting Statistical Analysis and Log Transaction Reports			X
Interpreting //DFSSTAT Reports		X	X
Interpreting IMS Monitor Reports			
Interpreting IMS Monitor Reports for DBCTL		X	

Table 3. Listing of System Utilities and Supported Environments (continued)

Utility or Report Name	Module Name	Supports DBCTL	Supports DCCTL
Interpreting IMS Monitor Reports for DCCTL			X

Summary of Changes

Changes to the Current Edition of This Book for V6

This edition, which is in softcopy only, includes technical and editorial changes.

Changes to This Book for V6

This book contains new and changed information about the following enhancements:

- Shared SDEPs
- New Time Stamp Format
- Database Image Copy 2 utility

Library Changes for Version 6

The IMS/ESA Version 6 library differs from the IMS/ESA Version 5 library in these major respects:

- *IMS/ESA Common Queue Server Guide and Reference*

  This new book describes the IMS Common Queue Server (CQS).
- *IMS/ESA DBRC Guide and Reference*

  This new book describes all the functions of IMS Database Recovery Control (DBRC).
- The IMS Application Programming summary books (*IMS/ESA Application Programming: Database Manager Summary*, *IMS/ESA Application Programming: Transaction Manager Summary*, and *IMS/ESA Application Programming: EXEC DLI Commands for CICS and IMS Summary*) are no longer included with the IMS library.
- The Softcopy Master Index is not included.
- All information about IRLM 1.5 and data sharing using IRLM 1.5 has been removed from the IMS V6 books. If you use IRLM 1.5, and want to migrate to using IRLM 2.1 and Sysplex data sharing, see *IMS/ESA Release Planning Guide*.
- The chapter that was titled ″Database Control (DBCTL) Interface″ in the *IMS/ESA Customization Guide* has been revised for Open Database Access (ODBA) and moved to ″Appendix A, Using the Database Resource Adapter (DRA)″ in the *IMS/ESA Application Programming: Database Manager*.

Part 1. Reorganization Utilities

Chapter 1. HISAM Reorganization Unload Utility (DFSURUL0)

Use the HISAM Reorganization Unload utility to:

- Unload a HISAM database.
- Create reorganized output, which you can use as input to either the Database Recovery utility or the HISAM Reorganization Reload utility.
- Format the index work data sets (that the the Prefix Resolution utility creates) so they can be used by the HISAM Reload utility. The HISAM Reload utility uses the data sets to create a secondary index or to merge records from the index work data setswith a shared secondary index, if one exists.

Figure 1 is a flow diagram of this utility.

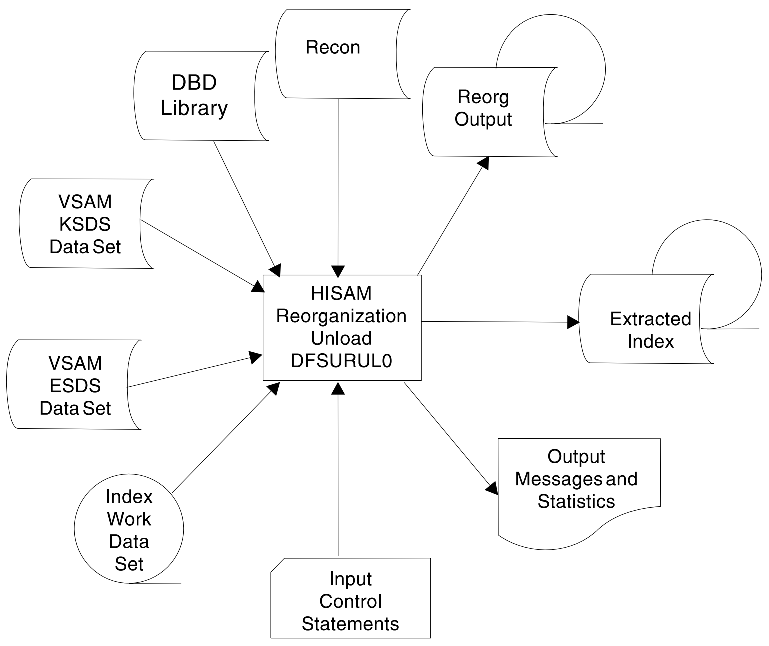


Figure 1. HISAM Reorganization Unload Utility

The output is blocked to the block size of the output device for devices other than 3380s. If the device is a 3380, a block size of 23 KB is used unless the logical record length is larger than 23 KB. If the logical record length is larger than 23 KB, the block size is 32 KB rounded down to an even multiple of the logical record length. Because the blocking factor is determined at execution time, you must use IBM standard labels on all output volumes.

The Utility Control Facility can also perform the functions of this utility.

Related Reading: For more information on the Utility Control Facility and a description of its operation, see "Chapter 30. Utility Control Facility (DFSUCF00)" on page 277.

In this Chapter:

- "Restrictions" on page 6
- "JCL Requirements" on page 6
- "Utility Control Statement" on page 9

- “Output Messages and Statistics” on page 11
- “Return Codes” on page 15
- “Examples” on page 15

Restrictions

The following restrictions apply to use of the HISAM Reorganization Unload utility:

- The HISAM Reorganization Unload utility cannot make structural changes or changes to database organizations. Use the HD Reorganization Unload and Reload utilities for these purposes.
- This utility cannot unload a HISAM database if the database contains logical child segments that have direct address logical parent pointers.
- This utility cannot unload a SHISAM database.
- To use this utility for recovery purposes, use the HISAM Reorganization Reload utility to reload the database prior to applying changes to the database. If you do not immediately reload the database prior to applying changes, the segments are reloaded into a different location. The logs that are created between unload and reload refer to the old location, making recoveryimpossible.
- If a write error has occurred for the database and the database has not been recovered, recovery must be performed before this utility is executed. If the database is registered with DBRC, and if this utility uses DBRC, the fact that a write error has occurred and recovery has not been performed is known to DBRC, and DBRC rejects the authorization request of this utility.
- If a new DBD is generated with new sizes, first unload the database using the new DBD. Then run DFSMS/MVS Access Method Services 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 databases can then be reloaded using the HISAM Reload utility. If you want to specify control interval or record sizes different from the DBD, then code CHNG=CARD in columns 50-80 of the Utility Control Statement and use a CHANGE statement control card to specify the new sizes. Otherwise, use CHNG=DBD in this field.

  Related Reading: For more information, refer to *DFSMS/MVS V1R4 Access Method Services*.
- 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 times.
- Do not use this utility to unload or reload a shared secondary index that has alias names that arenot registered to DBRC. Message DFS194W is issued if this is tried.
- Sequential buffering does not support this utility.

JCL Requirements

The HISAM Reorganization Unload utility is executed as a standard MVS job. The following JCL statements are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURUL0'
```

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

Related Reading: For additional information on executing a batch processing region in DBBATCH or DBLIBATCH, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized (because libraries that are not APF authorized are concatenated to IMS.RESLIB), you must include a DFSRESLB DD statement.

DFSRESLB DD
Points to an APF-authorized library that contains the IMS SVC modules.

IMS DD
Defines the library that contains the DBD that describes the database to be reorganized (DSN=IMS.DBDLIB,DISP=SHR). This data set must reside on a direct-access device.

SYSPRINT DD
Defines the message and statistics output data set. The data set can reside on a tape, direct-access device, or printer, or it canbe routed to the output stream. DCB parameters specified for this data set are RECFM=FBA and LRECL=133. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 133.

SYSIN DD
Defines the input control statement data set. This data set can reside on a tape, or adirect-access device, or it can be routed through the input stream.

vsamksds DD
Defines the VSAM KSDS that is to be reorganized. The ddname must matchthe name in the DBD that describes this data set. It must also appear on the utility control statement in the SYSIN data set of this job step. One DD statement of this type must be present for each VSAM KSDS that is to be reorganized.

This DD statement represents the primary data set of the HISAM database. In case of secondary index creation, you mustprovide one or more DD statements. Each DD statement represents a secondary index data set that is to be reorganized.

vsamesds DD
Defines the VSAM ESDS to be reorganized. The ddname must matchthe name in the DBD that describes this data set. One DD statement of this type must be present for each VSAM ESDS that is to be reorganized.

dataout1 DD
Defines the first copy of the reorganized output data set. One DD statement of this type is required for each 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 a tape or a direct-access device.

This output data set can be used as an image copy data set and is recorded in the RECON data set.

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, andthe second volume continues to thenormal end of job, a total rerun is not required.

The same requirements described for the dataout1 DD statement apply to dataout2.

indexwrkds DD
Describes the output data set ddname (DFSURIDX) from the Prefix Resolution program that contains secondary index information. This statement is required if the utility control statement is type 'X'; otherwise, it is optional. The ddname must match the name starting in position 40 of the control statement.

DFSEXTDS DD
Writes an unloaded version of the records that have been split out from a shared secondary index as specified in control statements. DFSEXTDS DD is optional and is required only if 'E' is specified in position 3 of the utility control statement (see “Utility Control Statement” on page 9). The DCB attributes are determined dynamically, based on the output device type and the VSAM LRECLs that are used. You must use standard labels.

DFSVSAMP DD
Describes the data set that contains the buffer information the DL/I buffer handler requires. This DD statement is required.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The data set can reside on a tape, ordirect-access device, or it canbe routed through the input stream.

SYSUDUMP or SYSABEND DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first Database Recovery Control (DBRC) RECON data set. This RECON1 data set must be the same RECON1 data set that the control region is using.

RECON2 DD
Defines the second DBRC RECON data set. This RECON2 data set must be the same RECON2 data set that the control region is using.

RECON3 DD
Defines the optional RECON data set DBRC uses when an error is encountered in RECON1 or RECON2. This RECON3 data set must be the same RECON3 data set thatthe control region is using.

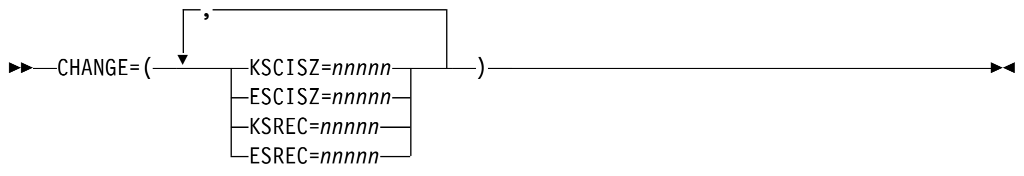
Restriction: If you are using dynamic allocation, do not use these RECON data set ddnames.

Utility Control Statement

Position	Description
1	Must contain either an 'R' or an 'X'. **R** Indicates a HISAM Reorganization Unload utility control statement **X** Indicates a secondary index reorganization control statement There is no default; if this position is left blank, an error message is generated.
2	Must contain a 1 or a 2, depending on the number of output copies required. There is no default; if this position is left blank, an error message is generated.
3	Must contain a 'M', 'E', 'R', or a blank for secondary index reorganization ('X' in position 1) control statements. If the value of this position is blank, the default is M. **M** Indicates that the index work data set created during either a database initial load or reorganization (using the Prefix Resolution utility) is to be merged into an existing secondary index or used to create a secondary index if the data set does not exist. **E** Indicates that the secondary index (defined by the constant in position 49 of this statement) is to be extracted from either a shared index database or from the index work data set (from the Prefix Resolution utility). **R** Indicates that those segments in the secondary index that match the constant in position 49 of this statement are to be replaced with matching segments found in the work data set. If the secondary index segments being replaced need to be saved, perform an extract function prior to the replace.
4-12	Must contain the name of the DBD that includes the ddnames of the VSAM data set group that is to be reorganized.
13-21	Must contain the KSDS ddname for the VSAM data set that is to be reorganized. The ddname is the primary data set name for the HISAM database and the secondary index ddname for the secondary index database. It must appear in the referenced DBD statement, and a corresponding DD statement must be provided.
22-30	Must contain the ddname of the primary output data set. A corresponding DD statement must be provided.
31-39	Must contain the ddname of the second copy of the reorganized output data set. If this field contains the ddname, a corresponding DD statement must be provided. This field must be blank if position 2 contains a 1.
40-48	Must contain the ddname of the secondary index work data set if this control statement is type 'X'.
49	Must contain the 1-byte constant specified in the DBD generation of the shared secondary index if 'E' or 'R' is specified in position 3 of this statement.

50-80 Can contain comments or specify control interval (CI) or record size changes. To specify the source of CI or record size changes, code CHNG=DBD or CHNG=CARD, where DBD tells the unload utility to use the DBD values for KSDS and ESDS CI and record sizes, and CARD tells the unload utility to use the values specified on the control statement that starts with CHANGE in column 1.

CHANGE Statement



This is an optional control statement. If you use this statement, you must specify at least one keyword.

CHANGE=
Identifies the CHANGE control statement.

KSCISZ
Specifies a new KSDS CI size in bytes. The size is specified in multiples of 512 bytes.

ESCISZ
Specifies a new ESDS CI size in bytes. The size is specified in multiples of 512 bytes.

KSREC
Specifies a new KSDS record size.

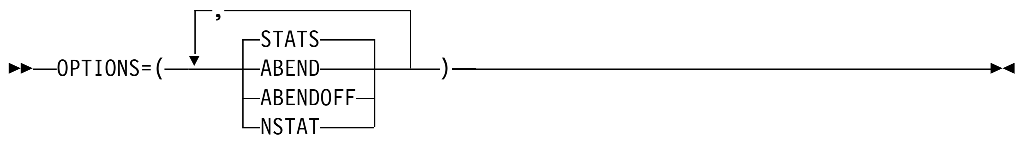
ESREC
Specifies a new ESDS record size.

nnnnn
Is a five-digit decimal value, including leading zeros if necessary. The maximum value is 32767.

Example: In this example, the CHANGE statement is used to specify a new KSDS CI size and a new ESDS CI size:

```
CHANGE=(KSCISZ=01024,ESCISZ=02048)
```

Options Statement



This is an optional control statement.

OPTIONS=
Identifies the OPTIONS control statement.

STATS
Provides statistics to the SYSPRINT data set and to the HISAM Reorganization Reload utility. STATS is the default.

ABEND
Turns on the ABEND function. If any condition arises causing termination of the run, terminates the run with abend U0359. A dump is printed if a SYSABEND or SYSUDUMP DD statement is supplied.

ABENDOFF
Turns off the ABEND function.

Exception:ABENDOFF is the initial default; however, if ABEND has been specified previously, it remains in effect until ABENDOFF is coded.

NSTAT
Specifies thatno statistics output be generated. If this parameter is used, the HISAM Reorganization Reload utility also ignores statistics output.

Output Messages and Statistics

The HISAM Reorganization Unload utility provides messages and statistics about the database contents for each data set group. In addition, the utility provides an audit trail. Figure 2 on page 12 is an example of the output messages and statistics this utility produces.

```
                              H I E R A R C H I C A L   I N D E X E D   S E Q U E N T I A L
                              D A T A   B A S E   R E O R G A N I Z A T I O N   U N L O A D


                            D A T A   S E T   G R O U P   S T A T I S T I C S
DATA BASE - DIVNTZ04
   PRIMARY  DD- DBHVSAM1
   OVERFLOW DD- DBHVSAM2

PRIMARY ROOTS                 OVERFLOW ROOTS           OVERFLOW DEPENDENTS
IN       OUT    DELETED       IN      DELETED          IN          OUT
3        3      0             0       0                2           2

TOTAL NUMBER OF RECORDS OUT =10
COPY 1 ON VOLUME(S)-  USER02
ROOT OVERFLOW CHAINS (#)       DEPENDENT OVERFLOW CHAINS (#) ROOTS W/OUT OVERFLOW CHAINS (BYTES)
NO.  LONGEST  SHORTEST  AVG    NO.  LONGEST SHORTEST  AVG    NO.  LONGEST SHORTEST AVG
0    0        0         0.00   2    1       1         1.00   0    0       0        0.00


                          S E G M E N T   L E V E L   S T A T I S T I C S

MAXIMUM  AVERAGE MAXIMUM  AVERAGE  SEGMENT SEGMENT TOTAL SEGMENTS  AVERAGE COUNT PER
TWINS    TWINS   CHILDREN CHILDREN NAME    LEVEL   BY SEGMENT TYPE DATA BASE RECORD
1        1.00    12       8.00      J1      1       3               1.00
2        1.00     2       0.66      J2      2       3               1.00
1        0.33     1       1.00      J3      3       1               0.33
1        1.00     0       0.00      J4      4       1               0.33
2        1.33     3       1.75      J5      2       4               1.33
1        1.00     2       0.75      J6      3       4               1.33
1        0.50     1       0.50      J7      4       2               0.66
1        0.50     0       0.00      J8      5       1               0.33
2        1.00     3       1.66      J9      2       3               1.00
1        1.00     0       0.00     J10      3       3               1.00
0        0.00     0       0.00     J11      4       0               0.00
2        0.66     0       0.00     J7P      3       2               0.66
0        0.00     0       0.00     J12      2       0               0.00
0        0.00     0       0.00     J13      3       0               0.00
0        0.00     0       0.00     J14      4       0               0.00
0        0.00     0       0.00     J13X     3       0               0.00
TOTAL SEGMENTS IN DATA SET GROUP=27   AVG DATA SET GROUP RECORD LENGTH=203 BYTES
```

Figure 2. Example of Output Statistics from the HISAM Reorganization Unload Utility

If you select any options for this execution, various messages are generated and appear immediately following the page heading.

Related Reading: For explanations of all numbered messages, refer to *IMS/ESA Messages and Codes*.

Following the message for each data set group, the heading `DATA SET GROUP STATISTICS` appears. The fields in this section are:

DATA BASE
Database name.

PRIMARY DD
VSAM KSDS ddname of the data set group.

OVERFLOW DD
VSAM ESDS ddname of the data set group.

PRIMARY ROOTS

Statistics related to root records in the VSAM KSDS:

IN	Number of old VSAM KSDS roots that were read
OUT	Number of new VSAM KSDS roots that were written
DELETED	Number of old VSAM KSDS roots that were deleted

OVERFLOW ROOTS

Statistics related to old roots in the VSAM ESDS:

IN	Number of old VSAM ESDS roots that were read
DELETED	Number of old VSAM ESDS roots that were deleted

OVERFLOW DEPENDENTS

Statistics related to dependent records in the VSAM ESDS:

IN	Number of old dependent records in the VSAM ESDS that were read
OUT	Number of new dependent records in the VSAM ESDS that were written

TOTAL NUMBER OF RECORDS OUT

Number of records, both VSAM KSDS roots and VSAM ESDS dependents, that were written. This total includes at least one header record. It might also include two or more statistics records—one or more at the beginning of the data set that was used as a table initialization record for the Reload program, and one or more at the end of the data set containing totals that were unloaded by segment type so that the HISAM Reload utility can compare the numbers reloaded with those unloaded.

A statistics table record consists of 20 bytes for each segment type in the DBD, plus a 28-byte header for each LRECL that is needed to contain the entire statistics record. The approximate number of LRECLs that are 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 written.

COPY 1 ON VOLUME(S) - *volser1*

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`*, is included.

ROOT OVERFLOW CHAINS (#)

Statistics dealing with root records in a VSAM KSDS that have overflow chains to root records in a VSAM ESDS:

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

DEPENDENT OVERFLOW CHAINS (#)
Statistics dealing with VSAM KSDS roots which had dependents in VSAM ESDS:

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

ROOTS WITHOUT OVERFLOW CHAINS (BYTES)
Statistics dealing with VSAM KSDS roots which had no dependents:

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

The heading “SEGMENT LEVEL STATISTICS” appears next. The fields in this section are:

MAXIMUM TWINS
Maximum number of segments of this type encountered under an immediate parent segment. At the root level, this value is always 1.

AVERAGE TWINS
Average number of segments of this type encountered under an immediate parent segment. This value is carried out to two decimal places.

MAXIMUM CHILDREN
Maximum number of child segments (at all subordinate levels) under a given parent.

AVERAGE CHILDREN
Average number of child segments (at all subordinate levels) under a given parent. This value is carried out to two decimal places. The lowest level segment in any hierarchic path has a value of zero in this field type.

SEGMENT NAME
Segment name to which this line of statistics applies.

SEGMENT LEVEL
The hierarchic level of this segment in the database.

TOTAL SEGMENTS BY SEGMENT TYPE
Count of the occurrences of this segment type in the entire database. The count field in the level 1 segment type reflects the total number of database records (root segments) in the database.

When a database is part of a shared secondary index database, the segment occurrence count always appears under the first segment name. All other segment name counts are zero.

AVERAGE COUNT PER DATABASE RECORD
A count of the average number of occurrences of this segment type within a given database record. The value is carried to two decimal places.

Following the individual segment type statistics for this data set group are the following two fields:

TOTAL SEGMENTS IN DATA SET GROUP
The total number of segments in the data set group.

AVERAGE DATA SET GROUP RECORD LENGTH
The average length in bytes of the portion of the database record stored in this data set group.

Return Codes

The HISAM Reorganization Unload utility produces return codes preceded (in the case of errors) by numbered messages to the SYSPRINT data set that more fully explain the results of program execution. The return codes are as follows:

Code	Meaning
0	All requested operations completed successfully
4	One or more operations did not complete successfully
8	Severe errors caused the job to terminate
12	A combination of return codes 4 and 8 occurred
16	The ddname SYSIN could not be opened

Examples

The examples in this section contain the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7
```

This comment line is for reference only, and is not required.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1  DD  DSNAME=RECON1,DISP=SHR
//RECON2  DD  DSNAME=RECON2,DISP=SHR
//RECON3  DD  DSNAME=RECON3,DISP=SHR
```

Example 1

In this example, an index work data set passed from the Prefix Resolution utility is used to create:

- Unloaded versions of two secondary indexes in a shared secondary index database
- An unloaded version of a third secondary index

```
//INDREOR  JOB 1,1,MSGLEVEL=1
//*
//STEP1   EXEC PGM=DFSRRC00,
//             PARM='ULU,DFSURUL0,,,1,,,,,,,,,N,N'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
```

```
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1330
//DBIDX1    DD DSNAME=IMS.INDX1,DISP=SHR
//DXOUT1    DD DSNAME=IMS.DBXOUT1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=SER=DXOUT1,LABEL=(,SL)
//DBIDX2    DD DSNAME=IMS.INDX2,DISP=SHR
//DXOUT2    DD DSNAME=IMS.DBXOUT2,DISP=(,KEEP),
//             UNIT=TAPE,VOL=SER=DXOUT2,LABEL=(,SL)
//NDXDS     DD DSNAME=IMS.NDXWDS,DISP=(OLD,DELETE)
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7
//SYSIN     DD *
OPTIONS=(NSTAT,ABEND)
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
                                                DATABASE
/*
```

The output of the example is used as input to the HISAM Reorganization Reload utility to create the secondary indexes. The MVS Access Method Services utility must have been run to create the secondary index databases.

DBRC and IRLM are turned off in the EXEC PARM statement. The OPTIONS statement specifies that statistics are not wanted and that any serious message causes an abend U0359.

Example 2

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 changed on the second OPTIONS statement.

```
//IDEREORG JOB 1,1,MSGLEVEL=1
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURUL0'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1330
//DBIDX2    DD DSNAME=IMS.INDX2,DISP=SHR
//DBIDX1    DD DSNAME=IMS.INDX1,DISP=SHR
//DBXOUT1   DD DSNAME=IMS.DBXOUT1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=SER=DXOUT1,LABEL(,SL)
//DFSEXTDS  DD DSNAME=IMS.EXTDS1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=SER=DEXTDS,LABEL=(,SL)
//NDXWDS1   DD DSNAME=IMS.XWDS1,DISP=(OLD,PASS)
//NDXWDS2   DD DSNAME=IMS.XWDS2,DISP=(OLD,PASS)
//NDXWDS3   DD DSNAME=IMS.XWDS3,DISP=(OLD,PASS)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7
//SYSIN     DD *
OPTIONS=(STATS,ABEND)
X1EDIX3DB01 DBIDX1   DBXOUT1          NDXWDS1  BUNLOAD INDEX EXTRACT
                                                THOSE MARKED WITH
                                                CONSTANT B INCLUDING
                                                THOSE ON WORK DATA SET
```

```
OPTIONS=(ABENDOFF,NSTAT)
X1RDIX4DB01 DBIDX2   DBXOUT1           NDXWDS2  BUNLOAD 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 3

In this example, JCL is provided to:

- Create two output data sets to be used to create two separate secondary indexes
- Merge two secondary indexes by merging an index work data set created by the Prefix Resolution utility with an existing secondary index (which was created previously as a shared secondary index having only one constant)
- Replace the one constant already in the shared secondary index with the constants in the index work data sets
- Extract a constant from a shared index and put it in a form that can be used by the HISAM Reorganization Reload utility to create another separate secondary index

Each operation is separated by an OPTIONS statement. 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'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1330
//DBIX1A    DD DSNAME=IMS.DBIX1A,DISP=OLD
//DBIX2B    DD DSNAME=IMS.DBIX2B,DISP=OLD
//DBOUT1    DD DSNAME=IMS.DBOUT1A,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,1))
//DBOUT2    DD DSNAME=IMS.DBOUT2A,DISP=(,KEEP),
//             VOL=SER=DBOUT2,LABEL=(,SL)
//INDXWDS1  DD DSNAME=IMS.INDXWDS1,DISP=(OLD,DELETE)
//INDXWDS2  DD DSNAME=IMS.INDXWDS2,DISP=(OLD,DELETE)
//DFSEXTDS  DD DSNAME=IMS.EXTSD1,DISP=(,KEEP),
//             VOL=SER=EXTDS1,LABEL=(,SL)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7
//SYSIN     DD *
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 AREPLACE ANY EXISTING
                                                 A'S WITH ONES FROM
                                                 WORK DATA SET
```

```
OPTIONS=STATS
X1EDI32XDB1 DBIX1A  DBOUT1             INDXWSD2 BTAKE B CONSTANTS FROM
                                                SHARED INDX AND/OR
                                                WORK DATA SET AND PUT
                                                OUT TO DFSEXTDS DD
                                                STATEMENT
/*
//DFSVSAMP    DD  *
1024,10
/*
```

Attention: An 'X' or an 'R' control statement must be supplied for all aliases contained in the shared index DBD. If the 'X' or 'R' statement is omitted for any alias, the actual data that the alias represents might be deleted if the shared secondary index database is deleted or redefined.

Chapter 2. HISAM Reorganization Reload Utility (DFSURRL0)

Use the HISAM Reorganization Reload utility to:

- Reload an HISAM database unloaded by the HISAM Reorganization Unload utility
- Create or merge secondary indexes from a reorganized output data set provided by the HISAM Reorganization Unload utility
- Reload a primary index of a HIDAM database unloaded by the HISAM Reorganization Unload utility

Figure 3 is a diagram of the HISAM Reorganization Reload utility.

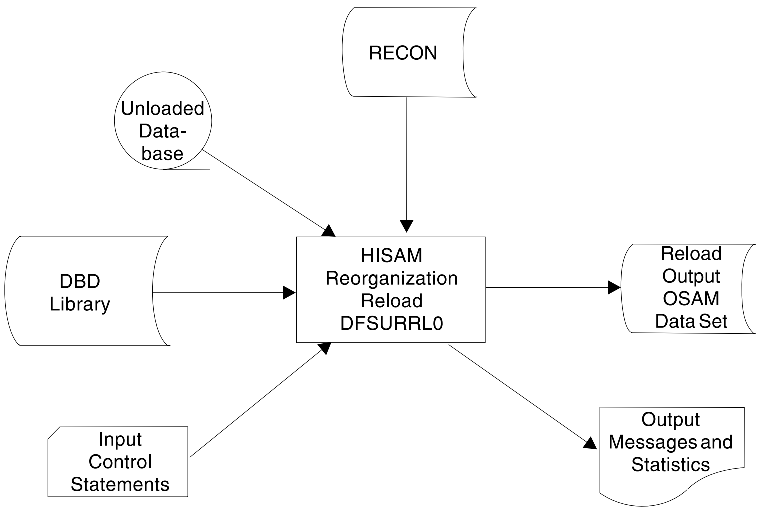

RECON
Unloaded Data-base
DBD Library
HISAM Reorganization Reload DFSURRL0
Reload Output OSAM Data Set
Input Control Statements
Output Messages and Statistics

Figure 3. HISAM Reorganization Reload Utility

Related Reading: The functions of this utility can also be performed by the Utility Control Facility. Refer to “Chapter 30. Utility Control Facility (DFSUCF00)” on page 277 for a description of its operation.

In this Chapter:

- “Restrictions”
- “JCL Requirements” on page 20
- “Utility Control Statement” on page 21
- “Output Messages and Statistics” on page 22
- “Return Codes” on page 25

Restrictions

The following restrictions apply when using the HISAM Reorganization Reload utility:

- You can only use this utility to make changes to logical record length and block size. Use the HD Reorganization Reload utility to make structural changes to a database.

HISAM Reorganization Reload

- You cannot use this utility to reorganize HISAM databases that are indexed by a secondary index or that contain segments with direct address pointers used in logical relationships. The HD Reorganization Unload and Reload utilities must be used instead.
- You must use the same IMS release to reload the database as was used to unload the database.

JCL Requirements

The HISAM Reorganization Reload utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURRL0'
```

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

Related Reading: See Member Name DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having libraries that are not APF authorized concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an APF authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBD that describes the database 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 to 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 be incremented by 1.

vsamout1 DD
Defines the VSAM KSDS 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 database space allocation can be increased by specifying a larger space parameter on this DD statement.

vsamout2 DD
Defines the 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 database space allocation can be increased by specifying a larger space parameter on this DD statement.

Requirement:VSAM databases require a DEFINE control statement to alter space.

SYSIN DD
Defines the input control information data set. The data set can reside on a tape, a direct-access device, or be routed through the input stream. This DD statement is not necessary if no utility control statements 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.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The data set can reside on a tape, direct-access device, or be routed through the input stream.

SYSABEND DD or SYSDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional data set used by DBRC when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, do not use these RECON data set ddnames.

Utility Control Statement

The HISAM Reorganization Reload utility has one optional control statement.

OPTIONS Statement

Options Statement

```
                   ┌─,──────────────┐
                   ▼                │
►►──OPTIONS=(──────┬─ABEND──────┬───┴──)──────────────────────────────►◄
                   ├─ABENDOFF───┤
                   ├─STATS──────┤
                   └─NSTAT──────┘
```

OPTIONS=
Identifies the OPTIONS control statement.

ABEND
Terminates with abend U0359 if any condition arises causing termination of the run. A dump is printed if a SYSABEND or SYSUDUMP DD statement is supplied.

ABENDOFF
Turns off the ABEND function. An abnormal condition causes program termination, but no abend code or dump is provided.

Exception:ABENDOFF is the initial default; however, if ABEND has been specified previously it remains in effect until ABENDOFF is encountered in the JCL code.

STATS
Compares the number loaded against the number provided by the HISAM Reorganization Unload utility, if statistics were provided by the HISAM Reorganization Unload utility

By specifying OPTIONS=STATS or by not using an OPTIONS statement, statistics are provided for each reload.

NSTAT
Causes the statistics provided by the HISAM Reorganization Unload utility to be ignored.

By specifying OPTIONS=NSTAT, no statistics are provided for this job step.

Output Messages and Statistics

The HISAM Reorganization Reload utility provides messages and statistics and an audit trail for each data set group reloaded. An example of the messages and statistics obtained from this utility, accompanied by explanatory information, is shown in Figure 4 on page 23.

```
                          H I E R A R C H I C A L   I N D E X E D   S E Q U E N T I A L
                          D A T A B A S E   R E O R G A N I Z A T I O N   R E L O A D


                                D A T A   S E T   G R O U P   S T A T I S T I C S
DATABASE - DIVNTZ04
PRIMARY  DD - DBHVSAM1
OVERFLOW DD - DBHVSAM2

DEPENDENT OVERFLOW CHAINS (#)
PRIMARY ROOTS OVERFLOW DEPENDENTS NO. LONGEST SHORTEST AVG
3             2        2          1   1       1.00     1

ROOT WITHOUT OVERFLOW CHAINS (BYTES)
NO.  LONGEST  SHORTEST  AVERAGE
1    0        0         0.00

           S E G M E N T   L E V E L   S T A T I S T I C S
SEGMENT      SEGMENT        TOTAL SEGMENTS BY SEGMENT TYPE
NAME         LEVEL          RELOADED        DIFFERENCE
J1             1                  3
J2             2                  3
J3             3                  1
J4             4                  1
J5             2                  4
J6             3                  4
J7             4                  2
J8             5                  1
J9             2                  3
J10            3                  3
J11            4                  0
J7P            3                  2
J12            2                  0
J13            3                  0
J14            4                  0
J13X           3                  0

TOTAL SEGMENTS IN DATA SET GROUP
UNLOADED            RELOADED          DIFFERENCE
27                27

DFS340I    DATABASE DIVNTZ04 HAS BEEN SUCCESSFULLY RELOADED BY FUNCTION SR
DFS339I    FUNCTION SR HAS COMPLETED NORMALLY RC=00
```

Figure 4. Example of Output Statistics for the HISAM Reorganization Reload Utility

If any options were selected for this execution, various messages are generated and appear immediately following the page heading.

Related Reading: An explanation of all numbered messages can be found in *IMS/ESA Messages and Codes*.

Statistics are normally provided on every execution of the HISAM Reorganization Reload utility. Because this increases reload time slightly, installations that are billed by processor time can suppress statistics recording. This is done by using the NSTATS parameter on the OPTIONS utility control statement.

Following the messages for each data set group, the heading “DATA SET GROUP STATISTICS” appears. The fields in this section are:

DATABASE
Database name

HISAM Reorganization Reload

PRIMARY DD
VSAM KSDS ddname of data set group

OVERFLOW DD
VSAM ESDS ddname of data set group

PRIMARY ROOTS
Number of roots loaded

OVERFLOW DEPENDENTS
Number of dependent records loaded

DEPENDENT OVERFLOW CHAINS (#)
Statistics dealing with roots with dependents chained into VSAM ESDS:

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

ROOTS WITHOUT OVERFLOW CHAINS (BYTES)
Statistics dealing with VSAM KSDS which have no VSAM ESDS dependent records:

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

The heading “SEGMENT LEVEL STATISTICS” appears next. The fields in this section are:

SEGMENT NAME
The segment name to which this line of statistics applies

SEGMENT LEVEL
The hierarchic level of this segment in the database

TOTAL SEGMENTS BY SEGMENT TYPE
The total number of segments reloaded and the difference between the number reloaded and unloaded

RELOADED	The total number of segments of this type unloaded
DIFFERENCE	This field is blank if the counts by reload and unload are equal. If they are not equal, the difference is printed. When a database is part of a shared secondary index base, the segment occurrence count always appears under the first segment name. All other segment name counts are zero. Although IMS is unable to distinguish which segment name matches the statistics, the count is correct.

Following the individual segment type statistics for this data set group is the section TOTAL SEGMENTS IN DATA SET GROUP. The fields are:

UNLOADED
The total number of segments unloaded by the HISAM Reorganization Unload utility (DFSURUL0)

RELOADED
The total number of segments reloaded by the HISAM Reorganization Reload utility (DFSURRL0)

DIFFERENCE
The difference, if any, between the previous two totals

Return Codes

One of the following return codes is provided at program termination:

Code	Meaning
0	All operations have successfully completed
4	One or more warning messages were issued
8	One or more operations did not complete successfully
16	A severe error causing program termination occurred

Chapter 3. HD Reorganization Unload Utility (DFSURGU0)

Use the HD Reorganization Unload utility to:

- Unload an HDAM, HIDAM, or HISAM database to an OSAM formatted data set
- Generate a data set with prefix information (if logical relationships exist)
- Make structural changes to a HISAM or HD database

There are no utility control statements for this utility.

Figure 5 is a flow diagram of the HD Reorganization Unload utility.

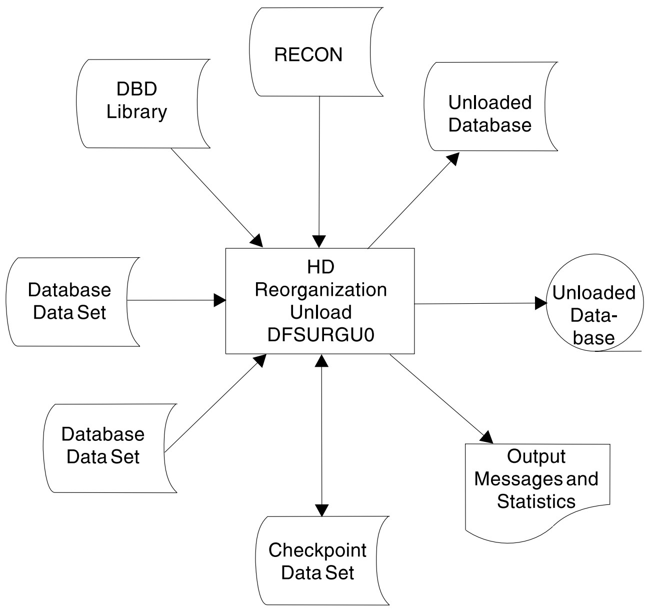


Figure 5. HD Reorganization Unload Utility

The functions of this utility can be performed by the Utility Control Facility.

Related Reading: Refer to “Chapter 30. Utility Control Facility (DFSUCF00)” on page 277 for a description of its operation.

In this Chapter:

- “Restrictions”
- “Recovery and Restart” on page 29
- “JCL Requirements” on page 29
- “Return Codes” on page 33
- “Examples” on page 34

Restrictions

The DBD of the database 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 database during the process of reorganization. The following rules and restrictions apply:

HD Reorganization Unload

- The HD Reorganization Unload utility must have been executed against the DBD describing the current structure of the database, and updates must not have been made since the unload.
- Logical record length and block size can be changed, the format can be changed from OSAM format to VSAM format, or both.
- When unloading an HDAM database, the randomizing module must be included in the JOBLIB.
- An existing segment type can be deleted from the DBD, provided all segments of this type were deleted from the database prior to execution of the HD Reorganization Unload utility.
- New segment types can be added to the new DBD, provided they do not change either the hierarchic relationships 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 to alter the data content of a segment.
- Existing segment lengths can be changed on fixed length segments. IMS cannot alter the data content, however, except to truncate data if the segment is made smaller.

  If the segment length is increased,binary zeros are used as fill characters for the added portion of the segment. It is the your responsibility to replace the extended portion of the segment through use of an application program running in update mode under IMS.
- The DL/I access method can be changed. OSAM format can be changed to VSAM format or VSAM format to 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 database contains logical relationships and if counter, LT, or LP pointers are changed, the Database Prereorganization utility must be rerun. If changing from physical to virtual pairing, all occurrences of the segment which will become virtual must be deleted.
- Do not use the HD Reorganization Unload utility:
  – When changing from bidirectional virtual pairing to bidirectional physical pairing, if any logical child segments have been deleted from either the physical or logical path but not from both paths.
  – When changing a real logical child from one logically related database to another.
  – When reorganizing a prime or secondary index, use the HISAM reorganization utilities for an index database.
  – If a write error has occurred for the database, and the database has not been recovered. The recovery must be executed before this utility is executed. If the database is registered with DBRC, and this utility uses DBRC, then the fact that a write error has occurred and recovery has not been performed is known to DBRC and DBRC rejects the authorization request of this utility.

To accomplish the unload operation, the utility functions as an application program and issues a series of unqualified GN calls to DL/I. A complete pointer integrity validation is not performed as a by-product of executing the Unload utility.

Related Reading: For further information on the use of the HD Reorganization Unload utility, see *IMS/ESA Administration Guide: Database Manager*.

Recovery and Restart

The following must be done before executing the HD Reorganization Reload utility:

1. Assemble and link-edit the new DBD into the IMS DBD library.
2. If adding new segment or deleting segments and the database contains logical relationships, rerun the Database Prereorganization utility against the new DBD.
3. If the DBD name is changed and the DBD contains logical relationships, rerun the Database Prereorganization utility against the new DBD.

Use the following utility execution sequence to recover invalid logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents in a databases.

1. Run the HD Reorganization Unload utility (DFSURGU0) against all databases involved in a logical relationship.
2. Run the Database Prereorganization utility (DFSURPR0) using the DBIL control statement:

   ```
   DBIL=database
   name1,database name2,...
   ```

   The database name is replaced by all database names that are involved in a logical relationship. If database name2 is involved in a logical relationship with database name3, database name3 must also be unloaded and reloaded.
3. Run the HD Reorganization Reload utility (DFSURGL0) to reload the databases using the control data set created by the Database Prereorganization utility as input.

   During the reload, the DBIL statement in the Database Prereorganization utility causes the work data set generator program (DFSDSEH0) to use the concatenated keys of the logical parents instead of the old addresses.
4. Run the Prefix Resolution utility (DFSURG10) to resolve the logical relationships defined for the databases.
5. Run the Prefix Update utility (DFSURGP0) to complete the logical relationship updates.

If adding a logical pointer that does not exist in a previous DBD or changing pointer options from symbolic to direct, use this same utility execution sequence, but run the new DBD after the HD Reorganization Unload utility.

Related Reading: For further information on using the HD Reorganization Unload utility, see “Modifying Your Database” in *IMS/ESA Administration Guide: Database Manager*. For information on scratching and allocating OSAM data sets, see “Allocation for OSAM Data Sets” in *IMS/ESA Installation Volume 1: Installation and Verification*.

JCL Requirements

The HD Reorganization Unload utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement

- DD statements defining inputs and outputs

The output from the HD Reorganization Unload utility is an operating system variable blocked sequential data set. Because output is blocked to the maximum size that the output device can handle, standard labels must be used on output volumes.

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGU0,dbdname'
```

The parameters ULU and DFSURGU0 describe the utility region. *dbdname* is the name of the DBD that describes the database to be reorganized. The normal IMS positional parameters such as SPIE, BUF, and DBRC can follow *dbdname*.

Related Reading: See Member Name DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBD that describes the database to be reorganized (that is, DSN=IMS.DBDLIB,DISP=SHR). This data set must reside on a direct-access device.

SYSPRINT DD
Defines the message and statistics output data set. The data set can reside on a tape, direct-access device, or printer, or be routed through the output stream. DCB parameters specified for this data set are RECFM=FBA and LRECL=121. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 121.

DFSUCKPT DD
Defines the data set to be used to take checkpoints. If checkpoints are not required, do not use this statement. This data set usually 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. Omit this statement if you are not attempting a restart. If you are attempting a restart, ensure that the statement references 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.

The DFSURSRT DD statement writes a special checkpoint record to the checkpoint data set and to the output data sets. 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 database. The statistics table records are read, and the main storage tables are properly initialized. The program then continues with normal processing.

DFSURGU1 DD
Defines the primary output data set. The data set can reside on either a tape or a direct-access device. This DD statement is required.

DFSURGU2 DD
Defines the secondary output data set. Use this DD statement if you are requesting two copies of the output. The data set can reside on either a tape or a direct-access device.

Multiple copies of the database can be produced. 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.

database DD
Defines the database data set to be reorganized. One statement must be present for each data set that is named in the DBD that describes the database being reorganized. The ddname must match the ddname in the DBD.

If this is a HIDAM database, 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 database. No DD statements are required for whatever secondary indexes are associated with this database.

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.

Related Reading: For additional information on control statement format and buffer pool structure, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The data set can reside on a tape, direct-access device, or be routed through the input stream. This statement is required.

DFSCTL DD
Describes the data set containing SBPARM control statements which request activation of sequential buffering. Conditional activation of sequential buffering might improve the buffering performance of OSAM DB data sets and reduce the job elapsed time.

Related Reading: For a description of SBPARM control statements, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS and the record length must be 80. The data set can reside on a direct-access device, a tape, or be routed through the input stream.

SYSABEND DD or SYSUDUMP DD
Define a dump data set. If either statement is supplied, any return code greater than 4 causes an abend U0347. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

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 is shown in Figure 6.

```
 H I E R A R C H I C A L   D I R E C T   D A T A B A S E   R E O R G A N I Z A T I O N   U N L O A D  PAGE 01

DFS343W    DDNAME DFSUCKPT WAS SPECIFIED AS DD DUMMY OR WAS OMITTED FOR FUNCTION DU
DFS342I    RESTART NOT REQUESTED, NORMAL PROCESSING BEGINS
DFS344W    DDNAME FOR SECOND COPY WAS NOT SUPPLIED, 1 COPY REQUESTED FOR FUNCTION DU
COPY 1 ON VOLUME(S) -  USER02
DFS340I    DATABASE DHONTZ04 HAS BEEN SUCCESSFULLY UNLOADED BY FUNCTION DU


                        D A T A B A S E   S T A T I S T I C S
SEGMENT LEVEL STATISTICS                          RECORD LEVEL STATISTICS

MAXIMUM  AVG    MAXIMUM  AVG      SEGMENT SEGMENT TOTAL SEGMENTS  AVG COUNT PER
TWINS    TWINS  CHILDREN CHILDREN NAME    LEVEL   BY SEG TYPE     DB RECORD
1        1.00   8        7.00     K1      1       3               1.00
1        0.66   3        2.50     K2      2       2               0.66
2        1.50   1        0.66     K3      3       3               1.00
1        0.66   0        0.00     K4      4       2               0.66
2        1.66   1        0.60     K5      2       5               1.66
1        0.60   0        0.00     K6      3       3               1.00
4        2.00   0        0.00     K8      2       6               2.00
0        0.00   0        0.00     K9      2       0               0.00
0        0.00   0        0.00     K10     3       0               0.00
0        0.00   0        0.00     K11     3       0               0.00
0        0.00   0        0.00     K12     4       0               0.00
0        0.00   0        0.00     K13     4       0               0.00
0        0.00   0        0.00     K14     3       0               0.00
0        0.00   0        0.00     K15     3       0               0.00

H I E R A R C H I C A L   D I R E C T   D B   R E O R G   U N L O A D PAGE 02

TOTAL SEGMENTS IN DATABASE=24    AVERAGE DATABASE RECORD LENGTH=300  BYTES
DFS339I    FUNCTION DU HAS COMPLETED NORMALLY RC=0
```

Figure 6. 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.

Related Reading: An explanation of all numbered messages can be found in *IMS/ESA Messages and Codes*.

The message “COPY 1 ON VOLUME(S) - *volser1*” appears 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*” appears. The heading “DATABASE STATISTICS” follows the messages.

The fields under the heading “SEGMENT LEVEL STATISTICS” are:

MAXIMUM TWINS
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
The average number of segments of this type encountered under an immediate parent segment. This value is carried out to two decimal places.

MAXIMUM CHILDREN
The maximum number of child segments (at all subordinate levels) under a given parent.

AVERAGE CHILDREN
The average number of child segments (at all subordinate levels) under a given parent. This value is carried out to 2 decimal places. The lowest level segment in any hierarchic path has 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 database. The segment descriptions are mapped from top to bottom, in the same order in which they were described in the DBD.

The fields under the heading “RECORD LEVEL STATISTICS” are:

TOTAL SEGMENTS BY SEGMENT TYPE
The count of the number of occurrences of this segment type in the entire database. The count field in the level 1 segment type reflects the total number of database records (root segments) in the database.

AVERAGE COUNT PER DATABASE RECORD
A count of the average number of occurrences of this segment type within a given database record. The value is carried to two decimal places.

Following the individual segment type statistics is a count of the total number of all segments in the database and the average database record length in bytes. The average database record length includes both the data length and the prefix size. The product of the number of segments at level 1 times the average database record length gives the total number of bytes in the database. Because all physically stored records might not use all available data positions, this figure can only be used as an approximation of the data set space required.

When variable-length segments are involved, the calculation of the average database record length is based on the maximum possible segment length.

Return Codes

One of the following return codes is provided at program termination:

Code	Meaning
0	Database unload completed successfully
4	One or more warning messages were issued
8	A serious error occurred or copy 1 has an I/O error
12	A combination of return codes 4 and 8 occurred
16	Database unload did not complete successfully

Examples

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1    DD DSNAME=RECON1,DISP=SHR
//RECON2    DD DSNAME=RECON2,DISP=SHR
//RECON3    DD DSNAME=RECON3,DISP=SHR
```

Example 1

In this example, a HIDAM database is to be reorganized using the checkpoint facility and two output copies. A restart is not requested. Two database DD statements are provided: one is for the HIDAM OSAM data set, and the other is for the Index database (VSAM) used with HIDAM. If this were the unload of a single data set group HDAM database, only one DD statement would be necessary for access to the database.

```
//HDREORG  JOB 1,1,MSGLEVEL=1
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGU0,DI32DB02'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1210
//DFSUCKPT  DD DSNAME=IMS.CHKPT,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=222222,SPACE=(TRK,(50))
//DFSURGU1  DD DSNAME=IMS.UNLOAD1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSURGU2  DD DSNAME=IMS.UNLOAD2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=TAPE21,LABEL=(,SL)
//HDPAYROL  DD DSNAME=DATABASE.PAYROLL,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0001
//HDINDEXO  DD DSNAME=DATABASE.INDEXO,DISP=OLD,
//             UNIT=SYSDA,VOLUME=SER=DB0003
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//DFSCTL    DD *
SBPARM ACTIV=COND
```

The HDPAYROL DD statement is for the OSAM data set of the HIDAM database. The HDINDEXO DD statement is for the index database.

Example 2

In this example, execution of Example 1 is 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',
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A DCB=BLKSIZE=1210
//DFSUCKPT  DD DSNAME=IMS.CHKPT,DISP=(OLD,KEEP),
//             UNIT=SYSDA,VOL=SER=222222
//DFSURSRT  DD DSNAME=IMS.CHKPT,DISP=(OLD,KEEP),
//             UNIT=SYSDA,VOL=SER=222222
//DFSURGU1  DD DSNAME=IMS.UNLOAD1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=(,,,2,SER=(TAPE11,TAPE12)),
//             LABEL=(,SL)
//DFSURGU2  DD DSNAME=IMS.UNLOAD2,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=(,,,2,SER=(TAPE21,TAPE22)),
//             LABEL=(,SL)
//HDPAYROL  DD DSNAME=DATABASE.PAYROLL,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0001
```

```
//HDINDEXO  DD DSNAME=DATABASE.INDEXO,DISP=OLD,
//             UNIT=SYSDA,VOLUME=SER=DB0003
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//DFSCTL    DD *
SBPARM ACTIV=COND
```

The DFSUCKPT DD statement and the DFSURSRT DD statement can reference the same data set. If restart is successful, the old checkpoint record is overwritten by the next checkpoint taken.

The primary and secondary output DD statements were changed to supply two volumes; only one volume was supplied by the previous execution of the utility. This assumes the previous abnormal termination was caused by an output I/O error that might, for example, have occurred at the end of the volume because there were no additional volumes available.

Because the program does not differentiate between various causes of termination, it positions the volume in use at the time the checkpoint was taken to the applicable checkpoint record. It then issues a force end of volume (FEOV) to cause volume-switching and continues with the new output volume.

To avoid considerable tape handling on restart of a multivolume output execution, remove the volumes that have completed normally 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.

Chapter 4. HD Reorganization Reload Utility (DFSURGL0)

Use the HD Reorganization Reload utility to:

- Reload an HDAM, HIDAM or HISAM database from a data set created by the HD Unload utility
- Create work data sets that are used as input to the logical relationship resolution utilities if the database that is reloaded includes logical relationships or secondary indexes

Sequence checking is performed on all applicable segments of the database records.

A flow diagram of the HD Reorganization Reload utility is shown in Figure 7.

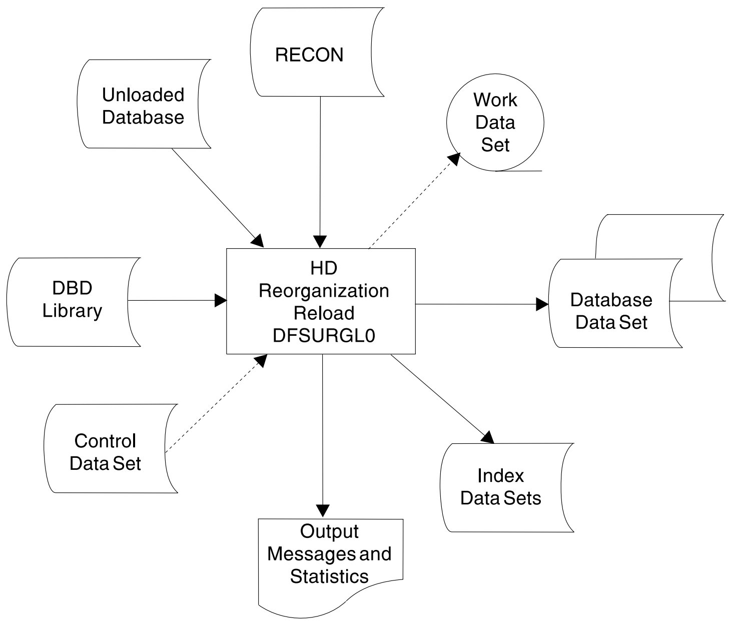


Figure 7. HD Reorganization Reload Utility

The functions of this utility can be performed by the Utility Control Facility.

Related Reading: Refer to “Chapter 30. Utility Control Facility (DFSUCF00)” on page 277 for a description of its operation.

In this Chapter:

Restrictions

The following restrictions apply when using the HD Reorganization Reload utility:

HD Reorganization Reload

- If logical relationships or secondary indexes exist, a work data set is created that must be used later as input to the Database Prefix Resolution utility to resolve any logical or secondary index relationships that exist. Refer to Restrictions in Chapter 3. HD Reorganization Unload Utility (DFSURGU0).
- Do the following before executing the HD Reorganization Reload utility:
  1. Assemble and link-edit the new DBD into the IMS/ESA DBD library.
  2. If adding new segments or deleting segments and the database contains logical relationships, rerun the Database Prereorganization utility against the new DBD.
  3. If the DBD name is changed and the DBD contains logical relationships, rerun the Database Prereorganization utility against the new DBD.
- For information on scratching and allocating OSAM data sets, see “Allocation for OSAM Data Sets” in *IMS/ESA Installation Volume 1: Installation and Verification*.
- Do not use the HD Reorganization Reload utility:
  – When changing from bidirectional virtual pairing to bidirectional physical pairing if any logical child segments have been deleted from either the physical or logical path, but not from both paths.
  – When changing a real logical child from one logically related database to another.
  – To reorganize a HISAM database that is an index to a HIDAM database. Use the HISAM Reorganization Reload utility instead.

Related Reading: For further information on using the HD Reorganization Unload utility, see “Modifying Your Database” in *IMS/ESA Administration Guide: Database Manager*.

JCL Requirements

The HD Reorganization Reload utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGL0,dbdname'
```

The parameters ULU and DFSURGL0 describe the utility region. *dbdname* is the name of the DBD which includes the database to be reloaded. The normal IMS/ESA positional parameters such as SPIE, BUF, and DBRC can follow *dbdname*.

Related Reading: See the Member Name DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMSESA.RESLIB, which contains the IMS/ESA nucleus and required

action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMSESA.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS/ESA SVC modules.

IMS DD
Describes the library containing the DBD referenced in the EXEC statement PARM field. (Normally, this is IMSESA.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 a tape or a direct-access device.

DFSURWF1 DD
Describes the work data set to be created during reload that is used as input by the Prefix Resolution utility (DFSURG10) to resolve logical or secondary index relationships. It can be specified as DUMMY if the database being reloaded is not involved in a logical relationship or with a secondary index.

The DCB parameters for the DD statement must include RECFM=VB and BLKSIZE specified to be the same as that specified for the work data set of the user’s initial load program or for the Database Scan utility (DFSURGS0).

Recommendation:A value of LRECL=900 is recommended, but a smaller value (as small as 300) can be used if no secondary indexes are present.

The data set must reside on either a tape or a direct-access device.

database DD
Defines the database 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 database. The ddname must match the ddname in the DBD.

If this is a HIDAM database, 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 database. No DD statements are required for any secondary indexes that are associated with this database.

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 Database Prereorganization utility (DFSURPR0). This DD statement must be included if logical relationships exist.

This data set must reside on either a 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.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS/ESA Buffer Pool” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.)

Recommendation: The buffer pool size affects performance when reloading a database. For best results, ensure that the buffer pool contains enough blocks to hold a database record (a root segment and all its dependents). Blocks already created are referenced when a segment is inserted, that is, pointed to be a parent or twin in a prior block. If many segments were inserted between these two segments, the buffer containing the parent or twin might have been written to the data set and must be read in again.

DFSCTL DD
Describes the data set containing SBPARM control statements which request activation of sequential buffering. Conditional activation of sequential buffering can improve the buffering performance of OSAM DB data sets and reduce the job elapsed time.

Related Reading: For a description of SBPARM control statements, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS and the record length must be 80. The data set can reside on a direct-access device, a tape, or be routed through the input stream.

SYSABEND DD or SYSUDUMP DD
Define a dump data set. If either statement is supplied, any return code greater than 0 causes an abend U0355. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

Output Messages and Statistics

The HD Reorganization Reload utility provides output messages and statistics. Figure 8 on page 41 is an example of the messages and statistics obtained from this utility.

```
H I E R A R C H I C A L   D I R E C T   DB   R E O R G  R E L O A D


                SEGMENT  LEVEL  STATISTICS

               TOTAL SEGMENTS BY SEGMENT TYPE
SEGMENT      SEGMENT
NAME         LEVEL           RELOADED     DIFFERENCE

K1               1                3

K2               2                2

K3               3                3

K4               4                2

K5               2                5

K6               3                3

K8               2                6

K9               2                0

K10              3                0

K11              3                0

K12              4                0

K13              4                0

K14              3                0

K15              3                0


               TOTAL  SEGMENTS  IN  DATABASE
          UNLOADED          RELOADED      DIFFERENCE

                 24               24
```

Figure 8. Example of Output Statistics from the HD Reorganization Reload Utility

Following the messages, the heading “SEGMENT LEVEL STATISTICS” appears. The fields are:

SEGMENT NAME
The segment name to which this line of statistics applies.

SEGMENT LEVEL
The hierarchic level of this segment in the database. The segments are mapped from top to bottom, in the same order they are described in the DBD, and in the same order they appear in the HD Reorganization Unload statistics.

The two fields under the heading “TOTAL SEGMENTS BY SEGMENT TYPE” are:

RELOADED
The number of occurrences of this segment type in the reload of the entire database.

DIFFERENCE
A blank field if the reload count equals the unload count for this segment. If it is

not blank, a '+' is to the right if there were more counted in reload than in unload; a '−' is there if there were more counted in unload than in reload.

Following the individual segment type statistics, the heading “TOTAL SEGMENTS IN DATABASE” appears. The fields are:

UNLOADED
The total number of all segments in the database counted by the HD Reorganization Unload utility.

RELOADED
The total number of all segments in the data base counted by the HD Reorganization Reload utility.

DIFFERENCE
A blank field if the counts by reload and unload are equal. If they are not equal, the difference is printed.

Return Codes

The following return codes are provided at program termination:

Code	Meaning
0	Database reload was completed successfully
4	There were no segments to reload
8	Reload count differs from unload count
16	Database reload did not complete successfully

Examples

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1   DD DSNAME=RECON1,DISP=SHR
//RECON2   DD DSNAME=RECON2,DISP=SHR
//RECON3   DD DSNAME=RECON3,DISP=SHR
```

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',
//STEPLIB    DD DSNAME=IMSESA.RESLIB,DISP=SHR
//DFSRESLB   DD DSNAME=IMSESA.RESLIB,DISP=SHR
//IMS        DD DSNAME=IMSESA.DBDLIB,DISP=OLD
//SYSPRINT   DD SYSOUT=A
//DFSUINPT   DD DSNAME=IMSESA.UNLOAD1,DISP=OLD,
//              UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSURWF1   DD DSNAME=IMSESA.WRKTAPE1,DISP=(NEW,KEEP),
//              UNIT=TAPE,VOL=SER=WKTAPE,LABEL=(,SL),
//              DCB=(BLKSIZE=1008,LRECL=900,RECFM=VB)
//HDSKILLS   DD DSNAME=DATABASE.SKILLS,DISP=(NEW,KEEP),
//              UNIT=SYSDA,VOL=SER=DB0002,SPACE=(CYL,(10,10))
//DFSURCDS   DD DSNAME=IMSESA.RLCDS,DISP=(OLD,KEEP),
//              UNIT=SYSDA,VOL=SER=IMSMSC
//DFSVSAMP   DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//DFSCTL     DD *
SBPARM ACTIV=COND
```

Example 2

This example shows the JCL for a HIDAM reorganization reload. The primary index database data sets are also described.

```
//HIRELOAD JOB 1,1,MSGLEVEL=1
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,HD32DB02',
//STEPLIB   DD DSNAME=IMSESA.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMSESA.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMSESA.DBDLIB,DISP=OLD
//SYSPRINT  DD SYSOUT=A
//DFSUINPT  DD DSNAME=IMSESA.UNLOAD1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSURWF1  DD DSNAME=IMSESA.WRKTAPE1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=WKTAPE,LABEL=(,SL)
//             DCB=(BLKSIZE=1008,LRECL=900,RECFM=VB)
//HDPAYROL  DD DSNAME=DATABASE.PAYROLL,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0001
//HDINDEXO  DD DSNAME=DATABASE.INDEXO,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=DB0003,SPACE=(CYL,(10,10))
//DFSURCDS  DD DSNAME=IMSESA.RLCDS,DISP=(OLD,KEEP),
//             UNIT=SYSDA,VOL=SER=IMSMSC
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//DFSCTL    DD *
SBPARM ACTIV=COND
```

Example 3

This example shows the JCL for a HIDAM VSAM database unload and reload.

```
//HDREORG  JOB 1,1,MSGLEVEL=1
//UNLOAD  EXEC PGM=DFSRRC00,PARM='ULU,DFSURGU0,DHVBTZ01'
//STEPCAT   DD DISP=SHR,DSN=IMSCAT1
//STEPLIB   DD DSNAME=IMSESA.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMSESA.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMSCT.V1,DBDLIB,DISP=SHR
//          DD DSNAME=IMSCT.V1,PSBLIB,DISP=SHR
//DFSURGU1  DD DSNAME=UNLOAD,DISP=(NEW,PASS),
//             UNIT=SYSDA,SPACE=(TRK,(10,5))
//PRINTDD   DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//DXSK0302  DD DSNAME=VVDX0302,DISP=OLD
//DXSK0301  DD DSNAME=VVDX0301,DISP=OLD
//DHSK0301  DD DSNAME=VVDH0301,DISP=OLD
//DFSVSAMP  DD *
2048,10
//DFSCTL    DD *
SBPARM ACTIV=COND
/*
//STP98   EXEC PGM=IDCAMS
//STEPCAT   DD DSNAME=IMSCAT1,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSABEND  DD SYSOUT=A
//VSA       DD UNIT=SYSDA,DISP=OLD,VOL=SER=VSIMSA
//SYSIN     DD *
 DELETE VVDH0301
 DELETE VVDX0301
 DELETE VVDX0302
 DEF CL (NAME(VVDX0301) CYL(1 1) RECSZ(16 16) VOL(VSIMSA) IXD-
      CISZ(2048) FSPC(25) KEYS(10 5))
 DEF CL (NAME(VVDH0301)   TRK(10 5) RECSZ(2041 2041) VOL(VSIMSA) NIXD-
  CISZ(2048)  )
 DEF CL (NAME(VVDX0302)   TRK(10 5) RECSZ(2041 2041) VOL(VSIMSA) NIXD-
  CISZ(2048)  )
//*
//RELOAD  EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,DHVBTZ01'
```

HD Reorganization Reload

```
//STEPCAT   DD DSNAME=IMSCAT|,DISP=SHR
//STEPLIB   DD DSNAME=IMSESA.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMSESA.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMSCT.V1,DBDLIB,DISP=SHR
//          DD DSNAME=IMSCT.V1,PSBLIB,DISP=SHR
//DFSUINPT  DD DSNAME=UNLOAD,DISP=(OLD,PASS),UNIT=SYSDA
//PRINTDD   DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//DHSK0301  DD DSNAME=VVDH0301,DISP=OLD
//DXSK0302  DD DSNAME=VVDX0302,DISP=OLD
//DXSK0301  DD DSNAME=VVDX0301,DISP=OLD
//DFSVSAMP  DD *
2048,10
//DFSCTL    DD *
SBPARM ACTIV=COND
/*
```

Chapter 5. Database Surveyor Utility (DFSPRSUR)

Use the Database Surveyor utility (DFSPRSUR) to scan all or part of an HDAM or HIDAM database and produce a report describing the physical organization of the database. This report can help you determine the need for reorganization. In addition, the Database Surveyor utility identifies the size and location of free space areas that can receive reorganized records during a partial database reorganization.

Related Reading: See "Chapter 6. Partial Database Reorganization Utility (DFSPRCT1 and DFSPRCT2)" on page 59 for more information on how to use the Database Surveyor utility output.

The Database Surveyor utility can run as a batch message processing (BMP) program against an online database, or as a batch program.

In this Chapter:

Input and Output

The database to be analyzed can be shared when Surveyor is executing as a BMP. Input utility control statements direct the utility to specific sections (key ranges or block number ranges) of a particular database. See "JCL Requirements" on page 46 and "Utility Control Statement" on page 48 for a detailed explanation of all required input.

A PSB containing a PCB for the database being surveyed must be defined for the use of the Database Surveyor utility. (More than one database PCB can be contained in this PSB.) The database PCB must contain SENSEG statements for all segments defined in the corresponding DBD. Ensure that the PSB specifies PROCOPT=G. When the Surveyor is to be executed as a BMP, OLIC=YES must be specified on the PSBGEN statement, and the PSB must be defined to the IMS DC control region.

The utility produces a Database Surveyor report as output. For each partition of a range specified, the Surveyor report contains statistics such as the distribution of the number of blocks accessed to read a database record, the average number of blocks accessed per record, the average size of a record, the total size of all records accessed, and the actual number of records read. Also, the report can indicate the total amount of free space, the distribution of contiguous free space areas by size, or the location of the largest areas of contiguous free space.

For each range specified, the utility divides the number of blocks in the secondary database into 10 parts and lists the portion of segments in each of the 10 parts that belong to the specified range. For example, if there are 56 blocks, the utility divides the number of blocks into 10 parts (ranging from 1 through 6, 7 through 12, ..., 55 through 56) and lists the portion of segments in each part that belongs to the specified range. See "Examples" on page 50 for an annotated sample Surveyor report.

JCL Requirements

The Database Surveyor utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

To run the Database Surveyor utility as a batch program that uses prebuilt blocks, specify:

```
PGM=DFSRRC00,PARM='DBB,DFSPRSUR,...'
```

To run the Database Surveyor utility as a batch program that does not use prebuilt blocks, specify:

```
PGM=DFSRRC00,PARM='DLI,DFSPRSUR,...'
```

To run the Database Surveyor utility as a batch message processing program, specify:

```
PGM=DFSRRC00,PARM='BMP,DFSPRSUR,...'
```

The normal IMS positional parameters can follow the program name in the PARM field.

Related Reading: See information for member name DBBBATCH, member name IMSBATCH, and member name DLIBATCH in the *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch or batch message processing program.

DD Statements

The following DD statements define the required and optional data sets.

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

SYSIN DD
Defines the input control data set for this program. The data set can reside on tape, on a direct-access device, or be routed through the input stream. LRECL and BLKSIZE must both be 80.

IMS DD
Defines the libraries containing the DBD and PSB that describe the database to be analyzed. These data sets must reside on a direct-access device. This statement is required and must always define the DBD library. The PSB library is only required when PARM=DLI is specified.

IMSACB DD
Defines the library containing the ACB that describes the database to be analyzed. This data set must reside on a direct-access device. This statement is required only when PARM=DBB is specified.

SYSPRINT DD
Defines the message and report output data set. The data set can reside on a tape, direct-access device, or printer, or be routed through the output stream.

DCB parameters specified for this data set are RECFM=FBM and LRECL=121. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 121.

This DD statement is required.

IEFRDER DD
Defines the IMS log data set. This statement is required when Surveyor executes as a batch program, but can be specified as DUMMY.

database DD
Defines the database to be analyzed. The ddname must match the ddname in the DBD. This statement is only required when Surveyor executes as a batch program. (When Surveyor executes as a BMP, the database data sets must be defined in the control region JCL.)

DFSVSAMP DD
Describes the data set that contains the buffer pool information required by the DL/I Buffer Handler.

Requirement: This DD statement is required when Surveyor executes as a batch program.

Related Reading: For additional information on control statement format and buffer pool structure, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DFSCTL DD
Describes the data set containing SBPARM control statements, which request activation of sequential buffering (SB).

Related Reading: For a description of SBPARM control statements, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS and the record length must be 80. The data set can reside on a direct-access device, a tape, or be routed through the input stream.

SYSABEND DD or SYSUDUMP DD
Define a dump data set. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

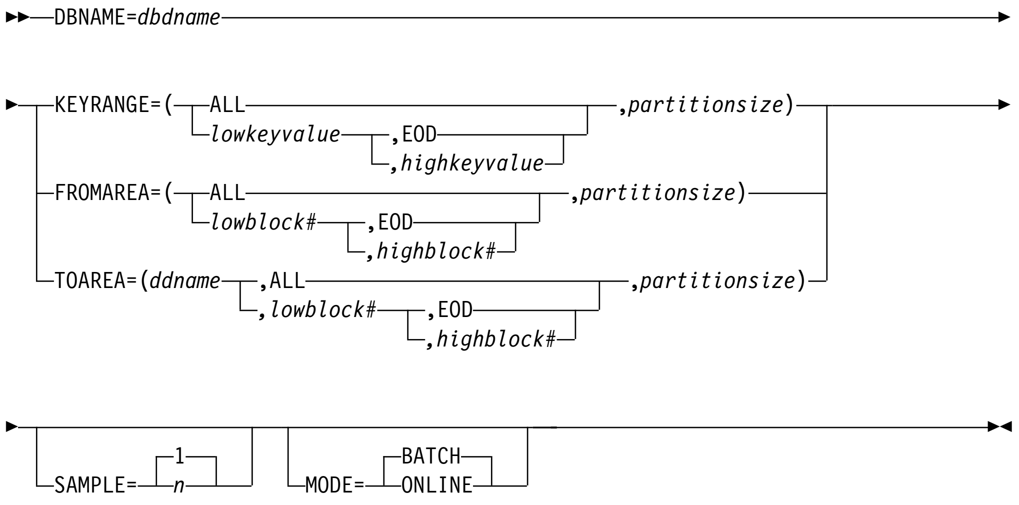
Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statement

Input statements are used to describe processing options for the Database Surveyor utility. Input statement must conform to the following:

- The first character of a statement must start before column 17.
- Multiple keywords cannot be specified on a single statement.
- If there are any blanks embedded within an operand, any characters following the blanks are assumed to be comments.
- Continuation statements are allowed when a keyword's operands extend beyond one input statement.
- There must be a nonblank character in column 72 to continue a statement.
- Comment-only statements are specified by placing an asterisk in column 1.

The format of the Database Surveyor utility control statement is:



DBNAME=
Specifies the database to be surveyed to find database distortions, free space, or both. This operand must be the name of a DBD with HD organization. DBNAME is required and must appear only once.

KEYRANGE=
Specifies the range of keys to be analyzed for database distortions. Only one KEYRANGE definition is allowed.

Restriction: If KEYRANGE is specified, FROMAREA or TOAREA cannot be specified on other input statements in the same job stream.

KEYRANGE is invalid if the database is HDAM. The operands are the root segment keys or generic keys, with a maximum of 255 bytes each. Keys can be expressed in hexadecimal by preceding the value with an X and enclosing them in quotation marks; for example:

```
KEYRANGE=(X'C8C5E7',X'D2C5E8',1000)
```

The high key value can be specified as "EOD" if the upper limit is the end of the database. The low key value can be specified as "ALL" and the high key value omitted, in which case, the range is the entire database.

The partition size is specified as the number of database records to be included in each partition of the range. The number specified must be from 1 to 9999. The Database Surveyor utility produces statistics for each partition of the range and also for the entire range.

FROMAREA=

Specifies one range of blocks to be analyzed for database distortions. Only one FROMAREA definition is allowed.

Restriction: If FROMAREA is specified, KEYRANGE or TOAREA cannot be specified on other input statements in the same job stream.

FROMAREA is invalid if the database is HIDAM. The operands are block numbers in the root addressable area. The high block number can be specified as “EOD” if the upper limit is the last block in the root addressable area. The low block number can be specified as “ALL” and the high block number omitted, in which case, the range is the entire root addressable area.

The partition size is specified as the number of blocks of root addressable area to be included in each partition of the range. The number specified must be from 1 to 9999. Surveyor produces statistics for each partition of the range and also for the entire range.

TOAREA=

Specifies one area of the database to be analyzed for free space. TOAREA keywords are defined to be a range within a data set group. Up to ten TOAREA definitions can be specified for a single database.

Restriction: If TOAREA is specified, KEYRANGE or FROMAREA cannot be specified on other input statements in the same job stream. *ddname* must be the name of a DD statement defining a data set in the database being surveyed.

The block numbers can be coded as any block number within the limits of the data set, ALL, or EOD. The low block number has a minimum value of 2. If the low block number is specified as less than 2, 2 is assumed. If the low block number is specified as ALL, with the high block number omitted, the range is the entire data set group. If the high block number is specified as EOD, the upper limit of the range is the last block of the data set group.

The partition size is specified as the number of blocks to be included in each partition of the range. The number specified must be from 1 to 9999. Surveyor produces statistics for each partition of the range and also for the entire range.

SAMPLE=

Specifies that the utility is to sample only a part of each range. The operand for this keyword is specified as a number between 1 and 1000. The operand specifies number of records for KEYRANGEs and FROMAREAs and number of blocks for TOAREAs. Every *n*th record (or block) is accessed and intervening records (or blocks) are ignored for analysis. For example, if *n*=10, then the first record/block, the 11th record/block, the 21st record/block, and so forth, is accessed and used for reporting database storage information. If the SAMPLE keyword is omitted, every record (or block) in the range is accessed.

MODE=

Specifies whether the Database Surveyor utility is run as a BMP (MODE=ONLINE) or batch program (MODE=BATCH). BATCH is the default.

Return Codes

The Database Surveyor utility produces the following return codes at program termination:

Code	**Meaning**
0	No errors were detected
4	Warning message was issued
8	Program terminated abnormally

Related Reading: See *IMS/ESA Messages and Codes* for explanations of all messages produced by the Database Surveyor utility.

Examples

The examples in this section contain the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1    DD DSNAME=RECON1,DISP=SHR
//RECON2    DD DSNAME=RECON2,DISP=SHR
//RECON3    DD DSNAME=RECON3,DISP=SHR
```

Example 1

This example shows the JCL and utility control statements required to execute DFSPRSUR as a batch program to analyze portions of a database, using the FROMAREA keyword.

```
//STSTN53  EXEC PGM=DFSRRC00,
//         PARM='DLI,DFSPRSUR,PRPSB23P,,,,,,,,,,,N,N',
//         REGION=512K
//IMS       DD DSN=IMSTESTG.I11X.PSBLIB,DISP=SHR
//          DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SNAPDD   DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//PR23DD1   DD  DSN=PR23RW00,DISP=SHR
//PR23DD2   DD  DSN=PR23A,DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN    DD  *
    MODE=BATCH
    DBNAME=PR23RW00
    FROMAREA=(001,EOD,1)
//DFSVSAMP DD  *
```

Figure 9 on page 51 is a sample FROMAREA partition report produced by running Example 1. One of these reports is produced for each partition.

```
     SURVEYOR FROMAREA PARTITION REPORT FOR DBD PR23RW00
     PARTITION BLOCK NUMBERS 2 TO 2
     TOTAL NUMBER OF ROOTS = 2
     TOTAL LENGTH OF SEGMENTS = 129
     AVERAGE LENGTH PER DBR =  64
     SEGMENTS OF A DATABASE RECORD (DBR) MAY SPREAD ACROSS SEVERAL BLOCKS
     FOR MANY REASONS. THIS TABLE SUMS THE NUMBER OF DATABASE RECORDS BY
     THE NUMBER OF BLOCKS THEY OCCUPY.

     # OF BLOCKS/SUM OF # DBR WHICH OCCUPY THIS # OF BLOCKS
     NO. BLK  1     2     3     4     5    6-8   9-11  12-14  15-17 > 17
     NO. DBR  0     2     0     0     0    0     0     0      0       0

     THIS TABLE SHOWS THE NUMBER AND LENGTH OF  SEGMENTS IN EACH TENTH OF
     EACH DSG FOR THIS  DATABASE.

DSG                   BLOCK     TOTAL LENGTH  NO.          AVG LENGTH   % AREA
DDNAME   BLKSIZE    LOW  HIGH   OF SEGMENTS   OF SEGMENTS  OF SEGMENTS  OCCUPIED
------   -------    ---------   ------------  -----------  -----------  --------
PR23DD1  4096       1-     15   53            2             26          .1
                    16-    30    0            0              0          .0
                    31-    45    0            0              0          .0
                    46-    60    0            0              0          .0
                    61-    75    0            0              0          .0
                    76-    90    0            0              0          .0
                    91-    105   0            0              0          .0
                   106-    120   0            0              0          .0
                   121-    135   0            0              0          .0
                   136-    EOD   0            0              0          .0
```

Figure 9. Surveyor-FROMAREA-Partition Report

The fields in the FROMAREA partition report are:

PARTITION BLOCK NUMBERS
Low and high block numbers in the partition

TOTAL NUMBER OF ROOTS
Total number of roots in the partition

TOTAL LENGTH OF SEGMENTS
Total length of all segments in the partition

AVERAGE LENGTH PER DBR
Average length of a database record

A table, which shows the relationship between the number of records and the number of blocks they are spread across, appears next. The fields in the table are:

NO. BLK
Heading line of number of blocks

NO. DBR
Number of DBRs spread across the number of blocks in heading above

A table that shows the characteristics of each 10th of each data set group (DSG) in the database being surveyed, appears next. The fields in the table are:

DSG DDNAME
ddname of DSG.

BLKSIZE
Block size of the DSG.

BLOCK LOW and HIGH
Low and high block numbers of this portion of the DSG.

TOTAL LENGTH OF SEGMENTS
Total length of all segments (found in partition being reported) which physically reside in this portion of this DSG.

NUMBER OF SEGMENTS
Number of segments (found in partition being reported) which physically reside in this portion of this DSG.

AVERAGE LENGTH OF SEGMENTS
Average length of segments (found in partition being reported) which physically reside in this portion of this DSG.

PERCENT OF AREA OCCUPIED
Percentage of area occupied by segments (found in partition being reported) that physically reside in this portion of this DSG. Segments belonging to database records outside the range being reported can physically reside within the DSG area being reported but are not reflected in this report.

Figure 10 is a sample FROMAREA range report produced by running Example 1. One of these reports is produced to summarize the entire range.

```
SURVEYOR FROMAREA RANGE REPORT FOR DBD PR23RW00
RANGE BLOCK NUMBERS       1 TO  10
TOTAL NUMBER OF ROOTS =         20
TOTAL LENGTH OF SEGMENTS =     991
AVERAGE LENGTH PER DBR =        49
SEGMENTS OF A DATABASE RECORD (DBR) MAY SPREAD ACROSS SEVERAL BLOCKS
FOR MANY REASONS.  THIS TABLE SUMS THE NUMBER OF DATABASE RECORDS BY
THE NUMBER OF BLOCKS THEY OCCUPY.
# OF BLOCKS/SUM OF # DBR WHICH OCCUPY THIS # OF BLOCKS

NO. BLK  1      2      3      4      5     6-8    9-11   12-14   15-17  > 17

NO. DBR  5      15     0      0      0     0      0       0      0       0
```

Figure 10. Surveyor-FROMAREA-Range Report

The fields in the FROMAREA Range report are:

RANGE BLOCK NUMBERS
Low and high block numbers in the range

TOTAL NUMBER OF ROOTS
Total number of roots in the range

TOTAL LENGTH OF SEGMENTS
Total length of all segments in the range

AVERAGE LENGTH PER DBR
Average length of a database record

A table, which shows the relationship between the number of records and the number of blocks they are spread across, appears next. The fields in the table are:

NO. BLK
Heading line of number of blocks

NO. DBR
Number of DBRs spread across the number of blocks in heading above

Example 2

This example shows the JCL and utility control statements required to execute DFSPRSUR as a batch program to analyze an entire database, using the TOAREA keyword.

```
//STSTN54  EXEC PGM=DFSRRC00,
//         PARM='DLI,DFSPRSUR,PRPSB23P,,,,,,,,,,N,N',
//         REGION=512K
//IMS       DD DSN=IMSTESTG.I11X.PSBLIB,DISP=SHR
//          DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SNAPDD   DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//PR23DD1   DD  DSN=PR23RW00,DISP=SHR
//PR23DD2   DD  DSN=PR23A,DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN    DD  *
    MODE=BATCH
    DBNAME=PR23RW00
    TOAREA=(PR23DD1,ALL,1)
    TOAREA=(PR23DD2,ALL,1)
//DFSVSAMP DD  *
```

Figure 11 is a sample partition report produced running Example 2. One of these reports is produced for each partition.

```
     SURVEYOR TOAREA PARTITION REPORT FOR DBD PR23RW00 DSG PR23DD1
PARTITION BLOCK NUMBERS   149 TO    149
TOTAL NUMBER OF FREE SPACE ELEMENTS   =          1
TOTAL AMOUNT OF FREE SPACE            =       4085
TOTAL NUMBER OF BLOCKS ACCESSED       =          1
AVERAGE AMOUNT OF FREE SPACE PER FSE  =       4085
PERCENT OF FREE SPACE TO TOTAL AREA   =       99.9
                    TEN LARGEST AREAS OF FREE SPACE
SIZE    NUMBER   LOCATION (FIRST TEN)
4085        1      149
```

Figure 11. Surveyor-TOAREA-Partition Report

The fields of the TOAREA partition report are:

PARTITION BLOCK NUMBERS
Low and high block numbers in the partition.

TOTAL NUMBER OF FREE SPACE ELEMENTS
Total number of free space elements found in the partition. The Database Surveyor utility calculates the free space available in blocks that have not been formatted by the IMS space managements routines. To make the calculation, each of these unformatted blocks is considered to contain one free-space anchor point (FSEAP) with one free-space element (FSE).

TOTAL AMOUNT OF FREE SPACE
Total amount of free space found in partition.

TOTAL NUMBER OF BLOCKS ACCESSED
Total number of blocks accessed in partition.

AVERAGE AMOUNT OF FREE SPACE PER FSE
Average amount of free space per free space element which was found in partition.

PERCENT OF FREE SPACE TO TOTAL AREA
Percentage of free space found in the partition blocks that were surveyed.

A table, which shows the 10 largest areas of free space, appears next. The fields in the table are:

SIZE
Size of free space element

NUMBER
Number of free space elements of the size indicated under SIZE which were found in this partition

LOCATION (FIRST TEN)
Block number of the first 10 free space elements of the size indicated under SIZE which were found in this partition

Figure 12 is a sample range report produced by running Example 2. One of these reports is produced to summarize the entire range.

```
     SURVEYOR TOAREA RANGE REPORT FOR DBD PR23RW00
RANGE BLOCK NUMBERS     2 TO    149
TOTAL NUMBER OF FREE SPACE ELEMENTS   =        148
TOTAL AMOUNT OF FREE SPACE            =     603949
TOTAL NUMBER OF BLOCKS ACCESSED       =        148
AVERAGE AMOUNT OF FREE SPACE PER FSE  =       4080
PERCENT OF FREE SPACE TO TOTAL AREA   =       99.7
```

Figure 12. Surveyor-TOAREA-Range Report

The fields of the TOAREA range report are:

RANGE BLOCK NUMBERS
Low and high block numbers in the range.

TOTAL NUMBER OF FREE SPACE ELEMENTS
Total number of free space elements found in the range. The Database Surveyor utility calculates the free space available in blocks that have not been formatted by the IMS space managements routines. To make the calculation, each of these unformatted blocks is considered to contain one free-space anchor point (FSEAP) with one free-space element (FSE).

TOTAL AMOUNT OF FREE SPACE
Total amount of free space found in the range.

TOTAL NUMBER OF BLOCKS ACCESSED
Total number of blocks accessed in the range.

AVERAGE AMOUNT OF FREE SPACE PER FSE
Average amount of free space per free space element which was found in the range.

PERCENT OF FREE SPACE TO TOTAL AREA
Percentage of free space found in the range that was surveyed.

Example 3

This example shows the JCL and utility control statements required to execute DFSPRSUR as a BMP to analyze portions of a database using the KEYRANGE keyword. The DD statements for the database being analyzed must be included in the IMS control region JCL.

```
//STSTE01  EXEC PGM=DFSRRC00,
//         PARM='DLI,DFSPRSUR,PRPSB01Y,,,,,,,,,,,N,N',
//         REGION=512K,
//         COND=EVEN
//IMS       DD DSN=IMSTESTG.I11X.PSBLIB,DISP=SHR
//          DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SYSUDUMP DD  SYSOUT=A                                                  40
//SNAPDD   DD  SYSOUT=A                                                  40
//PR01DD   DD  DSN=PR01RW00,DISP=SHR
//PR01IDD  DD  DSN=PR01I,DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7
//SYSIN    DD  *
       MODE=BATCH
       DBNAME=PR01RW00
       KEYRANGE=(000050,000100,10)
       SAMPLE=2
//DFSVSAMP DD   *
```

Figure 13 is a sample partition report produced by executing the JCL in Example 3. One of these reports is produced for each partition.

```
SURVEYOR KEYRANGE PARTITION REPORT FOR DBD PR01RW00
KEYRANGE =  '000050'
TO =        '000090'
TOTAL NUMBER OF ROOTS =             3
TOTAL LENGTH OF SEGMENTS =        454
AVERAGE LENGTH PER DBR =          151
SEGMENTS OF A DATABASE RECORD (DBR) MAY SPREAD ACROSS SEVERAL BLOCKS
FOR MANY REASONS.  THIS TABLE SUMS THE NUMBER OF DATABASE RECORDS BY
THE NUMBER OF BLOCKS THEY OCCUPY.
# OF BLOCKS/SUM OF # DBR WHICH OCCUPY THIS # OF BLOCKS
NO. BLK  1      2      3      4      5     6-8    9-11  12-14  15-17  > 17
NO. DBR  3      0      0      0      0     0      0     0      0       0

THIS TABLE SHOWS THE NUMBER AND LENGTH OF  SEGMENTS IN EACH TENTH OF
EACH DSG FOR THIS  DATABASE.
DSG              BLOCK    TOTAL LENGTH  NO. OF    AVG LENGTH   % AREA
DDNAME BLKSIZE  LOW  HIGH  OF SEGMENTS   SEGMENTS  OF SEGMENTS  OCCUPIED
------ -------  ---------- ------------  --------- -----------  -------------
PR01DD 4096     1-     15  454           5         90           .7
                16-    30  0             0         0            .0
                31-    45  0             0         0            .0
                46-    60  0             0         0            .0
                61-    75  0             0         0            .0
                76-    90  0             0         0            .0
                91-    105 0             0         0            .0
                106-   120 0             0         0            .0
                121-   135 0             0         0            .0
                136-   EOD 0             0         0            .0
```

Figure 13. Surveyor-KEYRANGE-Partition Report

The fields in the KEYRANGE partition report are:

KEYRANGE= and TO=
Low and high key values in the partition

TOTAL NUMBER OF ROOTS
Total number of roots in the partition

TOTAL LENGTH OF SEGMENTS
Total length of all segments in the partition

AVERAGE LENGTH PER DBR
Average length of a database record

A table, which shows the relationship between the number of records and the number of blocks they are spread across, appears next. The fields in the table are:

NO. BLK
Heading line of number of blocks

NO. DBR
Number of DBRs spread across the number of blocks in heading above

A table, which shows the characteristics of each 10th of each DSG in the database being surveyed, appears next. The fields in the table are:

DSG DDNAME
ddname of DSG.

BLKSIZE
Block size of the DSG.

BLOCK LOW and HIGH
Low and high block numbers of this portion of the DSG.

TOTAL LENGTH OF SEGMENTS
Total length of all segments (found in partition being reported) which physically reside in this portion of this DSG.

NUMBER OF SEGMENTS
Number of segments (found in partition being reported) which physically reside in this portion of this DSG.

AVERAGE LENGTH OF SEGMENTS
Average length of segments (found in partition being reported) which physically reside in this portion of this DSG.

PERCENT OF AREA OCCUPIED
Percentage of area occupied by segments (found in partition being reported) that physically reside in this portion of this DSG. Segments that belong to database records outside the range being reported can physically reside within the DSG area being reported but are not reflected in this report.

Figure 14 is a sample range report produced by running Example 3. One of these reports is produced to summarize the entire range.

```
SURVEYOR KEYRANGE RANGE REPORT FOR DBD PR01RW00
KEYRANGE ='000050'
TO ='000100'
TOTAL NUMBER OF ROOTS = 3
TOTAL LENGTH OF SEGMENTS = 454
AVERAGE LENGTH PER DBR = 151
SEGMENTS OF A DATABASE RECORD (DBR) MAY SPREAD ACROSS SEVERAL BLOCKS
FOR MANY REASONS. THIS TABLE SUMS THE NUMBER OF DATABASE RECORDS BY
THE NUMBER OF BLOCKS THEY OCCUPY.
# OF BLOCKS/SUM OF # DBR WHICH OCCUPY THIS # OF BLOCKS
NO. BLK  1     2     3     4     5     6-8    9-11   12-14   15-17 > 17
NO. DBR  3     0     0     0     0     0      0      0       0       0
```

Figure 14. Surveyor-KEYRANGE-Range Report

The fields in the KEYRANGE range report are:

KEYRANGE= and TO=
Low and high block numbers in the range

TOTAL NUMBER OF ROOTS
Total number of roots in the range

TOTAL LENGTH OF SEGMENTS
Total length of all segments in the range

AVERAGE LENGTH PER DBR
Average length of a database record

A table which shows the relationship between the number of records and the number of blocks they are spread across, appears next. The fields in the table are:

NO. BLK
Heading line of number of blocks

NO. DBR
Number of DBRs spread across the number of blocks in heading above

Chapter 6. Partial Database Reorganization Utility (DFSPRCT1 and DFSPRCT2)

Use the Partial Database Reorganization utility to reorganize user-specified ranges of an HDAM or HIDAM database.

Definition: A *range* is either a group of HIDAM records with continuous key values or a group of HDAM records with continuous relative block numbers. A range is specified using a low and high pair of key or block number values.

The Database Surveyor utility can be used to aid in the selection of the ranges to be reorganized.

Related Reading: For more information and samples of output, see “Chapter 5. Database Surveyor Utility (DFSPRSUR)” on page 45.

The Partial Database Reorganization utility, in one execution, performs operations similar to several other reorganization utilities, such as:

- Unloading, in hierarchic sequence, all root segments within a specified range and all segments dependent on those roots
- Reloading, in hierarchic order, those root and dependent segments into specified contiguous free space
- Resolving all pointers relating to the reloaded segments, including:
  - Logically related segments in the same database
  - Logically related segments in other databases
  - Physical twin pointers of roots at boundaries of selected ranges

To accomplish the reorganization, two steps are executed:

“Step 1” performs preorganization functions to check for user errors without requiring use of the database.

“Step 2” performs the unload/reload/pointer resolution functions with the database offline.

A detailed description of the input, output, and function of both steps appears under “Input and Output” on page 60.

In this Chapter:

Restrictions

The following restrictions apply when using the Partial Database Reorganization utility:

- Structural changes to the database are not allowed.

Partial Reorganization

- HISAM logically related databases cannot have a direct pointer in a logical child or logical parent segment.
- Up to 49 related databases can be processed.
- As many as 500 segment types can participate in a reorganization.
- Up to 500 scan and reload actions are allowed for pointer resolution.
- Up to 10 KEYRANGEs or FROMAREAs are allowed.
- Up to 10 TOAREAs are allowed after each FROMAREA or KEYRANGE.

Input and Output

Two steps are used in the Partial Database Reorganization utility. The input for each step is described in the following sections.

Several reports are produced by the utility. For samples and explanations of these reports see “Examples” on page 68.

Step 1 (Prereorganization)

The first step of partial database reorganization performs initialization functions consisting of:

Reading control statements that specify the range of records
Creating control tables for use during Step 2
Determining logically related databases that might require pointer resolution
Preparing a report

Step 1 requires, as input, the DBD of the database to be reorganized and utility control statements defining the record ranges and sort options.

The primary output is the set of control tables to be used by Step 2 to perform the partial reorganization. Optionally, PSB source statements can be produced to create a PSB for use in Step 2. (This PSB must contain PCBs for four required GSAM work data sets and must be assembled and link-edited before Step 2; see “PSB Example” on page 70.) After the PSB generation is done for the databases to be reorganized, that process does not have to be repeated for subsequent reorganizations of the same databases. Step 1 also produces two scan reports. The REQUIRED-SEGMENT-SCAN report lists any logically related segments of logically related databases that are automatically scanned during Step 2 processing. The OPTIONAL-SEGMENT-SCAN report lists any logically related segments of logically related databases for which a scan is optional. (To select an optional segment for scanning, provide a SCANSEG control statement as Step 2 input.)

The scan examines every occurrence of all segment types being scanned and builds a work record for each occurrence. Step 2 uses these work records to speed its location of the segment occurrences that need pointer updating. If a scan is not performed for the optional scan segments, the prefix update phase of Step 2 must follow pointer chains to locate segments to be updated.

Scanning the optional segments has advantages when:

- Other segments in the same database require the scan
- A large proportion of the optional scan segments contains pointers to required scan segments being moved in a logically related database

Additional Step 1 outputs include reports, possibly error messages, and a return code.

Step 2 (Unload/Reload/Pointer Resolution)

The second step reads the control tables produced by Step 1 and the user-supplied control statements. According to the specified record ranges, all segments (roots and their dependents) are unloaded in hierarchic order to an intermediate data set. The space within the database that these records occupied is freed. Again according to your specification, the records are reloaded into ranges of free space within the database and the new record locations saved in work records. Then, all logically related databases are scanned for pointers to the records that have been moved: work records are created to designate where pointer resolution is required. All work records are then sorted (by MVS SORT) according to database name and segment name. Finally, all pointers to the records having new locations are changed to point to the new locations.

Output from Step 2 includes the partially reorganized database, as well as a report of what was done and a return code. In the case of an unsuccessful execution, an error message is issued.

Recovery and Restart

The Partial Database Reorganization utility has restart capabilities. Checkpoints are taken before the unload/reload phase, at the end of each sort phase, and at the beginning of the prefix update phases. A restart can be performed from any checkpoint. Before restarting, you must use the Batch Backout utility to undo any changes made after the checkpoint was taken.

Related Reading: For more information, see "Chapter 21. Batch Backout Utility (DFSBBO00)" on page 195.

JCL Requirements

The Partial Database Reorganization utility is executed as two standard MVS jobs in an IMS batch processing region. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

Step 1

This section explains the EXEC statement and DD statements for Step 1. See "Step 2" on page 62 for the EXEC statement and DD statements for Step 2.

EXEC Statement

The EXEC statement describes the program to be run. The format of the statement is:

```
PGM=DFSPRCT1
```

DD Statements

The following DD statements define the required and optional input and output data sets for Step 1.

STEPLIB DD
Points to the IMS.RESLIB, which contains the IMS nucleus and required action modules.

IMS DD
Defines the library containing the DBDs that describe the database to be reorganized and all logically related databases.

SYSPRINT DD
Defines the message and report output data set. The data set can reside on a tape, a direct-access device, or a printer, or be routed through the output stream.

DCB parameters for this data set are RECFM=FBA and LRECL=121. BLKSIZE must be provided on the DD statement and must be a multiple of 121.

This DD statement is required.

SYSIN DD
Defines the input control data set for this job. The data set can reside on a tape, or direct-access device, or be routed through the output stream. The LRECL must be specified as 80.

SYSABEND DD or SYSUDUMP DD
Define a dump data set. If both statements are present, the last occurrence is used for the dump.

SYSPUNCH DD
Defines the data set that contains the PSB source statements if the user elected to have them produced.

DCB parameters for this data set are RECFM=FBS and LRECL=80. BLKSIZE must be provided on this DD statement and must be a multiple of 80.

DFSPRCOM DD
Defines the data set that contains the control tables that are to be used by Step 2.

Step 2

This section explains the EXEC statement and DD statements for Step 2. See “Step 1” on page 61 for the EXEC statement and DD statements for Step 1.

EXEC Statement

The EXEC statement describes the program to be run. The format is:

```
PGM=DFSRRC00,PARM=(DLI,DFSPRCT2,psbname)
```

DLI describes the region, DFSPRCT2 names the Step 2 program, and *psbname* is the name of the PSB that includes the database to be reorganized. The normal IMS positional parameters can follow the psbname in the PARM field.

Related Reading: See “Member Name DBBBATCH” or “Member Name DLIBATCH” in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing program.

DD Statements

The following DD statements define the required and optional input and output data sets for Step 2.

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library which contains the IMS SVC modules.

IMS DD
Defines the libraries containing the DBDs that describe the database to be reorganized and all logically related databases. This library must also contain the PSB named in the EXEC statement and the required GSAM DBDs.

DFSPRWF1 DD
Defines an intermediate work data set. Specify DISP=NEW for this data set.

DFSPRWF2 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=18 and RECFM=FB. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001. In this case, specify DISP=OLD. A GSAM DBD must be generated for this data set.

DFSPRWF3 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=18 and RECFM=FB. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001. In this case, specify DISP=OLD. A GSAM DBD must be generated for this data set.

DFSPRWF4 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=1000 and RECFM=VB. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001. In this case, specify DISP=OLD. A GSAM DBD must be generated for this data set.

DFSPRWF5 DD
Defines an intermediate work data set. This is a GSAM data set. DCB parameters must include LRECL=1000 and RECFM=VB. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001. In this case, specify DISP=OLD. A GSAM DBD must be generated for this data set.

DFSPRWF6 DD
Defines an intermediate work data set. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR001. In this case, specify DISP=OLD.

DFSPRWF7 DD
Defines an intermediate work data set. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR002. In this case, specify DISP=OLD.

DFSPRWF8 DD
Defines an intermediate work data set. Specify DISP=NEW for this data set, except when executing a restart from a checkpoint with a number greater than DFSPR003. In this case, specify DISP=OLD.

DFSPRWF9 DD
Defines an intermediate work data set. Specify DISP=NEW for this data set,

except when executing a restart from a checkpoint with a number greater than DFSPR004. In this case, specify DISP=OLD.

DFSPRWFA DD
Defines an intermediate work data set. Specify DISP=NEW for this data set.

SORTLIB DD
Defines the data set containing the load modules for the operating system SORT/MERGE program. This DD statement is required.

SORTWKnn DD
Defines intermediate storage data sets for the operating system SORT/MERGE program.

Related Reading: Refer to *S/360 OS Sort/Merge: Programmer's Guide* for information regarding specification and size of intermediate storage data sets. These DD statements are required.

IEFRDER DD
Defines the IMS log data set, which must reside on a direct-access device. This statement is required.

If this data set is specified as DD DUMMY, no checkpoint/restart capability is available.

SYSPRINT DD
Defines the message and report output data set. The data set can reside on a tape, a direct-access device, or a printer, or be routed through the output stream. This statement is required.

DCB parameters for this data set are RECFM=FBA and LRECL=121. BLKSIZE must be provided on this DD statement and must be a multiple of 121.

DFSPRCOM DD
Defines the data set created by Step 1 containing the control tables.

SYSIN DD
Defines the input control data set for this job. The data set can reside on a tape, or a direct-access device, or be routed through the output stream.

SYSOUT DD
Defines the message output data set for SORT/MERGE. The ddname is the one specified during generation of invoked sort (normally, the ddname is SYSOUT). This data set can reside on a printer, a tape, or a direct-access device, or be routed through the output stream. This DD statement is required.

database DD
Defines the database data sets to be reorganized, all logically related database data sets, and all secondary index data sets. The ddnames must match those specified in the DBD.

If this is a HIDAM database, DD statements for the index data sets must also be supplied. The ddnames must match those specified for the index data sets in the DBD.

These data sets must reside on a direct-access device. DISP=OLD is recommended.

DFSVSAMP DD
Describes the data set that contains the buffer pool information required by the DL/I buffer handler. This DD statement is required.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS/ESA Buffer Pool” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DFSCTL DD
Describes the data set containing SBPARM control statements, which request activation of sequential buffering (SB).

Related Reading: For a description of SBPARM control statements, see “SB Parameters (SBPARM) Control Statement” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS and the record length must be 80. The data set can reside on a direct-access device, a tape, or be routed through the input stream.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statements

Input statements are used to describe processing options for Partial Database Reorganization. Input statements must conform to the following:

- The first character on a statement must start before column 17.
- Multiple keywords cannot be specified on a single statement.
- If there are any blanks on the input statement, with the exception of blanks embedded within an operand, any characters following the blanks are assumed to be comments.
- Continuation statements are allowed when a keyword’s operands extend beyond one input statement.
- Continuation is allowed only between operands of a keyword and not in the middle of an operand, with the exception of KEYRANGE key operands.
- A nonblank character must be placed in column 72 to continue a statement.
- The first character of a continuation statement must start in column 16.
- Comment-only statements are specified by placing an asterisk in column 1.

Step 1 Utility Control Statement

Partial Reorganization

```
►►──DBNAME=dbdname──┬─KEYRANGE=(lowkeyvalue─┬─,EOD──────────┬─)─┬──────────────►
                    │                       └─,highkeyvalue─┘   │
                    └─FROMAREA=(lowblock#─┬─,EOD────────┬─)─────┘
                                          └─,highblock#─┘

►─TOAREA=(ddname─┬─,EOD───────┬──┬─,EOD────────┬─)──┬──────────────┬──────────►
                 └─,lowblock#─┘  └─,highblock#─┘    └─PSB=psbname──┘

►─┬─────────────────────┬───────────────────────────────────────────────────►◄
  └─SORTOPT=sortoptions─┘
```

DBNAME=
Specifies the database to be partially reorganized. This operand must be the name of a DBD with HD organization.

Requirement:DBNAME is required as the first keyword and must appear only once.

KEYRANGE=
Specifies one range of keys to be reorganized. At least one KEYRANGE must be provided if the database is HIDAM. Up to 10 KEYRANGEs are allowed. The operands are the root segment or generic keys, with a maximum or 255 bytes each. Keys are specified as either character or hexadecimal values. Character keys are the default. Keys can be expressed in hexadecimal by preceding the value with an X and enclosing the value in single quotation marks; for example, KEYRANGE=(X'C8C5E7', X'D2C5E8'). The high key value can be specified as EOD if the upper limit is the end of the database.

The KEYRANGE keyword must be immediately followed by a TOAREA keyword. KEYRANGE is invalid if the database is HDAM.

FROMAREA=
Specifies one range of blocks to be reorganized. At least one FROMAREA must be provided if the database is HDAM. Up to 10 of these keywords are allowed. The operands are block numbers in the root addressable area. The high block number can be specified as EOD if the upper limit is the last block in the root addressable area.

FROMAREA is invalid if the database is HIDAM.

The FROMAREA keyword must be immediately followed by a TOAREA keyword.

TOAREA=
Specifies one area of the database into which reorganized segments must be placed. TOAREA keywords are grouped in sets, with one occurrence per set for each data set group in the database being reorganized. One set must be provided for each KEYRANGE or FROMAREA specified. *ddnames* must be the name of a DD defining a data set in the database being organized.

If both low block number and high block number are specified, the rules are the same as for FROMAREA.

EOD can be the value for either the low block number or the high block number. When EOD is specified as the low block number, the segments are placed beginning at the end of the overflow area. In this case, no high block number is specified. When EOD is specified as the high block number, the TOAREA extends from the low block number to the end of the data set group.

PSB=
Specifies the name of a PSB to be generated in Step 1, to be used in Step 2. You can omit this keyword if a PSB has already been generated by a previous execution of Partial Database Reorganization, and a new PSB is not desired. This keyword can also be omitted if you provide a PSB that exactly follows the PSB example at the end of this section.

SORTOPT=
Specifies options for all sorts. SORTOPT is optional, and it can be specified only once.

Restriction: The character string located in quotes cannot exceed 69 characters.

Only those sort options which you want to use are entered in the operand string. No extra commas are needed for omitted SORTOPT operands. Default options for sorts in Partial Database Reorganization are the normal MVS sort option defaults.

Sort options that you can use with the Partial Database Reorganization utility are:

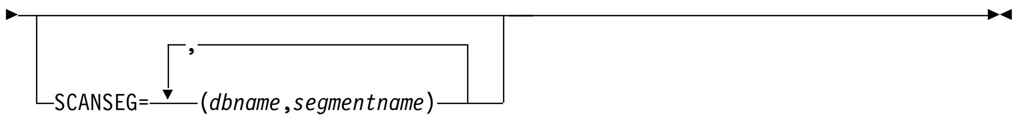
```
                                  ┌─MAX──┐             ┌─MAX──┐
►►──SORTOPT=─────┬─PEER─┬──,SIZE=─┴─XXXX─┴──,CORE=─┴─XXXX─┴──────────────────►
                 ├─BALN─┤
                 ├─OSCL─┤
                 ├─POLY─┤
                 └─CRCX─┘

►─┬─,FLAG=─┬─I─┬─┬─,DIAG──────────────────────────────────────────────────►◄
  │        └─U─┘ │
  └─,MSG=─┬─NO─┬─┘
          ├─CC─┤
          ├─CP─┤
          ├─AC─┤
          ├─AP─┤
          └─PC─┘
```

Related Reading: For a description of all the sort options that can be used with this utility, see *DFSORT Application Programming Guide*, SC33-4035.

Step 2 Utility Control Statement

```
►►──DBNAME=dbname──┬────────────────────────────┬─────────────────────────►
                   └─RESTART=checkpointidnumber─┘
```

Partial Reorganization



DBNAME=
Specifies the database to be partially reorganized. *dbname* must be the same dbname that was specified for Step 1.

RESTART=
Specifies that partial reorganization is to be restarted from the checkpoint named. If RESTART is specified it must appear only once. All other keywords except DBNAME are ignored.

SCANSEG=
Specifies segment types to be included in the scan process. One or more SCANSEG statements can be included in a single execution of Step 2. Only DBD names and segment names listed in the optional scan section of the Step 1 option report can be used as operands. (Segments listed as requiring the scan need not be specified with this statement; they are automatically scanned.)

If other segments in the same database require the scan, including the optional segments for the scan can result in improved performance.

Return Codes

This program provides return codes preceded, in the case of errors, by numbered messages to the SYSPRINT data set that more fully explain the results of program execution. The return codes are as follows:

Code	Meaning
0	All requested operations have completed successfully
4	Warning message issued
8	Severe errors causing job termination have occurred

Related Reading: For a detailed explanation of the numbered messages, see *IMS/ESA Messages and Codes*.

Examples

The examples in this section contain the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1   DD DSNAME=RECON1,DISP=SHR
//RECON2   DD DSNAME=RECON2,DISP=SHR
//RECON3   DD DSNAME=RECON3,DISP=SHR
```

Step 1 Example

The following example shows the JCL and utility control statements to run Step 1 of the Partial Database Reorganization utility.

```
//LOAD1    EXEC  PGM=DFSRRC00,
//         PARM='DLI,DFSDDLT0,PRPSB01L,,,,,,,,,,N,N',
//         REGION=380K
//PRINTDD  DD  SYSOUT=A
//PR01DD   DD  DSN=PR01RW00,DISP=SHR
//PR01IDD  DD  DSN=PR01I,DISP=SHR
//IMS       DD DSN=IMSTESTG.I11X.PSBLIB,DISP=SHR
//          DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//DFSVSAMP DD  *
//SYSIN    DD  *
//         DD  DSN=IMSQA.DBTDATA(PR01DT),DISP=SHR
//PTSTE17  EXEC  PGM=DFSPRCT1,COND=EVEN
//IMS       DD DSN=IMSTESTG.I11X.PSBLIB,DISP=SHR
//          DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A,DCB=(BLKSIZE=1210)
//SYSUDUMP DD  SYSOUT=A
//SYSPUNCH DD  DSN=&&DPSB10,SPACE=(TRK,(2,1)),UNIT=SYSDA,
//          DISP=(,PASS)
//DFSPRCOM DD  DSN=&&DPR10,SPACE=(TRK,(2,1)),UNIT=SYSDA,
//          DISP=(,PASS)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN    DD  *
    KEYRANGE=(000010,000020)
    TOAREA=(PR01DD,1,EOD)
    KEYRANGE=(000030,000040)
    TOAREA=(PR01DD,1,EOD)
    KEYRANGE=(000050,000060)
    TOAREA=(PR01DD,1,EOD)
    KEYRANGE=(000070,000080)
    TOAREA=(PR01DD,1,EOD)
    KEYRANGE=(000090,000100)
    TOAREA=(PR01DD,1,EOD)
    KEYRANGE=(000110,000120)
    TOAREA=(PR01DD,1,EOD)
    KEYRANGE=(000130,000140)
    TOAREA=(PR01DD,1,EOD)
    PSB=PTSTE17
```

The following are sample output reports from Step 1. Figure 15 contains the input statements.

```
          IMS/ESA PARTIAL REORGANIZATION - STEP 1 INPUT STATEMENTS
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
     DBNAME=PR07RW12
     KEYRANGE=(000100,000200)
     TOAREA=(PR07DD,005,007)
     PSB=PTSTN07
```

Figure 15. Partial Database Reorganization Step 1 Input Statements

Figure 16 on page 70 contains the range values.

```
IMS/ESA PARTIAL REORG - RANGE VALUES  FOR DBD - PR01RW00 137/89  PAGE   2
 KEYRANGE = '000100'
         TO '000200'
 TOAREA   =  00000005 TO 00000007       DDNAME = PR07DD
```

Figure 16. Partial Database Reorganization Range Values

Figure 17 contains the names of segments for required scanning.

```
IMS/ESA PARTIAL REORG - REQUIRED SEGMENT SCAN FOR DBD - PR01RW00 137/89 PAGE 3

                 NO SEGMENT FOUND FOR REQUIRED SCAN
```

Figure 17. Partial Database Reorganization Required Segment Scan

Figure 18 contains the names of segments for optional scanning.

```
IMS/ESA PARTIAL REORG - OPTIONAL SEGMENT SCAN FOR DBD - PR01RW00 137/89 PAGE 4

               NO SEGMENT FOUND FOR OPTIONAL SCAN
```

Figure 18. Partial Database Reorganization Optional Segment Scan

PSB Example

The following example shows the PSB source statements generated by Step 1 if the PSB keyword is included in the Step 1 input control statements.

```
PCB    TYPE=DB,NAME=PRO7RW12,KEYLEN=12,PROCOPT=GIR
   SENSEG   NAME=D,PARENT=0
   SENSEG   NAME=F,PARENT=D
PCB    TYPE=DB,NAME=PR071,KEYLEN=6,PROCOPT=G
   SENSEG   NAME=INDEX,PARENT=0
PCB    TYPE=DB,NAME=PRO7R,KEYLEN=12,PROCOPT=GIR
   SENSEG   NAME=A,PARENT=0
   SENSEG   NAME=C,PARENT=A
PCB    TYPE=GSAM,DBDNAME=DFSPRWF2,PROCOPT=L
PCB    TYPE=GSAM,DBDNAME=DFSPRWF3,PROCOPT=L
PCB    TYPE=GSAM,DBDNAME=DFSPRWF4,PROCOPT=L
PCB    TYPE=GSAM,DBDNAME=DFSPRWF5,PROCOPT=L
PSBGEN LANG=COBOL,CMPAT=YES,PSBNAME=PTSTN07
```

The JCL for assembling and link-editing the PSB, which must be executed before Step 2, is shown below:

```
//C      EXEC PGM=IFOX00,REGION=250K,PARM='LOAD,NODECK'
//SYSLIB   DD DSNAME=IMS.MACLIB,DISP=SHR
//SYSGO    DD DISP=(,PASS),UNIT=SYSDA,SPACE=(80,(200,50),RLSE)
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSUT1   DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(1700,(100,50))
//SYSUT2   DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(1700,(100,50))
//SYSUT3   DD UNIT=(SYSDA,SEP=(SYSLIB,SYSUT1,SYSUT2),
//            SPACE=(1700,(100,50))
//SYSIN    DD DSNAME=*.PTSTN07.SYSPUNCH,DISP=(OLD,DELETE)
//L      EXEC PGM=DFSILNK0,REGION=256K
//SYSLIN   DD DSNAME=*.C.SYSGO,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSNAME=IMS.PSBLIB(PTSTN07),DISP=SHR
//SYSUT1   DD DISP=(,DELETE),UNIT=SYSDA,SPACE=(1024,(100,10),RLSE)
```

DBD Examples

The following DBDs must be generated before executing Step 2.

```
DBD  NAME=DFSPRWF2,ACCESS=(GSAM,BSAM)                          *
DATASET  DD1=DFSPRWF2,RECFM=FB,RECORD=18,BLOCK=10
DBDGEN
FINISH
END

DBD  NAME=DFSPRWF3,ACCESS=(GSAM,BSAM)                          *
DATASET  DD1=DFSPRWF3,RECFM=FB,RECORD=18,BLOCK=10
DBDGEN
FINISH
END

DBD  NAME=DFSPRWF4,ACCESS=(GSMA,BSAM)
DATASET  DD1=DFSPRWF4,RECFM=VB,RECORD=1000,BLOCK=1
DBDGEN
FINISH
END

DBD  NAME=DFSPRWF5,ACCESS=(GSAM,BSAM)                          *
DATASET  DD1=DFSPRWF5,RECFM=VB,RECORD=1000,BLOCK=1
DBDGEN
FINISH
END
```

Step 2 Example

The following example shows the JCL and utility control statements required to run Step 2.

```
//STEP2   EXEC PGM=DFSRRC00
//             PARM=(DLI,DFSPRCT2,PTSTN07)
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSPRWF1  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(10,5))
//DFSPRWF2  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1)),
//             DCB=(RECFM=FB,LRECL=18,BLKSIZE=180)
//DFSPRWF3  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//             DCB=(RECFM=FB,LRECL=18,BLKSIZE=180)
//DFSPRWF4  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//             DCB=(RECFM=VB,LRECL=1000,BLKSIZE=1004)
//DFSPRWF5  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//             DCB=(RECFM=VB,LRECL=1000,BLKSIZE=1004)
//DFSPRWF6  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//DFSPRWF7  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//DFSPRWF8  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//DFSPRWF9  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//DFSPRWFA  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(01,1))
//SORTLIB   DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYC,1)
//IMS       DD DSNAME=IMS.PSBLIB,DISP=(SHR)
//          DD DSNAME=IMS.DBDLIB,DISP=SHR,
//             UNIT=SYSDA,VOL=SER=IMSQAW
//IEFRDER   DD DSNAME=IMS.LOG1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=222222
//SYSPRINT  DD SYSOUT=A,DCB=(BLKSIZE=1210)
//DFSPRCOM  DD DSNAME=*.PTSTN07.DFSPRCOM,DISP=(OLD,KEEP)
//SYSUDUMP  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//PR07DD    DD DSNAME=PR07RW12,DISP=OLD
//PR07RDD   DD DSNAME=PR07R,DISP=OLD
//PR07RIDD  DD DSNAME=PR07RI,DISP=OLD
//PR07IDD   DD DSNAME=PR07I,DISP=OLD
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

```
//SYSIN     DD *
   DBNAME=PR07RW12
//DFSVSAMP  DD *
1024,4
4096,8
//DFSCTL    DD *
SBPARM ACTIV=COND
/*
```

The following are sample output reports from Step 2. Figure 19 contains the input statements.

```
          IMS/ESA PARTIAL REORGANIZATION - STEP 2 INPUT STATEMENTS
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
     DBNAME=PR07RW12
```

Figure 19. Partial Database Reorganization Step 2 Input Statements

Figure 20, Figure 21 on page 73, and Figure 22 on page 73 contain unload statistics.

```
IMS/ESA PARTIAL REORG - UNLOAD STATS FOR RANGE 1 FOR DBD - PR07RW12 137/89 PG 2


                             *             SEGMENT STATISTICS              *       RECORD STATISTICS       *

SEGMENT    SEG   DSG   BLOCK   % OF SEG IN  SAME BLK AS:  AVEGRAGE  AVERAGE    AVERAGE   TOTAL     AVG SEG PER
NAME       LVL   NUM   SIZE    PHY-TWIN     PHY-PAR       TWINS     CHILDREN   LENGTH    SEGMENTS  DB RECORD
-------    ---   ---   -----   ----------------------     -------   --------   -------   --------  -----------
D          1     1     4096    79.3         N/A           N/A       2.2        44.0      29        1.0
F          2     1     4096    90.6         92.2          2.2       0.0        36.0      64        2.2

TOTAL SEGMENTS UNLOADED =     93

AVERAGE DATABASE RECORD LENGTH =     123.2

NUMBER OF ROOT ANCHOR POINTS PER BLOCK =     1

KEYRANGE = '000100'
        TO '000200'
```

Figure 20. Partial Database Reorganization Unload Statistics

The fields in the PARTIAL REORGANIZATION - UNLOAD STATISTICS report are:

SEGMENT NAME
The name of the segment type being unloaded/reloaded

SEG LVL
The hierarchic level number of the segment type being unloaded or reloaded

DSG NUM
The data set group number of the segment type being unloaded or reloaded

BLOCK SIZE
Block size of the data set group

% OF SEG IN SAME BLK AS PHY-TWIN
The percentage of segments of this type which occupy the same database block as the previously unloaded or reloaded physical twin segment

% OF SEG IN SAME BLK AS PHY-PAR
The percentage of segments of this type which occupy the same database block as the previously unloaded or reloaded physical parent segment

AVERAGE TWINS
The average number of physical twins within the twin families

AVERAGE CHILDREN
The average number of children of this segment type

AVERAGE LENGTH
The average length of segments of this type

TOTAL SEGMENTS
The total number of segments being unloaded or reloaded

AVE SEG PER DB RECORD
The average number of segments per database record

```
       IMS/ESA PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE 1     FOR DBD - PR07RW12      137/89     PAGE   3

      RANGE OF UNLOADED SEGMENTS

DATA SET       LOW BLOCK   HIGH BLOCK
GROUP NUMBER   NUMBER      NUMBER

2              2           5
```

Figure 21. Range of Unloaded Segments

The fields in the RANGE-OF-UNLOADED-SEGMENTS report are:

RANGE OF UNLOADED SEGMENTS
The actual range of the segments being unloaded. (For an HDAM database, this number is probably different from the range specified.)

DATA SET GROUP NUMBER
The data set group number of the segment type being unloaded or reloaded.

LOW BLOCK NUMBER
The lowest block number of the segments unloaded within the data set group.

HIGH BLOCK NUMBER
The highest block number of the segments unloaded within the data set group.

```
          IMS/ESA PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE 1     FOR DBD - PR07RW12      137/89     PAGE   4

                            DISTRIBUTION OF DATABASE RECORDS

               NUMBER OF    OBSERVED    PERCENT   CUMULATIVE   CUMULATIVE
               BLOCKS       FREQUENCY   TOTAL     PERCENT      REMAINDER
               1            27          93.10     93.10        6.90
               2            2           6.90      100.00       0.00

               MAXIMUM BLOCKS FOR A DATABASE RECORD =   2

               MEAN OBSERVED FREQUENCY             =  1.07
```

Figure 22. Distribution of Database Records

The fields in the DISTRIBUTION-OF-DATABASE RECORDS report are:

DISTRIBUTION OF DATABASE RECORDS
The distribution of the database records unloaded over the number of physical blocks. This report only tabulates 1 through 40 blocks; distributions over 40 blocks are accumulated in one entry.

NUMBER OF BLOCKS
The number of physical blocks occupied by a database record.

OBSERVED FREQUENCY
The number of database records observed for the distribution.

PERCENT TOTAL
The percentage of the database records observed over the total database records unloaded.

CUMULATIVE PERCENT
Total percentage up to this point.

CUMULATIVE REMAINDER
Total percentage remaining up to this point.

MAXIMUM BLOCKS FOR A DATABASE RECORD
The maximum number of blocks occupied by a database record.

MEAN OBSERVED FREQUENCY
The average number of blocks occupied by unloaded database records.

Figure 23 and Figure 24 on page 75 show reload statistics.

```
          IMS/ESA PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE 1     FOR DBD - PR07RW12      137/89     PAGE   5

SEGMENT   SEG   DSG    BLOCK   % OF SEG IN  SAME BLK AS:   RELOAD   DIFFERENCE
NAME      LVL   NUM    SIZE    PHY-TWIN     PHY-PAR        COUNT    RELOAD-UNLOAD
-------   ---   ---    -----   ------------------------    ------   -------------
D           1     1     4096        96.6         N/A         29          0
F           2     1     4096        98.4        98.4         64          0

TOTAL SEGMENTS RELOADED =        93
```

Figure 23. Partial Database Reorganization—Reload Statistics

The fields in Figure 23 are:

SEGMENT NAME
The name of the segment type being unloaded or reloaded

SEG LVL
The hierarchic level number of the segment type being unloaded or reloaded

DSG NUM
The data set group number of the segment type being unloaded or reloaded

BLOCK SIZE
Block size of the data set group

% OF SEG IN SAME BLK AS PHY-TWIN
The percentage of segments of this type which occupy the same database block as the previously unloaded or reloaded physical twin segment

% OF SEG IN SAME BLK AS PHY-PAR
The percentage of segments of this type which occupy the same database block as the previously unloaded or reloaded physical parent segment

RELOAD COUNT
The number of segments reloaded

DIFFERENCE RELOAD-UNLOAD
The difference between the number of segments unloaded and the number reloaded

TOTAL SEGMENTS RELOADED
The total number of segments reloaded for the specific range

```
         IMS/ESA PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE 1     FOR DBD - PR07RW12      137/89     PAGE   6

                  RANGE OF RELOADED SEGMENTS

DATA SET          LOW BLOCK   HIGH BLOCK     BYTE COUNT INSERTED
GROUP NUMBER      NUMBER      NUMBER         TO OVERFLOW

1                 5           6              N/A

DFS3000I SUCCESSFUL COMPLETION OF PARTIAL REORGANIZATION
```

Figure 24. Range of Reloaded Segments

The fields in the RANGE-OF-RELOADED-SEGMENTS report are:

DATA SET GROUP NUMBER
The data set group number of the segment type being reloaded

LOW BLOCK NUMBER
The lowest block number of the segments reloaded within the data set group

HIGH BLOCK NUMBER
The highest block number of the segments reloaded within the data set group

BYTE COUNT INSERTED TO OVERFLOW
The number of bytes inserted into the overflow area (HDAM only)

Figure 25 shows scan statistics.

```
           IMS/ESA PARTIAL REORGANIZATION - UNLOAD STATISTICS FOR RANGE 1     FOR DBD - PR07RW12      137/89     PAGE   7

                 DATABASE NAME    SEGMENT NAME     SCAN COUNT

                 PR05R            D                100

                 TOTAL SEGMENTS SCANNED =          100

DFS3000I SUCCESSFUL COMPLETION OF PARTIAL DATABASE REORGANIZATION
```

Figure 25. Partial Database Reorganization Scan Statistics

The fields in the PARTIAL-REORGANIZATION-SCAN-STATISTICS report are:

DATABASE NAME
The name of the database that was scanned

SEGMENT NAME
The name of the segment type that was scanned

SCAN COUNT
The number of segments scanned for this segment type

Chapter 7. Database Prereorganization Utility (DFSURPR0)

Use the Database Prereorganization utility to create a control data set to be used by the other logical relationship resolution utilities. This utility also indicates which databases and segments, if any, must be scanned by the Database Scan utility.

A flow diagram of the Database Prereorganization utility is shown in Figure 26.

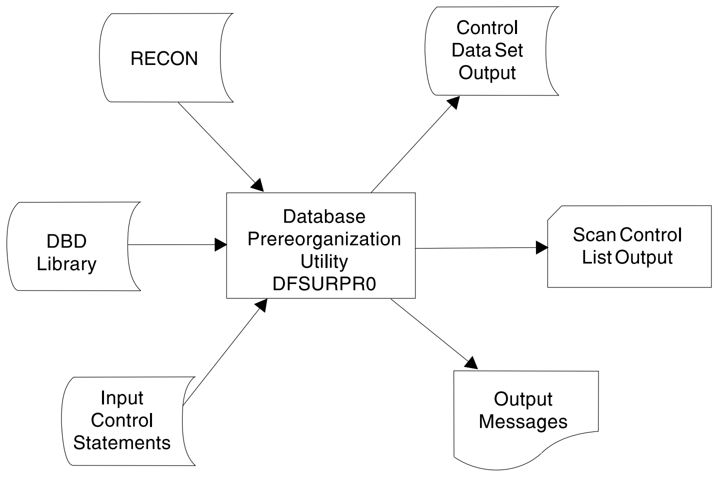


Figure 26. Database Prereorganization Utility

In this Chapter:

- “Restrictions”
- “Input and Output”
- “JCL Requirements” on page 78
- “Utility Control Statements” on page 79
- “Output Messages and Statistics” on page 81
- “Return Codes” on page 82
- “Example” on page 82

Restrictions

If secondary indexes exist when initially loading or reorganizing an indexed database, execute the Database Prereorganization utility against the indexed database to create a control data set used in the creation of a secondary index. You must also execute this utility when databases being loaded, reorganized, or both contain logical relationships.

Input and Output

The input to this utility is a data set that consists of the utility control statements that name the databases being loaded, reorganized or both. The DBDs that are used for the databases named on these statements must define each database 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.

Prereorganization

The output is a control data set that is used by the Database Scan utility, the Database Prefix Resolution utility, and the Database HD Reorganization Reload utility.

JCL Requirements

The Database Prereorganization utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURPR0'
```

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

Related Reading: See Member Name DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring*for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBDs which describe the databases named on the input control statements. The data set must reside on a direct-access device.

This DD statement is required.

SYSIN DD
Contains input control statements. The data set can reside on a a tape, or a direct-access device, or be routed through the input stream.

This DD statement is required.

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 BLKSIZE default, and the results are unpredictable.

SYSPRINT DD
Defines the message output data set. The data set can reside on a printer, a tape, a direct-access device, or be routed through the output stream.

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 BLKSIZE default, and the results are unpredictable.

This DD statement is required.

SYSPUNCH DD
Defines the punch-type output data set. The data set can reside on a printer, a tape, or a 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 BLKSIZE 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 database load time, by the Prefix Resolution utility and the Database Scan utility.

DCB parameters specified within this program are RECFM=FB and LRECL=1600. BLKSIZE must be provided on this DD statement.

This DD statement is required.

SYSABEND DD or SYSUDUMP DD
Define a dump data set. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statements

There are three utility control statements:

DBIL	Names databases that are to be initially loaded or reorganized
DBR	Names databases that are either being converted from DOS/VSE DL/I or being reorganized
OPTIONS	Specifies any optional information to be provided during reorganization or initial load of the database

DBIL Statement

Prereorganization

```
       ┌─,─────────────┐
▶▶──DBIL=─▼─database name─┴──┬──────────┬────────────────────────────▶◀
                             └─comments─┘
```

This utility control statement is used to identify databases that are to be initially loaded or reorganized. This statement is necessary for reorganization when there has been a DBD change affecting logical pointers or counters. You can provide one or more of these statements. The DBIL statement must conform to the following:

- Each DBD name must be left-justified and be a total 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 database is to be initially loaded, list only its DBD name on a DBIL control statement. (Do not list the HIDAM primary index or any secondary index DBD names.)

When structural changes affecting logical relationships are made to a database during a database reorganization process, the following must be performed prior to the HD reload step:

- Assemble and link-edit the new DBD into the IMS DBD library.
- Rerun the Database Prereorganization utility against the new DBD, specifying the database name on the DBIL= statement. You might need to specify DBIL for other logically related databases. See *IMS/ESA Administration Guide: Database Manager*for information that can help you determine whether you need to specify DBIL for other logically related databases. *IMS/ESA Administration Guide: Database Manager*also describes the necessary procedures for adding logical relationships.

Recommendation: Use the DBIL statement if a DBD change affects logical pointers or counters. This option must also be used to correct a pointer error.

DBR Statement

```
      ┌─,─────────────┐
▶▶──DBR=─▼─database name─┴──┬──────────┬─────────────────────────────▶◀
                            └─comments─┘
```

This utility control statement is used to identify databases that are either being converted from DOS/VSE DL/I or being reorganized. One or more of these statements can be provided. Each DBD name must be left-justified and be a total of 8 characters. If the DBD name is less than 8 characters, sufficient trailing blank characters must be provided to make up 8 characters. A blank must follow the last entry on each statement.

If a HISAM database 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 database 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 database is to be reorganized, list only its DBD name on a DBR control statement. Do not list the HIDAM primary index or secondary index DBD names.

DBR uses the old RBA value to resolve logical relationships of the database.

OPTIONS Statement

```
                  ┌─NOPUNCH─┐                             ┌─,ABENDOFF─┐
►►──OPTIONS=─(────┴─────────┴──┬───────┬──┬───────┬───────┴───────────┴──)──────────►◄
                  └─PUNCH───┘  └─,STAT─┘  └─,SUMM─┘       └─,ABEND────┘
```

This utility control statement indicates whether any optional information is to be provided during the reorganization or initial load of the database. These parameters are not positional and can be specified in any order. If more than one parameter is specified, include a comma between the parameters. Information specified on this statement affects output in the execution of the Prereorganization utility (DFSURPR0) and the Prefix Resolution utility (DFSURG10).

PUNCH
Causes the database scan list to be written to both the SYSPUNCH data set and SYSPRINT data set. This output is used as input to the Database Scan utility via the SYSIN data set.

NOPUNCH
Prevents the scan list from being written to the data set defined by the SYSPUNCH DD statement. NOPUNCH is the default.

STAT
Causes the Database Prefix Resolution utility (DFSURG10) to accumulate statistics on segments that are updated.

SUMM
Causes the Database Prefix Resolution utility (DFSURG10) to accumulate and print the number of times the message DFS878I was issued.

ABENDOFF
Turns off the abend function. This is the default. If ABEND is coded, it remains in effect within a job step until ABENDOFF is coded.

ABEND
Terminates with user abend 955, if any condition arises causing abnormal termination of the run. A dump is printed if a SYSABEND or SYSUDUMP DD statement is present.

Output Messages and Statistics

The output messages issued by this utility indicate the database operations that must be performed prior to execution of the Prefix Resolution and the Prefix Update utilities. For example:

- Databases listed after DBIL= in message DFS861I must be initially loaded.

Prereorganization

- Databases listed after DBR= in message DFS861I must be reorganized using the HD Reorganization Unload/Reload utilities. Databases listed after DBS= in message DFS862I must be scanned using the Database Scan utility.

Other outputs created by this utility are:

- A listing of the control statements that were provided as input
- An optional deck of scan control statements for use with the Database Scan utility
- Error messages
- A termination message

The functional identifier for the Database Prereorganization utility is PO.

Return Codes

The following return codes are provided at program termination:

Code	Meaning
0	No errors were detected
8	One or more error messages were issued

Example

This example shows the job control statements and utility control statement required to execute DFSURPRO for two databases that are to be initially loaded. The DBD names for the two databases are PAYR and SKILLINV.

```
//RLUTIL   JOB 1,1,MSGLEVEL=1
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURPR0'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1200
//DFSURCDS  DD DSNAME=IMS.RLCDS,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=IMSMSC,DCB=(BLKSIZE=1600),
//             SPACE=(CYL,1)
//SYSIN     DD *,DCB=BLKSIZE=80
DBIL=PAYRbbbb,SKILLINV
/*
```

Chapter 8. Database Scan Utility (DFSURGS0)

Use the Database Scan utility to scan databases that are not being loaded or reorganized. This utility locates segments containing logical relationships that are affected when other databases are loaded, reorganized, or both. For each segment affected, the utility generates one or more output records, depending on the relationships in which that segment is involved. The records are written to the DFSURWF1 output work data set allocated to this utility.

This utility generates a work data set that is used as one of the inputs to the Prefix Resolution utility.

Figure 27 shows a flow diagram of the Database Scan utility.

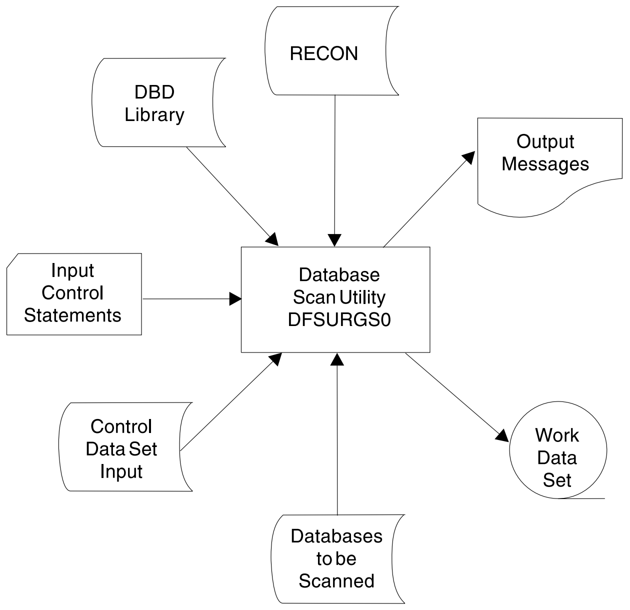


Figure 27. Database Scan Utility

The functions of this utility can be performed by the Utility Control Facility.

Related Reading: Refer to “Chapter 30. Utility Control Facility (DFSUCF00)” on page 277 for a description of its operation.

Recovery and Restart

If execution of the Database Scan utility is abnormally terminated for any reason other than a database I/O error, program execution can be resumed by a restart operation.

If execution of this utility is abnormally terminated because of a database I/O error, do the following:

1. Determine the cause of the error and take the necessary action to correct it.
2. Restore the database as it existed before execution of this utility.
3. Request a restart operation.

JCL Requirements

The Database Scan utility is executed as a standard MVS job. You must provide:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSURGS0'
```

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

Related Reading: See Member Name DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

The Database Scan utility can be passed a buffer size parameter on this statement. The buffer size is most useful if the block size of the database is such that more than 7KB is required to have two blocks in storage. This allows sequential GETs in one block while the second is being read in from a storage device.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBDs that describe the databases to be scanned, plus any logically related databases. The data set must reside on a direct-access device.

This DD statement is required.

SYSIN DD
Defines the input data set for this program. The data set can reside on a tape, or a direct-access device, or be routed through the input stream. This DD statement is only required if 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 specified on this DD statement. If BLKSIZE is not specified, there is no BLKSIZE default, and the results are unpredictable.

SYSPRINT DD
Defines the message output data set. The data set can reside on a printer, a tape, or a direct-access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB, LRECL=120. If BLKSIZE is specified, it must be a multiple of 120.

This DD statement is required.

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, a tape, or a direct-access device, or be routed through the output stream.

DFSURCDS DD
Defines the control data set for this program. It must be the output control data set generated by the Database Prereorganization utility. The data set must reside on either a tape or a direct-access device.

This DD statement is required.

DFSURSRT DD
Defines the data set to be used for restart purposes. This data set is not required if restart is not desired.

For restart processing, ensure that the DFSURSRT DD concatenation is the same as the DFSURWF1 DD concatenation from the previous scan. However, you can begin the DFSURSRT concatenation with the data set that contains the record identified in the `RSTRT` command. Examine the checkpoint information in SYSPRINT from the previous scan execution to determine which data set this is.

If the original DFSURWF1 was a single data set, specify the data set in the DFSURSRT DD statement.

database DD
References the database that is to be scanned as indicated by the Database Prereorganization utility. DD statements must be present for each database. 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 is supplied as one of the inputs to the Prefix Resolution utility, and as the output for the Initial Load and Database Scan utilities. The data set can reside on a tape or a direct-access device.

DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be specified on this DD statement. If BLKSIZE is not specified, there is no BLKSIZE default, and the results are unpredictable.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I Buffer Handler.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The data set can reside on a tape, direct-access device, or it can be routed through the input stream.

This DD statement is required.

DFSCTL DD
Describes the data set containing SBPARM control statements, which request activation of sequential buffering (SB).

Related Reading: For a description of SBPARM control statements, see “SB Parameters (SBPARM) Control Statement” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS and the record length must be 80. The data set can reside on a direct-access device, a tape, or be routed through the input stream. This DD statement is optional.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statements

The Database Scan utility has four utility control statements:

DBS	Names a database segment that is to be scanned by the Database Scan utility
CHKPT	Indicates that checkpoint operations are to be performed during operation of this program
RSTRT	Indicates a restart operation is to be performed by this program
ABEND	Indicates a storage dump is provided if any abnormal condition occurs during execution of this utility

DBS Statement

```
                                  ┌─,SEQ─┐
►►──DBS=database-name,segment-name─┤      ├──────────────────────────────►◄
                                  └─,SEG─┘
```

This utility control statement names a database segment that is to be scanned by the Database Scan utility. One or more of these statements can be provided. The database 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 Database Prereorganization utility is used and only those database names and segment names appearing in the control data set are accepted. (All databases provided on the scan list must be scanned prior to execution of the Prefix Resolution utility.)

If DBS control statements are not provided, the databases to be scanned are:

- Scanned sequentially, using unqualified GN calls for HISAM databases
- Scanned by segment, using GN calls qualified by segment names for HDAM and HIDAM databases

If DBS control statements are provided to the Database 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 are also ignored.

Even if a scan list is provided through DBS control statements, the control data set must still be provided to this utility because it contains other information that is required by the Database Scan utility.

The scan list contained in the SYSPUNCH output data set of the Database Prereorganization utility can be used as input to the Database Scan utility. The value of using the SYSPUNCH output as input to this utility is that, if multiple databases need to be scanned, they can be scanned in parallel with multiple executions of the Database Scan utility. This can be done by separating the SYSPUNCH output (DBS= statements) by database name and submitting scan control statements for each database to different executions of the utility.

The SEQ and SEG options specify the method to be used to scan a database. The SEQ option indicates that a database is to be scanned sequentially by using unqualified GN calls. The SEG option indicates that a database 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 database 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 database, the method specified on the last DBS control statement encountered for that database is used for the entire database. If neither option is specified, the SEQ option defaults for HISAM databases and the SEG option defaults for HDAM and HIDAM databases.

The efficiency of scanning an HD database can be improved by specifying the SEQ option if many segment types are to be scanned. Conversely, the efficiency of scanning a HISAM database 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 database to be scanned. In some cases, you must determine the best method by usage.

CHKPT Statement

```
               ┌─NO────┐
►►──CHKPT=─────┴─nnnnn─┴──────────────────────────────────────────────►◄
```

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 MVS 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. Save the checkpoint messages in case a restart operation is required.

RSTRT Statement

```
          ┌─NO───────────┐
►►──RSTRT=─┴─nnnnn,volser─┴──────────────────────────────────────►◄
```

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 MVS system console.

If the volume specified on this statement is not mounted, FEOVs are issued until the correct volume is made available. The volume specified on this statement must be identified by the DFSURSRT DD statement. The restart module reads records present on the volume identified by this DD statement until the checkpoint record numbered “nnnnn” is encountered.

After each record is read, it is written to the data set identified by the DFSURWF1 DD statement. When the correct checkpoint record is located, a message (DFS378I) is issued to the MVS system console indicating this program name, the checkpoint number, and the volume serial. Because the checkpoint record specified on the “RSTRT” control statement was written to the data set identified by the DFSURWF1 statement, the current volume available to that data set appears in the restart-completed message.

Processing continues from the restart point. The volume containing the specified checkpoint record, and any succeeding volumes that were written during a previous execution of this program, must not be included in the data set supplied to the Prefix Resolution utility (DFSURG10).

ABEND Statement

```
►►──ABEND─────────────────────────────────────────────────────────►◄
```

Include this optional utility control statement when a storage dump is required for diagnostic purposes. If included in the input stream, and if any abnormal condition arises during execution of this utility, any return code greater than zero causes an abend U0955 to be issued. The abend is issued at end of program execution, and a storage dump is provided at that time. If this control statement is included, a SYSUDUMP DD statement is also required.

Output Messages and Statistics

The output messages issued by this utility include DFS339I and DFS340I. The DFS339I message indicates when the scan processing started for the database named on the DBS utility control statement. Message DFS340I indicates that scan processing is completed.

Related Reading: Other output messages issued by this utility denote error conditions that are fully explained in *IMS/ESA Messages and Codes*.

Return Codes

The following return codes are provided at program termination:

Code	Meaning
0	No errors were detected
8	One or more error messages were issued

Example

For the example in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1    DD DSNAME=RECON1,DISP=SHR
//RECON2    DD DSNAME=RECON2,DISP=SHR
//RECON3    DD DSNAME=RECON3,DISP=SHR
```

This example shows the JCL required to scan the database defined by the HDRELTD DD statement. This database is logically related to one or more other databases which the user indicated on either DBIL or DBR control statements supplied to the Database Prereorganization utility. The information in the control data set from the Database Prereorganization utility is used in preference to control statements.

```
//RLUTIL   JOB 1,1,MSGLEVEL=1
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGS0'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1200
//DFSURWF1  DD DSNAME=IMS.URWF1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL),
//             DCB=(LRECL=300,BLKSIZE=1008,RECFM=VB)
//HDRELTD   DD DSNAME=DATABASE.DBRELATD,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0003,
//DFSURCDS  DD DSNAME=IMS.RLCDS,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSMSC
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
```

DD statements must be included for all logically related databases.

Chapter 9. Database Prefix Resolution Utility (DFSURG10)

Use the Database Prefix Resolution utility to accumulate the information generated on work data sets during the load, reorganization, or both of one or more databases. This utility produces an output data set that contains the prefix information needed to complete the logical relationships defined for the databases. Optionally, it produces an output data set containing information needed to create or update secondary index databases. There are no utility control statements for this utility.

Figure 28 is a flow diagram of the Database Prefix Resolution utility.

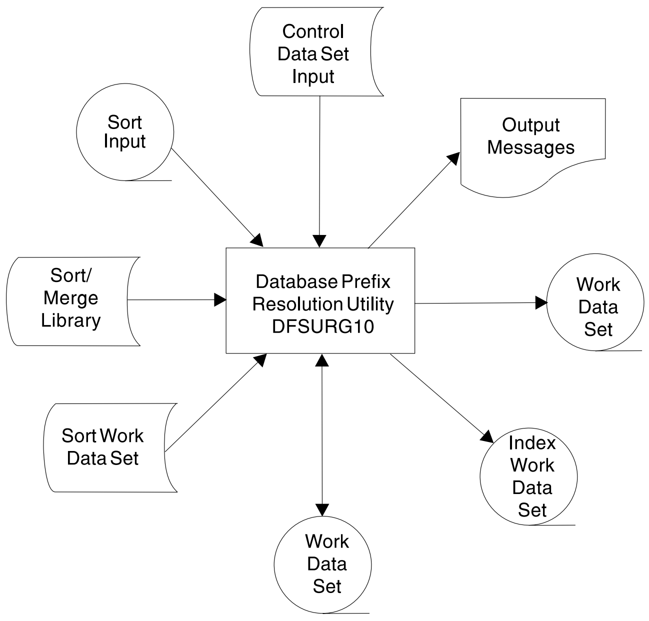


Figure 28. Database Prefix Resolution Utility

The functions of this utility can be performed by the Utility Control Facility.

Related Reading: Refer to “Chapter 30. Utility Control Facility (DFSUCF00)” on page 277 for a description of its operation.

Restrictions

The Database Prefix Resolution utility uses the MVS SORT/MERGE programs. Because the maximum sort field permitted by SORT/MERGE is 256 characters, the following restrictions apply:

- For any given logical parent/logical child pair, the sum of items 1 and 2 below must not exceed 200 characters (the balance of 56 characters is used by IMS for control purposes):
  1. The length of the logical parent’s concatenated key
  2. The length of the sequence field of the logical child as seen by its logical parent
- The sum must be computed once for the logical parent and once for the logical child. These summations are treated separately.

- One or more of the above quantities can be omitted from the summations as described below.
  - 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.
  - The logical child's sequence field length as seen by its logical parent must be included in the logical child's limit check if the logical child is being initially loaded and if it has a logical twin chain. Otherwise, it can be omitted.

    If the above limit check is not satisfied for either a logical parent or a logical child, the user can omit loading of the logical parent or logical child at initial database load time. The logical parent or logical child can then be inserted into the database at a later time by an application program operating in an update mode. Once a database is loaded, one or more of the components of the limit check can be omitted.

The Database Prereorganization utility performs the above limit check for logical parent/logical child combinations affected by an intended database initial load or reload. The limit check is a worst-case check. If the limit check fails for a logical parent/logical child combination, message DFS885 is issued.

Related Reading: Refer to *IMS/ESA Messages and Codes* for an explanation of the message.

IMS makes no assumption of sequence for unkeyed or nonunique keyed segments during initial database 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 might be inconsistent in successive runs of this program or in successive reorganization runs.

JCL Requirements

The Database Prefix Resolution utility is executed as a standard MVS job. You must supply the following:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

This utility attaches the Sort/Merge program. You can use the IMS-supported PARM field options available to the Sort/Merge program with this utility by specifying the desired options in the PARM field of the EXEC statement for this utility.

Related Reading: For a description of the options, refer to *S/360 OS Sort/Merge: Programmer's Guide*.

EXEC Statement

The EXEC statement must be in the form:

```
//STEP EXEC PGM=DFSURG10,REGION=nnnk,PARM='options'
```

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

The PARM field specifications are not positional. The PARM field options and defaults are:

```
►►──EXECPGM=DFSURG10,REGION=nnnk──,PARM='──┬─BALN─┬──┬───────┬──┬─,FILSZ=n──┬──────►
                                           ├─OSCL─┤  └─,CKPT─┘  ├─,SIZE=n───┤
                                           └─POLY─┘             ├─,FILSZ=En─┤
                                                                └─,SIZE=En──┘

         ┌─061440─┐        ┌─AP─┐
►─,CORE=─┼─nnnnnn─┼──,MSG=─┼─AC─┼──┬────────┬──┬─,DUMPWF1=─┬──┬─ALL'─┬────────►◄
         └─MAX────┘        ├─CC─┤  └─,ABEND─┘  ├─,DUMPWF2=─┤  └─n'───┘
                           ├─CP─┤              └─,DUMPWF=──┘
                           └─NO─┘
```

If OSCL is specified, a SIZE parameter must be specified.

If you do not specify the defaults indicated above for CORE and MSG, the Sort/Merge program determines the sort method to be used.

Only the keywords shown above are passed to the attached Sort/Merge program. For the MSG parameter, any valid Sort/Merge option is passed to the Sort/Merge program.

If the CKPT parameter is specified, the Prefix Resolution utility links to the Sort/Merge program instead of attaching, as previously stated.

The value of CORE must be a 6-digit figure and should be calculated based on the type of SORT and SORT devices used.

Specify ABEND 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.

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.

SIZE=n

n is the exact number of records in the input data set, excluding any changes to be made at exit E15. SM1 accepts with FILSZ or SIZE, but FILSZ is always to be preferred when its use is possible, because it allows better optimization.

If the number of records in the input data set is not the same as the value n specified, the program terminates with the value n placed in the IN field of the message IGH047A or IGH054I.

FILSZ=En or SIZE=En

n is the *estimated* number of records to be sorted. The value specified for n must be large enough to include both the input data set and any records added or deleted at exit E15.

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

Prefix Resolution

If the balanced disk technique is being used, the Sort/Merge program prints message IGH070I, which states either:

- The size of the file has not been specified
- The decimal number of records has not been specified

If this operand is omitted, the Sort/Merge program assumes that:

- If intermediate storage is tape, the input data set can be contained on one volume at the blocking factor used by the sort
- If intermediate storage is direct-access, the input data set fits into the space you have allocated

If possible, give an estimated file size, because this greatly improves SM1's optimization and hence performance.

The DUMPWF1, DUMPWF2, and DUMPWF parameters are diagnostic tools to be used only when you need to see the DFSURWF1 and DFSURWF2 workfile records exactly as output from the first or second executions of the SORT.

If DUMPWF1 is included, the specified number of records as produced by the first sort (the sorted DFSURWF1) is printed in SYSPRINT. Specification of DUMPWF2 requests the same service for sorted DFSURWF2. Both DUMPWF1 and DUMPWF2 requests can be included, but, if the same value of n is desired for both, DUMPWF is an equivalent specification. The value specified for *n* can contain up to 9 digits.

DD Statements

STEPLIB DD
Points to the IMS.RESLIB, which contains the IMS nucleus and required action modules.

SYSPRINT DD
Defines the message output data set for this program. The data set can reside on a printer, a tape, or a direct-access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB and LRECL=121. If BLKSIZE is provided on the SYSPRINT DD statement, it must be a multiple of 121.

This DD statement is required.

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, a tape, or a direct-access device, or be routed through the output stream.

SYSOUT DD
Defines the message output data set for SORT/MERGE. The data set can reside on a printer, a tape, or a direct-access device, or be routed through the output stream.

This DD statement is required.

SORTLIB DD
Defines a data set containing load modules for the operating system SORT/MERGE program.

This DD statement is required.

SORTWKnn DD
Defines intermediate storage data sets for the operating system SORT/MERGE program.

Related Reading: Refer to *S/360 OS Sort/Merge: Programmer's Guide* for information regarding specification of number and size of intermediate storage data sets.

This DD statement is required.

SORTIN DD
Defines the input data set for this program. It is referenced by the operating system Sort/Merge program and must conform to its JCL requirements. The data sets referenced by this DD statement must be the output work data sets produced during a database initial load, reload, or scan operation; those work data sets must be concatenated to form the SORTIN data set. If there are multiple data sets with unlike DCB attributes, the data set with the largest LRECL should be first in the concatenation.

This DD statement is required.

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 database load, database reload, or database scan. Ensure that BLKSIZE is the same as that specified on the DFSURWF1 DD statement for those programs. If BLKSIZE is not specified, there is no BLKSIZE default, and the results are unpredictable.

DFSURWF2 DD
Defines an intermediate sort work data set. The data set can reside on a tape or a direct-access device. The size of the data set is approximately the same as that of the input data set defined by the SORTIN DD statement.

DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on this DD statement. If BLKSIZE is not specified, there is no BLKSIZE default, and the results are unpredictable.

This DD statement is required.

DFSURWF3 DD
Defines the output work data set that contains all output data from this program. The output data set defined by this statement is supplied as input to the Prefix Update utility. The data set can reside on a tape or a direct-access device. Its size is 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 BLKSIZE default, and the results are unpredictable.

This DD statement is required.

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 is required.

DFSURIDX DD
Defines an output work data set which is used if secondary indexes are present in the DBDs being reorganized/loaded. This data set must be used as input to the HISAM Unload program (DFSURUL0) for creating, replacing, merging, or extracting secondary indexes (shared or unshared). This DD statement is required only if secondary indexes are present. DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on the DFSURIDX DD statement.

Output Messages and Statistics

If no errors are detected by this program, the only output message issued is a normal program termination message, unless STAT or SUMM is specified in the Database Prereorganization utility control statement. If either is specified, statistics are printed.

Return Codes

The following return codes are provided at program termination:

Code	**Meaning**
0	No errors were detected.
4	Returned when any of the following messages have been issued during program execution or the following messages have been preempted by the SUMM parameter during DB Prereorganization: DFS878, DFS885, DFS961
8	Returned when data required by prefix update is not written to the WF3 data set, or when one or more of the following messages has been issued during program execution: DFS852, DFS855, DFS857, DFS876, DFS877, DFS879, DFS880, DFS881
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 or higher	Returned by the MVS Sort/Merge program.

If either an 8, 12, or 16 return code is provided by the Prefix Resolution utility (DFSURG10), do not run the Prefix Update utility (DFSURGP0) because the input work data set required by DFSURGP0 might not have been completely generated by DFSURG10. Correct the errors indicated by the diagnostic messages and redo the database operations before attempting to run the Database Prefix Resolution utility again.

If return code 4 is provided, a legitimate error might or might not be present.

Related Reading: Refer to *IMS/ESA Messages and Codes* for an explanation of the DFS878 and DFS885 cautionary messages.

Example

The following example uses the Database 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 is used as input to the Prefix Update utility.

DFSURIDX is an output data set where the segments required to build a secondary index using the HISAM Unload/Reload utilities are written.

```
//PREFXRES EXEC PGM=DFSURG10
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A,DCB=BLKSIZE=1200
//SYSOUT   DD SYSOUT=A
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR,VOL=SER=SYSLIB,UNIT=SYSDA
//SORTWK01 DD  UNIT=SYSDA,SPACE=(1008,(60),,CONTIG)
//SORTWK02 DD  UNIT=SYSDA,SPACE=(1008,(60),,CONTIG)
//SORTWK03 DD  UNIT=SYSDA,SPACE=(1008,(60),,CONTIG)
//SORTIN   DD DSN=&&WF1,UNIT=SYSDA,DISP=(MOD,PASS)
//         DD DSN=&&WWF1,UNIT=SYSDA,DISP=(MOD,PASS)
//DFSURWF2 DD DSN=&&WF2,UNIT=SYSDA,SPACE=(1008,(30),,CONTIG),
//         DISP=(,PASS),DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//DFSURWF3 DD DSN=&&WF3,UNIT=SYSDA,SPACE=(1008,(30),,CONTIG),
//         DISP=(,PASS),DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//DFSURCDS DD DSN=*.LDJJK310.PREREORG.DFSURCDS,
//            UNIT=SYSDA,DISP=(OLD,PASS),
//            VOL=REF=*.LDJJK310.PREREORG.DFSURCDS
//DBHVSAM1 DD DSN=DIVNTZ04.JJXXS01K,DISP=SHR
//DBHVSAM2 DD DSN=DIVNTZ04.JJXXS01E,DISP=SHR
//HIDAM    DD DSN=DHONTZ04.JKXXI01O,DISP=SHR
//HIDAM2   DD DSN=DHONTZ04.JKXXI02O,DISP=SHR
//XDLBT04I DD DSN=DXINTZ04.JKXXS01I,DISP=SHR
/*
```

Chapter 10. Database Prefix Update Utility (DFSURGP0)

Use the Database Prefix Update utility to update the prefix of each segment whose prefix information was affected by a database load and/or reorganization. The updating is done using the output generated by the Prefix Resolution utility.

Figure 29 is a flow diagram of the Database Prefix Update utility.

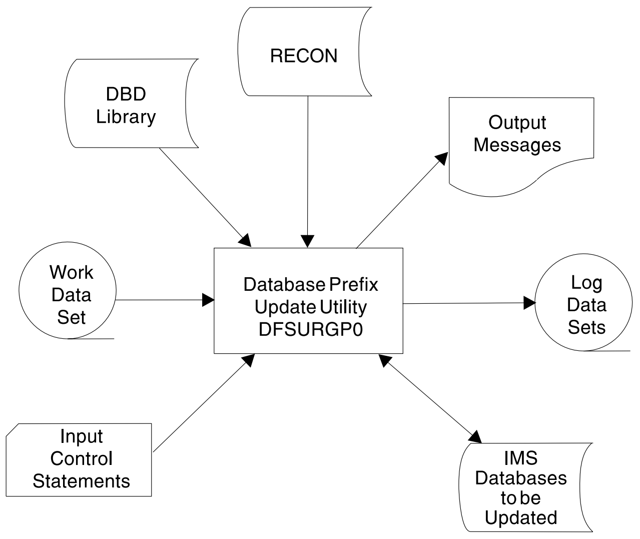


Figure 29. Database Prefix Update Utility

The functions of this utility can be performed by the Utility Control Facility, if required.

Related Reading: Refer to “Chapter 30. Utility Control Facility (DFSUCF00)” on page 277 for a description of its operation.

Output

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 database and segment physical location order by the Prefix Resolution utility. The prefix fields updated by this utility include the logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents.

Log output data sets are optionally created if log (IEFRDER, IEFRDER2) DD statements are included in the JCL. This log output can be used if a later database recovery is required. It is especially useful for the update of scanned databases. However, batch backout cannot be performed using the log output. If DBRC is active when prefix update executes and no log DD statements are supplied, a DBRC NOTIFY.REORG is automatically recorded in RECON for each data set updated.

Recovery and Restart

If execution of this program is abnormally terminated for any reason other than a database I/O error, program execution can be resumed by requesting a restart action or by re-executing the step using the original input work data set.

If execution of this program is abnormally terminated because of a database I/O error, determine the cause of the I/O error and rectify it. Then restore the database as it existed before execution of this program; and reexecute the program, using the original input work data set.

JCL Requirements

The Database Prefix Update utility is executed as a standard MVS job. You must supply:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
//STEP    EXEC PGM=DFSRRC00,PARM='ULU,DFSURGP0'
```

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

Related Reading: See Member Name DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBDs that describe the databases that was loaded and/or reorganized. The data set must reside on a direct-access device.

This DD statement is required.

SYSIN DD
Defines the data set that is to contain input control statements. The data set can reside on a tape, or a 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 BLKSIZE default, and the results are unpredictable.

SYSPRINT DD
Defines the message data set. The data set can reside on a printer, a tape, or a direct-access device, or be routed through the output stream.

This DD statement is required.

DCB parameters supplied by the program are RECFM=FB and LRECL=120. If BLKSIZE is specified, it must be multiple of 120.

SNAPDD
Defines the snap output data set for this program. This statement is required only if the SNAP control statement is specified in the SYSIN data stream. The data set can reside on a printer, a tape, a direct-access device, or be routed through the output stream.

DCB parameters specified within this program are RECFM=FB, LRECL=120. BLKSIZE must be provided on this DD statement and must be a multiple of LRECL. If BLKSIZE is not specified, there is no BLKSIZE 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, a tape, a direct-access device, or be routed through the output stream.

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 is required.

database DD
References the databases 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 database that has logical relationships. The ddname must match the ddname indicated in the DBD. If a HIDAM database is operated upon with this utility, DD statements must be supplied for its primary index database.

This data set must reside on a direct-access device.

IEFRDER DD
Describes the system log data set created during prefix update. The data set usually resides on a tape; however, a direct address volume can be used. This statement is optional. Include it only when log output is desired.

IEFRDER2 DD
Describes the secondary system log data set created during prefix update. The data set usually resides on a tape; however, a direct address volume can be used. This statement is optional. Include it only when dual log output is desired.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I Buffer Handler.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring* .

The data set can reside on a tape, direct-access device, or be routed through the input stream.

This DD statement is required.

DFSCTL DD
Describes the data set containing SBPARM control statements, which request activation of sequential buffering (SB).

Related Reading: For a description of SBPARM control statements, see “SB Parameters (SBPARM) Control Statement” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS and the record length must be 80. The data set can reside on a direct-access device, a tape, or be routed through the input stream.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statements

There are four utility control statements that you can use with the Database Prefix Update utility. They are:

CHKPT	Indicates checkpoint operations are performed during program operation
RSTRT	Indicates restart operation is performed by this program
SNAP	Indicates IMS control blocks are printed to the SNAPDD data set for diagnostic information
ABEND	Indicates a user abend 955 is issued and a storage dump is taken if any abnormal condition occurs while this utility is running

CHKPT Statement

```
               ┌─NO──┐
►►──CHKPT=─────┴─YES─┴─────────────────────────────────────────────────────►◄
```

This utility control statement indicates that checkpoint operations are to be performed during operation of this program. The Prefix Resolution utility automatically writes records on the DFSURWF3 data set as a service for the Prefix Update utility, and these records are used as checkpoints. As each checkpoint record is encountered, a checkpoint message (DFS867) is issued to the MVS 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. Save the checkpoint messages in case a restart operation is required.

RSTRT Statement

```
          ┌─NO───────────┐
►►──RSTRT=─┴─nnnnn,volser─┴──────────────────────────────────────────►◄
```

This control statement indicates that a restart operation is to be performed by this program. *nnnnn* is a 5-digit decimal number and *volser* is the volume serial identifier of the input volume containing the checkpoint record numbered *nnnnn.* The two parameters, *nnnnn* and *volser,* are obtained from the checkpoint messages that were issued to the MVS system console.

If the volume specified on this control statement is not mounted, FEOVs are issued until the correct volume is made available. When restart is completed, a message (DFS378) is issued indicating the checkpoint number, volume serial, and program name. Processing continues from the restart point.

SNAP Statement

```
►►──SNAP=──┬─STATUS─┬────────────────────────────────────────────────►◄
           └─nnnn───┘
```

This optional utility control statement indicates that IMS control blocks are to be printed to the SNAPDD data set for diagnostic information.

The STATUS parameter causes snaps to be taken whenever an abnormal status code is returned by DL/I or an abnormal return code is returned from the buffer handler.

nnnn specifies that a snap is taken after *nnnn* number of calls are made to the buffer handler. As many as 25 SNAP statements with *nnnn* specified are accepted. *nnnn* must be in the range from 1 to 9999999 with no blanks or commas embedded. The significant digits are required, but blanks cannot appear between the equal sign and the first digit of the relative call number.

A one-to-one correlation exists between the buffer handler calls and the records from the DFSURWF3 data set.

ABEND Statement

```
►►──ABEND────────────────────────────────────────────────────────────►◄
```

Include this optional utility control statement when a storage dump is required for diagnostic purposes. If included in the input stream, and if any abnormal condition arises during execution of this utility, any return code greater than zero causes user abend 955 to be issued. The abend is issued at end of program execution, and a storage dump is provided at that time. If this control statement is included, a SYSUDUMP DD statement is also required.

Output Messages and Statistics

If no errors are detected by this program, the only output message issued is a normal program termination message that indicates the number of records processed.

Return Codes

The following return codes are provided at program termination:

Code	**Meaning**
0	No errors were detected.
8	One or more error messages were issued. The messages contain details on the errors and are printed as part of the system output.

Examples

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1  DD  DSN=RECON1,DISP=SHR
//RECON2  DD  DSN=RECON2,DISP=SHR
//RECON3  DD  DSN=RECON3,DISP=SHR
```

Example 1

This example shows the JCL required to execute DFSURGP0 to update the five databases defined by the DBHVSAM1, DBHVSAM2, HIDAM, HIDAM2, and XDLBT04I DD statements.

```
//PREFIXUP EXEC PGM=DFSRRC00,PARM='ULU,DFSURGP0'
//IMS      DD DSN=IMSTESTG.IMS310.I11X.DBDLIB,DISP=SHR
//         DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//IEFRDER  DD DSN=DBRCIMS.LOG3,DISP=(,KEEP),VOL=SER=USER02,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=4096
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//DFSURWF3 DD DSN=&&WF3,UNIT=SYSDA,DISP=(OLD,PASS)
//DBHVSAM1 DD DSN=DIVNTZ04.JJXXS01K,DISP=SHR
//DBHVSAM2 DD DSN=DIVNTZ04.JJXXS01E,DISP=SHR
//HIDAM    DD DSN=DHONTZ04.JKXXI01O,DISP=SHR
//HIDAM2   DD DSN=DHONTZ04.JKXXI02O,DISP=SHR
//XDLBT04I DD DSN=DXINTZ04.JKXXS01I,DISP=SHR
//DFSVSAMP DD *
2048,4
IOBF=(8192,4)
/*
```

Example 2

Figure 30 on page 105 is an example of reorganizing two logically related databases. The two databases are DIVNTZ02 (a HISAM VSAM database) and DHVNTZ02 (a HIDAM VSAM database) with an index VSAM database (DXVNTZ02).

```
//DBREORG2  JOB A
//JOBLIB    DD DSN=IMS.RESLIB,DISP=SHR
//JOBCAT    DD DSN=VCATREC,DISP=SHR
//*
//*******************************************************************
//*  -DBDNAME-      -DDNAME-       -DSNAME-      -ACCESS-
//*   DIVNTZ02      DBHVSAM1       JDSG1RC       HISAM VSAM (PRIME)
//*      "          DBHVSAM2       JDSG1RCO      HISAM VSAM (OVERFLOW)
//*   DHVNTZ02      HIDAM          KDSG1RC       HIDAM VSAM (GROUP1)
//*      "          HIDAM2         KDSG2RC       HIDAM VSAM (GROUP2)
//*   DXVNTZ02      XDLBT04I       KINDXRC       INDEX VSAM
//*******************************************************************
//*
//*******************************************************************
//*   PREREORGANIZATION
//*******************************************************************
//*
//PREREORG EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURPR0,,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,;
//         UNIT=SYSDA,
//         DISP=(,PASS,DELETE),
//         SPACE=(TRK,(10,10),RLSE),
//         DCB=(BLKSIZE=1600)
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//SYSIN    DD *
OPTIONS=(NOPUNCH,STAT,SUMM)
DBR=DIVNTZ02
DBR=DHVNTZ02
/*
//*******************************************************************
//*      UNLOAD THE HIDAM DATABASE - DHVNTZ02
//*******************************************************************
//*
//UNLOAD1  EXEC PGM=DFSRRC00,REGION=1024K,COND=(1,LT),
//         PARM='ULU,DFSURGU0,DHVNTZ02,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,ASN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,;
//         UNIT=SYSDA,DISP=(OLD,PASS)
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//DFSURGU1 DD DSN=&&ULD1A,DISP=(NEW,PASS,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(5,1))
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//*******************************************************************
//*      UNLOAD THE HISAM DATABASE - DIVNTZ02
//*******************************************************************
//*
//UNLOAD2  EXEC PGM=DFSRRC00,REGION=1024K,COND=(1,LT),
//         PARM='ULU,DFSURGU0,DIVNTZ02,,1,,,,,,,,,Y,N'
```

Figure 30. Example of Reorganizing Two Logically Related Databases (Part 1 of 5)

```
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,;
//         UNIT=SYSDA,DISP=(OLD,PASS)
//DBHVSAM1 DD DSN=JDSG1RC,DISP=SHR
//DBHVSAM2 DD DSN=JDSG1RCO,DISP=SHR
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//DFSURGU1 DD DSN=&&ULD1B,DISP=(NEW,PASS,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(5,1))
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
/*
//*******************************************************************
//*      SCRATCH AND REALLOCATE THE VSAM DATABASES
//*******************************************************************
//*
//SCRATCH  EXEC  PGM=IDCAMS,COND=(1,LT)
//SYSPRINT DD SYSOUT=*
//VSAMDD   DD UNIT=SYSDA,DISP=SHR,VOL=SER=RECRES
//SYSIN    DD *
 DELETE  JDSG1RC   PURGE FILE(VSAMDD)
 DELETE  JDSG1RCO  PURGE FILE(VSAMDD)
 DELETE  KDSG1RC   PURGE FILE(VSAMDD)
 DELETE  KDSG2RC   PURGE FILE(VSAMDD)
 DELETE  KINDXRC   PURGE FILE(VSAMDD)
 DEFINE CLUSTER (NAME (JDSG1RC) -
          CYLINDERS (2,1) -
          VOL (RECRES) -
          FREESPACE (30,20) -
          SHAREOPTIONS (3,3) -
          RECSZ (200,200) -
          KEYS (5,6) -
          UNIQUE SPEED) -
        DATA (NAME(JDSG1RC1) -
          CISZ (1024)) -
        INDEX (NAME(JDSG1RC2) -
          CISZ (1024)) -
        CATALOG (VCATREC)
 DEFINE CLUSTER (NAME (JDSG1RCO) -
          CYLINDERS (1,1) -
          VOL (RECRES) -
          SHAREOPTIONS (3,3) -
          RECSZ (200,200) -
          NIXD -
          UNIQUE) -
        DATA (NAME(JDSG1RC3) -
          CISZ (512)) -
        CATALOG (VCATREC)
```

Figure 30. Example of Reorganizing Two Logically Related Databases (Part 2 of 5)

```
 DEFINE CLUSTER (NAME (KDSG1RC) -
          CYLINDERS (2,1) -
          VOL (RECRES) -
          SHAREOPTIONS (3,3) -
          RECSZ (2041,2041) -
          NIXD -
          UNIQUE) -
        DATA (NAME(KDSG1RC1) -
          CISZ (2048)) -
        CATALOG (VCATREC)
 DEFINE CLUSTER (NAME (KDSG2RC) -
          CYLINDERS (1,1) -
          VOL (RECRES) -
          SHAREOPTIONS (3,3) -
          RECSZ (505,505) -
          NIXD -
          UNIQUE) -
        DATA (NAME(KDSG2RC1) -
          CISZ (512)) -
        CATALOG (VCATREC)
 DEFINE CLUSTER (NAME (KINDXRC) -
          CYLINDERS (1,1) -
          VOL (RECRES) -
          FREESPACE (30,20) -
          SHAREOPTIONS (3,3) -
          KEYS (5,5) -
          RECSZ (12,12) -
          SPEED -
          UNIQUE) -
        DATA (NAME(KINDXRC1) -
          CISZ (512)) -
        INDEX (NAME(KINDXRC2) -
          CISZ (1024)) -
        CATALOG (VCATREC)
//*
//*********************************************************************
//*      RELOAD THE HIDAM DATABASE - DHVNTZ02
//*********************************************************************
//*
//RELOAD1  EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURGL0,DHVNTZ02,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSUINPT DD DSN=&&ULD1A,DISP=(OLD,DELETE),UNIT=SYSDA
//DFSURWF1 DD DSN=&&WF1A,UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(2,1)),
//            DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
```

Figure 30. Example of Reorganizing Two Logically Related Databases (Part 3 of 5)

```
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
//*
//*******************************************************************
//*      RELOAD THE HISAM DATABASE - DIVNTZ02
//*******************************************************************
//*
//RELOAD2  EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURGL0,DIVNTZ02,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSUINPT DD DSN=&&ULD1B,DISP=(OLD,DELETE),UNIT=SYSDA
//DFSURWF1 DD DSN=&&WF1B,UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(2,1)),
//            DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//DBHVSAM1 DD DSN=JDSG1RC,DISP=SHR
//DBHVSAM2 DD DSN=JDSG1RCO,DISP=SHR
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
//*
//*******************************************************************
//*     PREFIX RESOLUTION
//*******************************************************************
//*
//PREFRES  EXEC PGM=DFSURG10,REGION=1024K
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSOUT   DD SYSOUT=A
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//SORTIN   DD DSN=&&WF1A,UNIT=SYSDA,DISP=(OLD,DELETE)
//         DD DSN=&&WF1B,UNIT=SYSDA,DISP=(OLD,DELETE)
//DFSURWF2 DD DSN=&&WF2,;
//         UNIT=SYSDA,
//         DISP=(,DELETE),
//         SPACE=(CYL,(2,2)),
//         DCB=BLKSIZE=1008
//DFSURWF3 DD DSN=&&WF3,;
//         DISP=(,PASS),
//         UNIT=SYSDA,
//         SPACE=(CYL,(1,1),RLSE),
//         DCB=(RECFM=VB,LRECL=900,BLKSIZE=13030,BUFNO=8)
```

Figure 30. Example of Reorganizing Two Logically Related Databases (Part 4 of 5)

```
//DFSURCDS DD DSN=&&URCDS,UNIT=SYSDA,DISP=(OLD,PASS)
//DFSURIDX DD DUMMY,DCB=BLKSIZE=1008
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//*
//*******************************************************************
//*     PREFIX UPDATE
//*******************************************************************
//*
//UPDATE   EXEC PGM=DFSRRC00,REGION=1024K,
//         PARM='ULU,DFSURGP0,,,1,,,,,,,,,Y,N'
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSURWF3 DD DSN=&&WF3,;
//         DISP=(OLD,DELETE),
//         UNIT=SYSDA
//DBHVSAM1 DD DSN=JDSG1RC,DISP=SHR
//DBHVSAM2 DD DSN=JDSG1RCO,DISP=SHR
//HIDAM    DD DSN=KDSG1RC,DISP=SHR
//HIDAM2   DD DSN=KDSG2RC,DISP=SHR
//XDLBT04I DD DSN=KINDXRC,DISP=SHR
//RECON1   DD DSN=RNC.RRCON1,DISP=SHR
//RECON2   DD DSN=RNC.RRCON2,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
//
```

Figure 30. Example of Reorganizing Two Logically Related Databases (Part 5 of 5)

Chapter 11. MSDB Maintenance Utility (DBFDBMA0)

Use the MSDB Maintenance utility to create an MVS sequential data set for MSDB initial load and to maintain the MSDBs. This utility is an offline utility that runs in an MVS batch region.

The utility inserts, replaces, deletes, and modifies main storage databases in the MSDBINIT file. All of these actions can be performed on different MSDBs in a single run of the utility.

Figure 31 shows the input and output data sets used by the MSDB Maintenance utility for initializing and altering MSDBs.

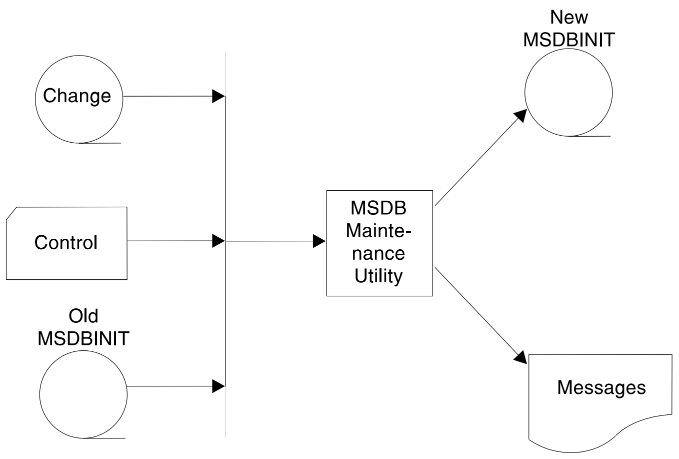


Figure 31. MSDB Maintenance Utility Input and Output Data Sets

Using the MSDB Maintenance Utility

The general action of the MSDB Maintenance utility is as follows:

1. Copy records from the old MSDBINIT file to the new one as long as the MSDB name in the old MSDBINIT record is not equal to the name in the control file.
2. When the MSDB name in the old MSDBINIT file is equal to the name in the control file, perform the actions requested in the control file.
3. Repeat steps 1 and 2 until the input files are exhausted.

Inserting MSDBs

To insert MSDBs:

- You must supply the change data set either in card format or in MSDBINIT format or both, and it must contain data for all segments to be formatted into new MSDBs.
- The control data set must contain ACTION statements, specifying MODE=INSERT for each MSDB to be inserted.
- The old MSDBINIT data set, if present, must not contain a record for any MSDB to be inserted.

The MSDBs are read from the change data set, formatted, and written to the new MSDBINIT data set by the utility.

Replacing MSDBs

To replace MSDBs:

- The change data set must be supplied and can be either in card format or in MSDBINIT format or both, must name MSDBs that exist on the old MSDBINIT data set, and must supply data for all segments.
- The control data set must contain an ACTION statement, specifying MODE=REPLACE for each MSDB to be replaced to appear in the new MSDBs.
- An old MSDBINIT data set must be supplied and must contain at least one record for each MSDB to be replaced.

Except for the MSDBs specified in the change data set, all the MSDBs are copied from the old MSDBINIT and written to the new MSDBINIT. When one of the specified MSDBs is reached, it is read and formatted from the change data set, rather than copied from the old MSDBINIT.

Deleting MSDBs

To delete MSDBs:

- The change data set is not required.
- The control data set must contain an ACTION statement specifying MODE=DELETE for each MSDB to be deleted.
- An old MSDBINIT data set must be supplied.

Except for the MSDBs specified in the change data set, all the MSDBs are copied from the old MSDBINIT and written to the new MSDBINIT. The specified MSDBs are merely read over and ignored.

The three functions described above are used to insert, replace, or delete entire MSDBs. The MODIFY function is used to insert, replace, delete, and alter segments within MSDBs. It can be used to alter one or more fields in specified segments or across a range of segments by key.

Modifying MSDBs

To modify MSDB segments:

- The change data set can be either in card format or MSDBINIT format or both and must supply the MSDB name, key field name, and, optionally, data for the segment to be modified.
- The control statement data set must contain ACTION statements specifying MODE=MODIFY for each MSDB to be modified.
- An old MSDBINIT data set must be supplied and must contain at least one record for each MSDB to be modified.

The utility compares the change record data set with the old MSDBINIT. If a change record with data is supplied for an existent key, the record is modified as specified by the change records. If a change record without data is supplied for an existent key, the record is deleted. If a change record with data is supplied for a nonexistent key, the record is inserted. Any record for which no change record is supplied is not modified.

Restrictions

The MSDB Maintenance utility must be executed in a separate job step.

The program cannot switch from one utility to another within the same job step.

Input and Output

The MSDB Maintenance utility uses a variety of input data sets depending on the particular function being performed; but the primary output from all functions is a sequential data set containing MSDBs. This data set, defined on MSDBOLD and MSDBNEW statement, can become input for a system startup or restart process that requires the MSDBs to be loaded.

The MSDB Maintenance utility uses the following input:

- A change data set in card format or MSDBINIT record format containing MSDB names, record keys, data, and other information as described in "MSDB Change Data Set" on page 116, unless you are only deleting MSDBs.
- A control data set containing a run statement and one or more action statements specifying the function to be performed. See "Utility Control Statements" on page 115.
- An old MSDBINIT data set containing formatted MSDBs produced by a previous execution of the MSDB Maintenance utility, the Dump Recovery utility, or, possibly, a user-written routine, unless you are only inserting new MSDBs.

The MSDB Maintenance utility produces the following output:

- An MSDBINIT data set (MSDBNEW).
- A printed summary of utility action or error statements (MSDBPRT).
- A data set (MSDBPUN) containing input control statements that can be used to update the PROCLIB. The data set contains a statement for each record in the DBFMSDBX member of the IMS.PROCLIB. The statements can be used with the IEBUPDTE utility to update the member in the library.

If the MSDBs are so critical to the Fast Path applications that IMS cannot run without them, you can specify the MSDBABND= parameter in the first statement of the DBFMSDBx member. You must type the ABEND= parameter, without blanks, within columns 1 and 72 of this statement. You can specify one of the following values to control the conditions under which the IMS control region must abend during loading of the MSDBs. Each of the following statements must be added manually to the MSBDPUN data set described in the output of this utility.

The format is:

```
MSDBABND=[Y/C/I/A/B]
```

Y Causes the IMS control region to abend if an error occurs during MSDB loading in system initialization, making all of the MSDBs unusable. Errors include open failures on the MSDBINIT data set and I/O errors, as well as errors in the MSDB definition.

C Causes the IMS control region to abend if an error occurs during the writing of the MSDBs to the MSDBCPn data set in the initial checkpoint after IMS startup.

I Causes the IMS control region to abend if an error occurs during the initial loading MSDBs in system initialization, making one or more of the MSDBs unusable. Errors include open failures on the MSDBINIT data set and I/O errors, as well as errors in the MSDB definition.

A Causes the IMS control region to abend if an error occurs during the MSDB loading in system initialization, as described in MSDBABND=I, or during the writing of MSDBs to the MSDBCPn data set (in the initial checkpoint after IMS startup).

B Causes the IMS control region to abend if an error occurs during the MSDB loading in system initialization, as described in MSDBABND=Y, or during the writing of MSDBs to the MSDBCPn data set (in the initial checkpoint after IMS startup).

JCL Requirements

The following are required to run the MSDB Maintenance utility:

- An EXEC statement
- DD statements defining input and output

EXEC Statement

This statement can either specify a procedure that contains the required JCL be in the form:

```
PGM=DBFDBMA0
```

Requirement: The utility requires at least 512KB of virtual storage.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules.

MSDBACB DD
Defines the IMS ACB library. This data set must reside on a direct-access device.

MSDBOLD DD
Describes the sequential MSDBINIT data set that contains MSDBs from a previous utility run.

MSDBNEW DD
Describes the sequential data set that contains the new MSDBs after the current utility run.

MSDBCTL DD
Describes a data set that contains control statements.

MSDBCCHG DD
Describes a data set that contains change records in statement format. This data set must be sequenced by DBD name and key.

MSDBLCHG DD
Describes a data set that contains change records in MSDBINIT record format. This data set must be sequenced by DBD name and key.

MSDBPRT DD
Describes a message data set for printed output.

MSDBPUN DD
Describes an output data set that contains change statements in IEBUPDTE format for adding or replacing member DBFMSDBX in IMS.PROCLIB.

Utility Control Statements

Requirement: The MSDB Maintenance utility requires two types of control statements called the run statement and the action statement. These are coded in columns 1 through 71.

Continuations are not permitted. Comments can be included as illustrated in "Examples" on page 118.

Run Statement

One run statement is used for the entire utility execution. The format of the run statement is:

```
►►──PROC──=──suffix────────────────────────────────────────────────────►◄
```

PROC=
Specifies the 1-byte suffix for the MSDB start procedure (DBFMSDBx).

Action Statement

One action statement is required for each MSDB to be included in the utility execution. Any MSDB in the input MSDBINIT data set that is not referred to in an action statement is written to the new MSDBINIT data set with no change. Action statement keywords can be separated by commas or blanks. The format of the action statement is:

```
►►──DBN=dbname,MODE=──┬─INSERT──┬──┬────────────────┬──┬───────────────┬──────►
                      ├─REPLACE─┤  └─,INCR=number─┘  └─,FIXED─┬─Y─┬─┘
                      ├─DELETE──┤                             └─N─┘
                      └─MODIFY──┘

►──┬─────────────┬─────────────────────────────────────────────────────►◄
   └─,TRACE=YES─┘
```

DBN=*dbname*
Specifies the same MSDB name that is specified in the DBDGEN.

MODE=
Designates the action against the specified MSDB:

INSERT
Specifies that a new MSDB is to be built from MSDB change records and placed in MSDBNEW. If INSERT is specified, all change records for the MSDB to be inserted must contain segment data, and the old MSDBINIT file (MSDBOLD), if present, must contain no records with the same MSDB name.

REPLACE
Specifies that all records of this MSDB be removed and new records from the change record data set be inserted. If REPLACE is specified, all change records for the affected MSDB must contain segment data.

DELETE
Specifies that the MSDB is to be deleted.

Restriction: If DELETE is specified, a change record cannot appear for the MSDB to be deleted.

MODIFY
Specifies that individual records in the MSDB are to be modified.

The utility compares the change record data set with the old MSDBINIT data set. If a change record with data is supplied for an existent key, the record is modified as specified by the change records. If a change record without data is supplied for an existent key, the record is deleted. If a change record with data is supplied for a nonexistent key, the record is inserted. Any record for which no change record is supplied is not modified.

Restriction:A range of records cannot be inserted using the TO= keyword.

INCR=
Specifies the decimal number of empty segments to be reserved in a dynamic MSDB for future inserts. Change records are not required for a dynamic MSDB. Ensure that this number does not exceed the number of logical terminals defined.

FIXED=
Specifies whether this MSDB is to be page-fixed (Y) or not page-fixed (N). Y is the default.

TRACE=YES
Turns on the MSDB Maintenance utility module entry/exit trace, which is used as a problem determination aid.

Restriction:A record (key) cannot appear more than once for the same MSDB in the change data set.

Dynamic MSDBs can be empty, but other types of MSDBs must have at least one record.

MSDB Change Data Set

The MSDB change data set contains keywords and operands that specify the changes to be applied to an MSDB. The data set contains variable-length records (segments) that are inserted, replace the old records, or modify individual records within an MSDB. The data set does not contain change records for the DELETE operation. Change records can be supplied in either a MSDBINIT record format data set or in a card format data set or both. Records must be in sequence by MSDB and, within each MSDB, by key.

MSDBINIT Record Format

Related Reading: This format is the same as described in the figure “MSDBINIT Record Format” in *IMS/ESA Administration Guide: Database Manager*, except that the field defined as an MSDB segment is optional for the MODIFY operation.

For the MODIFY operation do the following:

- To replace a segment, supply a record with an existent key in the KEY field and the desired segment content in the MSDB segment field.
- To insert a segment, supply a record with a nonexistent key in the KEY field and the desired segment content in the MSDB segment field.
- To delete a segment, supply a record with an existent key in the KEY field and no data in the MSDB segment field.

MSDBINIT Card Format

Each statement can contain data in columns 1 through 71 of 80-byte records. Comments can be used by placing an asterisk in column 1 of each comment statement. Remarks can appear in the same statement as keywords as shown in the following examples.

Restriction: The equal sign (=) is a reserved symbol for keywords and cannot be used in the remarks.

Each statement can contain the following keywords separated by commas or blanks.

```
►►──DBN=name,KEY=──┬─value1─┬──┬──────────────────────┬──┬────────────────────┬──────────►
                   └─value2─┘  └─,TO=──┬─value1─┬─────┘  └─,FIELD=fieldname───┘
                                       └─value2─┘

►──┬───────────────────────┬──────────────────────────────────────────────────────────────►◄
   └─,DATA=──┬─value1─┬────┘
             ├─value2─┤
             ├─value3─┤
             └─NULL───┘
```

DBN=
: Specifies the MSDB name. The name must be the same as the name in the DBDGEN.

KEY=
: Specifies the key of the record to be acted upon. See the description of value1 and value2 under the DATA keyword.

  If your MSDB uses an LTERM name for the segment key (REL=TERM, FIXED, or DYNAMIC specified on the DBDGEN), the only valid value for the field is the LTERM name. This operand must be 8 bytes in length. If the LTERM name is shorter than 8 bytes, pad the name with trailing blanks.

TO=
: Is an optional keyword that enables the manipulation of a range of segment keys within a single statement. The key that TO= refers to must have a higher value than the key that KEY= specifies. TO= is valid only if the action statement specifies MODIFY. See the description of value1 and value2 nder the DATA keyword.

FIELD=
: Is an optional keyword that specifies which fields are to be modified. FIELD= is valid only for INSERT, REPLACE, and MODIFY. Each FIELD= keyword must be followed by its associated DATA= keyword.

DATA=
: Specifies the contents of a field or segment. If a FIELD= keyword precedes DATA=, it specifies the contents of a field. If FIELD= does not precede DATA=, it specifies the entire contents of a segment.

  value1
  : Is a character field within quotation marks. This field can be continued by breaking the field at column 71, inserting a nonblank character in column 72, and continuing the field in column 16 of the next statement.

value2
Is the hexadecimal representation of the character field within quotation marks and is preceded by an 'x'. This field can be continued by breaking the field at column 71, inserting a nonblank character in column 72, and continuing the field in column 16 of the next statement.

value3
Is an arithmetic field supplied as a signed or unsigned decimal number within parentheses instead of quotation marks. The utility converts this number to the format matching the field type. Leading zeros can be omitted.

NULL
Specifies that the entire segment (except for the sequence field) or the specified field is set to a null value.

Character and hexadecimal fields are set to blanks (X'40'). Arithmetic fields are set to a zero value.

If a field is given more than one definition, the arithmetic definition prevails.

Return Codes

The following return codes are provided:

Code	Meaning
0	Utility executed successfully
4	Error—error message printed
8	Unable to open print data set

Examples

The following examples show the JCL necessary to execute the MSDB Maintenance utility. The control data set and the change record data set are used as input.

Example 1

Example 1 creates two new MSDBs.

```
//STEP1   EXEC PGM=DBFDBMA0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605)
//MSDBCTL   DD *
  PROC=M
  DBN=MSDBLM01 MODE=INSERT FIXED=YES
  DBN=MSDBLM06 MODE=INSERT INCR=4 FIXED=NO
/*
//MSDBACB   DD DSNAME=IMS.ACBLIB,DISP=SHR
//MSDBNEW   DD DSNAME=XXXX,DISP=(NEW,CATLG),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBPUN   DD SYSOUT=B
//MSDBCCHG  DD *
   DBN=MSDBLM01
//*                  CREATE TWO NEW MSDBS
//*           BUILD NONTERMINAL-RELATED MSDB LM01
       KEY='0001'                               CHARACTER KEY
               FIELD=FIELDC01 DATA='RP'         CHARACTER DATA
               FIELD=FIELDH01 DATA=X'9999'      HALFWORD DATA (HEX)
```

```
              FIELD=FIELDF01 DATA=X'9FFFFFFFF'    FULLWORD DATA (HEX)
              FIELD=FIELDP01 DATA=(1)            DECIMAL DATA (ARITH)
              FIELD=FIELDP02
                             DATA=X'9C'               HEX
              FIELD=FIELDX03 DATA=NULL           NULL FIELD
       KEY='0002'
              FIELD=FIELDC01 DATA='RP'
              FIELD=FIELDH01 DATA=(-1)           HALFWORD TO ARITH -1
              FIELD=FIELDF01 DATA=(-1)           FULLWORD TO ARITH -1
              FIELD=FIELDP01 DATA=(1)            DECIMAL TO ARITH  +1
//*                           LET FIELDP02 DEFAULT TO ZERO
//*                           LET FIELDP03 DEFAULT TO ZERO
//*                           LET FIELDX03 DEFAULT TO BLANKS
//*          BUILD DYNAMIC TERMINAL RELATED MSDB LM06 WITH ONE UNUSED SEGMENT
   DBN=MSDBLM06
       KEY='LTERM01 '
              FIELD=FIELDSEQ DATA='0001'
              FIELD=FIELDH01 DATA=(0)
              FIELD=FIELDX03 DATA=NULL
       KEY='LTERM02 '
              FIELD=FIELDSEQ DATA='0002'
              FIELD=FIELDH01 DATA=(0)
              FIELD=FIELDX03 DATA=NULL
       KEY='LTERM03 '
              FIELD=FIELDSEQ DATA='0003'
              FIELD=FIELDH01 DATA=(0)
              FIELD=FIELDX03 DATA=NULL
/*
```

Example 2

Example 2 deletes one MSDB, modifies a second as described in the comments, and replaces a third. All unmentioned MSDBs and unmentioned segments in MSDBLM04 are copied from MSDBOLD to MSDBNEW.

```
//STEP2   EXEC PGM=DBFDBMA0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605)
//MSDBCTL   DD *
  DBN=MSDBLM03 MODE=DEL
  DBN=MSDBLM04 MODE=MOD
  DBN=MSDBLM05 MODE=REP
  PROC=Y
//MSDBACB   DD DSNAME=IMS.ACBLIB,DISP=SHR
//MSDBNEW   DD DSNAME=XXXX,DISP=(NEW,CATLG),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBOLD   DD DSNAME=IMS.MSDBLM01(0),DISP=SHR
//MSDBPUN   DD SYSOUT=B
//MSDBCCHG  DD *
   DBN=MSDBLM04
//*             MODIFY THE MSDB BY DELETING ONE SEGMENT, CHANGING
//*             THE TRANSACTION LIMIT IN THE DESIGNATED SEGMENTS,
//*             AND COPYING THE REMAINDER OF THE FIELDS FROM THE
//*             OLD FILE.
       KEY='LTERM01 '                    DELETE THIS SEGMENT
       KEY='LTERM02 '                    MODIFY 1 FIELD IN THIS SEGMENT.
              FIELD=FIELDP03 DATA=(1500) SET MAX TRANSACTION LIMIT
       KEY='LTERM03 ',TO='LTERM09 '      MODIFY 1 FIELD IN EACH SEGMENT
//*                                      WITHIN THE RANGE OF KEYS
              FIELD=FIELDP03 DATA=(750)  SET MAX TRANSACTION LIMIT
   DBN=MSDBLM05
//*           INSERT 2 SEGMENTS, 1 FIELD IN EACH SEGMENT INITIALIZED
//*           ALL OTHER FIELDS ARE SET TO NULL VALUES COPIED
       KEY='LTERM03 '
```

```
               FIELD=FIELDP01 DATA=(-2)   SET TIME ZONE FACTOR
       KEY='LTERM05 '
               FIELD=FIELDP01 DATA=(+1)   SET TIME ZONE FACTOR
/*
```

Example 3

Example 3 merges two MSDB files. The MSDBs on MSDBLCHG are merged with the MSDBs on MSDBOLD.

```
//*            MERGE TWO MSDB FILES TO COMBINE ALL MSDBS INTO
//*            ONE INITIAL LOAD FILE.
//*
//STEP3   EXEC PGM=DBFDBMA0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605)
//MSDBLCHG  DD DSNAME=MSDBUT32,DISP=SHR
//MSDBOLD   DD DSNAME=MSDBUT31,DISP=SHR
//MSDBCTL   DD *
  DBN=MSDBLM04 MODE=INSERT         FROM CHANGE FILE
  DBN=MSDBLM05 MODE=INSERT         FROM CHANGE FILE
//MSDBACB   DD DSNAME=IMS.ACBLIB,DISP=SHR
//MSDBNEW   DD DSNAME=MSDBUT33,DISP=(NEW,CATLG),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBPUN   DD SYSOUT=B
/*
```

Chapter 12. DEDB Initialization Utility (DBFUMIN0)

Use the DEDB Initialization utility to initialize one or more data sets of one or more areas of a DEDB. This utility executes in an MVS batch region. The DEDB must have been previously allocated using IDCAMS. Because only one DEDB can be specified as a member in an ACBLIB DD statement, only one DEDB can be initialized at a time.

After the data sets have been initialized and the DEDB areas have been formatted to the DBDGEN specifications, a user-written program issues INSERT calls to load the data.

Once the data has been loaded, run the Database Image Copy utility. If a system failure occurs before the data is accessed, the image copy is needed for recovery. If an image copy is not taken prior to accessing the data and a failure occurs, the data set must be redefined and the DEDB Initialization utility must be rerun.

Restriction: The Online Database Image Copy utility does not support DEDBs, and there is no facility for recovering the DEDBs until the batch Image Copy has been executed.

Restrictions

The following restrictions apply when using the DEDB Initialization Utility:

- When you want to initialize the multiple area data sets of an area, the area must be registered in the DBRC RECON data set and these area data sets must be in an unavailable status in the RECON data set.
- When there are multiple area data sets of an area, all area data sets must be initialized with one successful execution of DBFUMIN0. To add additional area data sets, use the Data Set Create utility (DBFUMRI0).
- You must execute the Initialization utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.
- This utility cannot be stopped and restarted. It must be rerun from the beginning.
- If SMS-managed storage classes are used for fastpath areas and the define cluster spans the area across multiple volumes, you must allocate the area to a guaranteed space storage class. DBFUMIN0 will not format out any portion of the area that resides on a candidate volume.

Input and Output

The DEDB Initialization utility uses the following input:

- The ACB generated for the specific DEDB (access to the DBD member of ACBLIB is required)
- DBRC RECON data set if the area is registered and is in recovery-needed status
- DBRC RECON data set if the area is registered and the area data set is in unavailable status
- A control data set that specifies which areas are to be initialized

The utility produces the following output:

- A data set that contains output messages and statistics

DEDB Initialization

- Formatted areas
- DBRC RECON data set if the area is registered and is not in recovery-needed status
- DBRC RECON data set if the area is registered and the area data set is in available status

JCL Requirements

The following are required:

- An EXEC statement
- DD statements that specify the inputs and outputs

EXEC Statement

The EXEC statement must be in the form:

```
//STEP    EXEC PGM=DBFUMIN0
```

The DBNAME from the DMCB is used as the password for the DEDB areas.

The EXEC statement for the DBFUMIN0 utility allows you to specify either the DBRC=N parameter or no parameter at all, when the batch subsystem has no DBRC interface. Under these circumstances, the RECONn DD statements are not required. But, if the SYSGEN IMSCTRL macro parameter has DBRC=FORCE, then an error message (DFS0044I DBRC REQUIRED FOR THIS EXECUTION) is displayed.

When DBRC=Y is specified in the EXEC statement, the RECONn DD statements are required, unless dynamic allocation is being used.

When the DBRC parameter in the EXEC statement is not specified, DBRC is set according to the DBRC parameter in the IMSCTRL macro. The default for the batch subsystem in the IMSCTRL macro is NO.

DD Statements

STEPCAT DD
Describes a private VSAM user catalog that is searched first. This DD statement must be included if the defined areas are cataloged in a user catalog.

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules.

ACBLIB DD
Describes the DBD member of the ACBLIB that contains information on the databases to be initialized.

This DD statement is required.

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set. This RECON data set must be included if the area is registered in the DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

SYSPRINT DD
Describes a data set that contains error messages (if errors occur during processing) and statistics (if the utility runs to successful completion).

ddname DD (for DEDBs) or areaname DD
Describes a data set for each area that is to be initialized. If an area is not registered in the RECON data set, the ddname of this statement must be the same as the area name.

If an area is registered in the RECON data set, the ddname of this statement must be the same as the ADS name found in the ADS list of the RECON data set. The area of the ADS must be set as recovery-needed because all ADSs of this area are in unavailable status.

Requirement:A DD statement is required for each area and for each ADS to be initialized.

All data sets must be previously defined in the VSAM catalog.

DISP=OLD, UNIT, and VOL parameters must be specified if the DD statement describes a multivolume data set.

CONTROL DD
Describes the input control statement data set. This data set must have a logical record length of 80 bytes and have fixed-length blocks. See “Utility Control Statement” for an explanation of the required format.

Utility Control Statement

The control statement for the DEDB Initialization utility specifies either the name of the area to be initialized, or specifies that the entire DEDB is to be initialized. This statement must reside in the data set defined by the CONTROL DD statement.

If you are specifying areas, you can only specify one area on a control statement. However, you can specify more than one area by using multiple control statements.

```
►►──AREA=──┬─areaname─┬──────────────────────────────────────────────►◄
           └─ALL──────┘
```

areaname
Specifies the name of the area to be initialized. The area name must be 1 to 8 alphanumeric characters (A-Z, 0-9, #, @, $) long. Only one area name can be specified for each control statement.

ALL
Specifies that all areas in the DEDB are to be initialized. ALL must be specified at the first record in the CONTROL data set. If ALL is specified, the remainder of this control statement and all following control statements are ignored.

Return Codes

The DEDB Initialization utility provides one of the following return codes:

Code	**Meaning**
0	Initialization was successful
4	Error in parameters
8	ACB or DBRC processing error
12	Error in processing data set information
16	Error during format processing
20	SYSPRINT error

Related Reading: See *IMS/ESA Messages and Codes* for explanations of the messages accompanying all nonzero return codes.

Example

The example in this section contains the following comment line above the CONTROL DD statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the example in this section, if you are using DBRC without dynamic allocation,you must add the following DD statements to the sample JCL:

```
//RECON1   DD  DSNAME=RECON1,DISP=SHR
//RECON2   DD  DSNAME=RECON2,DISP=SHR
//RECON3   DD  DSNAME=RECON3,DISP=SHR
```

This example shows sample JCL to initialize a DEDB.

```
//INITDB9  JOB
//*
//*       INITIALIZE DEDB DATABASE       (DATAB01)
//*
//JOBCAT    DD DSNAME=DCLCATLG,DISP=SHR
//*
//TESTIT  EXEC PGM=DBFUMIN0,REGION=800K
//ACBLIB    DD DSNAME=IMS.ACBLIB(DATAB01),DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DB01AR01  DD DSNAME=DB01AR01,DISP=OLD,
//             VOL=SER=VOL111
//DB01AR02  DD DSNAME=DB01AR02,DISP=OLD,
//             VOL=SER=VOL111
//DB01AR03  DD DSNAME=DB01AR03,DISP=OLD,
//             VOL=SER=VOL222
//DB01AR04  DD DSNAME=DB01AR04,DISP=OLD,
//             VOL=SER=VOL222
//DB01AR05  DD DSNAME=DB01AR05,DISP=OLD,
//             VOL=SER=VOL333
//DB01AR06  DD DSNAME=DB01AR06,DISP=OLD,
//             VOL=SER=VOL333
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//CONTROL  DD  *
ALL          ALL AREAS INITIALIZED
/*     END OF DBFUMIN0
```

Chapter 13. DEDB Sequential Dependent Scan Utility (DBFUMSC0)

You can use this utility online to scan and copy sequential dependent segments to a sequential data set. You can then process this data set offline using your own programs. You might do a statistical analysis, for example. This utility runs in the DB utility type dependent region.

With this utility you can:

- Specify a range of segments to be copied. If the length exceeds the block size of the SCANCOPY DD data set minus 8 bytes, the utility terminates with an error message. If an exit routine is not specified, the scan utility takes the default, and the segment contents pass through the specified range unchanged. If a range limit is not specified, all dependent segments are scanned and copied. Range limits are described under “Input and Output”.
- Specify an exit routine, other than the one provided by IMS, which can change the segment content and length.
- Specify a loadmodule which can:
  - Change the sort criteria.
  - Sort for a specific record number and average record size.
  - Use a specific type of workspace for sorting.
  - Specify that sort criteria not be used at all. The version 6.1 format CI subset will be processed without sorting.

Restrictions

You must execute the DEDB Sequential Dependent Scan utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.

This utility fails if an in-doubt segment is encountered and 'INDOUBT' is not specified on a SYSIN control statement.

Related Reading: See *IMS/ESA Customization Guide* for information on how to write an exit routine, and a description of the DEDB Sequential Dependent Scan utility exit routine (DBFUMSE0) that is supplied as the default exit routine.

The DEDB Sequential Dependent Scan utility does not decompress data.

Data containing Version 6.1 format CI sequential segments are allocated to each IMS partner. The stopping location for the V6.1 format CI sequential segments is a CI boundary rather than the end of a specific segment. This ensures that a subsequent utility can specify the same CI and RBA stop point and process exactly the same segments.

Input and Output

For input, the DEDB Sequential Dependent Scan utility uses a data set that contains the input parameters supplied by commands (see “Appendix A. Summary of DEDB Utility Commands” on page 329 for more detail on these commands).

DEDB Scan

There are three ways to specify the range of sequential dependent segments to process. The starting location can be specified using the STARTRBA, STARTROOT, or STARTSEQ command. The stopping location can be specified using the STOPRBA, STOPROOT, or STOPSEQ command. If a range is not specified, the scan starts with the oldest current sequential dependent segment in the area and ends with the newest one.

STARTRBA is used to specify the 4-byte RBA of the segment or the 8-byte combination of cycle number and RBA. The cycle number is returned by the POS call and is the number that DEDB uses as a prefix to the RBA. It produces a value that is used only once within the life of the area.

If the cycle number is specified as a nonzero value, the scan utility checks to see that it matches the current cycle number for the RBA in question. If the cycle number is not the same as the current cycle number for that RBA, or the RBA is not in the range in which sequential dependents are currently being stored, an error message is printed and no data is scanned. If the cycle number is not specified or is specified as zero, the current cycle number for that RBA is used. If the start RBA and the stop RBA are specified without cycle numbers, the stop RBA could actually be lower than the start RBA because of the wraparound from the highest RBA back to the lowest RBA. However, the cycle number for the stop RBA would be higher. If the RBA specified is not the address of the beginning of a segment, the scan starts with the next segment.

If you start the scan with STARTROOT or STARTSEQ, it is assumed that marker segments are used. A marker segment is a special sequential dependent segment that has a unique field value. Marker segments are maintained by running an application program that inserts them as sequential dependents.

A root segment that has an inserted marker segment must be specified to the utility. You do this by specifying the key of the root segment on the STARTROOT command. If the sequential dependent is not the most recently inserted sequential dependent for the root, specify information for the utility to use to build a segment search argument (SSA) to search for the desired segment. You build an SSA by specifying a field name, a comparison operator, and a field value on the STARTSEQ command. The comparison rules are the same as for a normal DEDB SSA. The field value specified must be the same as for a normal DEDB SSA. The field value specified must be the same length as the field in the DEDB. If the root does not exist in the indicated area, or the indicated segment is not found, an error message is printed and no data is scanned.

An example of the use of marker segments is:

- If your user installation does not operate on a 24-hour basis, you might want to process each day’s transactions at the end of the day. Select particular root segments to which special sequential dependent segments are to be added. The segments have a field value that contains the day’s date.
- An application program is run that inserts the segments. These segments are marker segments that mark the end of each day’s processing and can be used to control the scan range. The utility scans on the unique field value (date) of the marker segments.

The three ways to specify the stop RBA are exactly the same as the ways to specify the start RBA, except that STOPRBA, STOPROOT, and STOPSEQ commands are substituted for STARTRBA, STARTROOT, and STARTSEQ. The last segment scanned is the last segment that begins at or before the stop RBA. If the stop RBA is the same as the start RBA, then no more than one segment is scanned. If the stop RBA refers to an earlier segment than the start RBA, an error message is printed and no data is scanned. Start RBA and stop RBA do not have to be specified by the same methods.

The DEDB Sequential Dependent Scan utility can produce the following output:

- A data set that contains a copy of sequential dependent segments (an exit routine can be used to limit this data set to specific segments)
- A data set that contains the in-doubt segments (these segments are not passed to the exit routine)
- A data set that contains output messages and statistics

Recovery and Restart

You cannot restart the DEDB Sequential Dependent Scan utility. If you use the restart (REST) parameter, the utility attempts an initial run from the beginning of the area defined by the restart parameter. You can use the Database Recovery, Database Image Copy, and Database Change Accumulation utilities, and the system logs for recovery purposes.

The Scan utility can be stopped at the end of an area by using the IMS `/STOP REGION` command.

JCL Requirements

The following are required:

- An EXEC statement
- DD statements defining inputs and outputs

The SCAN utility can contain sortwork data definition JCL statements to provide sort work space. The sortwork DD statements can be dynamically allocated by the “SORT” product. The SYSOUT DD statement used by SORT for sysprint can be dynamically allocated by the caller and passed to SORT. The SORTSETUP *exit_routine_name* parameter statement can provide an exit to tailor requirements. The SCAN utility default is to SORT the SDEP segments.

EXEC Statement

The EXEC statement executes the DEDB Sequential Dependent Scan utility. This statement can either specify the FPUTIL procedure which contains the required JCL, or it can be in the form:

```
PGM=DFSRRC00
```

Related Reading: See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on FPUTIL.

DD Statements

STEPLIB DD
Describes the library that contains the DEDB Sequential Dependent Scan utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. This statement is required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

SCANIDT DD
Specifies that in-doubt segments be written to the SCANIDT data set, provided INDOUBT is specified on a SYSIN control statement.

SYSIN DD
Describes the input control data set that contains the utility control statement.

ALLFMNEW
Indicates that all CIs in the area use the format from Version 6 or later versions. If IMS detects any IMS/ESA Version 5 and earlier format CIs, the utility abends. This parameter can be used to identify the existence of any old format CIs.

Note: If SORTSETUP is used, ALLFMNEW should be specified only when it is certain that there are no old-format segments left in the SDEP section

NOSORT
Specifies that the sequential dependent segments are not sorted. These segments are passed directly to the user segment. For areas that currently have a SHARELVL of 0 or 1, SORT is not invoked. If the area has previously been SHARELVL 2 or 3, there may be islands of SDEPs that will be returned out of sequence. NOSORT can be abbreviated with NS or NOS.

NSQCI
Specifies that IMS partners will give up their current SDEP CI and any preallocated CI RBAs during the next commit interval. This option is suitable only if all sharing IMS systems are inserting at a similar rate.

QUITCI
Specifies that all IMS partners will give up their current SDEP CI and any preallocated CI RBAs immediately. This option is useful if there is an IMS partner that processes a low volume of transactions and therefore takes an extended period of time to fill its preallocated CIs. QUITCI forces all in-use but only partially-filled CIs to be written to the ADS and then released, which in turn enables the delete utility to process the entire CI and advance the logical beginning (DMACXVAL) as far as possible, ensuring that sufficient space is available when the SDEPs cycle around. Using the QUITCI parameter guarantees that no new SDEPs will be inserted in the QUITCIs. The next utility run can therefore start from the CI following the last UHWM, assuming that the default STOP is used.

SORTSETUP
Identifies the name of a user-written routine to be called before SORT is invoked, allowing different SORT parameters to be passed instead of the segment timestamp. The DBFUMSC0 source code contains the default sort setup, as well as the sort input and output exits.

STARTIME
Specifies an explicit start time in the format: YYYY.DDD HH:MM:SS ±hh:mm where YYYY=year, DDD=day of year, HH=hour, MM=minute, SS=second, and hh:mm=offset to UTC which, when added to the UTC, gives local time. This start time is changed to storeclock format. The hh:mm=offset field must be provided and the format of this string is fixed, i.e., only one blank space can exist between the SS=second parameter and the hh:mm=offset parameter as shown below:

```
TYPE   SCAN
AREA   <area name>
STARTIME=C'1998.181 12:30:25 +02:00'
```

Note: Each MVS partner must set its clock to the same time as the other partner MVSs in the complex or the conversion to storeclock will not be the same from one MVS partner to another.

STOPTIME
Specifies an explicit stop time in the format: YYYY.DDD HH:MM:SS ±hh:mm where YYYY=year, DDD=day of year, HH=hour, MM=minute, SS=second, and the hh:mm=optional offset to UTC which when added to the UTC gives local time. This stop time is changed to storeclock format. If the offset field is omitted, it is assumed to be the current, local MVS offset.

V5COMP
Indicates that all scan and delete operations follow the rules of IMS/ESA Version 5, including:

- Place START and STOP on a segment boundary, without regard to timestamps.
- Scan starts reading from the segment specified in the START parameter. It does not read any CIs between the DMACXVAL CI and the CI containing the START segment.
- Delete reads only the STOP CI to determine the logical end. It does not read from DMACXVAL CI.
- Sort is invoked when V5COMP is chosen unless NOSORT is specified.

Using V5COMP allows you to produce results from the utilities that are identical to those of IMS/ESA Version 5. The performance is also similar. However, sets of segments scanned and deleted will not be the same with shared SDEPs because of the intermingling of time stamps across the CIs.

Related Reading: See “Appendix A. Summary of DEDB Utility Commands” on page 329 for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

SCANCOPY DD
Describes the scan output data set that contains sequential dependent segments in variable-length, blocked records.

Restriction: This cannot be the SYSOUT data set.

One SCANCOPY data set is produced. It contains output from a scan of either one area or multiple areas.

Example

This example shows the JCL and utility control statements for executing the DEDB Scan utility.

```
//SCAN2   EXEC  FPUTIL,RGN=500K,DBD=DEDBJN03,REST=00
//*
//*       DBD=DBDNAME AS TARGET DATABASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//*
//SCANCOPY  DD DSNAME=SCAN203,DISP=(NEW,PASS,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(NCP=5,BLKSIZE=2048),
//             SPACE=(TRK,5)
//*
//*       SCANIDT DD IS USED TO WRITING INDOUBT SEGMENTS TO
//*       THE SCAN204 DATA SET
//*
//SCANIDT   DD DSNAME=SCAN204,DISP=(NEW,PASS,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(NCP=5,BLKSIZE=2048),
//             SPACE=(TRK,5)
//SYSIN     DD *
*---------------------------------------------------------------*
*         USE THE STOPROOT WITH STOPSEQ TO LIMIT THE            *
*         RANGE OF THE SCAN UP TO THE LAST DAY ACCUMULATION     *
*         OF SEGMENTS.                                          *
*---------------------------------------------------------------*
*
* COMMAND     OPERATOR          COMMENT
*
*                               SET ERROR OPTION
 ERRORACTION  SCANRUN
*                               ONLINE SEQ DEP SCAN UTILITY
 TYPE         SCAN
*                               THE TARGET DATABASE IS
*                               DBDNAME=DEDBJN03
*
*                               THE TARGET DEDB AREA
 AREA         DB3AREA0
*                               THE NO. OF BUFFERS
 STOPROOT=C'R301102A'
*                               STOP ON SEQ DEP SEGMENT - JULIAN DATE
 STOPSEQ      FIELD=SDFLDDAT,OP='<=',VALUE=C'273'
*                               SCAN INDOUBT SEGMENTS
 INDOUBT
*                               THE EXIT PROGRAM NAME IS
 EXIT         EXITLDM3
```

Chapter 14. DEDB Sequential Dependent Delete Utility (DBFUMDL0)

Use the DEDB Sequential Dependent Delete utility online to logically delete sequential dependents within a specified limit of a DEDB area. This utility runs in the DB utility type dependent region. After the dependent segments are deleted, the utility resets the segment boundaries, and the freed space in the area is available for reuse.

Restrictions

You must execute the DEDB Sequential Dependent Delete utility in a separate job step.

The program cannot switch from one utility or one database to another within the same job step.

Input and Output

The input to the DEDB Sequential Dependent Delete utility consists a data set that contains the input parameters supplied by commands.

Related Reading: See "Appendix A. Summary of DEDB Utility Commands" on page 329 for details on these commands.

Sequential dependents must be deleted beginning with the oldest current SDEP so there is no start option parameter.

You can use the `STOPRBA`, `STOPROOT`, and `STOPSEQ` commands to specify the limit of sequential dependent segments to delete. If a dependent segment limit (stop RBA) is not specified, the DEDB Sequential Dependent Delete utility deletes all the current sequential dependents in the area, starting with the oldest segment.

`STOPRBA` is used to stop at the 4-byte RBA of the segment or the 8-byte combination of cycle number and RBA. The DEDB uses the cycle number as a prefix to the RBA, to get a value that is used only once within the life of the area. The address specified for the `STOPRBA` command must fall on an SDEP boundary; otherwise, the utility will abend.

If the cycle number is specified as a nonzero value, the delete utility checks to see that it matches the current cycle number for the RBA in question. If the cycle number is not the same as the current cycle number for that RBA or the RBA is not in the range in which sequential dependents are currently being stored, an error message is printed and no data is deleted. If the cycle number is not specified or is specified as zero, the current cycle number for that RBA is used. If an RBA is specified that is not the address of the beginning of a segment, the deletion starts with the next segment.

If you use the `STOPROOT` and `STOPSEQ` commands to specify the stop RBA, it is assumed that marker segments are used. A marker segment is a special sequential dependent segment that has a unique field value. Marker segments are maintained by running an application program that inserts them as sequential dependents.

DEDB Delete

A root segment that has an inserted marker segment must be specified to the utility. This is done by specifying the key of the root segment on the `STOPROOT` command. If the sequential dependent is not the most recently inserted sequential dependent for the root, specify information for the utility to use to build a segment search argument (SSA) to search for the desired dependent. You build an SSA by specifying a field name, a comparison operator, and a field value on the `STOPSEQ` command. The comparison rules are the same as for a normal DEDB SSA. The field value specified must be the same length as the field in the DEDB. If the root does not exist in the indicated area or the indicated segment is not found, an error message is printed and no data is deleted.

The DEDB Sequential Dependent Delete utility produces the following output:

- An area with freed space
- A data set that contains output messages and statistics

Recovery and Restart

You cannot restart the DEDB Sequential Dependent Delete Utility. If you use the restart (REST) parameter, the utility attempts an initial run from the beginning of the area defined by the restart parameter.

For recovery purposes you can use any or all of the following:

- Database Recovery utility
- Database Image Copy utility
- Database Change Accumulation utility
- The system logs

JCL Requirements

To execute the DEDB Sequential Dependent Delete utility you must provide:

- A JOB statement
- An EXEC statement
- DD statements that specify the inputs and outputs

EXEC Statement

This statement can either specify the FPUTIL procedure which contains the required JCL, or be in the form:

```
PGM=DFSRRC00
```

Related Reading: See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on FPUTIL.

DD Statements

STEPLIB DD
Describes the library that contains the DEDB Sequential Dependent Scan utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. This statement is required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement.

ALLFMNEW
Indicates that all CIs in the area use the format from Version 6 or later versions. If IMS detects any IMS/ESA Version 5 and earlier format CIs, the utility abends. This parameter can be used to identify the existence of any old format CIs.

Note: If SORTSETUP is used, ALLFMNEW should be specified only when it is certain that there are no old-format segments left in the SDEP section

NOSORT
Specifies that the sequential dependent segments are not sorted. These segments are passed directly to the user segment. For areas that currently have a SHARELVL of 0 or 1, SORT is not invoked. If the area has previously been SHARELVL 2 or 3, there may be islands of SDEPs that will be returned out of sequence. NOSORT can be abbreviated with NS or NOS.

NSQCI
Specifies that IMS partners will give up their current SDEP CI and any preallocated CI RBAs during the next commit interval. This option is suitable only if all sharing IMS systems are inserting at a similar rate.

QUITCI
Specifies that all IMS partners will give up their current SDEP CI and any preallocated CI RBAs immediately. This option is useful if there is an IMS partner that processes a low volume of transactions and therefore takes an extended period of time to fill its preallocated CIs. QUITCI forces all in-use but only partially-filled CIs to be written to the ADS and then released, which in turn enables the delete utility to process the entire CI and advance the logical beginning (DMACXVAL) as far as possible, ensuring that sufficient space is available when the SDEPs cycle around. Using the QUITCI parameter guarantees that no new SDEPs will be inserted in the QUITCIs. The next utility run can therefore start from the CI following the last UHWM, assuming that the default STOP is used.

SORTSETUP
Identifies the name of a user-written routine to be called before SORT is invoked, allowing different SORT parameters to be passed instead of the segment timestamp. The DBFUMSC0 source code contains the default sort setup, as well as the sort input and output exits.

STARTIME
Specifies an explicit start time in the format: YYYY.DDD HH:MM:SS ±hh:mm where YYYY=year, DDD=day of year, HH=hour, MM=minute, SS=second, and hh:mm=offset to UTC which, when added to the UTC, gives local time. This start time is changed to storeclock format. The hh:mm=offset field must be provided and the format of this string is fixed, i.e., only one blank space can exist between the SS=second parameter and the hh:mm=offset parameter as shown below:

```
TYPE   SCAN
AREA   <area name>
STARTIME=C'1998.181 12:30:25 +02:00'
```

Note: Each MVS partner must set its clock to the same time as the other partner MVSs in the complex or the conversion to storeclock will not be the same from one MVS partner to another.

STOPTIME
Specifies an explicit stop time in the format: YYYY.DDD HH:MM:SS ±hh:mm where YYYY=year, DDD=day of year, HH=hour, MM=minute, SS=second, and the hh:mm=optional offset to UTC which when added to the UTC gives local time. This stop time is changed to storeclock format. If the offset field is omitted, it is assumed to be the current, local MVS offset.

V5COMP
indicates that all scan and delete operations follow the rules of IMS/ESA Version 5, including:

- START and STOP on a segment boundary, without regard to timestamps.
- Scan starts reading from the segment specified in the START parameter. It does not read any CIs between the DMACXVAL CI and the CI containing the START segment.
- Delete reads only the STOP CI to determine the logical end. It does not read from DMACXVAL CI.
- Sort is invoked when V5COMP is chosen unless NOSORT is specified.

Using V5COMP allows you to produce results from the utilities that are identical to those of IMS/ESA Version 5. The performance is also similar. However, sets of segments scanned and deleted will not be the same with shared SDEPs because of the intermingling of time stamps across the CIs.

Related Reading: See “Appendix A. Summary of DEDB Utility Commands” on page 329 for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

Example

Figure 32 on page 135 shows the JCL for the DEDB Scan utility. Figure 33 on page 135 shows the utility control statements for executing the DEDB Delete utility. When you use the two utilities, the Scan utility retrieves data and the Delete utility removes it.

```
//SCAN    EXEC FPUTIL,RGN=500K,
//           DBD=DEDBJN02,REST=00
//*
//*       DBD=DBDNAME AS TARGET DATABASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//*
//SCANCOPY  DD DSNAME=SCAN202,DISP=(NEW,PASS,DELETE),
//           UNIT=SYSDA,VOL=SER=IMSDCL,
//           DCB=(NCP=5,BLKSIZE=2048),
//           SPACE=(TRK,5)
//SYSIN     DD *
*---------------------------------------------------------------*
*         THE STOPRBA  COMMAND WILL LIMIT THE RANGE OF THE    *
*         SCAN.                                               *
*---------------------------------------------------------------*
*
* COMMAND    OPERATOR          COMMENT
*
*                              SET ERROR OPTION
 ERRORACTION SCANRUN
*                              ONLINE SEQ DEP SCAN UTILITY
 TYPE        SCAN
*                              THE TARGET DATABASE IS
*                              DBDNAME=DEDBJN02
*
*                              THE TARGET DEDB AREA IS
 AREA        DB2AREA1
*                              STOP ON RBA
 STOPRBA     X'29E98' LAST SEQ DEP RBA (102A03-2)
*                              THE EXIT PROGRAM NAME IS
 EXIT       EXITLDM3
*
```

Figure 32. Sample JCL for the DEDB Scan Utility

```
//DELETE EXEC  FPUTIL,RGN=500K,
//       DBD=DEDBJN02,REST=00
//*
//*      DBD=DBDNAME AS TARGET DATABASE FOR THIS UTILITY RUN
//*      REST=RESTART NUMBER FOR THIS RUN
//*
//SYSIN   DD *
*----------------------------------------------------------------*
*           DELETE  DEDB JN02 USING THE 'STOPROOT' TO LIMIT   *
*           THE SCOPE OF THE DELETE.                           *
*----------------------------------------------------------------*
*
* COMMAND    OPERATOR          COMMENT
*
*                              SET ERROR OPTION
 ERRORACTION SCANRUN
                               ONLINE SEQ DEP DELETE UTILITY
 TYPE        DELETE
*                              THE TARGET DATABASE IS
*                              DBDNAME=DEDBJN02
*                              THE TARGET DEDB AREA IS
 AREA         DB2AREA1
*                              STOP ON ROOT SEGMENT KEY
 STOPROOT=C'R211304D'
*                              LAST ROOT WITH A SEQ DEP SEGMENT
```

Figure 33. Sample JCL for the DEDB Delete Utility

Chapter 15. High-Speed DEDB Direct Reorganization Utility (DBFUHDR0)

Use the High-Speed DEDB Direct Reorganization utility (DBFUHDR0) to remove external storage fragmentation and to sequence the root and direct dependent segments in the control intervals (CIs). This utility runs in the DB utility dependent type region.

CIs in the root addressable portion of each area are grouped into units of work (UOWs). The Reorganization utility reorganizes one UOW at a time: each UOW is reorganized in real storage and written back to the original UOW as a single unit of recovery. This use of real storage gives improved performance and reduced logging.

Each UOW being reorganized is held exclusively by the Reorganization utility until it is reorganized and written back to its permanent location. It cannot be accessed by other programs during this period. The standard IMS Fast Path commit process is used to write only those CIs that are changed. This use of storage gives improved performance and reduced logging.

The entire UOW is committed or aborted as a unit, including the independent overflow (IOVF) control and data CIs. The UOW can be recovered in the same fashion as any other online transaction, as necessary.

The keyword for the `TYPE` command used to invoke the Reorganization utility is `REORG`, along with the synonyms `HSR`, `HSREORG`, `DR` and `R`. The following commands are applicable to the Reorganization utility: `AREA`, `BUFNO`, `STARTUOW`, `STOPUOW` and `TYPE`. For the Reorganization utility, `BUFNO` specifies a number of buffer sets, rather than a number of buffers.

Definition:A *buffer set* is a set of buffers large enough to hold a complete UOW. For example, a UOW of 16 CIs would require a buffer set of 16 buffers, each buffer being large enough to hold one CI.

How the Reorganization Utility Uses the BUFNO Command

For the Reorganization utility, `BUFNO` specifies buffer sets rather than buffers. If `BUFNO` is not specified, the default is three buffer sets.

A specification of four or more buffers on the `BUFNO` command allows the utility to use asynchronous read ahead, because two buffer sets are used for root addressable portion input.

Definition:*Asynchronous read ahead* means that the utility can read the next UOW while reorganizing the current UOW. This can result in significant performance benefits if the independent overflow portion of the area is located on a different physical device than the root addressable portion.

If both the IOVF portion and the root addressable portion are on the same physical device, then your performance benefits may not be as significant.

The Reorganization utility dynamically extends the private buffer pool to obtain more buffer sets if waits for buffers are experienced. The number of extensions allowed is equal to the original BUFNO specification. This dynamic extension of the buffer pool allows the utility to maximize performance with a minimum number of buffer sets.

The minimum number of buffer sets required to reorganize a UOW is determined by the number of independent overflow CIs that must be accessed:

- The root addressable portion input UOW
- The reorganized root addressable portion UOW for output
- The independent overflow CIs that are freed or allocated until commit point is reached and the UOW is completely reorganized

The number of buffers required for independent overflow CIs is unpredictable. If enough buffers are not available, the UOW cannot be reorganized. Therefore, the utility uses the dynamic pool extensions to obtain more buffers as needed.

Experience will determine the best BUFNO specification for each area. The most efficient specification is when no extensions are required and an excess of buffer sets are not specified.

Buffers are obtained dynamically from a private buffer pool that is created when the utility begins processing. This buffer pool can be extended as necessary, up to the limit specified on the BUFNO command. Each buffer pool extension is one buffer set.

The buffer pool is permanently page-fixed for performance reasons. Since the buffer sets are permanently page-fixed, real storage is required. If your real storage is limited, the BUFNO specification becomes important. This page-fixing could be a consideration for how many copies of the Reorganization utility you run simultaneously. Although the storage does not affect the number of copies of the utility you can run, running too many can degrade overall system performance.

For each area that is being reorganized, a private buffer pool is created and page-fixed. The amount of real storage used can be significant. For example, an area with UOWs altering 45 CIs, each CI being 16KB in size, using the default BUFNO specification, would require 2160KB of real storage, not counting any buffer pool extensions. The default of three should suffice for all but the most extreme cases. However, for maximum performance, you may have to adjust for your environment.

Related Reading: See “Command Descriptions” on page 330 for more information on how to use the BUFNO command, and how to use it for the other DEDB utilities.

Restrictions

You must execute the High-Speed DEDB Direct Reorganization utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.

Input and Output

The High-Speed DEDB Direct Reorganization utility uses a data set that contains input parameters supplied by DEDB commands as input.

Related Reading: See “Appendix A. Summary of DEDB Utility Commands” on page 329 for more detail on these commands.

Using the `STARTUOW` command, you can reorganize a part of the root addressable portion of the area by specifying at which UOW to start reorganizing. The `STOPUOW` command specifies the last UOW to reorganize. The first UOW in the area is number 0, the second one is number 1, the third one is number 2, and so on. If `STARTUOW` is not specified, the utility starts reorganizing with the first UOW in the area and continues to the specified `STOPUOW`. If `STOPUOW` is not specified, the utility starts at the specified start UOW and processes until the end of the area. If both the start UOW and the stop UOW are specified, the start UOW must have a lower or equal value than the stop UOW. If only one UOW is being reorganized, then the start UOW and the stop UOW must be the same value. If an invalid value for either UOW is specified, an error message is printed and no data is reorganized. The UOW number can be entered in either decimal or hexadecimal format.

The High-Speed DEDB Direct Reorganization utility produces the following output:
- An area that contains logically sequenced UOWs with no storage fragmentation
- A data set that contains output messages and statistics

Statistics are collected during the execution of the utility and are passed back to the user in the SYSPRINT data set. The statistics include:
- UOW reorganization activity
  - Number of UOWs requested to be reorganized
  - Number of UOWs actually reorganized
  - Number of UOWs skipped because anchor points were empty
  - Number of UOWs that failed to be reorganized. The reason for the failure is given for first five failures.
- Private buffer set usage
  - Total buffer sets allocated
  - Number of times the private buffer pool was extended
  - Number of buffer sets used for the root addressable portion input
  - Number of buffer sets used for output (includes reorganized root addressable portion and IOVF data CI usage)
- Space reclamation activity
  - Number of IOVF data CIs freed and the number that were reused
  - Number of IOVF data CIs newly allocated
  - Total number of free IOVF data CIs in the area, and the number of free CIs as a percentage of the total number of CIs

Recovery and Restart

If the Reorganization utility terminates abnormally, the database or area does not need any cleaning up to maintain data integrity. The UOW in process at the time of failure is remembered and the utility restarts at the failure point when next invoked against the area unless another utility is executed against the area in the interim. But, if IMS or MVS terminates abnormally while the utility is executing, the failure point is not remembered.

No cleanup is required after an IMS or system failure during execution of the Reorganization utility. The only requirement is to ensure that the last UOW that was reorganized and committed is completely written back to DASD. This is normal Fast Path REDO processing, and it is done by emergency restart or the forward recovery utility when IMS is being cold started.

High-Speed DEDB Reorganization

For the Reorganization utility, no difference in processing exists between`REST=00` and `REST=01`.

Related Reading: For a description of the `REST=` parameter, refer to the *IMS/ESA Installation Volume 2: System Definition and Tailoring*, “FPUTIL Procedure”.

JCL Requirements

EXEC
Executes the Reorganization utility. This statement can be in the form `PGM=DFSRRC00` or specify the FPUTIL procedure, which contains the required JCL.

Related Reading: See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on the FPUTIL procedure.

DD Statements

STEPLIB DD
Describes the library that contains the Reorganization utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. This statement is required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statements.

Related Reading: For a description of how to write the control commands and operands, see “Appendix A. Summary of DEDB Utility Commands” on page 329.

The DEDB online utilities use QSAM to read the SYSIN data set. The input can be blocked or unblocked, fixed length or variable length. The records are interpreted as lines of input statements or commands, and the characters are in EBCDIC. The records may be between 80 and 120 characters long. If the necessary data fits onto the input line, the records can be shorter than 80 characters.

SYSPRINT DD
Describes the output data set that contains messages and statistics.

Error Processing

Table 4 summarizes the I/O error handling by the Reorganization utility and the area status after an I/O error.

Table 4. Summary of I/O Error Handling by Reorganization utility

Error Type	Action	Area Status
Read error (on either the root addressable portion UOW or on the independent overflow control interval)	UOW fails, statistics are gathered, and utility processing continues if not a serious error	Usable
Write error	Processing continues	Usable

Return codes at program termination:

Code	**Meaning**
0	Utility executed as requested
4	SYSPRINT error or failure to reorganize the requested number or UOWs
8	Error in parameter analysis or errors occurred; see utility output

Error messages accompany all nonzero return codes.

Example

This example shows the JCL and utility control statements for executing the Reorganization utility.

```
//ORGDB01 EXEC FPUTIL,RGN=500K,
//        DBD=DEDBJN01,REST=00
//*
//*       DBD=DBDNAME AS TARGET DATABASE FOR THIS UTILITY RUN
//*       REST=RESTART NUMBER FOR THIS RUN
//*
//SYSIN     DD *
************************************************************************
*          HIGH SPEED DEDB ROOT REORGANIZATION UTILITY
************************************************************************
*
* COMMANDS and OPERATORS     COMMENTS
*
 TYPE HSREORG
*                            SET ERROR OPTION
 ERROR HALT
*                            THE TARGET DATABASE IS
*                            DBDNAME DEDBJN01
*                            THE TARGET AREA IS
 AREA DB1AREA1
 GO
*                            THE TARGET DATABASE IS
*                            DBDNAME DEDBJN01
*                            THE TARGET AREA FOLLOWS
 AREA DB1AREA2
 GO
```

The following example shows the utility’s output:

```
  IMS/FP DEDB UTILITY                                                     PAGE:   1

  *******************************************************************       00055200
  *       ONLINE DEDB REORG   UTILITY.                              *       00055300
  *       REORG AREA DB21AR1                                        *       00055400
  *******************************************************************       00055500
   TYPE HSR                                                                 00055600
  *             THE TARGET DATA BASE IS                                     00055700
  *DBDNAME DEDBJN21                                                         00055800
  *             THE TARGET DATA BASE AREA IS                                00055900
   AREA DB21AR1                                                             00056200
   BUFNO=4
  *ERROR TEST
   GO                                                                       00056300
   AREA DB21AR1 HAS      15 UOW'S.
      EACH UOW HAS  10 CI'S;   9 AP CI'S,   1 DOVF CI'S.
      EACH CI IS   1K, ONE BUFFER SET REQUIRES      10K OF STORAGE
```

```
    INDEPENDENT OVERFLOW HAS       3 OVERFLOW UNITS, A TOTAL OF     347 DATA CI'S
 STATS FOR REORG OF AREA DB21AR1 :
    # OF UOWS REQUESTED TO REORG=      15; LOW UOW       0, HIGH UOW      14
    # OF UOWS ACTUALLY REORG'D=       6; LOW UOW       0, HIGH UOW      14
       # OF UOWS SKIPPED=       9, ALL ANCHOR POINT CI'S WERE EMPTY
    # OF IOVF CI'S FREED BY REORG:     141 -     135 (REUSED) =       6
    # OF IOVF CI'S ALLOCATED:       6 (NEW) +     135 (REUSED) =     141
    # OF FREE IOVF CI'S AFTER REORG=     206, PERCENT FREE=  59
    PRIVATE BUFFER SET INITIAL ALLOCATION=   4, EXTENSION COUNT=   4
       RAP INPUT BUFFER SETS=   2, OUTPUT/IOVF BUFFER SETS=   6
       UOW       6 USED  28 BUFFERS FOR IOVF I/O, THE HIGHEST USAGE
    PERFORMANCE STATS: # OF ASYNCHRONOUS READ AHEAD I/O'S=       7,
                       # OF WAITS FOR UOW LOCKS=       0,
                       # OF WAITS FOR PRIVATE BUFFERS=       3
IMS/FP DEDB UTILITY                                                     PAGE:   2

DFS2657I UTILITY EXECUTED AS REQUESTED
```

Part 2. Backup Utilities

Chapter 16. Database Image Copy Utility (DFSUDMP0)

Use the Database Image Copy utility to create an as-is image copy of a database. The output from the Database Image Copy utility is used as input to the Database Recovery utility.

Related Reading: For other ways to create a copy of an online database, see"Chapter 17. Database Image Copy 2 Utility (DFSUDMT0)" on page 155 or "Chapter 18. Online Database Image Copy Utility (DFSUICP0)" on page 161.

The frequency of creating image copies is determined by your recovery requirements. The minimum requirement is that a copy be created immediately after a database is reorganized, reloaded, or initially loaded. Because database recovery is done on a physical replacement basis, a reloaded data set is not physically the same as it was before unload.

Related Reading: For more information on using the Database Image Copy utility, see *IMS/ESA Operations Guide*.

Figure 34 is a flow diagram of the Database Image Copy utility.

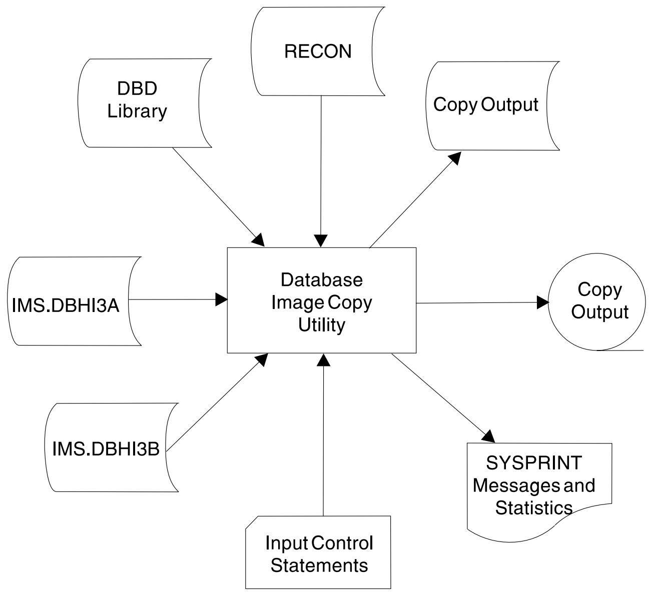


Figure 34. Database Image Copy Utility

The functions of this utility can be performed under control of the Utility Control Facility, if required.

Related Reading: Refer to "Chapter 30. Utility Control Facility (DFSUCF00)" on page 277 for a description of its operation.

Using the Database Image Copy Utility in an RSR Environment

Recommendation: When you run this utility in a Remote Site Recovery (RSR) environment, you should add a global service group (GSG) name to the parameter list on the EXEC statement.

Image Copy

The GSG parameter is `GSGNAME=gsgname`, where gsgname is a 1 to 8-character name. If you do not add the GSGNAME parameter, the GSG name defined during IMS system definition is used.

There are three cases in which you must send database image copies to the tracking site: before database tracking begins, after a time-stamp (partial) recovery for a tracked database at the active site, and after a database reorganization at the active site.

After performing a database reorganization at the active site, you must send an image copy of each reorganized database to the tracking site. It is important that you send these image copies as soon after the database reorganization as possible. If an unplanned takeover should occur before one of these image copies arrives at the tracking site, there will be a delay in getting that database, and all logically related databases, back into production.

If you determine that the image copy for any one of a set of logically related databases will not be available after a remote takeover, all of the logically related databases must be set back to the point before the database reorganization. This requires time-stamp recoveries for any databases that had already had the images copies applied. Thus, all the updates to those databases made at the old active site after the reorganization are discarded.

Some databases are not connected to others by an explicit logical relationship but are related implicitly by the processing done by a particular application. These databases need to be managed manually after a remote takeover, especially if you are applying image copies of them at the tracking site, because RSR does not know about their implicit interrelationships.

Other image copies (not resulting from one of the database reorganization utilities or from time-stamp recovery) should also be sent to the tracking site as soon as possible so that database recoveries can be as efficient as possible. But these image copies are not as important for RSR as the ones created by database reorganizations and time-stamp recoveries.

Restrictions

The following restrictions apply when using the Database Image Copy utility:

- HSAM, GSAM, and MSDB databases cannot be copied with0 this utility.
- The database being copied must not be updated while the Database Image Copy utility is being run. Issue the `/DBR` or `/DBD` command for the database to be copied. Wait for message DFS0488I *before* starting the Database Image Copy utility. This procedure does not apply if you specify CIC on the EXEC statement. (The CIC parameter selects concurrent image copy processing.)
- If updates are made to the database while the Database Image Copy utility is making a copy of a VSAM KSDS, do not rely on the image copy for a successful recovery of the database.
- To use this utility with multiple DEDB area data sets, the area name specified in the control statement must be registered in the DBRC RECON data set.
- Invoke an image copy immediately after running batch jobs that update the database without logging. You can then maintain the integrity of your database if a recovery is required. You can recover using log tapes up to the start of the batch jobs, and then reprocess the batch jobs. However, the resulting database might not be bit-for-bit identical to the database after the previous batch run, although it is logically identical. Batch jobs might not be repeatable; assume that

they are not. If the database is not bit-by-bit identical, log tapes created after the previous execution of the batch jobs would not be valid after the reprocessing. Therefore, the recovery process must not include the sequence of starting with an image copy, applying log tapes, reprocessing unlogged batch executions, then applying more log tapes.

- If a write error occurred for a database that has not been recovered, the recovery utility must be executed before running the Database Image Copy utility. If the database is registered with DBRC, and Database Image Copy uses DBRC, then DBRC knows that a write error occurred without recovery and will fail the Image Copy execution.
- The utility cannot be restarted.

Input and Output

Multiple data sets or areas can be copied on mixed DASD devices with one execution of the Image Copy utility. For the convenience of operations, copy all data sets of a database at the same time. For DEDBs, you can specify multiple area data sets of an area as input to the Image Copy utility if the area is registered in the DBRC RECON data set.

The output from the Database Image Copy utility is used as input to the Database Recovery utility.

You can create one or two output image copies. The advantage of specifying two copies is that if an I/O error occurs during copy execution, the utility continues to completion on the other copy. Performance is somewhat diminished, but a total rerun is not necessary.

Performance can be enhanced by providing additional buffers through the appropriate job control statements. The optimum number of buffers depends on your particular requirements.

Because logical record lengths and blocking factors are calculated at execution time, standard labels must be used on all output copies created by the Image Copy utility.

The first record on the output copy is a dump header record. The header record includes information such as data set identification, creation date, and time. The creation date and time are required by the Database Recovery utility for verification of input.

When a database is stopped and DBDGEN is run to insert and/or delete one or more areas, image copies must be taken for all areas that follow the inserted and/or deleted areas before the database is started again.

When creating an image copy of a shared secondary index, only specify the first DBD. This copies the entire data set.

JCL Requirements

The Database Image Copy utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement

- DD statements that define inputs and outputs

EXEC Statement

This statement can either invoke a cataloged procedure containing the required statements or be in one of the forms below:

```
►►──EXEC──────────────────────────────────────────────────────────────────────►

►──┬─PGM=DFSUDMP0,PARM=──┬─'DBRC=Y,─┬─────┬─┬────────────────┬─'─┬─┬────────►◄
   │                     │          └─CIC─┘ └─GSGNAME=gsgname─┘   │ │
   │                     └─'DBRC=N'─────────────────────────────────┘ │
   └─PGM=DFSRRC00,PARM=─┬─'ULU,DFSUDMP0'─────────────────────────────┘
                        └─'UDR,DFSUDMP0,dbdname'─┘
```

In the first form, the Image Copy utility is executed independently of the IMS region controller. This is the normal execution mode. The CIC parameter is used to select concurrent processing for copying OSAM and VSAM ESDS database data sets.

If you use the first form, PARM='DBRC=' can be specified as Y or N to override the specification of DBRC= on the IMSCTRL macro statement made during IMS system definition. DBRC is used during the execution of this utility if DBRC=Y was specified on the IMSCTRL macro statement during IMS system definition, unless overridden by the DBRC=N on this utility's EXEC statement. DBRC=N means that DBRC is not used for this execution of this utility and DBRC should not be used to generate the JCL. If you specify CIC, you must specify DBRC=Y.

The DBRC sharelevel must be specified in order to run CIC. You can use sharelevel 1, 2, or 3.

The DBRC sharelevel is specified on the `INIT.DB` command.

The access level is set during initialization of the database during system generation using the DATABASE macro.

Restrictions:
- CIC is not supported for VSAM KSDS database data sets.
- CIC cannot be specified for databases or areas registered with sharelevel 0.
- CIC cannot be run against a database with an access level of EX (EXCLUSIVE).
- CIC cannot be run against a nonrecoverable database. If a CIC of a nonrecoverable database is attempted, it fails with the message DSP0090I.
- | Sysplex timer is required ifusing CIC and block level data sharing.

Related Reading: For information on the `INIT.DB` command, see *IMS/ESA DBRC Guide and Reference* .

For information on changing the access level of a database, see *IMS/ESA Operations Guide*.

`GSGNAME` is a 1 to 8-character optional parameter to identify the global service group. See "Using the Database Image Copy Utility in an RSR Environment" on page 145 for more details.

If DBRC=N was specified during IMS system definition, DBRC is not used during the execution of this utility unless overridden by DBRC=Y on this utility's EXEC parameter. Specification of DBRC=Y means that DBRC is used for this execution of this utility.

If DBRC=FORCE was specified during IMS system definition, it cannot be overridden by the DBRC= parameter on the EXEC statement of this utility. DBRC is always used during the execution of this utility. If you attempt to override DBRC=FORCE, message DFS044I is issued and a nonzero return code is returned.

The second and third forms are maintained for upward compatibility. In these forms, the Image Copy utility is executed using the Region Controller program (DFSRRC00).

ULU
Specifies a load/unload region.

UDR
Specifies a recovery region. When PARM=UDR is specified, a valid dbdname is required, but is ignored by the Image Copy utility.

If either of these two forms is used, the normal IMS positional parameters can follow.

Related Reading: See member names DLIBATCH and DBBBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional parameters that can be specified in executing a batch processing region.

If you changed DBRC parameters through a control blocks generation, but have not relinked the IMS nucleus, then the two forms of execution are different. If executed independently of the IMS region controller, the new values are loaded from their control blocks module out of IMS.RESLIB. If executed as an IMS region, the old values within the IMS nucleus are used.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBD that describes the database to be dumped. This is usually DSNAME=IMS.DBDLIB. The data set must reside on a direct access volume.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct access volume, or printer, or be routed through the output stream (SYSOUT).

SYSIN DD
Defines the input control statement data set. The data set can reside on a tape, direct-access volume, or be routed through the input stream (DD * or DD DATA).

Datain DD
Defines the input data set to be dumped. The ddname on this statement must be the same as the name in the DBD that describes this data set; the ddname must also appear on the utility control statement. One DD statement of this type must be present for each data set to be dumped. The data set must reside on a direct access volume. If this is a VSAM data set, performance improves when the AMP parameter is used to specify additional VSAM buffers. For OSAM, DCB=BUFNO=n can be specified, where n is the number of blocks/CI per track. The minimum block size for the data set is 69; smaller data sets are padded with blanks.

Areain DD
For multiple DEDB area data sets, up to seven areain DD statements can be specified. If the area is registered in the RECON data set, the ddname specified in each areain DD statement must not be the area name but must match the names registered in the ADS list of the target area. If the area is not registered, the ddname specified in the areain DD statement must be the area name (ddname operand in the DBD area macro).

Dataout1 DD
Defines the first copy of the dumped output data set. One DD statement is required for each data set to be dumped. The ddname can be any 1- to 8-character string, but the ddname must appear in the associated utility control statement. The output device must be either direct-access or tape. Standard labels must be used. The BLKSIZE used is the largest multiple of the logical record length that does not exceed the maximum BLKSIZE. If a BLKSIZE is specified in the JCL, that BLKSIZE is considered the maximum. For devices other than the 3380, the default maximum BLKSIZE is the BLKSIZE that was specified as the maximum for that device in the MVS I/O generation. For 3380s, the maximum BLKSIZE is 23KB; if the logical record exceeds 23KB, the maximum is 32KB.

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 a direct-access device. Standard labels must be used. The default for BLKSIZE is the maximum capacity of the output device.

If either of the two output copies has open problems (message DFS301A) or fails the first PUT to either output data set (message DFS319A), the current control statement is terminated and the next control statement is processed.

Once the utility has proceeded beyond the first PUT, all I/O errors to either output data set have an RC=08, but the utility continues to copy to the remaining output data set. Each image copy control statement is treated as an independent copy, with the final return code being the highest received for the job.

Recon1 DD
Defines the first DBRC RECON data set. This Recon1 data set must be the same Recon1 data set that the control region is using.

Recon2 DD
Defines the second DBRC RECON data set. This recon2 data set must be the same recon2 data set that the control region is using.

Recon3 DD
Defines the third DBRC RECON data set. This recon3 data set must be the same recon3 data set that the control region is using.

Do not use these RECON data set ddnames if you have specified dynamic allocation using the DFSMDA macro.

Utility Control Statement

The utility control statement for the Database Image Copy utility has a fixed format using the positions described below:

Position	**Description**
1	Field id This must be the character D. The D identifies the statement as a Database Image Copy data set utility control statement.
2	Number of copies This must be a 1 or a 2, depending on the number of copies required.
3	Blank or the character I If coded I, an image copy of an index of a KSDS is requested and position 13 must reference the KSDS ddname. The I option is not valid for OSAM data sets. If position 3 is blank, and if position 13 specifies the ddname for the KSDS, an image copy of the KSDS is obtained. Position 3 must be blank. Image copy and recovery of an imbedded index of a KSDS are not possible. However, a normal full recovery of the KSDS rebuilds an embedded index and the KSDS data area.
4-11	dbdname This must be the name of the physical DBD that includes the name of the data set to be dumped.
13-20	INPUT ddname This must be the ddname of the input data set or area name to be dumped. It must appear in the referenced DBD, and a corresponding DD statement must have been provided.
22-29	OUTPUT ddname This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.
31-38	COPY ddname 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.
40-80	Comments can be placed in positions 40 through 80.

Return Codes

The Database Image Copy utility provides the following return codes:

Code	**Meaning**
0	All operations completed successfully
4	Warning messages were issued

8	One or more operations were not successful
16	Severe errors caused the job to terminate before completing all operations

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

Examples

The examples in this section contain the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1   DD  DSNAME=RECON1,DISP=SHR
//RECON2   DD  DSNAME=RECON2,DISP=SHR
//RECON3   DD  DSNAME=RECON3,DISP=SHR
```

Example 1

In this example, the data set with the ddname DBHI3A is to be copied from the database named DI32DB01. The output data set ddname is DBAOUT1.

```
//DBDUMP   JOB
//*
//STEP1   EXEC PGM=DFSUDMP0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DBHI3A    DD DSNAME=IMS.DBHI3A,DISP=SHR
//DBAOUT1   DD DSNAME=IMS.DBAOUT1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP1,LABEL=(,SL)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
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 database named DI32DB01. Two copies of the data set DBHI3A are to be created.

```
//DBDUMP   JOB
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DBHI3A    DD DSNAME=IMS.DBHI3A,DISP=SHR
//DBHI3B    DD DSNAME=IMS.DBHI3B,DISP=SHR
//DBAOUT1   DD DSNAME=IMS.DBAOUT1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP1,LABEL=(,SL)
//DBAOUT2   DD DSNAME=IMS.DBAOUT2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP2,LABEL=(,SL)
//DBBOUT1   DD DSNAME=IMS.DBBOUT1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP3,LABEL=(,SL)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

```
//SYSIN     DD *
D2 DI32DB01 DBHI3A   DBAOUT1  DBAOUT2  DATA SET 1-DUMP 1+2
D1 DI32DB01 DBHI3B   DBBOUT1           DATA SET 2-DUMP 1
```

Example 3

In this example, the area with the area name AREANAM1 is to be copied from the database named DI32DB01. The ddnames and dsnames in the ADS list for the area are DDNAME1, DDNAME2, DDNAME3 and DSNAME1, DSNAME2, DSNAME3.

The output data set ddname is DBAOUT1.

```
//DBDUMP   JOB
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DDNAME1   DD DSNAME=DSNAME1,DISP=SHR
//DDNAME2   DD DSNAME=DSNAME2,DISP=SHR
//DDNAME3   DD DSNAME=DSNAME3,DISP=SHR
//DBAOUT1   DD DSNAME=IMS.DBAOUT1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP1,LABEL=(,SL)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
D1 DI32DB01 AREANAM1 DBAOUT1           DUMP DUAL DEDB AREA
```

Chapter 17. Database Image Copy 2 Utility (DFSUDMT0)

Use the Database Image Copy 2 utility to take image copies of IMS databases by using the concurrent copy function of the Data Facility Storage Management Subsystem (DFSMS).

The concurrent copy function of DFSMS is a hardware and software solution that allows you to back up a database or any collection of data at a point in time and with minimum down time for the database. The database is unavailable only long enough for DFSMS to initialize a concurrent copy session for the data, which is a very small fraction of the time that the complete backup will take. For more information on DFSMS, see *DFSMS/MVS V1R3 DFSMSdss Storage Administration Guide*, SC26-4930, or *DFSMSdss Storage Administration Reference*, SC26-4929.

Once the concurrent copy session has been established, the copy is said to be logically complete. This concurrent copy process creates what is called the **logical copy**. After the concurrent copy initialization, updates may be resumed while DFSMS is reading the data and creating an output copy. The copy that is made will not include any of the update activity; it will be as if the backup were made instantaneously when it was requested.

A **physical copy** is made when DFSMS copies the logical copy to the output medium. This copy is registered with DBRC as either a SMSNOCIC or SMSCIC image copy. Depending upon the size of the data set, the physical copy may take time to produce. The physical copy is used by DBRC for recovery.

By using the DFSMS concurrent copy function the Database Image Copy 2 utility increases database availability. The utility can copy a database that is either stopped or active. If the database is stopped, it can be restarted after the logical copy is complete, and database updating can continue. The database is available to IMS without waiting for the physical copy to be complete. Note that nonrecoverable databases must be stopped before the utility is run.

You also have the option to wait until the physical copy is complete before releasing the database for update. This is useful in cases where an image copy is required for a specific purpose (like end-of-month processing). In this case, it is safer to wait for the physical copy before restarting the database.

Related Reading: For more information on the Database Image Copy 2 utility see *IMS/ESA Operations Guide*.

In this Chapter:

Restrictions

The following restrictions apply when using the Database Image Copy 2 utility:

- HSAM, GSAM or MSDB databases cannot be copied with this utility.

- Databases must be registered with DBRC to take advantage of the Database Image Copy 2 utility. DBRC must be active when the Database Image Copy 2 utility is run.
- Databases and area data sets that are to be copied must reside on hardware that supports the DFSMS concurrent copy feature (such as a 3990 Storage Control Model 3, extended function with licensed internal code) or an equivalent device. For further information on using the concurrent copy feature with 3990 storage control devices, see *IBM 3990 Storage Control Reference (Models 1, 2, and 3)*, GA32-0099.
- Databases that are specified as nonrecoverable must be stopped for the Database Image Copy 2 utility. Because they do not create logs, nonrecoverable databases cannot be active during an image copy.
- The database data set or area must be free of errors (EQEs or EEQEs) to be processed by the utility. If multiple area data sets (MADs) exist for an area, at least one area data set must be free of errors.
- VSAM data sets must be cataloged in an integrated catalog facility (ICF) catalog and must satisfy one of the following:
  - SMS-managed
  - Cataloged using an alias (i.e., the high-level qualifier(s) of the VSAM data set name is an alias for the catalog)
  - Cataloged in the master catalog
- A KSDS data set can be copied while it is being updated only if the data set is SMS-managed and BWO(TYPEIMS) was specified on the AMS DEFINE or ALTER. KSDSs that are not SMS-managed or for which BWO(TYPEIMS) was not specified must be stopped before executing the utility.
- Multiple executions of the utility are required to copy multiple database data sets or areas. One data set is copied per execution.
- Routines coded for this utility must not call module DFSRRC00. Doing so results in an abend.

Input and Output

The data sets to be copied must reside on a storage device that supports DFSMS concurrent copy.

If multiple area data sets (MADS) exist for an area, all the area data sets should be specified as input; the utility will select an error-free area data set to copy.

An image copy created by the utility is in DFSMS dump format, rather than standard batch image copy format. The copy is registered with DBRC as an SMSNOCIC or SMSCIC image copy, depending on the parameters specified when the image copy was taken. If more than two output copies are created, only the first two are registered with DBRC.

The output from the Database Image Copy 2 utility is used as input to the Database Recovery utility.

You can create up to four output image copies; however, only two will be registered in RECON. The advantage of specifying two or more copies is that if an I/O error occurs during copy execution, the utility continues to completion on the remaining copies.

DFSMS requires that you do not specify the BUFNO keyword for either the input or output data sets. For further information on input and output requirements see the *DFSMSdss Storage Administration Reference*, SC26-4929, or the *DFSMS/MVS V1R3 DFSMSdss Storage Administration Guide* , SC26-4930.

When a database is stopped and DBDGEN is run to insert and/or delete one or more areas, image copies must be taken of all areas that follow the inserted and/or deleted areas before the database is started again.

When creating an image copy of a shared secondary index, only specify the first DBD. This copies the entire data set.

JCL Requirements

The Database Image Copy 2 utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that define inputs and outputs

EXEC Statement

The EXEC statement can either invoke a cataloged procedure containing the required statements or be in the following form:

```
►►──EXEC──PGM=DFSRRC00,PARM=──┬─'ULU,DFSUDMT0'──────────┬──────────────────────►◄
                              └─'UDR,DFSUDMT0,dbdname'──┘
```

In this statement:

ULU
Specifies a load/unload region.

UDR
Specifies a recovery region. When PARM=UDR is specified, a valid dbdname is required, but is ignored by the Image Copy 2 utility.

Related Reading: See member names DLIBATCH and DBBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring*for additional parameters that can be specified in executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBD that describes the database to be dumped. This is usually DSNAME=IMS.DBDLIB. The data set must reside on a direct access volume.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct access volume, or printer, or be routed through the output stream (SYSOUT).

SYSIN DD
Defines the input control statement data set. The data set can reside on a tape, direct-access volume, or be routed through the input stream (DD * or DD DATA).

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. The data set must reside on hardware that supports the concurrent copy function.

For multiple DEDB area data sets, up to seven datain DD statements can be specified. The ddname specified in each datain DD statement must not be the area name but must match the names registered in the ADS list of the target area.

dataout1 DD
Defines the first copy of the dumped output data set. The ddname can 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. For additional information on blocksize and other considerations see *DFSMS/MVS V1R3 DFSMSdss Storage Administration Guide*, SC26-4930.

dataout2 DD
Required only if the associated utility control statement requests two or more copies of the dump. The name must appear in the control statement. The name must be that of either a tape or a direct-access device. For additional information on blocksize and other considerations see *DFSMS/MVS V1R3 DFSMSdss Storage Administration Guide*, SC26-4930.

dataout3 DD
Required only if the associated utility control statement requests three or more copies of the dump. The name must appear in the control statement. The name must be that of either a tape or a direct-access device. For additional information on blocksize and other considerations see *DFSMS/MVS V1R3 DFSMSdss Storage Administration Guide*, SC26-4930.

dataout4 DD
Required only if the associated utility control statement requests four copies of the dump. The name must appear in the control statement. The name must be that of either a tape or a direct-access device. For additional information on blocksize and other considerations see *DFSMS/MVS V1R3 DFSMSdss Storage Administration Guide* , SC26-4930.

RECON1 DD
Defines the first DBRC RECON data set. This RECON1 data set must be the same RECON1 data set that the control region is using.

RECON2 DD
Defines the second DBRC RECON data set. This RECON2 data set must be the same RECON2 data set that the control region is using.

RECON3 DD
Defines the third DBRC RECON data set. This RECON3 data set must be the same RECON3 data set that the control region is using.

Do not use these RECON data set ddnames if you have specified dynamic allocation using the DFSMDA macro.

Utility Control Statement

The utility control statement for the Database Image Copy 2 utility has a fixed format using the positions described below:

Position	**Description**
1	Unused
2	Number of copies This can be from 1 to 4.
3	Ignored
4-11	dbdname This must be the name of the physical DBD that includes the name of the data set to be dumped.
13-20	INPUT ddname This must be the ddname of the input data set or area name to be dumped. It must appear in the referenced DBD, and a corresponding DD statement must have been provided.
22-29	OUTPUT ddname This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.
31-38	OUTPUT2 ddname This is the ddname of the second copy of the dumped data set. This ddname must be specified and the DD statement for this ddname must be provided if position 2 contains `2`, `3`, or `4`.
40–47	OUTPUT3 ddname This is the ddname of the third copy of the dumped data set. The ddname must be supplied and a DD statement for this ddname must be provided if position 2 contains a `3` or a `4`. This data set is not registered with DBRC.
49-56	OUTPUT4 ddname This is the ddname of the fourth copy of the dumped data set. The ddname must be supplied and a DD statement for this ddname must be provided if position 2 contains a 4. This data set is not registered with DBRC.
58	DBACC (database access) Specifies whether databases can be updated during the image copy. `X` (exclusive) indicates that the database cannot be updated during the logical copy phase. The image copy is recorded in the RECON as an SMSNOCIC image copy. `S` (shared) indicates that the database can be updated during the image copy phase. The image copy is recorded in the RECON as an SMSCIC image copy. The default is `S`. Note that `X` must be specified for a nonrecoverable database or the image copy execution will fail.

	Note that an SMSNOCIC image copy can be used as input to the IMS/ESA System Utilities/Database Tools (DBT) HD Pointer Checker or DEBD Pointer Checker while an SMCIC cannot.
59	DBREL (database release) If the DBACC field (position 58) is X, this field specifies when the database is made available for update processing. This field is ignored when the DBACC field is specified as S. P specifies that the database is available for update after the physical copy completes. L specifies that the database is available for update after the logical copy is complete. L is the default. If a database is flagged in the RECON data set as "image copy needed," DBRC will not allow update processing until the physical copy is complete.
60-80	Comments can be placed in positions 60 through 80.

Example

The following example shows the JCL for the Database Image Copy 2 utility.

```
//T0COPY   JOB
//STEP1    EXEC   PGM=DFSRRC00,PARM='ULU,DFSUDMT0'
//STEPLIB  DD     DSN=IMS.RESLIB,DISP=SHR
//DFSRESLB DD     DSN=IMS.RESLIB,DISP=SHR
//IMS      DD     DSN=IMS.DBDLIB,DISP=SHR
//RECON1   DD     DSN=RECON1,DISP=SHR
//RECON2   DD     DSN=RECON2,DISP=SHR
//RECON3   DD     DSN=RECON3,DISP=SHR
//SYSPRINT DD     SYSOUT=A
//DBIN     DD     DSN=IMS.DBIN,DISP=SHR
//OUTPUTD1 DD     DSN=IMS.DBAOUT1,DISP=(NEW,KEEP),
//                UNIT=TAPE,VOL=SER=DMPNO1,LABEL=(,SL)
//OUTPUTD2 DD     DSN=IMS.DBAOUT2,DISP=(NEW,KEEP),
//                UNIT=TAPE,VOL=SER=DMPNO1,LABEL=(,SL)
//OUTPUTD3 DD     DSN=IMS.DBAOUT3,DISP=(NEW,KEEP),
//                UNIT=TAPE,VOL=SER=DMPNO1,LABEL=(,SL)
//OUTPUTD4 DD     DSN=IMS.DBAOUT4,DISP=(NEW,KEEP),
//                UNIT=TAPE,VOL=SER=DMPNO1,LABEL=(,SL)
//SYSIN    DD     *
 4 DBDNAMEX DBIN     OUTPUTD1 OUTPUTD2 OUTPUTD3 OUTPUTD4 XLCOMMENTSHERE
```

Figure 35. Database Image Copy 2 Utility JCL

Chapter 18. Online Database Image Copy Utility (DFSUICP0)

Use the Online Database Image Copy utility to create an as-is image copy of the database while it is being updated by the online system. The image copy is used for recovery purposes. The frequency of creating image copies must be determined by your requirements for timely recovery. The minimum requirement is that a copy be created immediately after a database is reorganized, reloaded, or initially loaded. Because database recovery is done on a physical replacement basis, a reloaded data set is not physically the same as it was before unload.

This utility runs as a batch message processing program (BMP).

Related Reading: For more information on the Online Database Image Copy utility, see *IMS/ESA Operations Guide* .

Restrictions

The following restrictions apply when running the Online Database Image Copy utility:

- HSAM and GSAM databases cannot be copied.
- MSDBs and DEDBs cannot be copied.
- The index portion of a VSAM KSDS cannot be copied without copying the entire KSDS.
- This utility requires a PSB that names the database to be copied and contains the OLIC=YES operand. This PSB can contain one or more PCBs and must be defined by an APPLCTN macro in the online system definition. The PSBGEN LANG= keyword must not specify PL/I. Refer to *IMS/ESA Utilities Reference: System* for details on constructing a PSB for use with this utility.

  If the data set to be copied is an OSAM data set of a secondary data set group and sequential buffering is used to accelerate the dumping process, then the PSB must contain at least one SENSEG for a segment of this secondary data set group.
- Take an image copy immediately after running batch jobs that update the database without logging. This allows you to maintain the integrity of your database in the event that recovery is required. You can recover using log tapes up to the start of the batch jobs, and then reprocess the batch jobs. However, the resulting database might not be bit-by-bit identical to the database after the previous batch run, although it is logically identical. Batch jobs might not be repeatable; assume that they are not. If the database is not bit-by-bit identical, log tapes created after the previous execution of the batch jobs would not be valid after the reprocessing. Therefore, the recovery process might not include the sequence of starting with an image copy, applying log tapes, reprocessing unlogged batch executions, then applying more log tapes.
- If the database is defined to DBRC as nonrecoverable, image copies can still be made for recovery purposes. See *IMS/ESA Utilities Reference: System* for more information on how the image copy is used to recover nonrecoverable databases. When making an image copy of a nonrecoverable database, the database access must be set to READ or READ ONLY status. To do this, the Master Terminal Operator (MTO) can issue a `/DBD` command. This utility fails if concurrent updates to the database are possible.
- If the online database image copy is created while an application program is updating the same database, the changes made by the program need to be

applied when the database is recovered. This is done using the Database Recovery utility. The time stamp of the first log data set required for recovery is printed on SYSOUT.

- Cannot be used in a data sharing environment if any other IMS subsystem has the database open for update access.
- Use the Concurrent Image Copy utility (DFSUDUMP0 with the CIC parameter) instead of the Online Database Image Copy utility.

Output

The output from the Online Database Image Copy utility is used as input to the Database Recovery utility.

You have 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 is somewhat diminished in this instance, but a total rerun is not necessary.

Because default block sizes are calculated at execution time if you do not specify them in the JCL, standard labels must be used on all output copies created by the Online Database Image Copy utility.

The utility prints the time stamp of the system log when the utility starts and again when it completes.

The first record on the output copy is a dump header record. It includes information such as data identification, creation date, and time. The creation date and time are required for subsequent use by the Database Recovery utility to verify input.

Recovery and Restart

Two optional DD statements let you restart an image copy job after a system or utility failure. If the statements are not included in the JCL for the Online Database Image Copy utility, no checkpoint/restart functions are available.

The DFSUCKPT DD statement defines the data set to which the utility writes checkpoint information during execution. The checkpoint information includes volume serial number and relative record numbers. By default, if the DFSUCKPT DD statement is included, the Online Database Image Copy utility writes a checkpoint for every 5000 records copied. You can override this checkpoint interval by specifying a different interval on the control statement. The checkpoint interval is described in “Utility Control Statement” on page 165.

The DFSURSRT DD statement defines a checkpoint data set to be used to restart the job. DFSUCKPT and DFSURSRT can define the same data set.

To use the restart function, the DD statements defining the image data sets must be coded with DISP=KEEP or CATLG and nonspecific serial numbers. The disposition of KEEP or CATLG ensures that the volume rewinds properly in the event of a restart. Not coding specific volume serial numbers allows the operator to mount multiple volumes in any order during a restart.

Restriction: The Online Image Copy utility cannot be restarted if it failed due to a power failure or to an out of space condition on DASD because QSAM records are not written from the QSAM buffers. Under these circumstances the online image copy SSID must be deleted from the RECON data set using the DBRC commands `CHANGE.SUBSYS` and `DELETE.SUBSYS`. After the old SSID has been deleted from the RECON another online image copy job may be submitted.

JCL Requirements

The Online Database Image Copy utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

The DD statements for the data sets to be copied are in the control region job control language data set and not in the copy job itself.

EXEC Statement

This statement can either invoke a cataloged procedure containing the required statements or can be in the form:

```
//STEP    EXEC PGM=DFSRRC00,PARM='BMP,DFSUICP0,psbname,,destname'
```

BMP and DFSUICP0
Describe the utility region.

psbname
Is the name of a PSB that has been described by an APPLCTN macro in the online system definition, specifies the database to be copied, and contains the OLIC=YES parameter.

destname
Is the output destination for critical error messages (the default destination is the MVS console).

The normal IMS positional parameters can follow.

Related Reading: For additional parameters that can be specified in executing a batch message processing region, see "Member Name IMSBATCH" in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBD that describes the database to be dumped. This is usually DSNAME=IMS.DBDLIB. The data set must reside on a direct-access volume.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct-access volume, or printer, or be routed through the output stream (SYSOUT).

SYSIN DD
Defines the input control statement data set. The data set can reside on a tape, direct-access volume, or be routed through the input stream (DD * or DD DATA).

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 can be any 1- to 8-character string, but the ddname must appear in the associated utility control statement. The output device must be either direct-access or tape. Standard labels must be used.

A user block size can be specified in the BLKSIZE subparameter of the DCB operand; the utility rounds the block size down to an even multiple of the database data set logical record size plus 8, rounded to the next doubleword. The block size specified must be larger than the logical record length and smaller than or equal to the device maximum. If a block size is not specified, the maximum block size of the device, rounded down to an even multiple of the LRCL, is used.

Exception: If the device is a 3380, a block size of 23KB is used unless the logical record length is larger than 23KB. If the logical record length is larger than 23KB, the block size is 32KB rounded down to an even multiple of the logical record length.

Restriction: The logical record length cannot be specified in the DCB operand.

If the restart function is used, ensure that the disposition of the image data sets is KEEP or CATLG.

dataout2 DD
Is required only if the associated utility control statement requests two copies of the dump. dataout2 DD has the same requirements as dataout1. The block size specified for dataout2 can be different from that specified for dataout1.

DFSUCKPT DD
Defines the optional checkpoint data set to which the utility writes checkpoint information. A single track on a direct-access device is required. Data set characteristics are specified by the utility.

DFSURSRT DD
Defines the optional restart data set, indicating that a previous checkpoint is to be used for restarting the job.

RECON1 DD
Defines the first DBRC RECON data set. This RECON1 data set must be the same RECON1 data set that the control region is using.

RECON2 DD
Defines the second DBRC RECON data set. This RECON2 data set must be the same RECON2 data set that the control region is using.

RECON3 DD
Defines the third DBRC RECON data set. This RECON3 data set must be the same RECON3 data set that the control region is using.

If you are using dynamic allocation, do not use these RECON data set ddnames.

DFSCTL DD

Describes the data set containing SBPARM control statements that request activation of sequential buffering (SB). Conditional activation of SB can improve the buffering performance of OSAM DB data sets and reduce the job time.

Related Reading: For a description of SBPARM control statements, see "SB Parameters (SBPARM) Control Statement" in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The DFSCTL file can be either a sequential data set or a member of a PDS. The record format must be either F, FB, or FBS, and the record length must be 80. The data set can reside on a direct access device, a tape, or be routed through the input stream. This DD statement is optional.

Utility Control Statement

The utility control statement for the Online Database Image Copy utility is fixed format using the positions described below:

Position	**Description**
1	Statement ID This must be the character 'D'. The D identifies the statement as an Online Database Image Copy utility control statement.
2	Number of copies This must be a 1 or 2, depending on the number of copies required.
3	This must be blank.
4-11	dbdname This must be the name of the physical DBD that includes the name of the data set to be dumped.
13-20	Input ddname This must be the ddname of the input data set to be dumped. It must appear in the referenced DBD. A corresponding DD statement does not need to be provided to DFSUICP0.
22-29	Output ddname This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.
31-38	Copy ddname 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.
40-43	Checkpoint This can be a 4-digit number specifying a checkpoint interval. This field is checked only if a DFSUCKPT DD statement is included in the JCL used to execute this utility. If this field is blank, nonnumeric, or zero, the default of 5000 is used (and, in the last two cases, a warning message indicating invalid field format is issued).
44-80	Comments can be placed in positions 44-80.

Return Codes

The Online Database Image Copy utility provides the following return codes:

Code	Meaning
0	All operations completed successfully
4	Warning messages were issued
8	One or more operations were not successful
12	An error occurred during restart
16	Severe errors have caused the job to terminate without completing all operations

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

An error received on the checkpoint data set during utility execution causes a message to be printed, but the utility continues. No further checkpoints are taken.

An error received on the restart data set during restart causes a message to be printed and the utility to abnormally terminate.

Example

The example in this section contains the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1   DD  DSNAME=RECON1,DISP=SHR
//RECON2   DD  DSNAME=RECON2,DISP=SHR
//RECON3   DD  DSNAME=RECON3,DISP=SHR
```

This example shows the JCL and utility control statements used to copy four OSAM data set groups associated with a HIDAM database, with checkpoints taken.

```
//DLICEX2  JOB
//*
//OLIC    EXEC PGM=DFSRRC00,PARM='BMP,DFSUICP0,HHTASK41'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//DFSUCKPT  DD DSNAME=OLIC2.CKPT2,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=IMSQAW,SPACE=(TRK,(1,1))
//DMP1      DD DSNAME=OLIC2.IMAG1.OSAM,DISP=(NEW,KEEP),
//             UNIT=TAPE,LABEL=(1,SL)
//DMP2      DD DSNAME=OLIC2.IMAG2.OSAM,DISP=(NEW,KEEP),
//             UNIT=TAPE,LABEL=(1,SL)
//DMP3      DD DSNAME=OLIC2.IMAG3.OSAM,DISP=(NEW,KEEP),
//             UNIT=TAPE,LABEL=(1,SL)
//DMP4      DD DSNAME=OLIC2.IMAG4.OSAM,DISP=(NEW,KEEP),
//             UNIT=TAPE,LABEL=(1,SL)
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
```

```
D1 DH41SK01 DHSK0101 DMP1              1000
D1 DH41SK01 DHSK0102 DMP2              1000
D1 DH41SK01 DHSK0103 DMP3              2000
D1 DH41SK01 DHSK0104 DMP4              1000
/*
//DFSCTL    DD *
SBPARM ACTIV=COND
/*
```

Part 3. Recovery Utilities

Chapter 19. Database Change Accumulation Utility (DFSUCUM0)

Use the Database Change Accumulation utility to streamline the recovery information you provide to the Database Recovery utility. This utility takes information from log data sets; output is in the form of a sequential data set. Utility processing involves:

1. Eliminating all nondatabase change records
2. Specifying a purge date (or dates) to eliminate all database records before that date
3. Sorting the acceptable database change records
4. Combining all database change records that update the same database physical record

The resulting records are sequenced by data set within the database. This utility invokes the Sort/Merge program, which is an execution prerequisite. This utility also sorts as a minor field RBA or key.

This utility can be executed several times over a period of time to incorporate additional database changes and to delete changes that are no longer useful.

Related Reading: For more information on using the Database Change Accumulation utility, see *IMS/ESA Operations Guide*.

The Database Change Accumulation utility can be run independently of IMS/ESA. When this utility is run with MVS/ESA, the output is compressed. This compression is achieved using MVS/ESA services.

Figure 36 on page 172 depicts the sources of input to the Database Change Accumulation utility and the output created by this utility.

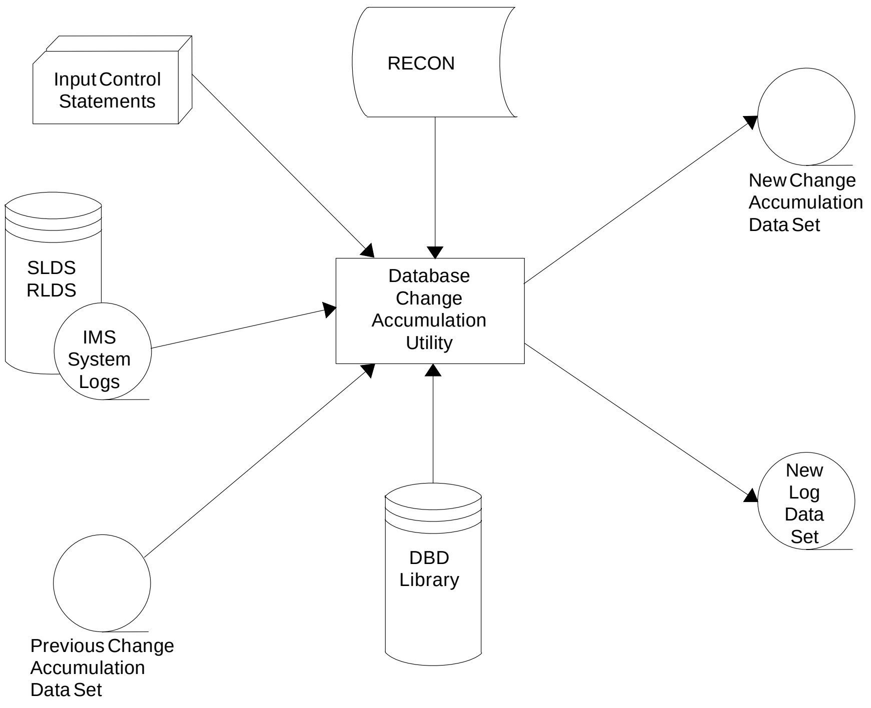

Input Control Statements
RECON
New Change Accumulation Data Set
SLDS RLDS
IMS System Logs
Database Change Accumulation Utility
New Log Data Set
DBD Library
Previous Change Accumulation Data Set

Figure 36. Database Change Accumulation Utility

Restrictions

The following restrictions apply when using the Data Base Change Accumulation utility:

- When using DBRC (Database Recovery Control), the creation of an optional log output data set using this utility is not recorded in the RECON data set. The creation of other output data sets is recorded in the RECON data set.
- The Database Change Accumulation utility cannot be restarted.

Input and Output

The input to the Database Change Accumulation utility consists of:

- All SLDS or RLDS created since either the last image copy utility execution or the last run of this utility. This can include the new log output data sets resulting from previous executions of this utility.
- The previous database Change Accumulation utility data set. This would be the output from the last execution of this utility.
- The DBD library, that is normally called IMS.DBDLIB.
- Control statements (ID, DB0 and DB1) that specify any purge dates and how the database log records are to be processed. See Figure 36 and “Utility Control Statements” on page 175.

Output from the Database Change Accumulation utility consists of:

- A new Change Accumulation utility sequential data set. This data set contains the combined database records for the database/data sets identified on a DB0 control statement. For IMS/ESA, this data set is compressed.

- A new log output data set that contains database records identified by the database/data sets on a DB1 control statement.

The new log output data set can be used to select records for critical databases. The new log is then used as input to individual change accumulation runs to obtain an accumulated change data set with changes for a particular database. This would minimize the time otherwise required to recover a database by eliminating the need to pass unwanted records from other databases.

The new log output data set cannot be used as input to the Database Backout utility. However, the new log output data set can be used as input to the Database Recovery Utility.

JCL Requirements

The Database Change Accumulation utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

EXEC statement

This statement can either invoke a cataloged procedure containing the required statements, or be in one of the following forms:

```
►►──EXEC──PGM=DFSUCUM0──,PARM='CORE=──┬─ssssss─┬──,DBRC=──┬─Y─┬──────────────────►
                                      └─MAX────┘          └─N─┘

           ┌─Y──┐
►─,HSSP=───┴─N──┴───'──,REGION=rrrK────────────────────────────────────────────►◄
```

CORE=*ssssss*|MAX
: Is the amount of storage in bytes that Sort/Merge can use for this application. The program defaults to 204800 bytes if no value is specified. If CORE=MAX is specified, the installation default value for the sort is used.

DBRC
: Can be specified as Y or N to override the specification of DBRC= on the IMSCTRL macro statement made during IMS system definition. If you specify DBRC=Y on the JCL EXEC statement, DBRC is used during execution of this utility. If you specify DBRC=N on the JCL EXEC statement, DBRC is not used during execution of this utility.

  If DBRC=FORCE was specified during IMS system definition, DBRC is always used during execution of this utility. If you attempt to override DBRC=FORCE, message DFS044I is issued and a nonzero return code is issued.

HSSP
: Can be specified as Y or N. You use HSSP=N only when you have no Fast Path log records, and both change accumulation and log records are input on a single tape drive. The default is HSSP=Y.

REGION=*rrrK*
: Is the region size. Ensure that the region size specified is the *ssssss* value plus

100KB, or 200KB if no *ssssss* value is specified. The region size required is dependent on many variables, including input and output buffer requirements, number of database/data sets specified on control statements, and the size of DBDs.

DD Statements

Lowercase ddnames on DD statements are supplied by you and can be any valid DD name.

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules.

SYSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct-access volume, or printer, or be routed through the output stream (SYSOUT). DCB parameters specified for this data set are RECFM=FBA and LRECL=121. If BLKSIZE is provided on the SYSPRINT DD statement, it must be a multiple of 121.

IMS DD
Defines the library containing the DBDs that describe all databases to be accumulated. This is usually DSNAME=IMS.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 output stream (SYSOUT). 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
Defines 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.

Related Reading: For specification of number and size of intermediate storage data sets, refer to *S/360 OS Sort/Merge: Programmer's Guide*.

DFSUCUMN DD
Defines the new accumulated change output data set. The data set can reside on a tape or a direct access volume. The block size specified must be at least 64 and less than or equal to the device maximum (or 32760). If no block size is specified, the default block size is the device maximum. If the device is a 3380, a block size of 23476 bytes is used. The logical record length cannot be overridden by the DCB operand.

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 a 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 block size specified must be at least 64 and less than or equal to the device maximum (or 32760). If no block size is specified, the default block size is the device maximum. If the device is a 3380, a block size of 23476 is used. The logical record length and RECFM=VBS cannot be overridden by the DCB operand.

The DCB parameters specified within this program are RECFM=VBS and LRECL=32760. They cannot be modified. The output is blocked to the maximum device capacity. Standard labels must be used.

DFSUDD2 DD
Defines the new reformatted sort output log data set. It is used for diagnostic purposes only and is required if the SO control statement is specified. The output can reside on a tape or a direct-access volume and is blocked to either the device maximum or 32760, whichever is smaller. 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 a tape or a direct-access volume.

Multiple log data sets can be used as input by concatenating the data sets. The log input must be in the sequence in which it was created. DBRC verifies that the log data sets are in chronological order, according to their START TIME.

SYSIN DD
Defines the control statement data set. The data set can reside on a tape, direct-access volume, or be routed through the input stream (DD * or DD DATA).

RECON1 DD
Defines the first DBRC RECON data set. This RECON1 data set must be the same RECON1 data set that the control region is using.

RECON2 DD
Defines the second DBRC RECON data set. This RECON2 data set must be the same RECON2 data set that the control region is using.

RECON3 DD
Defines the third DBRC RECON data set. This RECON3 data set must be the same RECON3 data set that the control region is using.

If you are using dynamic allocation, do not use these RECON data set ddnames.

Utility Control Statements

Four types of utility control statements are used by the Database Change Accumulation utility: ID, DB0, DB1, and SO statements. You can use all or any combination of these statements.

The Database Change Accumulation utility has the following limitations in the absence of utility control statement information:

Change Accumulation

If no utility control statements are present, all database log records are to be sorted and combined to produce a new change accumulation data set. No purge date and a maximum prime key length of 10 bytes are assumed.

If no ID control statement is present, a maximum prime key length of 10 is assumed.

If no DB0 or DB1 control statements are present, all database log records are to be sorted and combined to produce a new change accumulation tape. No purge date is assumed.

Restriction: If a DB0 statement is used, you cannot also use a DB1 *ALL statement. If a DB0 *ALL statement is used, you cannot also use a DB1 statement.

Attention: UTC, Universal Coordinated Time, will default to the current MVS offset. It is recommended that Change Accumulation always be executed using DBRC in order to insure that the correct UTC offset is calculated for all input data.

Use of Purge Date and Time

If change accumulation records (or an input log) span an Image Copy time or a reorganization time, the input log contains change records that were made *before* and after the image copy or reorganization time. A purge date and time corresponding to the image copy time or reorganization time *must* be specified. This ensures that database change records that are not valid in a recovery are eliminated.

You can specify one purge date and time for all database log records being processed. The purge date and time can be specified either for all database log records that update a particular database or for all database log records that update a data set within a database. You can also specify multiple purge dates and times, and these can be different for different data sets within a database.

If a purge date is specified without the associated time, the time defaults to zeros.

For input log tapes, the change accumulation utility compares the purge date specified to the date and time in each database update record. If the purge date is later than the date and time in the update record, then the input log record is dropped.

For previous change accumulation tapes, the utility compares the purge date specified to the date and time in the DFSUCUM0 records. If the purge date is later than the date and time in the DFSUCUM0 record, then the input log record is dropped.

The date and time in these records represents the latest update contained in the database block. For this reason, it is possible that there is an update in that record which is earlier than the purge date, and that update is not dropped, because the latest update in that same record is after the purge date specified. This is particularly important if a change accumulation spans a reorganization. You must always specify a purge date the first time a database is accumulated following a reorganization, so that old records are correctly discarded, and DFSUCUM0 blocks are correctly discarded.

In the following example, processing steps are listed to show the use of a purge date and time. No old accumulated change data set exists to be updated. (All the processing steps represent one sequence for one example. The intervening comments are for clarification only.)

Load Database 1 (DB1)

Process - Log 1

Process - Log 2

Accumulate (Log 1, Log 2) into Accumulation Log-A

Process - Log 3

Process - Log 4

Reorganize IDB1

Process - Log 5

Process - Log 6

Accumulate (Log 5, Log 6) into Accumulation Log-B

- For the accumulation run the purge date and time are not needed unless the Accumulation Log-A is input.
- Accumulation Log-B contains only the change log records since DB1 was reorganized.

Load DB2

Process DB1 and DB2 - Log 7

Process DB1 and DB2 - Log 8

Reorganize IDB2

Accumulate (Log 7, Log 8) into Accumulation Log-C

- Purge date and time are now needed to eliminate previous log records for DB2 because DB2 has been reorganized. Otherwise, database records that existed before the reorganization are accumulated, and might be impossible to purge, due to later updates to the same block. A change accumulation data set containing update records that existed before a reorganization, if used as input to a recovery, would destroy the database.

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 database 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.

ID Statement

Use the optional ID statement 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. This utility control statement is fixed format using the positions described below:

Position	Description
1-2	Statement ID Positions 1 and 2 must contain the characters ID. These identifies the statement as a Database Change Accumulation utility control statement.
3-30	Positions 3 through 30 must remain blank.
31-33	max seq length Positions 31 through 33 contain the maximum length root sequence field contained within the log records to be processed

as a result of a DB0 control statement. The maximum sequential length value for this field is the length of the sequence field on the root segment. This value is used to pad the sequence field with binary zeros for sorting purposes. If there are no VSAM KSDSs to be processed, specify this value as 4, the length of the relative block number field. This value must be in the range 1 through 236 and must be left-justified or supplied with leading zeros. The default value for this entry is 10.

41-45 max lrecl

Positions 41 through 45 contain the maximum length of a database change type log record plus the maximum sequence length specified in positions 31 through 33. The default value is 4351. Database change type log records created by IMS (other than Fast Path) are limited to a maximum of 4096 bytes and the maximum sequence length is 255 bytes. Specify this parameter when you expect Fast Path database change type records to be written to the log. For Fast Path, this parameter must be equal to the maximum control interval size for any area plus 255 plus the MAX SEQ length minus four (if a MAX SEQ length greater than four was specified). The value must be left-justified or supplied with leading zeros.

The MAX LRECL size might change with different releases of IMS.

46-80 Comments can be placed in positions 46-80.

DB0 Statement

Use the optional DB0 statement to describe which records are to be accumulated for output to the new change accumulation data set. One or more of these statements can be included. The DB0 statements are generally used to indicate what is to be accumulated from the input log data sets into the output accumulation data set. Input from the old accumulation data set for any databases or data sets that are specified in a DB0 statement are included in the output accumulation data set, unless it should be purged due to a purge date in the DB0 statement. All other input from the old accumulation data set for any other databases or data sets is carried forward to be included in the output accumulation data set, if DBRC is not active. If DBRC is active, these records are dropped. The options that are available are described below.

Each combination of database name/ddname can appear on one control statement only.

The DB0 utility control statement is fixed format using the positions described below:

Position **Description**

1-3 Statement ID

Positions 1 through 3 must contain the characters DB0. This identifies the statement as a Database Change Accumulation utility control statement.

4-11 dbname

Positions 4 through 11 can contain a database name or a ƀ*ALL where position 4 is blank and positions 5 through 8 contain the characters *ALL. If *ALL is specified, all records are accumulated, the purge date and time in positions 12 through 20 are applied to all records, and no DB1 statements can be specified. If DBRC is being used, the *ALL option of DB0 control statements is not valid. If a database name and ddnames are supplied, the purge date is applied only to those records whose database name/ddname combinations match. If no ddnames are supplied, the purge date applies to all records that match the database name.

12-37 Purge date and time. If a purge date and time are specified, all records that match a database name/ddname description and dated before the purge date are eliminated. If an old accumulated change input is supplied, all records that match the database name/ddname description and dated before the purge date are not merged into the new change accumulation data set. If this field is blank, no purge date is used. If any fields are coded, all the fields for time and date must be coded.

Related Reading: See *IMS/ESA DBRC Guide and Reference* , SES1-2061-00, in the chapter on Command Syntax, under Parameters, for a detailed discussion of the types of standard timestamp formats which may be used in this field.

38-72 ddnames

Positions 38 through 45, 47 through 54, 56 through 63, and 65 through 72 can contain 1 to 4 ddnames which,combined with the database name supplied, make up the record identification. All records matching this identification are sorted and accumulated and have the purge date and time applied to them. As many combinations of database 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 database name. All ddnames must be left-justified; unused positions must be blank. If no ddnames are specified but a dbname is specified in positions 4 through 11, all changes to the database are written to the accumulation change data set. The names specified must be the same names used for the DD1 keyword in the dataset or area statement for the appropriate DBDGEN. For Fast Path input, this can be the ddname or area name, whichever was used during the DBDGEN. Refer to the DBDGEN to determine which name was used.

73-80 Filler/Ignored

DB1 Statement

Use this statement 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 database change records are not written to the output data set. Any number of DB1 statements describing database name/ddname combinations can be included.

Each combination of database name/ddname can appear on one control statement only.

Change Accumulation

The DB1 utility control statement is fixed format using the positions described below:

Position	**Description**
1-3	Statement ID Positions 1 through 3 must contain the characters DB1. This identifies the statement as a Database Change Accumulation utility control statement.
4-11	dbname Positions 4 through 11 can contain a database name, ƀ*ALL or b*OTHER. If *ALL is specified, all records are written to the new log data set, and no DB0 control statements can 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.
12-37	Purge date and time. All records matching a record identification combination and dated before the purge date are eliminated. Related Reading: See *IMS/ESA DBRC Guide and Reference* , SES1-2061-00, in the chapter on Command Syntax, under Parameters, for a detailed discussion of the types of standard timestamp formats which may be used in this field.
38-72	ddnames Positions 38 through 45, 47 through 54, 56 through 63, and 65 through 72 can contain 1 to 4 ddnames. These names, combined with the database name, make up the record identification. All records matching this identification and dated after the purge date are written to the new log data set. All ddnames supplied must be left-justified; unused positions must be blank. If no ddnames are specified but a dbname is specified in positions 4 through 11, all changes to the database are written to the new database log data set. The names specified must be the same names used for the DD1 keyword in the data set or area statement for the appropriate DBDGEN. For Fast Path input, this can be the ddname or area name, whichever was used during the DBDGEN. Refer to the DBDGEN to determine which name was used.
73-80	Filler/ignored

SO Statement

Use the SO statement to request the reformatted output logs from the Sort program to be printed to the Sort output data set. This function is a diagnostic aid to help in problem determination.

There are four options that can be specified: no options, DBNAME, DSID or the RBA/RBN. Any combination of the DBNAME, DSID or the RBA/RBN options can be specified. If more than one option is chosen, all must be satisfied before a record is written. If no options are chosen, all sorted, reformatted records are written.

The SO utility control statement is fixed format using the positions described below:

Position	**Description**
1-2	Statement ID Positions 1 and 2 must contain the characters SO. This identifies the statement as a Database Change Accumulation utility control statement. If no other options are chosen, all sorted, reformatted records are written.
3	Position 3 must be blank.
4-11	dbname or blank Positions 4 through 11 can contain a database name or be left blank. If a dbname name is specified, all records with that dbname are written.
12-14	dsid or blank Positions 12 through 14 can contain a database data set ID or be left blank. If a database data set ID is specified, all records with that data set ID are written.
15	Position 15 must be blank.
16-25	RBA or RBN Positions 16 through 25 can contain the RBA (relative byte address) or the RBN (relative byte number) of a record or of many records. If a record contains the RBA or RBN specified, all records with that RBA or RBN are printed. The RBA or RBN must be written in hexadecimal.

Output Messages and Statistics

Message IEC161I is a normal message during an IMS database recovery of a VSAM data set.

Return Codes

The Database Change Accumulation utility provides the following return codes:

Code	**Meaning**
0	All operations completed successfully
4	Warning messages were issued
8	One or more operations were not successful
16	The Sort/Merge program was not successful

There might be other return codes returned by Sort/Merge if the ERRET=ABEND option was specified when Sort/Merge was installed.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

Examples

The examples in this section contain the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1   DD  DSNAME=RECON1,DISP=SHR
//RECON2   DD  DSNAME=RECON2,DISP=SHR
//RECON3   DD  DSNAME=RECON3,DISP=SHR
```

Example 1

In this example, two database logs are to be accumulated. No old accumulated change data set exists to be updated. The first database (DI32DB01) is to be accumulated in its entirety, and all records before day 175 of year 85 and before 1200 hours are to be eliminated. The second database (DI32DB02) is to be accumulated selectively.

The database (DI32DB02) data set with the ddname DDI3IA is to be accumulated, and all records before day 173 of year 85 and before 1500 hours are to be eliminated. The database (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 85 and before 1500 hours are to be eliminated. All other records are to be written out to the new log data set.

```
//ACCUM1   JOB
//*
//STEP1   EXEC PGM=DFSUCUM0,PARM='CORE=512000'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SORTLIB   DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD DUMMY,DCB=BLKSIZE=100
//DFSUCUMN  DD DSNAME=IMS.CUM1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=CUMTAP
//DFSUDD1   DD DSNAME=IMS.NEWLOG,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=LOGTAP
//DFSULOG   DD DSNAME=IMS.LOG1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE1
//          DD DSNAME=IMS.LOG2,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE2
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
DB0DI32DB01851751200000
DB0DI32DB02851731500000     DDI3IA
DB1DI32DB02851731500000     DDI3OA
DB1 *OTHER
/*
```

Example 2

In this example, all database change records are to be accumulated. The maximum root segment sequence field length is specified as 4 bytes, because all log records reflect HD-type organizations. There are no 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 85 are to be eliminated. An old change accumulation data set is to be merged with the new change accumulation data set. The purge date is applied to the old accumulation data set. DFSUDD1 (new log output data set) is defined as DUMMY because a DB1 control statement is not specified.

```
//ACCUM2   JOB
//*
//STEP1   EXEC PGM=DFSUCUM0,PARM='CORE=512000'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SORTLIB   DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD DSNAME=IMS.CUM1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=CUMTAP
//DFSUCUMN  DD DSNAME=IMS.CUM2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=CUMTP2
//DFSUDD1   DD DUMMY
//DFSULOG   DD DSNAME=IMS.LOG1,DISP=OLD.
//             UNIT=TAPE,VOL=SER=LTAPE3
//          DD DSNAME=IMS.LOG2,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE4
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
ID                             004
DB0 *ALL   852000000000
/*
```

Example 3

In this example (which is similar to example 2), two database logs are to be accumulated. No old accumulated change data set exists to be updated. The first database (DI32DB01) is to be accumulated in its entirety, and all records before day 175 of year 85 and before 1200 hours are to be eliminated. The second database (DI32DB02) is to be accumulated selectively.

The database (DI32DB02) data set with the ddname DDI3IA is to be accumulated, and all records before day 173 of year 85 and before 1500 hours are to be eliminated.

The database (DI32DB02) data set with the ddname DDI3OA is to have its records written to the new log data set, and all records before day 173 of year 85 and before 1500 hours are to be eliminated.

The database (DI32DB01) data set 001 is to have all its sorted reformatted records (with an RBN of 100) written to the sort output data set.

All the log records passed to the SORT routine with database names of DI32DB01, data set ID 001, and an RBA of 100 is written to IMS.NEWLOG2.

All other records are to be written to the new log data set IMS.NEWLOG1, which is used for diagnostics only.

```
//ACCUM1   JOB
//*
//STEP1   EXEC PGM=DFSUCUM0,PARM='CORE=512000'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SORTLIB   DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD DUMMY,DCB=BLKSIZE=100
//DFSUCUMN  DD DSNAME=IMS.CUM1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=CUMTAP
//DFSUDD1   DD DSNAME=IMS.NEWLOG1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=LOGTAP
//DFSUDD2   DD DSNAME=IMS.NEWLOG2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=LOGTAP
//DFSULOG   DD DSNAME=IMS.LOG1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE1
//          DD DSNAME=IMS.LOG2,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE2
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *

DB0DI32DB01851751200
DB0DI32DB02851731500000     DDI3IA
DB1DI32DB02851731500        DDI3OA
DB1 *OTHER
SO DI32DB01001 0000000100
/*
```

Chapter 20. Database Recovery Utility (DFSURDB0)

Use the Database Recovery utility to recover a physically damaged data set in an IMS database. This utility does not provide a means of recovery from application logic errors; it is your responsibility to ensure the integrity of the data in the database.

This utility performs recovery by updating a copy of the database with the changes logged since the copy was made.

The Database Recovery utility is executed in a special IMS batch region. This allows database recovery to be run independently of the IMS system.

The Database Recovery utility can be used with multiple DEDB area data sets, but the utility recovers one area data set at a time. If multiple-copy DEDB area data set support is required, the DEDB Area Data Set Create utility must be used to create the additional copies.

Requirement: To recover a MADS, the MADS must be made unavailable and set to recovery needed.

Related Reading: For more information on the Database Recovery utility, see *IMS/ESA Operations Guide*.

Figure 37 is a flow diagram of the Database Recovery utility.

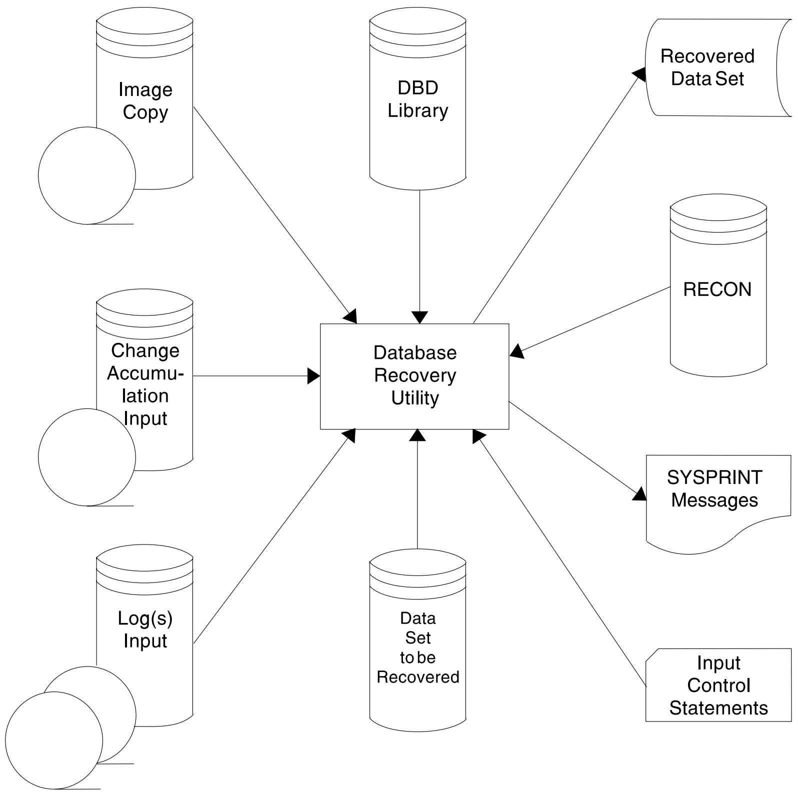


Figure 37. Database Recovery Utility

Using the Database Recovery Utility in an RSR Environment

You can use the Database Recovery utility at a Remote Site Recovery (RSR) tracking site. If you use the `GENJCL.RECOV` command to generate the JCL, then image copy and change accumulation data sets are included.

Once all the recovery jobs have completed for a group of databases, you must issue the `/START DATABASE|AREA|DATAGRP` command to initiate online forward recovery (OFR) to prepare the databases for normal tracking.

OFR is only available on a DLT tracking subsystem for databases tracked at the database readiness level.

If you recover a database to a time-stamp at the active site, you must take an image copy of the database before the database can be used again. You must then register the image copy with DBRC at the tracking site (using NOTIFY.IC) and perform a full recovery of the database.

- At the active site:
  1. Perform the time-stamp recovery.
  2. Take an image copy of the database.
  3. Send the image copy to the tracking site.
- At the tracking site:
  1. Receive the image copy.
  2. Use the `NOTIFY.IC` command to register the image copy with DBRC.
  3. Run the recovery utility to apply the image copy (and possibly any change accumulation data).
  4. Issue the `/START DATABASE|AREA|DATAGRP` command make the database ready for tracking. This causes OFR to apply changes from the log and causes normal tracking to resume.

Restrictions

The following restrictions apply when you run the Database Recovery utility:

- Only one data set is recovered for each execution.
- SYSPRINT can be blocked but must be a multiple of 121.
- Do not use HD reload as input to recovery. There is no guarantee that physical location of segments after a second reload will match log records created from updates to a database after the first reload.
- Do not use output from the Log Merge utility as input to recovery.
- If an area is registered in the DBRC (Database Recovery Control) RECON data set, the following restrictions apply:
  – The ddname and dsname in the data set1 DD statement must match the names registered in the ADS (area data set) list of the target area.
  – The target area must be marked “recovery-needed” in the RECON.
- If the target area is not registered in RECON and RECON has a NOFORCE attribute, the ddname in the data set1 DD statement must match the target area name.
- If the target area is not registered in RECON and RECON has a FORCER attribute, this utility terminates with an error message.
- When recovering a shared secondary index, specify only the first DBD. The utility recovers the entire data set.

- For full recovery of an OSAM data set, the OSAM data set must be scratched and reallocated. Reallocate the data set in the recovery step's JCL. Because the Database Recovery utility does not use OSAM to build the output data set, preallocating the primary data space across multiple volumes is not supported.
- If the HISAM unload data set is to be used as the DFSUDUMP data set input to the Database Recovery utility, the database must be reloaded *immediately* following the unload. This procedure ensures that the Database Recovery utility ignores any records on the IMS system log created before the unload/reload operation. This is necessary because the HISAM unload tape, when processed by the Database Recovery utility, reorganizes the database. Reloading after unload creates an equivalent copy of the database. Start a new change accumulation tape after unload, or purge the reloaded database of log entries prior to the unload tape.
- For recovery of a VSAM data set, the VSAM data set must be deleted and redefined before executing the Database Recovery utility. It is normal to receive VSAM information message IEC161I 072-053 (DATA SET WAS EMPTY) when recovering a VSAM data set.
- If concurrent image copy (CIC) is used, the minimum DBRC sharelevel for CIC is 1. The sharelevel must be specified to run CIC. See *IMS/ESA Administration Guide: System* for the levels of data sharing that the `DBRC INIT.DB` command issues.

Input and Output

The Image Copy input, Change Accumulation input, and Log input are all optional. If there is no Image Copy input, the data set to be recovered is used as input.

The copy of the database to be supplied to the Database Recovery utility can be:

- An image copy created by the Database Image Copy utility (DFSUDMP0) or the Online Database Image Copy utility (DFSUICP0). (Neither of these utilities can be used for HSAM or GSAM databases; therefore, the Database Recovery utility cannot be used with HSAM or GSAM databases.)
- An image copy created by the Image Copy 2 Utility can also be used as input.
- A HISAM unload data set created by the HISAM Reorganization Unload utility (DFSURUL0).

If you have already copied the database with a copy created by an MVS utility, the image copy input data set is not required. For a KSDS, any utility which preserves the KSDS data content can be used to create the copy (IMS accesses a KSDS by key, and therefore has no dependency upon RBA sequence within the KSDS). For all other dataset types, the physical sequence of the content of the dataset must be preserved. Failure to copy the dataset correctly can result in failure of the database recovery process. Be sure to use only utilities that copy the dataset correctly, or use the IMS provided Image Copy utilities.

Multiple logs can be provided by concatenating the logs in date and time sequence.

The changes to the database to be supplied as input must be one of the following:

- An SLDS or RLDS created by IMS during normal execution
- An accumulation of the changes on the log created by the Database Change Accumulation utility (DFSUCUM0)
- A combination of both the SLDS and the log created by the Database Change Accumulation utility (the changes in the SLDS must be more recent than the changes in the Change Accumulation log)

- One or more partial image copies created by the HSSP option

  Definition: An unsuccessful execution of the HSSP image copy option (with update intent) produces a *partial image copy*. A partial image copy consists of the “after” images of the records of the area up to the point of the HSSP processing failure.

If the Online Database Image Copy utility is used to back up a database data set, changes made concurrent with and subsequent to the image copy might be required. The image copy can be a dump created by the Batch Database Image Copy utility or the Online Database Image Copy utility. In the case of HISAM, the image copy can be a reorganization dump created by the HISAM Reorganization Unload utility. The Online Database Image Copy utility provides on SYSOUT the volume serial number of the first log tape that might contain applicable changes.

Take an image copy immediately after running batch jobs that update the database without logging. This allows you to maintain the integrity of your database in the event that recovery is required. You can recover using log tapes up to the start of the batch jobs, and then reprocess the batch jobs. However, the resulting database might not be bit-by-bit identical to the database after the previous batch run, although it is logically identical. Batch jobs might not be repeatable; assume that they are not. If the database is not bit-by-bit identical, log tapes created after the previous execution of the batch jobs would not be valid after the reprocessing. Therefore, the recovery process might not include the sequence of starting with an image copy, applying log tapes, reprocessing unlogged batch executions, then applying more log tapes.

If the target data set fails to open because of a failure during the current or a previous run, scratch and reallocate the data set before rerunning this utility.

The new log output data set can be used as input to the Database Recovery Utility.

It is recommended that Database Recovery always be executed with DBRC in order to calculate the correct UTC, Universal Coordinated Time, offsets using the DBRC THT, Time History Table. If DBRC is not used then the MVS offset will be used as the default.

JCL Requirements

The Database Recovery utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

This statement can either invoke a cataloged procedure containing the required statements, or be in the form:

```
//STEP    EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,dbdname'
```

UDR
Specifies a recovery region

dbdname
Is the name of the DBD that includes the data set to be recovered

The normal IMS positional parameters such as BUF and SPIE can follow dbdname.

Related Reading: For additional parameters that can be specified in executing a batch processing region, see “Member Name DLIBATCH” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

If DBRC=FORCE was specified on the IMSCTRL macro statement during IMS system definition, DBRC is used during the execution of this utility.

If DBRC=YES was specified on the IMSCTRL macro statement during IMS system definition, DBRC is used during the execution of this utility unless it is overridden by specifying N in the positional parameter for DBRC on the execute statement.

If DBRC=NO was specified during IMS system definition, do not use DBRC during the execution of this utility and do not use it to generate the JCL. DBRC must be used if an SMSCIC or SMSNOCIC image copy is input to recovery.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBD that describes the database data set to be recovered. This is usually DSNAME=IMS.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 (SYSOUT). SYSPRINT can 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 be routed through the input stream (DD * or DD DATA).

DFSUDUMP DD
Defines the image copy input data set, if any, to be used for recovery. It can be a data set created by either the Batch Database Image Copy utility, the Online Database 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 the ddname is not DFSUDUMP, the ddname must also be included in position 22 of the utility control statement. If the USEDBDS or USEAREA keyword is specified on the `GENJCL.RECOV` command, the DFSUDUMP DD statement generated is DUMMY. The data set can reside on a tape or a direct-access volume.

DFSUCUM DD
Defines the accumulated change input data set. If no accumulated change input is supplied, this statement must be coded DD DUMMY. This data set can reside on a tape or a direct-access volume.

DFSULOG DD
Defines the log change input. If no log changes are to be applied, this statement must be coded as DD DUMMY. This data set can reside on a tape or a direct-access volume.

Multiple logs can be used as input by concatenating the data sets. This requires that the DD statements be in date and time sequence.

Related Reading: See “Example 1” on page 194 for details. DBRC verifies that the log data sets are in chronological order according to their STOP TIME.

DFSUSNAP DD
Defines a SNAP output data set that is used to write specific areas of storage for diagnostic purposes where appropriate. The data set can be defined as SYSOUT=* or with the following attributes:

```
DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
```

data set1 DD
Defines the data set to be recovered. The ddname must be the same as the one in the DBD that describes this data set. It must also be in the utility control statement.

For DEDBs, this DD statement defines the area data set of the area to be recovered and the ddname must be the same as the one in the DBD that describes this area.

If an area is registered in the DBRC RECON data set, the ddname and dsname must match the names registered in the ADS list of the target area. If an area is not registered in the DBRC RECON data set and the DBRC RECON data set has the NOFORCER attribute, the ddname must be the same as the area name and must be in the utility control statement. If an area has multiple area data sets (MADS) and the SMSNOCIC or SMSCIC image copy created by the Database Copy 2 utility is used is input to recovery, DD statements for all the area data sets should be included. The area data set that will be recovered is determined from the contents of the image copy 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 only change accumulation input is used
- If log input is used
- For recovering a VSAM ESDS with data from HISAM unload as input
- For recovery when a null image copy data set is used as input

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The data set can reside on a tape, direct-access device, or be routed through the input stream (DD * or DD DATA).

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence is used for the dump.

RECON1 DD
Defines the first DBRC RECON data set. This RECON1 data set must be the same RECON1 data set that the control region is using.

RECON2 DD
Defines the second DBRC RECON data set. This RECON2 data set must be the same RECON2 data set that the control region is using.

RECON3 DD
Defines the third DBRC RECON data set. This RECON3 data set must be the same RECON3 data set that the control region is using.

Do not use these RECON data set ddnames if you are using dynamic allocation.

DFSVDUMP DD
The DFSVDUMP DD statement is generated as DUMMY.

Utility Control Statement

The Database Recovery utility uses the following utility control statements:

- ABEND
- NOSEQCK
- (S) Database Recovery

These statements can be used separately or together.

Related Reading: See *IMS/ESA DBRC Guide and Reference* for additional parameters you might need to include in your JCL. These keywords are described in the JCL-statement descriptions.

ABEND Statement

Use this utility control statement to request that the utility terminate with a user abend 302 when an abnormal condition is encountered. A storage dump is provided if a SYSUDUMP DD statement is supplied. If this statement is omitted, the Database Recovery utility issues error messages for any abnormal condition encountered and continues processing.

This control statement must come before the Database Recovery utility control statement.

The format of this control statement is with ABEND starting in column 1.

NOSEQCK Statement

Use this utility control statement to request that the utility not perform sequence checking on the input logs. Use this statement with extreme caution, because recovery might not be possible if logs do not sequence properly.

This control statement must come before the Database Recovery utility control statement.

The format of this control statement is with NOSEQCK starting in column 1.

(S) Database Recovery Statement

The utility control statement for the Database Recovery utility has a fixed format using the field positions described below:

Position	Description
1	Statement ID The ID of the Database Recovery utility control statement. It must be the character 'S'.
2	This position must be blank.
3	This position must be blank.
4-11	dbdname The name of the DBD that describes the database containing the data set to be recovered. This name must also appear in the PARM field of the EXEC statement.
12	This position must be blank.
13-20	Data set or area ddname The ddname of the data set or area name to be recovered. It must be the same as the ddname in the DBD and data set1 DD statement.
21	Blank
22-29	Input ddname The ddname of the data set used as the image copy input. If this field is blank, the ddname 'DFSUDUMP' is the default.
30	Blank
31-55	The time stamp specified when the RCVTIME parameter is input on the `GENJCL.RECOV` command. Otherwise, these positions are blank.

Position	Description
31-42	This position specifies the date and time, in the format yydddhhmmsst, where yy=year, ddd=day of year, hh=hour, mm=minute, ss=second, t=tenths of a second.
43	This position specifies the sign of the offset, + or -.
44-47	This position specifies the offset from the UTC, in the format HHMM.
48-55	Blanks

or . . .

Position	Description
31-47	This position specifies the punctuated time stamp, in the format yy.ddd hh:mm:ss.t, where yy=year, ddd=day of year, hh=hour, mm=minute, ss=second, t=tenths of a second.
48	This position specifies the sign of the offset, + or -.
49-53	This position specifies the offset from the UTC, in the format HH:MM.
54-55	Blanks

or . . .

	31-49	This position specifies the punctuated time stamp with a four-year digit, in the format yyyy.ddd hh:mm:ss.t, where yyyy=year, ddd=day of year, hh=hour, mm=minute, ss=second, t=tenths of a second.
	50	This position specifies the sign of the offset, + or -.
	51-55	This optional position specifies the offset from the UTC, in the format HH:MM. If the position is omitted the default is derived from the current MVS offset value (CVTLDTO).
56	Blank.	
57	C, if the USEDBDS parameter is specified on the `GENJCL.RECOV` command; otherwise, blank.	

Related Reading: See the `GENJCL.RECOV` command in *IMS/ESA DBRC Guide and Reference*for further information.

Output Messages and Statistics

It is normal to receive a VSAM information message, IEC161I 072-053, when recovering a VSAM data set.

Return Codes

The Database Recovery utility provides the following return codes:

Code	**Meaning**
0	All operations completed successfully.
4	Warning messages were issued.
8	More than one operation was not successful.
16	Severe errors caused the job to terminate before completing all operations.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

Examples

The examples in this section contain the following comment line above the SYSIN statement to aid in column alignment.

```
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
```

This comment line is for reference only.

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL:

```
//RECON1   DD  DSNAME=RECON1,DISP=SHR
//RECON2   DD  DSNAME=RECON2,DISP=SHR
//RECON3   DD  DSNAME=RECON3,DISP=SHR
```

Example 1

This example shows the JCL to recover an HDAM OSAM data set with a ddname of DBHD3B in a database named DD32DB01. Input is provided from an image copy data set and multiple system log data sets. The ddname on the image data set is not defaulted to DFSUDUMP in the control statement.

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

```
//RECOVERY JOB
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,DD32DB01'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DUMPDS    DD DSNAME=IMS.DBBOUT1,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP3,LABEL=(,SL)
//DFSUCUM   DD DUMMY
//DFSULOG   DD DSNAME=IMSLOG.MONDAY,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=LOG1,LABEL=(,SL)
//          DD DSNAME=IMSLOG.TUESDAY,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=LOG2,LABEL=(,SL)
//DBHD3B    DD DSNAME=IMS.DBHD3b,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=DBASE2,
//             SPACE=(CYL,(20,10))
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
S  DD32DB01 DBHD3B   DUMPDS
/*
```

Example 2

This example shows the JCL to recover a DEDB area with an area name of AREANAM1 in a database named DI32DB01. Input is provided from an image copy data set and change accumulation data sets. DDNAME1 and DSNAME1 are the ddname and dsname in the ADS list of the area to be recovered. Additional parameters specify YES (Y) for DBRC and a GSGNAME is supplied.

```
//RECOVERY JOB
//*
//STEP1   EXEC PGM=DFSRRC00,
// PARM='UDR,DFSURDB0,DI32DB01,,,,,,,,,,Y,,,,,,,GSGNAME1'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//IMS       DD DSNAME=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP  DD DSNAME=IMS.DBAOUT1,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP1,LABEL=(,SL)
//DFSUCUM   DD DSNAME=IMS.CUM1,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=DBCUM1,LABEL=(,SL)
//DFSULOG   DD DUMMY
//DDNAME1   DD DSNAME=DSNAME1,DISP=OLD
//DFSVSAMP  DD DSNAME=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//RECON1    DD DSNAME=RECON1,DISP=SHR
//RECON2    DD DSNAME=RECON2,DISP=SHR
//RECON3    DD DSNAME=RECON3,DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN    DD  *
S  DI32DB01 AREANAM1
/*
```

Chapter 21. Batch Backout Utility (DFSBBO00)

Use the Batch Backout utility to recover databases to a point before a program was initiated or to a checkpoint or sync point. The Batch Backout utility operates as a normal IMS batch job using the PSB of the program whose errors are to be backed out.

The usefulness of this utility is dependent on the type of errors encountered, the configuration of the IMS system, and whether the proper sequence of recovery steps are taken.

Related Reading: The circumstances under which to run this utility are described in *IMS/ESA Operations Guide*.

Figure 38 is a flow diagram for the Batch Backout utility.

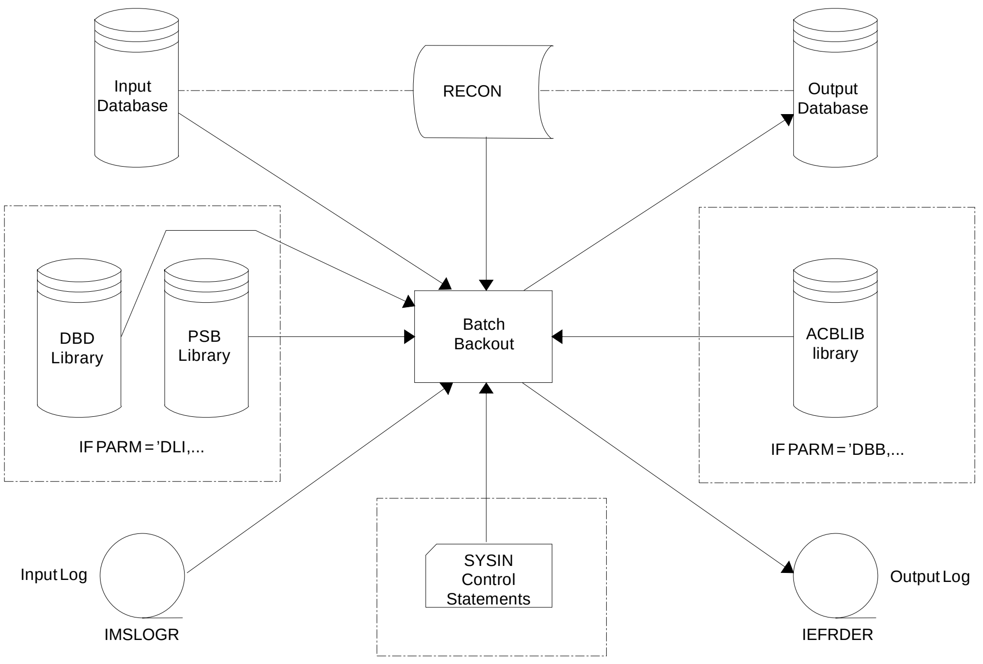


Figure 38. Data Set Requirements for the Batch Backout Utility

Using the Batch Backout Utility in an RSR Environment

Batch Backout at a Remote Site Recovery (RSR) active site calls the transport manager to send log data to the tracking site, just like other IMS batch jobs. When Batch Backout informs DBRC about completed backouts, it records the same information in a log record sent to the tracking site, so the tracking site RECON data set is updated.

Restriction: Batch backout cannot be used at the tracking site.

After a remote takeover, you can use the utility at the new active site to perform backouts for any uncommitted changes for applications run at the old active site. In general, you use the utility at the new active site as you would at the old active site if there had been no remote takeover.

Batch Backout

However, if the first restart of the new active subsystem is an emergency cold start (/ERE COLDSYS), you must first run the Batch Backout utility (before the emergency restart) with the COLDSTART statement, regardless of whether you had already done it at the old active site. All databases associated with in-flight units of work are inhibited from authorization until they have been backed out.

After a remote takeover, you must run the utility to back out uncommitted changes for any batch jobs that did not complete at the old active site. You can do this immediately after the tracking system has shutdown or defer it to a more convenient time.

You must use the Batch Backout utility to take care of any previous backout that had been deferred past a cold start at the old active site. You can do this immediately after the tracking system has shutdown or defer it to a more convenient time.

Restrictions

The following restrictions apply when using this utility:

- The release level of Batch Backout must match the release level of the control region which created the input log data sets.
- If a write error has occurred for an unregistered database (or for any database, if DBRC is not used) and the database has not been recovered, then recovery must be completed before this utility is run. Without the assistance of DBRC, no check for successful completion of recovery is possible. If you attempt to backout without doing a recovery first, the results are unpredictable. With registered databases, recovery prior to backout is required only if backout fails to complete due to an I/O error. If backout completes normally, recovery can be postponed to a more convenient time.
- If the IMS region that created the log being used as input, used DBRC and IRLM, Batch Backout must also use DBRC and IRLM.
- Batch Backout does not back out database updates if other applications have had access to them. For an online database, updates are made available to other applications as soon as the MPP, BMP, or IFP making the updates reaches a sync point. When a batch job using IRLM issues a checkpoint, all updates it has made become available to applications in other IMS systems. In the above cases only updates made since the last sync point can be backed out. Batch Backout backs out updates to any specified checkpoint for a batch job not using IRLM. Be sure no other application has had access to the database updates before using this option.
- The logs used must not have been created before a reorganization.
- When you run Batch Backout for a PSB that references Fast Path databases, you must specify a 'DBB' region, and you must use the same ACBLIB that was used with the online application.
- This utility does not back out nonrecoverable databases unless the input log was created by an IMS batch job (and not the result of archiving that log).
- When using DBRC=C, you must ensure that no other application has modified the same databases as the job you are backing out after it completed. You must also provide a BYPASS LOGVER Utility Control statement in your SYSIN data set. No log checking will be done for DBRC=C.

Input and Output

The input to the Batch Backout utility consists of:

- Log data sets (SLDS and/or OLDS; tape and/or DASD) containing database updates to be backed out. If the updates to be backed out are from a batch job, the complete log from a single run of that job must be provided. If the database updates to be backed out were done by an IMS DB/DC or DBCTL subsystem, enough log data sets must be included for Batch Backout to recognize that it has all the updates completed for that UOR. If there are no unrecovered I/O errors for any database being backed out, log data sets from unsuccessful restart attempts are not needed.

  For dynamic backout failures, Batch Backout needs all records between the X'5607' record, indicating the beginning of the UOR, and the X'07' record, indicating the end of the UOR. In some cases, the backout can be done using a log containing an IMS checkpoint taken after the dynamic failure. If the problem that caused the dynamic backout failure interferes with data collection for the checkpoint, the checkpoint data cannot be used for the backout.

  For in-flight and in-doubt UORs, Batch Backout needs all records between the X'5607' record, indicating the beginning of the UOR, to (but not including) the next restart.

  If the BYPASS LOGVER utility control statement is used, include all of the log data sets between the data set that was active at the beginning of the UOR, and the data set that was active when all databases affected by that UOR were stopped. If the `/START` command was entered for any of the affected databases after they have been stopped, include all the data sets to that point.

  If either the ACTIVE or the COLDSTART utility control statement is used, all the log data sets must be included from the last sync point or application checkpoint to the restart. For the first execution of Batch Backout with the COLDSTART or the ACTIVE control statement before a restart, the input log must include an IMS checkpoint and the beginning of every non-BMP application active at termination.
- The databases with updates that need to be backed out. All data sets related to databases in the PCBs used for the updates must be provided.
- Optional control statements can be provided to determine what is to be backed out. See “Utility Control Statements” on page 200 for more information.

Batch Backout uses information from DBRC to validate the input log. Batch Backout avoids performing backouts that have completed or that are done using an in-progress restart.

The output from the Batch Backout utility consists of:

- The input databases with the changes made by the incomplete transactions or jobs backed out to the last sync point (or, if the CHKPT statement was used for a batch job that did not use IRLM, backed out to the specified checkpoint).
- An output log that is to be saved for input to the Database Recovery Utility, in case a forward recovery has to be done on either of the backed out databases.

The output log from the Batch Backout Utility might be needed as input for the Change Accumulation utility (if the time of the Batch Backout run falls within the period covered by the Change Accumulation).

Do not use this log as input to subsequent runs of the Batch Backout utility.

JCL Requirements

The Batch Backout utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

This statement must be in the form:

```
//stepname EXEC PGM=DFSRRC00,
//                  PARM='DBB,DFSBBO00,psbname,nnn'
```

or

```
//stepname EXEC PGM=DFSRRC00,
//                  PARM='DLI,DFSBBO00,psbname,nnn'
```

The first form uses prebuilt blocks. The second form does not use prebuilt blocks.

psbname
Is the name of the PSB used by the program to be backed out.

nnn
Is the number of 1KB blocks required for the database buffer pool.

Related Reading: See “Member Name DBBBATCH or DLIBATCH” in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for an explanation of other parameters that can be specified in the EXEC statement.

The value 'C' must be specified for the DBRC parameter when backing out a normally terminated batch job that used DBRC but did not use IRLM.

Restriction:A normally terminated job that used IRLM cannot be backed out.

If this parameter is used for a batch job whose DBRC subsystem record is marked abnormally terminated, batch backout functions as if 'Y' had been specified for the DBRC parameter.

DD Statements

Lowercase ddnames on DD statements are supplied by you and can be any valid ddname.

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Describes the IMS.PSBLIB and IMS.DBDLIB data sets, (PARM='DLI,...'). These data sets must reside on a direct-access volume.

IMSACB DD
Describes the IMS.ACBLIB data set, (PARM='DBB,...'). This data set must

reside on a direct-access volume. If GSAM files are used, you must include the IMS DD statement. The IMS DD statement concatenates the IMS.PSBLIB and IMS.DBDLIB libraries.

IMSLOGR DD
Describes the input log file. It can reside on a tape or DASD.

Attention:

Do not reference the model DSCB name in the IMSLOGR or IMSLOGxx DD statements. Specifying the DSCB name in the DCB causes Batch Backout to process only the first volume of a multivolume log data set. Backout completes normally, but the database might be damaged.

IMSLOGxx
Describes additional input logs. The data sets can be OLDS and/or SLDS. The suffix xx can be any alphanumeric characters.

IMSLOGxx DD statements specify the input log data sets. If there is only one input log data set, it is specified by the IMSLOGR DD statement. If there are multiple input log data sets, they must be listed in the order they were created with IMSLOGR as the ddname for the oldest log.

IEFRDER DD
Describes the system log created during backout. The data set resides on a tape or DASD.

IEFRDER2 DD
Describes the secondary system log created during backout. The data set resides on a tape or DASD. Include this statement only when dual log output is desired.

database DD
Describes the DD statements for the database data sets of the PCB requiring backout. This data set must reside on a direct-access volume, and can be dynamically allocated. If you are using dynamic allocation for the databases required by the PSB referenced in the EXEC statement, the database DD statements are not required.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I buffer handler. This DD statement is required.

Related Reading: For additional information on control statement format and buffer pool structure, see “Defining the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The data set can reside on a tape, a direct-access device, or be routed through the input stream (DD * or DD DATA).

This dataset also describes the parameters used to attach to the Coupling Facility (CF) in a Sysplex environment. The CFNAMES statement must match the entire CFNAMES statement of the IMS being backed out. Include CFIRLM, CFVSAM, and CFOSAM as described by the failed IMS. (See *IMS/ESA Installation Volume 2: System Definition and Tailoring* , “Defining Coupling Facility Structure names for Sysplex Data Sharing”.)

RECON1 DD
Defines the first DBRC RECON data set.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set as the control region is using.

If you are using dynamic allocation, do not use these RECON data set ddnames.

SYSIN DD
Defines the data set that contains the utility control statements. If no utility control statements are needed, this statement can be omitted. This data set can reside on a tape, a direct-access volume, or be routed through the input stream (DD * or DD DATA). The data set must be a physical sequential data set. The record format must be F, and the record length must be 80.

DFSCTL DD
Is an optional statement that points to the statement image file containing the SB control statements. This can be either a sequential data set or a member of a PDS. The record format must be either F, FB or FBS and the record length must be 80.

The SBIC control statement must be provided in the //DFSCTL input data set of the executed program in order to create the SB image capture log records necessary as input to this utility.

Related Reading: For more information, see the section about SB control statements in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Utility Control Statements

Eight optional utility control statements can be used by the Batch Backout utility:

- ABEND
- ABENDMSG
- ACTIVE
- BYPASS LOGVER
- BYPASS SEQVER
- CHKPT
- COLDSTART
- READBACK

You can provide control statements to the utility by including them in the SYSIN data set. See the SYSIN DD description above.

If no utility control statements are used for the backout of an IMS batch job, all database updates for the PSB named in the EXEC statement that occurred after the last checkpoint on the input log are backed out. If the IMS batch job did not issue any checkpoints, all database updates since the “IMS started” log record are backed out. If the input log does not contain either an “IMS started” record or a checkpoint record, the backout request is rejected.

If the input log comes from an IMS DB/DC or DBCTL subsystem and no control statements are used, the Batch Backout utility performs only backouts identified in the RECON backout record. If in-flight and in-doubt UORs are identified in the RECON backout record as a result of a previous Batch Backout run with either the COLDSTART or the ACTIVE control statement, Batch Backout backs out the UORs before restart has completed only when an appropriate control statement is specified. See the ACTIVE and COLDSTART control statement descriptions.

If DBRC is not used in a Batch Backout run in which the input log comes from an IMS DB/DC or DBCTL subsystem and no control statements are used, Batch Backout backs out only dynamic backout failures.

Recommendation: If your virtual storage area is limited, or if you receive an error indicating that not enough virtual private area storage is available, then use the READBACK statement. The READBACK statement is not needed for IMS batch input logs, however, since these jobs use READBACK automatically.

ABEND Statement

Use the ABEND statement to produce a U507 abend dump when a SYSUDUMP or SYSABEND DD statement is provided. This U507 abend is unconditional and it occurs at the end of the Batch Backout run.

This utility control statement has a fixed format. The positions are described below:

Position	Description
1-5	Statement ID Positions 1 through 5 must be the characters ABEND. These characters define the control statement.
6	This must be blank.
7-80	Positions 7 through 80 can contain comments.

ABENDMSG Statement

Use the ABENDMSG statement to produce a U507 abend dump when a SYSUDUMP or SYSABEND DD statement is provided. This U507 abend is issued when specific failures occur. This abend is always preceded by the failing message.

This utility control statement has a fixed format using the positions described below:

Position	Description
1-8	Statement ID Positions 1 through 8 must be the characters ABENDMSG. These characters define the control statement.
9	This must be blank.
10-80	Positions 10 through 80 can contain comments.

ACTIVE Statement

Use the ACTIVE statement to tell the utility to back out only in-flight UORs. This statement causes the utility to back out uncommitted updates from an incomplete batch message processing program (BMP) before an ERE NOBMP is done. If the ERE NOBMP has completed and DBRC is active, the utility performs the backout without the ACTIVE statement.

The ACTIVE control statement is valid for input logs from IMS DB/DC and DBCTL environments only. When this statement is used, Batch Backout compiles a list of all in-flight and in-doubt UORs from the information in the input log and passes that list to DBRC if DBRC does not already have that information.

This utility control statement has a fixed format using the positions described below:

Position	**Description**
1-6	Statement ID Positions 1 through 6 must be the characters ACTIVE. These characters define the control statement.
7-80	Positions 7 through 80 can contain comments.

BYPASS LOGVER Statement

Use the BYPASS LOGVER statement to allow backouts to be performed when RECON information indicates that the input log is invalid or backouts are not needed. The BYPASS LOGVER statement must also be used when DBRC=C is specified. If the BYPASS LOGVER statement is used when Batch Backout would normally pass a list of in-flight and in-doubt UORs for other PSBs to DBRC, as described in the ACTIVE and COLDSTART statements, Batch Backout does not compile the list. Batch Backout will inform DBRC of its activity concerning the PSB being backed out. The BYPASS LOGVER statement is valid for input logs from an IMS batch job or from IMS DB/DC and DBCTL environments.

This utility control statement has a fixed format using the positions described below:

Position	**Description**
1-13	Statement ID Positions 1 through 13 must be the characters BYPASS LOGVER. These characters define the control statement.
14-80	Positions 14 through 80 can contain comments.

BYPASS SEQVER Statement

Use the BYPASS SEQVER to inhibit the log record sequence check. The BYPASS SEQVER statement is valid for input logs from an IMS batch job or from IMS DB/DC and DBCTL environments.

This utility control statement has a fixed format using the positions described below:

Position	**Description**
1-13	Statement ID Positions 1 through 13 must be the characters BYPASS SEQVER. These characters define the control statement.
14-80	Positions 14 through 80 can contain comments.

CHKPT Statement

Use the optional CHKPT statement to identify an earlier checkpoint to backout. This control statement is valid only if the input log is from an IMS batch job that did not use IRLM.

Do not use the CHKPT statement when backing out a BMP. The Batch Backout utility always uses the first checkpoint that it comes to reading backward from the end of the log when backing out BMPs.

This utility control statement has a fixed format using the positions described below:

Position	Description
1-5	Statement ID Positions 1 through 5 must be the characters CHKPT. These characters define the control statement.
6	This position must be blank.
7-14	Checkpoint ID This is the 8-character checkpoint ID supplied to IMS with the CHKPT call. The ID is displayed as part of message DFS681I at the time the CHKPT call is made.
15	This position must be blank.
16-80	Positions 16 through 80 can contain comments.

COLDSTART Statement

Use the COLDSTART statement to back out all full-function databases with incomplete UORs, using the PSB named in the EXEC statement. When COLDSTART is specified, deferred backouts, in-flight UORs, and indoubt UORs are backed out for the PSB named.

The COLDSTART control statement is valid for input logs from IMS DB/DC and DBCTL environments only. When the COLDSTART statement is used, Batch Backout compiles a list of all in-flight and all in-doubt UORs from the information in the input log and passes that list to DBRC if DBRC does not have the information.

The COLDSTART statement is used to back out in-flight and in-doubt UORs before a COLDBASE or COLDSYS restart.

If you wait until after a COLDBASE restart to perform the backouts, use the COLDSTART statement when one of the following conditions is true:

Batch Backout is executed without DBRC.

The BYPASS LOGVER statement is used for the Batch Backout run.

If you wait until after a COLDSYS restart to back out an in-flight or indoubt UOR, use the COLDSTART statement to the Batch Backout utility when one of the following conditions is true:

- Batch Backout is executed without DBRC.
- The BYPASS LOGVER statement is used for the Batch Backout run.
- You have not specified the ACTIVE or COLDSTART statement in a run of Batch Backout, which causes a list of in-flight and indoubt UORs to pass to DBRC.

This utility control statement has a fixed format using the positions described below:

Position	Description
1-9	Statement ID Positions 1 through 9 must be the characters COLDSTART. These characters define the control statement.
10	This position must be blank.
11-80	Positions 11 through 80 can contain comments.

READBACK Statement

Use the READBACK statement to tell the utility to perform the backout by reading the log backwards, instead of saving the database changes in virtual storage during the forward reading of the log.

The READBACK statement is not needed for IMS batch input logs, since those jobs use read backward only.

Position	Description
1-8	Statement ID Positions 1 through 8 must be the characters READBACK. These characters define the control statement.
9	This position must be blank.
10-80	Positions 10 to 80 can contain comments.

Return Codes

The Batch Backout utility provides the following return codes. Each return code causes a message to be printed. The message that corresponds to each return code is shown below.

Code	Meaning
0	Backout successful (DFS395I).
4	PSB incorrect (DFS396I).
8	Unable to open database (DFS397I).
12	Database I/O error (DFS398I).
16	Buffer pool too small (DFS399I).
20	Unable to open input log (DFS400I).
24	Call to DBRC failed (DFS401I).
28	No database record in log (DFS888I). If the input logs come from an online IMS subsystem, this code indicates that no UOR fitting the backout criteria was found in the logs; check to make sure that you have indicated the correct logs and the correct PSB name. If the PSB name and logs are correct, you can review the logs to verify that the records were committed successfully, and that batch backout is unnecessary.
32	No block size for IMSLOGR DD statement (DFS890I).
36	Invalid record on input log (DFS894I).
40	Unexpected record encountered (DFS896A).
48	Specified CHKPT not found (DFS958I).
52	Specified CHKPT not within last schedule (DFS959I).
60	Input log was specified as DD DUMMY or multiple log DD statements were specified, but correct ddname and/or order was not specified (DFS2296A).
64	Backout processing incomplete for PSB (DFS2298A).
68	Log sequence error found in input log (DFS3278I).

72	Invalid option statement in SYSIN (DFS898A).
80	A control statement was found in the SYSIN data set that is not compatible with the type of input log.
88	Backout incomplete for PSB psbname databases (DFS3283A).
96	Two different control statements were found in the SYSIN data set. They cannot be used together.
108	Either the log data sets are not in correct order in the JCL, or the log data set indicated in the message has incorrect data.
112	The RECON backout record does not identify any pending backout fitting the criteria implied by the input control statements and the EXEC PARM (DFS3290I).
120	Batch Backout was given a control statement which implies that it is to perform the same backout that an in-progress restart performs (DFS3292I).
124	Batch Backout performed a backout which the RECON backout record did not show as necessary (DFS3293W).
128	Input log not valid for backout (DFS3294A).
132	The READBACK control statement was used. A system checkpoint indicates that a backout is needed, but the original log data is not in the log data sets provided as input (DFS3295A).
140	You have specified DBRC=C without including a BYPASS LOGVER Utility Control statement in your SYSIN data set (DFS3296A).

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

The following return codes are issued by DFSBCKIO during backout initialization and cause ABENDU007I to be issued from DFSXBAT0:

Code	Meaning
20	Unable to open IMSLOGR DD (DFS400I) or Permanent I/O error occurred (DFS319I).
56	Incorrect PSB on EXEC parameters (DFS428I).
60	IMSLOGR DD DUMMY or device type not recognized (DFS2296A).
68	DBRC is required for this execution of backout (DFS044I).
72	IRLM is required for this execution of backout (DFS045I).
76	Unable to locate X'42' log record (DFS091I).
80	A normally completed job cannot be backed out if IRLM was active (DFS173I).

Example

The following example shows the JCL for backout of database changes made by a program that uses PSB PLVAPZ12.

```
//BTCHBO   JOB
//JOBLIB    DD DSNAME=IMS.RESLIB,DISP=SHR
//JOBCAT    DD DSNAME=VCATSHR,DISP=SHR
//*
```

```
//EXAMPLE EXEC PGM=DFSRRC00,REGION=156K,
//             PARM='DBB,DFSBBO00,PLVAPZ12,008,1,,,,,,,IMSS,,Y,Y,IRLM'
//SYSABEND  DD SYSOUT=A
//IMSACB    DD DSNAME=IMS.ACBLIB,DISP=SHR
//IEFRDER   DD DSNAME=LGBKOUTF,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=000000,
//             DCB=(RECFM=VB,BLKSIZE=8192,LRECL=8188,BUFNO=12),
//             SPACE=(CYL,(1,1))
//IMSLOGR   DD DSNAME=DSHR.OLDSP0,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSQAD
//IMSLOG00  DD DSNAME=DSHR.OLDSP1,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSQAD
//IMSLOG01  DD DSNAME=DSHR.OLDSP2,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSQAD
//DFSVSAMP  DD *
VSRBF=512,50
VSRBF=1024,50
VSRBF=2048,50
//DBHVSAM1  DD DSNAME=DSHR.FJVHSG1K,DISP=SHR
//DBHVSAM2  DD DSNAME=DSHR.FJVHSG1E,DISP=SHR
//HIDAM     DD DSNAME=DSHR.FKVHIG1E,DISP=SHR
//HIDAM2    DD DSNAME=DSHR.FKVHIG2E,DISP=SHR
//XDLBT04I  DD DSNAME=DSHR.FKVHIIXK,DISP=SHR
//RECON1    DD DSNAME=RCONDFI1,DISP=SHR
//RECON2    DD DSNAME=RCONDFI2,DISP=SHR
```

Chapter 22. MSDB Dump Recovery Utility (DBFDBDR0)

Use the MSDB Dump Recovery utility to create a new copy of MSDBINIT. This utility employs either the dump data set (MSDBDUMP) or the checkpoint data sets (MSDBCP1 and MSDBCP2) to accomplish this. In an XRF environment, the checkpoint data sets can be either MSDBCP1 and MSDBCP2 or MSDBCP3 and MSDBCP4. DBFDBDR0 is an offline utility that runs in an MVS batch region.

Requirement: You must run the same release level of the MSDB Dump Recovery utility as you used to create the MSDB dump or checkpoint data set.

If the system terminates abnormally and emergency restart is unable to recover the MSDBs, the Dump Recovery utility is used in RECOVERY mode to reconstruct the MSDBs. The MSDBs are reconstructed using the checkpoint data sets and the associated system log. From this input the utility produces a new copy of the MSDBINIT data set. Whenever a new MSDBINIT is needed, containing one or more selected MSDBs, the MSDB Dump Recovery utility is used in UNLOAD mode with the MSDBDUMP data set as input.

To unload MSDBs:

- The MSDBDUMP data must be supplied
- The IMS log data set is not used
- The control data set must specify UNLOAD

The MSDBs are unloaded onto the MSDBINIT data set at the same level of change as they were dumped.

To reconstruct MSDBs:

- The MSDB checkpoint data sets (MSDBCP1 and MSDBCP2) must be supplied. The utility determines which is the older valid image copy and recovers from that point.

  In an XRF environment, the MSDB checkpoint data sets (MSDBCP1, MSDBCP2, MSDBCP3, and MSDBCP4) must be supplied. The utility chooses the later pair of MSDB checkpoint data sets. Within that pair, the utility determines which is the older valid image copy, and recovers from that point.
- The IMS log data set containing MSDB changes logged since the older checkpoint data set was created can be supplied.
- Ensure that the control data set specifies RECOVERY.

If both the checkpoint data set DD statements specify the same data set, the utility recovers from the checkpoint in that data set.

Because the format of the MSDBDUMP data set is identical to that of the checkpoint data sets, the user can force recovery from the MSDBDUMP data set by specifying that data set in both checkpoint DD statements.

The MSDBs are reconstructed on the MSDBINIT data set by applying all the logged changes to the image copy data set.

Input and Output

Figure 39 shows the input and output data sets used by the MSDB Dump Recovery utility for unloading and reconstructing MSDBs.

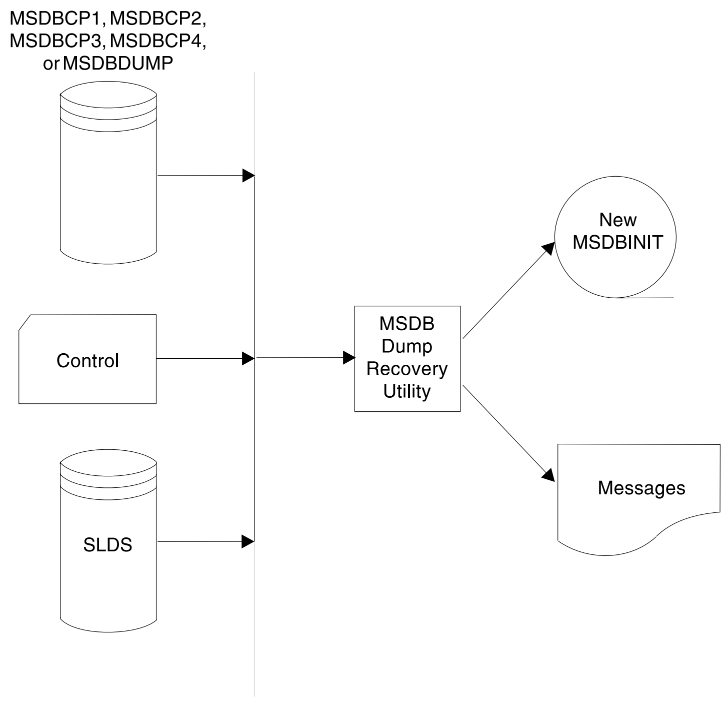


Figure 39. MSDB Dump Recovery Utility Input and Output Data Sets

The input data sets are:

- An MSDB dump data set (MSDBDUMP) or the MSDB checkpoint data sets (MSDBCP1 and MSDBCP2). The dump data set is created by the `/DBDUMP` command. A checkpoint data set is written automatically at each system checkpoint: the output data set alternates on successive checkpoints between MSDBCP1 and MSDBCP2.

  In an XRF environment, the MSDB checkpoint data sets are MSDBCP1, MSDBCP2, MSDBCP3, and MSDBCP4. The output data set alternates on successive checkpoints between either MSDBCP1 and MSDBCP2 or MSDBCP3 and MSDBCP4.
- An IMS system log data set (optional).
- A control data set (optional). See “Utility Control Statements” on page 209.

The primary output is a new MSDBINIT data set. A message data set is also produced.

JCL Requirements

The following are required to run the MSDB Dump Recovery utility:

- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

This statement can either specify a procedure that contains the required JCL, or be in the form:

```
PGM=DBFDBDR0
```

Requirement: The utility requires at least 512KB of virtual storage.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules.

MSDBDUMP DD
Describes a data set that contains a dump of all or selected MSDBs.

MSDBCP1 DD
Describes a checkpoint data set that contains an image copy of all MSDBs.

MSDBCP2 DD
Describes a checkpoint data set that contains an image copy of all MSDBs.

MSDBCP3 DD
Describes a checkpoint data set in an XRF environment that contains an image copy of all MSDBs.

MSDBCP4 DD
Describes a checkpoint data set in an XRF environment that contains an image copy of all MSDBs.

IEFRDER DD
Describes the data set that contains the old IMS system log, which is used for recovery operations.

This data set can reside on multiple volumes. The volumes containing the checkpoint for the older of the two image copies and all later volumes must be mounted and specified in the DD statement.

MSDBCTL DD
Describes a file containing control statements in 80-byte records. This statement is optional.

MSDBPRT DD
Describes the message data set. It can reside on a tape, a direct-access device, or a printer, or be routed through the output stream.

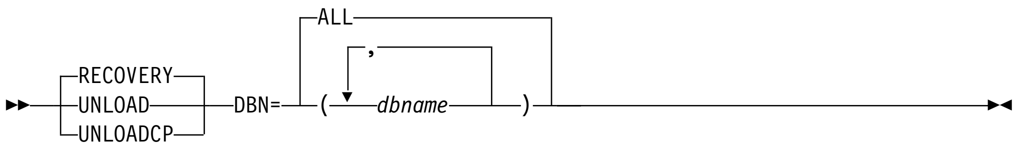
MSDBINIT DD
Describes the data set that contains the unloaded or reconstructed MSDBs after the utility executes.

Utility Control Statements

The control statements for the Dump Recovery utility specify the input, and whether the utility unloads or reconstructs the MSDBs. These control statements are read from the MSDBCTL data set.

The control statements are free form. The control information is contained in columns 1 through 71. The list of MSDB names can be continued by inserting a comma after the last name of the statement, inserting a nonblank character in column 72, and continuing the list in column 16 of the next statement. (Columns 1 through 15 must be blank.) Comments can be included as illustrated in the examples following this section.

Utility Control Statements



RECOVCP
Specifies that the failing MSDBCPx dataset is to be recreated from the remaining error-free MSDBCPx dataset; or, in the case of an XRF-generated IMS system, from the MSDBCPx dataset that contains the most recent check-pointed data.

RECOVERY
Specifies that the MSDBs are to be reconstructed from the older MSDB checkpoint data set (MSDBCP1 or MSDBCP2) and the associated IMS system log.

To force a recovery from a particular checkpoint data set, both MSDBCP1 and MSDBCP2 DD statements must specify the same data set.

UNLOAD
Specifies that the MSDB dump data set defined by the MSDBDUMP DD statement is to be unloaded onto an MVS sequential data set (MSDBINIT). No updates from the log are applied.

UNLOADCP
Specifies that the MSDB checkpoint data set defined by the MSDBCPX DD statement is to be unloaded onto an MVS sequential data set (MSDBINIT). No updates from the log are applied.

DBN=ALL
Specifies that all MSDBs in the data set are to be unloaded or reconstructed, according to the operation specified.

DBN=(dbname,...)
Specifies which MSDBs in the data set are to be unloaded or reconstructed, according to the operation specified.

If DBN= is not supplied, DBN=ALL is assumed.

If no control statements are supplied, defaults are RECOVERY and DBN=ALL.

Return Codes

The MSDB Dump Recovery utility provides the following return codes:

Code	**Meaning**
0	Utility executed successfully
4	Error—error message printed
8	Unable to open print data set
12	I/O error in print routine

Examples

The following examples show the JCL and utility control statements needed to execute the MSDB Dump Recovery utility.

Example 1

Example 1 unloads all MSDBs from the MSDBDUMP data set onto the MSDBINIT data set.

```
//UN101   EXEC PGM=DBFDBDR0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSNAME=IMS.MSDBLM02,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBDUMP  DD DSNAME=IMS.LMDMP,DISP=SHR     MSDB DUMP
//*                             UNLOAD ALL MSDB'S
//MSDBCTL   DD *
  UNLOAD  DBN=ALL
/*
```

Example 2

Example 2 reconstructs the MSDBs from the older checkpoint data sets and the IMS system log.

```
//RC101   EXEC PGM=DBFDBDR0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSNAME=IMS.MSDBLM03,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBCP1   DD DSNAME=IMS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2   DD DSNAME=IMS.LMCP2,DISP=SHR      CHECKPOINT # 2
//IEFRDER   DD DSNAME=IMS.LMLOG,DISP=SHR      IMS OLDS/SLDS
//*                             RECOVER ALL MSDB'S
//MSDBCTL   DD *
   RECOVERY DBN=ALL
/*
```

Example 3

Example 3 unloads one particular MSDB from the MSDBDUMP data set onto the MSDBINIT data set.

```
//UN201   EXEC PGM=DBFDBDR0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSNAME=IMS.MSDBLM02,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBDUMP  DD DSNAME=IMS.LMDMP,DISP=SHR     MSDB DUMP
//*                             UNLOAD ONLY MSDB 05
//MSDBCTL   DD *
  UNLOAD  DBN=(MSDBLM05)
/*
```

Example 4

Example 4 reconstructs one particular MSDB from the older checkpoint data set and the IMS system log.

```
//RC202   EXEC PGM=DBFDBDR0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSNAME=IMS.MSDBLM03,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBCP1   DD DSNAME=IMS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2   DD DSNAME=IMS.LMCP2,DISP=SHR      CHECKPOINT # 2
//IEFRDER   DD DSNAME=IMS.LMLOG,DISP=SHR      IMS OLDS
//*                             RECOVER ONLY MSDB 05
//MSDBCTL   DD *
  RECOVERY DBN=(MSDBLM05)
/*
```

Example 5

Example 5 reconstructs one particular MSDB from the older checkpoint data set of the later pair and the IMS system log.

```
//RC202   EXEC PGM=DBFDBDR0
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSNAME=IMS.MSDBLM03,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBCP1   DD DSNAME=IMS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2   DD DSNAME=IMS.LMCP2,DISP=SHR      CHECKPOINT # 2
//MSDBCP3   DD DSNAME=IMS.LMCP3,DISP=SHR      CHECKPOINT # 3
//MSDBCP4   DD DSNAME=IMS.LMCP4,DISP=SHR      CHECKPOINT # 4
//IEFRDER   DD DSNAME=IMS.LMLOG,DISP=SHR      IMS OLDS
//*                             RECOVER ONLY MSDB 06
//MSDBCTL   DD *
  RECOVERY DBN=(MSDBLM06)
/*
```

Example 6

Example 6 shows the JCL for recreating a new MSDBCP1 dataset using the RECOVCP utility control statement.

```
  //RECOVCP   EXEC PGM=DBFDBDR0
  //STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
  //SYSUDUMP  DD SYSOUT=A
  //MSDBPRT   DD SYSOUT=A
  //@               MESSAGE PRINT FILE
  //NEWCP     DD DSNAME=IMS.LMCP1,         new CHECKPOINT DS #1
  //             DISP=(NEW,CATLG,DELETE),
  //             UNIT=SYSDA,VOL=SER=IMSDCL,
  //             DCB=(BLKSIZE=2048,RECFM=F,LRECL=2048),
  //             SPACE=(CYL,1)
  //MSDBCP1   DD DSNAME=IMS.LMCP2,DISP=SHR     CHECKPOINT DS #2
  //MSDBCP2   DD DSNAME=IMS.LMCP2,DISP=SHR     CHECKPOINT DS #2
  //@      IF XRF ENVIRONMENT, UNCOMMENT NEXT TWO LINES
  //@MSDBCP3   DD DSNAME=IMS.LMCP3,DISP=SHR    CHECKPOINT DS #3
  //@MSDBCP4   DD DSNAME=IMS.LMCP4,DISP=SHR    CHECKPOINT DS #4
  //IEFRDER   DD DSNAME=IMS.LMLOG,DISP=SHR     IMS OLDS/SLDS
```

```
//@                              RECOVER ALL MSDB'S
//MSDBCTL   DD *
   RECOVCP
/@
```

Note: In the above example, `//@` denotes a JCL comment card and `/@` denotes the EOF.

Chapter 23. DEDB Area Data Set Create Utility (DBFUMRI0)

Use the DEDB Area Data Set Create utility to create one or more copies from multiple DEDB area data sets during online transaction processing. Application programs can continue during the copying process. The DEDB Initialization utility is not required. During the copying process, application programs can continue. Two or more data sets with defective control intervals (CIs) can be used by this utility to produce a copy free of defective CIs. Writes requested from application programs are performed to both the **available** and the new data sets.

The need for multiple area data sets is reduced for VSO areas. Once a CI from a VSO area is put into an MVS data space, it is available to applications as long as IMS is active. The CI is retrieved from the data space for all read requests, thus eliminating read errors. Updates to the CI are written to both the data space and to DASD. If a write error occurs during the DASD write, the CI remains available for subsequent reads from and updates to the data space.

Despite the reduced need for multiple copies, you can use this utility to create additional copies of a VSO area. The performance of the utility depends on the number of CIs present in the data space at the time the utility is executed. If the area was preloaded or is heavily accessed, all or many of the CIs can be retrieved from the data space (rather than from DASD) and copied to the new area data sets, which improves read time.

Any CIs not present in the data space when the utility is started are copied into the data space as they are read from DASD. When the utility terminates, all CIs for the area are in the data space, thus eliminating the need for any future reads from DASD.

The sequential dependent (SDEP) part of an area is not kept in the data space, so SDEPs must still be read from DASD to be copied to a new area data set.

The write function of the utility is not affected by the Virtual Storage Option.

Restrictions

The following restrictions apply when using the DEDB Area Data Set Create utility:

- Prior to starting this utility, the new data sets must be defined to DBRC using the DBRC `INIT.ADS` command. The new area data set (ADS) must be unavailable in the DBRC RECON data set.
- You must execute the Create utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.
- The data set is defined by the VSAM definition. The size of the CI must be identical to that defined for the other data sets of the same area. If the allocated space for the data set is not equal to or greater than that defined for the other data sets of the same area, the data set is extended by the primary allocation until enough space has been allocated. Secondary allocation is ignored for Fast Path data sets. If the data set is defined with multiple volumes, no space is allocated on the next volume until all space on the previous volume has been used.
- If a read error is detected in a CI of an available data set, another data set in the same area is used. If all the available data sets have read errors in the same CI, this utility terminates.

- If a write error is detected in the new data set, this utility terminates.
- A /STO REGION issued while the DEDB Area Data Set Create utility is running is not processed until the format phase ends.
- In a data sharing environment, if the DEDB Area Data Set Create utility is running in the data sharer, the takeover process is suspended until the format phase ends.

Recommendation: For XRF users, to prevent the last two restrictions from occurring, preformat the new ADS using DBFUMIN0 with DBRC=N. This causes the DEDB Area Data Set Create utility to skip the format phase. DBRC=N on DBFUNIN0 is required to prevent the new ADS from becoming available in DBRC before the copy phase of the DEDB Area Data Set Create utility is processed.

Input and Output

The DEDB Area Data Set Create utility uses the following input:

DBRC RECON data set

A data set that contains the input parameters supplied by commands (see “Appendix A. Summary of DEDB Utility Commands” on page 329 for more detail on DEDB online utility commands)

The DEDB Area Data Set Create utility produces the following output:
- One or more copies of AVAILABLE data sets
- A data set that contains output messages and statistics

Recovery and Restart

Restriction: This utility cannot be restarted. It must be rerun from the beginning.

The utility can be stopped before processing has completed by using the `/STOP REGION` command.

You can use the Database Recovery, Database Image Copy, Change Accumulation utilities, and/or the system logs for recovery purposes.

JCL Requirements

The following are required to run the DEDB Area Data Set Create utility:
- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

This statement can either specify the FPUTIL procedure which contains the required JCL or be in the form:

```
PGM=DFSRRC00
```

Related Reading:
- See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on FPUTIL.
- See “Appendix A. Summary of DEDB Utility Commands” on page 329 for more information on the DEDB online utility commands.

DD Statements

STEPLIB DD
Describes the library that contains the DEDB Data Set Create utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. This statement is required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement.

Related Reading: See “Appendix A. Summary of DEDB Utility Commands” on page 329 for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

Examples

The following examples provide sample JCL for creating one, two, or five new area data sets.

Example 1

Example 1 shows sample JCL and utility control statements for creating one new area data set.

```
//CREATE1  JOB
//*
//UTL1    EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN     DD *
TYPE CREATE
AREA DB23AR3
DDNAME DB23AR33
GO
/*
```

Example 2

Example 2 shows sample JCL and utility control statements for creating two new area data sets.

```
//CREATE2  JOB
//*
//UTL2    EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN     DD *
TYPE CREATE
AREA DB23AR3
DDNAME DB23AR33
DDNAME DB23AR34
GO
/*
```

Example 3

Example 3 shows sample JCL and utility control statements for creating five new area data sets.

```
//CREATE4  JOB
//*
//UTL4    EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN     DD *
TYPE CREATE
AREA DB23AR3
DDNAME DB23AR33
DDNAME DB23AR34
DDNAME DB23AR35
DDNAME DB23AR36
DDNAME DB23AR37
GO
/*
```

Chapter 24. DEDB Area Data Set Compare Utility (DBFUMMH0)

Use the DEDB Area Data Set Compare utility to compare the physical records of two or more area data sets (ADSs) of an area. This online utility compares the control intervals (CIs) of:

- The root addressable part and the independent overflow part
- Reorganized unit of work (UOW)
- Sequential part, if the sequential dependent segment is defined in the AREA statement at DBDGEN time

In case of unequal comparison, full dumps of up to 10 unmatched records are printed onto the media specified by the SYSPRINT DD statement. Comparison processing then continues until the end of the area data sets.

At the end of the comparing process, a message is printed on a SYSPRINT data set that shows the number of unmatched CIs and the number of identical CIs. This utility also checks the error queue element (EQE) in each specified area data set and prints the EQE status of an area on the SYSPRINT data set.

The comparison of each CI is done only when the to-be-compared CI count is equal to or greater than two. The to-be-compared CI count is decreased by one each time an I/O error or an EQE is detected.

This utility can be stopped immediately by a `/STOP REGION` command.

Restrictions

The following restrictions apply to the DEDB Area Data Set Compare Utility:

- You must execute the DEDB ADS Compare utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.
- This utility does not compare:
  - The control CIs (first and second CI of an area)
  - All CIs in a residual part, when the space defined at DBDGEN is smaller than that defined by the VSAM definition
- The DEDB Area Data Set Compare utility terminates if:
  - The to-be-compared area data set or area is stopped by a `/STOP AREA` or `/STOP ADS` command
  - The to-be-compared area data set or area is stopped by an internal stop command
  - The to-be-compared area data set count has decreased to one

Input and Output

The DEDB Area Data Set Compare utility uses a data set that contains the input parameters supplied by commands as input.

Related Reading: See "Appendix A. Summary of DEDB Utility Commands" on page 329 for more details on DEDB online utility commands.

DEDB Compare

The DEDB Area Data Set Compare utility produces the following output:

- A printed dump of up to 10 unmatched records on the SYSPRINT data set
- An error queue element (EQE) status report on the SYSPRINT data set
- A message showing the number of unmatched CIs and the number of identical CIs, also printed out on the SYSPRINT data set

Recovery and Restart

This utility can be stopped immediately by a `/STOP REGION` command.

The Compare utility cannot be restarted. If the restart (REST) parameter is used, it is ignored.

You can use the Database Recovery, Database Image Copy, Database Image Copy 2, Change Accumulation utilities, and/or the system logs for recovery purposes.

JCL Requirements

The following are required to run the DEDB Area Data Set Compare utility:

- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

This statement can either specify the FPUTIL procedure that contains the required JCL or be in the form:

```
PGM=DFSRRC00
```

Related Reading:

- For more information on the DEDB online utilities commands, see “Appendix A. Summary of DEDB Utility Commands” on page 329.
- For more information on FPUTIL, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DD Statements

STEPLIB DD
Describes the library that contains the DEDB Area Data Set Compare utility.

STEPCAT DD
Describes a private VSAM user catalog that is searched first. Required if the defined areas are cataloged in a user catalog.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

SYSIN DD
Describes the input control data set that contains the utility control statement.

Related Reading: See “Appendix A. Summary of DEDB Utility Commands” on page 329 for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains output messages and statistics.

Examples

The following examples provide sample JCL for comparing one, seven, or all available area data sets.

Example 1

Example 1 shows sample JCL and utility control statements for comparing two area data sets.

```
//COMP1   JOB
//*
//UTL101 EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN    DD *
TYPE COMPARE
AREA DB23AR2
DDNAME DB23AR21
DDNAME DB23AR22
GO
/*
```

Example 2

Example 2 shows sample JCL and utility control statements for comparing seven area data sets.

```
//COMP2   JOB
//*
//UTL102 EXEC FPUTIL,DBD=DEDBJN23
//*
//*       DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN    DD *
TYPE COMPARE
AREA DB23AR2
DDNAME DB23AR21
DDNAME DB23AR22
DDNAME DB23AR23
DDNAME DB23AR24
DDNAME DB23AR25
DDNAME DB23AR26
DDNAME DB23AR27
GO
/*
```

Example 3

Example 3 shows sample JCL and utility control statements for comparing all available area data sets of an area.

```
//COMP3   JOB
//*
//UTL103 EXEC FPUTIL,DBD=DEDBJN23
//*
//*      DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN    DD *
TYPE COMPARE
AREA DB23AR2
GO
/*
```

Example 4

The following is an example of the error queue element (EQE) status report:

```
AREA EQE STATUS

         RBA     ADS01    ADS02    ADS03    ADS04
 1. 00000800       *
 2. 00001400       *                          *
 3. 00001800                *
 4. 00001C00       *                 *
 5. 00002000       *                          *
 6. 00002400                                  *
 7. 00002800       *        *        *        *
 8. 00002C00                                  *
 9. 00003000                                  *
10. 00003400                                  *

 EQE COUNT         5        2        2        7

 NO DATA AVAILABLE FOR CI STARTING AT 00000800
```

Example 5

The following is a sample printed dump of the unmatched records:

```
UNMATCHED CI AT     AREA=AREA01    RBA=00000800

ADS02    0000  F0F1F2F3..................C1C2C3C4  *0123......ABCD*
ADS03    0000  F0F1F2F3..................A1C2C3C4  *0123.......BCD*
ADS04    0000  F0F1F2F3..................C1C2C3C4  *0123......ABCD*

ADS02    0020  F4F5F6F7..................C5C6C7C8  *4567......EFGH*
ADS03    0020  00000000..................C5000000  *..........E...*
ADS04    0020  F4F5F6F7..................C5C6C7C8  *4567......EFGH*

ADS02    03C0  F4F5F6F7..................C5C6C7C8  *4567......EFGH*
ADS03    03C0  00000000..................C5000000  *..........E...*
ADS04    03C0  F4F5F6F7..................C5C6C7C8  *4567......EFGH*

ALL      03E0  D1D2D3D4..................E2E3E4E5  *JKLM......STUV*

DFS3753I  COMPARISON NOT PERFORMED FOR ADS=ADS01
          REASON: NO DATA AVAILABLE DUE TO EQE
```

Part 4. Conversion Utilities

Chapter 25. MSDB-to-DEDB Conversion Utility (DBFUCDB0)

DBFUCDB0 (MSDB-to-DEDB Conversion utility)

Use the MSDB-to-DEDB Conversion utility to convert an existing main storage database (MSDB) to a data entry database (DEDB). You can also use the utility to fall back from a DEDB to an MSDB. The utility is an offline utility and runs in an MVS batch region.

Conversion

Only nonterminal-related MSDBs can be converted into DEDBs.

Only one MSDB can be converted for each execution of the utility. Each MSDB is converted into a single DEDB area. Each MSDB segment is converted into a level '01' segment.

No consistency checking of fields occurs between the MSDB and DEDB DBD.

There is no checking for the proper ascending order of the RAP RBA values output by the conversion utility in DBT unload format.

Conversion Process

Conversion requires the use of the MSDB Dump Recovery utility and either the Database Tools (DBT) DEDB Unload/Reload utility or an equivalent utility.

To perform the conversion:

1. Create a DEDB DBD for the MSDB to be converted.
2. Run the DBDGEN and ACBGEN utilities to create a new ACBLIB.
3. Run the MSDB Dump Recovery utility to create the MSDBINIT data set. MSDBINIT contains the MSDB segments that are to be loaded into the new DEDB area.
4. Run the MSDB-to-DEDB Conversion utility with the CONVERT option specified. This creates a data set in DBT unload format for each MSDB that is being converted.
5. Sort the data set in DBT unload format for each MSDB by ascending order of the RAP RBA values.
6. Run the DBT DEDB reload utility to load these data sets into a DEDB.

DBD Changes Required for Conversion

You must make the following DBD changes for MSDB-to-DEDB conversion:

- Change the ACCESS parameter in the MSDB DBD from MSDB to DEDB
- Specify the RMNAME parameter for the randomizer name
- Specify an AREA statement for the DEDB area

You do not need to change the BYTES parameter because the single value that has already been specified for the MSDB implies a fixed-length segment.

Related Reading: For more information on this topic, see *IMS/ESA Administration Guide: Database Manager*.

Fallback

Fallback requires use of the MSDB Maintenance utility and a DEDB unload/reload utility (such as the DBT DEDB Unload/Reload utility).

To perform fallback:

1. Run the unload utility to unload the DEDB. This creates the input for the MSDB-to-DEDB Conversion utility.
2. Run the MSDB-to-DEDB Conversion utility with the FALLBACK option specified. This creates the MSDBLCHG data set, which contains change records that will be input to the MSDB Maintenance utility.
3. Run the MSDB Maintenance utility to create a new MSDBINIT data set. MSDBINIT can then be used to reload the MSDB into storage.

JCL Requirements

EXEC
Executes the MSDB-to-DEDB Conversion utility. This statement must be in the form:

```
//STEPNAME EXEC PGM=DBFUCDB0
```

DD Statements

STEPLIB DD
Defines IMS.RESLIB, which contains the IMS nucleus and required action modules.

MACBLIB DD
Defines the **MSDB** IMS ACB library. In this library, the DMB for the database to be converted specifies it as an MSDB.

DACBLIB DD
Defines the **DEDB** IMS ACB library. In this library, the DMB for the database to be converted specifies it as a DEDB.

MSDBINIT DD
Defines the MSDBINIT data set containing MSDB segments that will be loaded into the new DEDB area. This data set is used as input when the CONVERT option has been specified. This DD statement is required **only** for the CONVERT option, not for the FALLBACK option.

DURDBDFN DD
Defines a data set that will contain a formatted copy of the DMB that is used by the DBT reload processor.

MSDBLCHG DD
Defines a data set that will contain change records in MSDBINIT record format. This data set will be used as input to the MSDB Maintenance utility to create an MSDBINIT data set. (The MSDBINIT data set is used to reload the MSDB into storage when fallback is being performed.) MSDBLCHG is the output data set when the FALLBACK option has been specified. This DD statement is required **only** for the FALLBACK option, not for the CONVERT option.

areaname **DD**
Defines a data set that will contain MSDB converted data that will be used by the reload utility to load the new DEDB area. One DD statement is required for

each AREA name in the DBDGEN input control statement. The ddname on the JCL statement **must** be the same as the AREA name in the input control statement.

SYSIN DD
Defines the input control statement data set. See “Utility Control Statements” for a description of the control statement format.

SYSOUT DD
Defines the output message data set used for messages from the DFSORT program.

SORTWKnn DD
Defines the intermediate storage data sets used by the DFSORT program.

Related Reading: See *DFSORT Application Programming Guide* for more information on how to code SORTWKnn DD statements.

SYSPRINT DD
Defines the output data set containing error messages, return codes, and the summary report for this utility.

Utility Control Statements

This utility requires two control statements. They are:

TYPE=CONVERT|FALLBACK
Indicates the execution mode for the utility, conversion or fallback. **TYPE** starts in column 1.

CONVERT
Indicates the utility will be in conversion mode. An MSDB will be converted to a DEDB.

FALLBACK
Indicates the utility will be in fallback mode. A DEDB will be converted back to an MSDB.

MSDB=*xxxxxxxx***,DEDB=***yyyyyyyy*
Indicates which database is to be processed. The conversion utility will convert one database at a time. This control statement starts in column 1.

The format of this statement is:

```
MSDB=xxxxxxxx,DEDB=yyyyyyyy
```

For the CONVERT option, the MSDB parameter is the name of the source MSDB to be converted. The DEDB parameter is the name of the destination DEDB. The DEDB and MSDB name do not have to be the same.

For the FALLBACK option, the MSDB parameter is the name of the destination MSDB for the fallback procedure. The DEDB parameter is the source DEDB for fallback. The DEDB and MSDB name do not have to be the same. The AREA parameter in the DBD contains the name of the area in the DEDB that is to be unloaded.

Summary Report

This utility generates a summary report when it completes execution. The report is sent to the SYSPRINT data set.Figure 40 shows an example of the report.

```
               CONVERSION UTILITY SUMMARY REPORT

 TYPE OF CONVERSION = CONVERT/FALLBACK

 TOTAL NUMBER OF RECORDS PROCESSED FOR AREA XXXXXXXX IS NNNN

 XXXXXXXX = the name of the area that was processed
 NNNN     = the total number of segments processed for the area
```

Figure 40. MSDB-to-DEDB-Conversion-Utility-Summary Report

Using a Utility Other Than DBT Unload/Reload

The MSDB-to-DEDB conversion utility uses the DBT Unload/Reload utility. You can use any equivalent product to unload and reload your DEBDs.

The conversion utility provides an exit routine that does the actual conversion of the MSDB record into DBT format. The main routine (DBFUCDB0) reads an MSDB record from the MSDBINIT data set and calls the exit routine to do the conversion. The DBFUCDX0 exit routine converts the MSDB record to DBT format and returns the converted record back to the main routine. The main routine writes the converted record to the output data set.

You can change this exit routine to format the MSDB record to a format other than DBT.

The fallback option also has an exit routine (DBFUCDX1) that converts the unloaded records to MSDBINIT format. This routine **must** be changed if a format other than DBT is used.

Parameter List for the Convert Exit Routine

The exit routine for converting to unload format (DBFUCDX0) is called with a parameter list. The first word in the list is the function code. The function codes are:

- **0**, for the INIT function. This function gets called once, at the start of the conversion process. This function can be used to do any special setup processing required.
- **4**, for the CONVERT function. This function contains the address of the input MSDB record to convert and the output location for the converted record. This function code has other information along with it. The following list shows which address is in the words following the function code.

Word	Contents
2	Address of MSDBINIT record to convert
3	Address of output area for the converted record
4	Address of DMCB mapped by DBFDMCB
5	Address of DMAC mapped by DBFDMAC
6	Address of BHDR mapped by DBFBMSDB

- **8**, for the TERM function. This function gets called once, at the end of the conversion process. This function can be used to do any special cleanup processing required.

Parameter List for the Fallback Exit Routine

The exit routine for executing the fallback procedure (DBFUCDX1) converts a record in unloaded format to MSDBINIT format and is called with a parameter list. The first word in the list is the function code. The function codes are:

- **0**, for the INIT function. This function gets called once, at the start of the conversion process. This function can be used to do any special setup processing required.
- **4**, for the FALLBACK function. This function contains the address of the input MSDB record to convert and the output location for the converted record. This function code has other information along with it. The following list shows which address is in the words following the function code.

Word	Contents
2	Address of input record to convert
3	Address of output area for the converted record

- **8**, for the TERM function. This function gets called once, at the end of the conversion process. This function can be used to do any special cleanup processing required.

Procedure for Using a New Exit Routine

You can use your own exit routine for these conversions. The process is as follows:

1. Write a new exit routine to convert MSDBINIT records to new unload format. This routine must have the same name as the routine provided (DBFUCDX0).
2. Link-edit the new DBFUCDX0 into the conversion utility load module.
3. Prepare JCL to execute the Unload/Reload utility.

Use the same procedure for the FALLBACK option. The new exit routine must be called DBFUCDX1.

Error Processing

User abend codes are not generated.

These return codes are provided:

Code	Meaning
1	The TYPE= statement is missing or invalid
2	The database statement is invalid
3	The MSDB ACB library (MACBLIB) indicates that the database specified on the input control statement is not an MSDB
4	The MSDB specified on the input control statement is not a nonterminal-related MSDB
5	The DEDB ACB library (DACBLIB) indicates that the database specified on the input control statement is not a DEDB
6	The MSDB= member specified was not found in the MSDB ACB library (MACBLIB)

7	The DEDB= member specified was not found in the new DEDB library (DACBLIB)
8	An error was detected loading randomizer module *xxxxxxxx*
9	The SYSIN DD statement is missing
10	The SYSPRINT DD statement is missing
11	The MSDBINIT DD statement is missing
12	The AREADCB DD statement is missing
13	The MACBLIB DD statement is missing
14	The DACBLIB DD statement is missing
15	The MSDBLCHG DD statement is missing
16	The AREAIN DD statement is missing

Related Reading: For an explanation of the messages generated by this utility, see *IMS/ESA Messages and Codes* .

Examples

The following examples show the JCL necessary to run this utility for both conversion and fallback.

Example 1 (Conversion)

This is an example of an MSDB being converted to a DEDB.

The first step in the conversion process is to create an MSDBINIT data set from either the MSDBDUMP or MSDBCPx (checkpoint) data sets. This is done using the MSDB Dump Recovery utility. In this example, a new MSDBINIT data set is created containing the MSDB named SAVINGS. After the MSDB Dump Recovery utility has completed, the MSDB-to-DEDB Conversion utility can be executed.

The MSDB-to-DEDB Conversion utility uses the MSDBINIT data set created by the MSDB Dump Recovery utility as the input source for the MSDB segments.

The SYSIN control statement indicates that the execution mode for the utility is CONVERT. The database control statements are also included in the SYSIN stream. The MSDB SAVINGS is being converted to a DEDB named SAVINGS. The DEDB SAVINGS has an area named SAVE1, which will be loaded with the segments from MSDBINIT.

The MSDB.ACBLIB data set contains the DMB specifying that the database SAVINGS is an MSDB. The DEDB.ACBLIB data set contains the DMB specifying that the database SAVINGS is a DEDB.

The DBT.FORMAT data set contains the output from the conversion. It contains a record, in DBT Unload format, for each segment of the MSDB SAVINGS.

The next step is the execution of the DFSORT program or an equivalent program. The input control statements in the SYSIN stream sort the records of the DBT.FORMAT data set by ascending RAP RBA values. When this step is complete, the DBT.FORMAT data set may be deleted.

The DBT.FORMAT.SAVE1 data set contains the output from the sort and is now usable by the DBT Reload utility. See *IMS SU/DBT DEDB Unload/Reload User's Guide* for a description of this utility.

The output data set DBT.FORMAT.SAVE1 will be used as input to the DBT Reload utility to load the new DEDB area. The new DEDB area is SAVE1.

```
//RECOVERY EXEC  PGM=DBFDBDR0
//*  EXECUTE THE DUMP RECOVERY UTILITY USING MSDBDUMP AS INPUT
//STEPLIB  DD  DSN=IMS.RESLIB,DISP=SHR
//MSDBINIT DD  DSN=MSDBINIT,DISP=(,KEEP,DELETE),
//             UNIT=SYSDA,VOL=SER=IMS333,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026)
//MSDBDUMP DD  DSN=IMS.MSDBDUMP,DISP=OLD
//MSDBCTL  DD  *
  UNLOAD  DBN=(SAVINGS)
//
//CONVERT  EXEC  PGM=DBFUCDB0
//*  EXECUTE MSDB-TO-DEDB CONVERSION UTILITY WITH THE CONVERT OPTION
//STEPLIB  DD  DSN=IMS.RESLIB,DISP=SHR
//MACBLIB  DD  DSN=MSDB.ACBLIB,DISP=SHR
//DACBLIB  DD  DSN=DEDB.ACBLIB,DISP=SHR
//MSDBINIT DD  DSN=IMS.MSDBINIT,DISP=SHR
//DURDBDFN DD  DSN=DURDBDFN,DISP=(,CATLG,DELETE),UNIT=SYSDA,
//             VOL=SER=333333,SPACE=(TRK,(1,1))
//SAVE1    DD  DSN=DBT.FORMAT,DISP=(,CATLG,DELETE),UNIT=SYSDA,
//             VOL=SER=IMS333,SPACE=(CYL,(1,1)),
//             DCB=(BLKSIZE=13030,RECFM=VB)
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK02 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK03 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
TYPE=CONVERT
MSDB=SAVINGS,DEDB=SAVINGS
/*
//SORT     EXEC PGM=SORT
//* EXECUTE SORT UTILITY TO ORDER RBAS IN DBT.FORMAT DATA SET
//STEPLIB  DD   DSN=program.lib,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SORTIN   DD   DSN=DBT.FORMAT,DISP=SHR
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT  DD   DSN=DBT.FORMAT.SAVE1,DISP=(,CATLG,DELETE),
//              UNIT=SYSDA,VOL=SER=IMS333,SPACE=(CYL,(1,1)),
//              DCB=(BLKSIZE=13030,RECFM=VB)
//SYSIN    DD   *
SORT FIELDS=(5,77,CH,A)  /* Control statements to order RAP RBA */
/*
//DELETE   EXEC PGM=IEBGENER
//* EXECUTE IEBGENER TO DELETE DATA SET DBT.FORMAT
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD DUMMY
//SYSUT1    DD DISP=(OLD,DELETE,KEEP),
//             DSN=DBT.FORMAT
//SYSUT2    DD DUMMY
/*
```

Figure 41. JCL for Converting an MSDB to a DEDB (Part 1 of 2)

```
//RELOAD   EXEC  PGM=FABCUR3
//*  EXECUTE THE DBT RELOAD UTILITY TO LOAD AREA SAVE1
//SAVE1    DD    DSN=SAVE1,DISP=OLD
//DURDATA  DD    DSN=DBT.FORMAT.SAVE1,DISP=OLD
//DURDBDFN DD    DSN=DURDBDFN,DISP=OLD
//DURIWRK  DD
//DURAUDIT DD
//SYSPRINT DD
//SYSIN    DD
```

Figure 41. JCL for Converting an MSDB to a DEDB (Part 2 of 2)

Example 2 (Fallback)

This is an example of a DEDB being converted back to an MSDB.

The first step in the fallback procedure is to unload the DEDB using the DBT Unload utility.

Related Reading: See *IMS SU/DBT DEDB Unload/Reload User's Guide* for a description of this utility.

The DEDB in the fallback procedure is named SAVINGS. It will be converted into an MSDB. The area name is SAVE1.

The next step is execution of the MSDB-to-DEDB Conversion utility. The input control statements in the SYSIN stream indicate that this execution will be a fallback from an MSDB to a DEDB.

The MSDB.ACBLIB data set contains the DBD specifying that the DMB for SAVINGS is an MSDB. The DEDB.ACBLIB data set contains the DBD specifying that the DMB for SAVINGS is a DEDB.

MSDBLCHG is the output data set from the conversion. It contains all of the segments from SAVINGS. The format is the same as the format of the MSDBINIT data set. MSDBLCHG will be used as input to the MSDB Maintenance utility.

The next step is to execute the MSDB Maintenance utility using the IMS.MSDBLCHG data set as input. The MSDB Maintenance utility adds the records from the MSDBLCHG data set to the MSDBINIT data set.

The MSDBINIT data set can be used to bring the new MSDB segments online.

```
//UNLOAD   EXEC  PGM=FABCUR1
//* EXECUTE THE DBT UNLOAD UTILITY TO UNLOAD AREA SAVE1.
//STEPLIB  DD    DSN=IMS.RESLIB,DISP=SHR
//OLDACB   DD    DSN=MSDB.ACBLIB,DISP=SHR
//NEWACB   DD    DSN=DEDB.ACBLIB,DISP=SHR
//SAVE1    DD    DSN=SAVE1,DISP=OLD
//DURD0010 DD    DSN=DBT.SEGMENTS.SAVE1,DISP=(,CATLG,DELETE)
//DURDBDFN DD
//DURS0010 DD
//DURAUDIT DD
//SYSPRINT DD
//SYSIN    DD
//
//
//SORT     EXEC  PGM=SORT
//*  SORT THE OUTPUT OF THE UNLOAD
//SORTIN   DD    DSN=DBT.SEGMENTS.SAVE1,DISP=OLD
//SORTOUT  DD    DSN=DBT.FORMAT.SAVE1,DISP=(,CATLG,DELETE)
//SYSIN
//SORTWKnn
//
//
//FALLBACK EXEC  PGM=DBFUCDB0
//*  EXECUTE THE CONVERSION UTILITY WITH THE FALLBACK OPTION
//STEPLIB  DD  DSN=IMS.RESLIB,DISP=SHR
//MACBLIB  DD  DSN=MSDB.ACBLIB,DISP=SHR
//DACBLIB  DD  DSN=DEDB.ACBLIB,DISP=SHR
//MSDBLCHG DD  DSN=IMS.MSDBLCHG,DISP=(,CATLG,DELETE),
//             VOL=SER=IMS333,SPACE=(CYL,(1,1)),
//             DCB=(LRECL=13026,BLKSIZE=13030,RECFM=VBT)
```

Figure 42. JCL for Converting a DEDB Back to an MSDB (Part 1 of 2)

```
//SAVE1    DD  DSN=DBT.FORMAT.SAVE1,DISP=OLD
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK02 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK03 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
TYPE=FALLBACK
MSDB=SAVINGS,DEDB=SAVINGS
/*
//MAINT    EXEC  PGM=DBFDBMA0
//* EXECUTE THE MSDB MAINTENANCE UTILITY
//STEPLIB  DD  DSN=IMS.RESLIB,DISP=SHR
//MSDBPRT  DD  SYSOUT=A
//MSDBOLD  DD  DSN=OLD.MSDBINIT,DISP=OLD
//MSDBLCHG DD  DSN=IMS.MSDBLCHG,DISP=OLD
//MSDBACB  DD  DSN=MSDB.ACBLIB,DISP=OLD
//MSDBNEW  DD  DSN=NEW.MSDBINIT,DISP=(,CATLG,DELETE)
//MSDBCTL  DD  *
 DBN=SAVINGS MODE=INSERT
```

Figure 42. JCL for Converting a DEDB Back to an MSDB (Part 2 of 2)

Part 5. Report and Test Utilities

Chapter 26. Database-Monitor Report Print Utility (DFSUTR30)

The Database-Monitor Report Print utility (DFSUTR30) is an offline program that produces reports summarizing information collected by the DB Monitor (DFSMNTB0) during the execution of the IMS batch system. This utility produces the following reports:

- VSAM-Buffer-Pool report
- VSAM-Statistics report
- Database-Buffer-Pool report
- Program-I/O report
- DL/I-Call-Summary report
- Distribution-Appendix report
- Monitor-Overhead report

Related Reading: For examples of the output of each of these reports, see "Chapter 27. Interpreting DB-Monitor Reports" on page 239.

Each field in the reports is explained, followed by a summary of how you can use the report. Many of these reports are also provided by the IMS monitor, which is described in *IMS/ESA Administration Guide: System*. Where the same report is produced by both the DB and IMS monitor, the description of the report in this book is applicable for both.

Restrictions

The DFSUTR30 utility 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.

JCL Requirements

The DB-Monitor Report Print utility runs in batch mode, with one job step for each monitor trace period. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

The EXEC statement specifies the program name. The form of this statement is:

```
PGM=DFSUTR30,REGION=256K
```

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules.

SYSPRINT DD
Specifies the output data set, usually SYSOUT=A.

SYSUT1 DD
Specifies the input data set which is a labelled data set written by monitor module DFSMNTB0. It can be a separate data set (ddname= and dsname=IMSMON) or the system log (dsname=IMSLOG).

ANALYSIS DD
Specifies the Analysis Control data set. This file must be in card image format. Use DD DUMMY for default parameters (no distribution report), and input is the first trace interval on the tape.

Analysis Control Data Set

The Analysis Control data set has three record types that allow you 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.

The FILE= specification does not refer to OS files. It refers to trace intervals recorded within an OS file.

Example

The following example shows the JCL for a complete set of reports on the first trace interval from a tape with a serial number of IMSDA1:

```
//TRACE    JOB 969,6014),CHAPMAN,MSGLEVEL=(1,1),CLASS=A
//*
//STEP1   EXEC PGM=DFSUTR30,REGION=256K
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//SYSUT1    DD DSNAME=IMSMON,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=IMSDA1
//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
```

In the example the first two intervals are not changed, but remain as 0 and 1, respectively.

Chapter 27. Interpreting DB-Monitor Reports

This section describes the DB-Monitor reports and how to use them:

VSAM-Buffer-Pool Report

The VSAM-Buffer-Pool report gives you processing information about a specific VSAM subpool. One report is produced for each subpool you specify. For subpools within a VSAM local shared resource pool, reports are produced in ascending order based on subpool buffer size. If both index and data subpools exist in a given shared resource pool, index subpool reports follow data subpool reports. Reports for subpools in different shared resource pools are always produced in the order the shared resource pools are specified. You specify shared resource pools and subpools in control statements processed when IMS is initialized. In a batch system, the control statements are put in the DFSVSAMP data set. In an online system, they are put in the IMS.PROCLIB data set with the DFSVSMnn member name.

The VSAM-Buffer-Pool report has no meaning for HSAM or SHSAM databases, because neither of these databases can use VSAM as the access method.

Fields in the Report

Figure 43 is an example of a VSAM-Buffer-Pool report.

```
              V S A M   B U F F E R   P O O L
                                                                             FIX INDEX/BLOCK/DATA          N/Y/N
                                                                             SHARED RESOURCE POOL ID       VPL/1
                                                                             SHARED RESOURCE POOL TYPE         D
                                                                             SUBPOOL ID                        1
                                                                             SUBPOOL BUFFER SIZE            2048
                                                                             NUMBER  HIPERSPACE BUFFERS        0
                                                                             TOTAL BUFFERS IN SUBPOOL          4

                                                                   17:08:15           17:10:16
                                                                   START TRACE        END   TRACE        DIFFERENCE

   NUMBER OF RETRIEVE BY RBA CALLS RECEIVED BY BUF HNDLR           0                  0                  0
   NUMBER OF RETRIEVE BY KEY CALLS                                 0                  0                  0
   NUMBER OF LOGICAL RECORDS INSERTED INTO ESDS                    0                  0                  0
   NUMBER OF LOGICAL RECORDS INSERTED INTO KSDS                    0                  0                  0
   NUMBER OF LOGICAL RECORDS ALTERED IN THIS SUBPOOL               0                  0                  0
   NUMBER OF TIMES BACKGROUND WRITE FUNCTION INVOKED               0                  0                  0
   NUMBER OF SYNCHRONIZATION CALLS RECEIVED                        18566              27923              9357
   NUMBER OF WRITE ERROR BUFFERS CURRENTLY IN THE SUBPOOL          0                  0                  0
   LARGEST NUMBER OF WRITE ERRORS IN THE SUBPOOL                   0                  0                  0
   NUMBER OF VSAM GET CALLS ISSUED                                 0                  0                  0
   NUMBER OF VSAM SCHBFR CALLS ISSUED                              0                  0                  0
   NUMBER OF TIMES CONT INT REQUESTED ALREADY IN POOL              229956             349375             119419
   NUMBER OF CONTR INT READ FROM EXTERNAL STORAGE                  1078               1229               151
   NUMBER OF VSAM WRITES INITIATED BY IMS                          33                 64                 31
   NUMBER OF VSAM WRITES TO MAKE SPACE IN THE POOL                 0                  0                  0
   NUMBER OF VSAM READS FROM HIPERSPACE BUFFERS                    0                  0                  0
   NUMBER OF VSAM WRITES FROM HIPERSPACE BUFFERS                   0                  0                  0
   NUMBER OF FAILED VSAM READS FROM HIPERSPACE BUFFERS             0                  0                  0
   NUMBER OF FAILED VSAM WRITES FROM HIPERSPACE BUFFERS            0                  0                  0
```

Figure 43. VSAM Buffer-Pool Report

The meaning of the various fields in the report is as follows:

FIX INDEX/BLOCK/DATA
This field indicates the fix options for the index buffers/data buffer prefix/data buffers for this subpool.

SHARED RESOURCE POOL ID
This field is a four character, VSAM local-shared-resource-pool ID provided at subpool definition time. This ID is specified in the DFSVSAMP or IMS.PROCLIB data set control statements.

Related Reading: Refer to *IMS/ESA Installation Volume 2: System Definition and Tailoring* for details about these control statements.

SHARED RESOURCE POOL TYPE
This field indicates whether this subpool contains index (I) or data (D) buffers.

Related Reading: Refer to *IMS/ESA Installation Volume 2: System Definition and Tailoring* for details about this control statement.

SUBPOOL ID
This field tells you the subpool ID. A unique subpool ID is assigned for each different database buffer size in each VSAM local shared resource pool you specify in the DFSVSAMP or IMS.PROCLIB data set control statements.

Related Reading: Refer to *IMS/ESA Installation Volume 2: System Definition and Tailoring* for details about these control statements.

SUBPOOL BUFFER SIZE
This field tells you the size, in bytes, of buffers in this subpool. You can specify buffers of different sizes. A subpool contains all buffers of the same size. All subpools grouped together make up the buffer pool. Buffer size is specified by you in the control statements for the DFSVSAMP or IMS.PROCLIB data sets. For example, suppose you specified:

```
//DFSVSAMP DD *
⋮
POOLID=BB
VSRBF=4096,4,D,HSR,10
VSRBF=8192,4,D,HSO,20
POOLID=FF
VSRBF=4096,4,I
VSRBF=8192,4,D
⋮
/*
```

This gives you four subpools and four VSAM-Buffer-Pool reports. The headings on the reports would look like this:

Table 5. Report #1

SHARED RESOURCE POOL ID:	BB
SHARED RESOURCE POOL TYPE:	D
SUBPOOL ID:	1
SUBPOOL BUFFER SIZE:	4096
HIPERSPACE BUFFERS:	10
TOTAL BUFFERS IN SUBPOOL:	4

Table 6. Report #2

SHARED RESOURCE POOL ID:	BB
SHARED RESOURCE POOL TYPE:	D
SUBPOOL ID:	2
SUBPOOL BUFFER SIZE:	8192

Table 6. Report #2 (continued)

HIPERSPACE BUFFERS:	20
TOTAL BUFFERS IN SUBPOOL:	4

Table 7. Report #3

SHARED RESOURCE POOL ID:	FF
SHARED RESOURCE POOL TYPE:	D
SUBPOOL ID:	1
SUBPOOL BUFFER SIZE:	8192
HIPERSPACE BUFFERS:	0
TOTAL BUFFERS IN SUBPOOL:	4

Table 8. Report #4

SHARED RESOURCE POOL ID:	FF
SHARED RESOURCE POOL TYPE:	I
SUBPOOL ID:	2
SUBPOOL BUFFER SIZE:	4096
HIPERSPACE BUFFERS:	0
TOTAL BUFFERS IN SUBPOOL:	4

NUMBER OF HIPERSPACE BUFFERS
This field indicates the number of hiperspace buffers specified at subpool definition time.

TOTAL BUFFERS IN SUBPOOL
This field tells you how many buffers are in the specified subpool.

START TRACE, END TRACE, and DIFFERENCE
The start trace and end trace fields tell you the time the DB Monitor program was *last* started and stopped. The time is generated by the time-of-day clock. Clock times are read as follows:

```
Clock time = hh.mm.ss
```

where:

hh	Hours 0 through 23
mm	Minutes
ss	Seconds

If the DB Monitor program was on during an entire batch run, the start and end trace times are when the batch run started and stopped. If the DB Monitor program was turned on and off more than once in the same batch run, the start and end trace times are when the monitor was *last* started and stopped.

The numbers under the start and end trace fields are cumulative numbers for the entire batch run. If the monitor was only turned on and off once, the start trace number is zero. If the monitor was turned on and off more than once, the start trace numbers are the cumulative numbers when the monitor was last turned on; the stop trace numbers are the cumulative numbers when the monitor was last turned off.

The difference column tells you the difference between the cumulative numbers in the start and end trace fields. If the monitor was only turned on and off once, the difference column contains the same numbers as the end trace column.

NUMBER OF RETRIEVE BY RBA CALLS

This field tells you how many retrieve by relative byte address (RBA) calls were issued for this subpool. Retrieve by RBA calls are calls issued internally by DL/I. One retrieve by RBA call is issued for each direct-address pointer that must be followed in searching for a segment. For example, a GN call for a dependent segment in an HDAM database uses a series of RBA calls to search for the dependent segment, one call for each direct-address pointer it follows.

If you want to know the exact sequence of a search when a retrieve by RBA call is used, you can record the sequence by turning on the buffer handler trace and using a SNAP call to see the trace records. You can turn on the buffer handler trace using the DL/I= operand on the OPTIONS statement for the DFSVSAMP or DFSVSMnn data set. The SNAP call can be issued from the application program or by using the DFSDDLT0 test program.

One call from an application program can generate more than one retrieve by RBA call. The retrieve by RBA call might or might not require an I/O operation. Because the number in this field does not reflect the number of I/O operations to access a segment, do not use it to judge VSAM performance.

NUMBER OF RETRIEVE BY KEY CALLS

This field tells you how many retrieve by key calls were issued for this subpool. Retrieve by key calls are calls issued internally by DL/I. The calls are issued to search a KSDS using a key as a qualification (where key is equal to or greater than X). For example, a GU call for a root segment in a HIDAM database causes DL/I to issue a retrieve by key call to access the index segment pointing to the requested root segment.

If you want to know the exact sequence of a search when key calls are used, you can record the sequence by turning on the buffer handler trace and using a SNAP call to see the trace records. You can turn on the buffer handler trace using the BHTRACE= operand on the OPTIONS statement for the DFSVSAMP or DFSVSMnn data set. The SNAP call can be issued from the application program or by using the DFSDDLT0 test program.

One call from an application program can generate more than one retrieve by key calls. The retrieve by key calls might or might not require an I/O operation. Because the number in this field does not reflect the number of I/O operations to access a segment, do not be use it to judge VSAM performance.

NUMBER OF LOGICAL RECORDS INSERTED INTO ESDS

This field tells you how many of the logical records in your ESDS that were previously empty now contain segments. When a dependent segment is inserted into an ESDS in a HISAM or HIDAM database, the segment might not fit into a logical record that already contains other segments. In this case, the segment is put into a new ESDS logical record. When a dependent segment is inserted into a logical record in an ESDS in a HISAM database, other segments in the same logical might need to be shifted into a new ESDS logical record to make room for the segment being inserted.

Look at this field from one report to the next. It helps you determine when you are running out of logical records in the primary space you have allocated. It is best to avoid using logical records from secondary space because this space is probably not close to the primary space. The distance between the two areas of space might cause extra seek time and therefore poor performance. In general, it is best to reorganize your database before you need to use secondary space.

NUMBER OF LOGICAL RECORDS INSERTED INTO KSDS

This field tells you how many of the logical records in your KSDS that were previously empty now contain segments. HISAM databases use a new logical record when a root segment is inserted. HIDAM index databases use a new logical record for the index segment created when a root segment is inserted. Secondary index databases use a new logical record when a new pointer segment is inserted.

Look at this field from one report to the next. It helps you determine when you are running out of logical records in the primary space you have allocated. It is best to avoid using logical records from secondary space because this space is probably not close to the primary space. The distance between the two areas of space might cause extra seek time and therefore poor performance. In general, it is best to reorganize your database before you need to use secondary space.

NUMBER OF LOGICAL RECORDS ALTERED IN THIS SUBPOOL

This field tells you how many logical records, while in the buffer pool, were marked as altered. When a segment is inserted or replaced in a logical record, the logical record in the buffer is marked as altered until it is written back to the database.

NUMBER OF TIMES BACKGROUND WRITE FUNCTION INVOKED

If you have specified use of the background write function, this field tells how many times the function was used. The background write function, at intervals, writes buffers containing modified data back to the database. It does this so buffers are available for use when an application program needs them. Without background write, if an application program wants to read data into a buffer that already contains modified data, the application program has to wait while the contents of the buffer are written back to the database. The number of times background write is invoked is the same on each subpool report produced, during a given execution of the monitor, for a given local shared resource pool. Once invoked, the background write function writes buffers from all subpools within a local shared resource pool.

Background write is specified in the BGWRT= operand of the OPTIONS statement for the DFSVSAMP or DFSVSMnn data set.

NUMBER OF SYNCHRONIZATION CALLS RECEIVED

This field tells you how many requests were made to write all altered buffers in the local shared resource pool while the monitor was on. CHKP and STAT calls are typical requestors of this.

NUMBER OF PERM WRT ERROR BUFFS NOW IN THE SUBPOOL

This field tells you how many buffers are currently “frozen” in storage because a permanent I/O error occurred when writing them to the database. When a VSAM write operation results in a permanent I/O error, the affected buffers are frozen in storage until the data set is closed or, in the online system, until the system is shut down. Once the data set is closed or the system shut down, the buffers are written to the log and the number in this field returns to zero.

Ensure that the operator performed database recovery of the affected data set before you ever receive this report.

LARGEST NUMB OF PERM ERR BUFFS EVER IN THE SUBPOOL

This field tells you how many buffers *were* frozen in storage when the condition described in the previous field occurred.

NUMBER OF VSAM GET CALLS ISSUED

This field tells you how many times VSAM GET calls were issued. VSAM GET calls are calls issued internally by DL/I. The GET call might be satisfied by data in the buffer pool or it might require that data be read into the buffer pool. Because the number in this field does not reflect the number of I/O operations required to access a segment, do not use it to judge VSAM performance.

NUMBER OF VSAM SCHBFR CALLS ISSUED

This field tells you how many times the HD space management routine issued calls to search for space in which to insert segments.

If, from one monitor report to the next, the number in this field is increasing, it means that space for storing new segments is not available in the most desirable location. Eventually, this means you must reorganize your database to improve performance. In reorganizing, pay special attention to the operands affecting database space (the bytes operand in the RMNAME= keyword in the DBD statement and the fbff and fspf operands in the FRSPC= keyword in the DATASET statement).

NUMBER OF TIMES CONT INT REQUESTED ALREADY IN POOL

This field tells you how many times a logical record was found in a CI that was already in the buffers. When this occurs, no I/O operations are required to access the desired segments.

If you are trying to improve performance, increase the number of buffers you have allocated. If you increase the number of buffers, you can monitor this field to see if the number in it increases, which indicates improved performance.

NUMBER OF CONTR INT READ FROM EXTERNAL STORAGE

This field tells you how many times a logical record was *not* found in a CI that was already in the buffers. When this occurs, an I/O operation is required to read the CI containing the logical record into the buffer pool. Because performance is always better when fewer I/O operations are performed, you might want to increase the number of buffers you have specified to see how that affects the number in this field. Specifying more buffers keeps more CIs (and therefore logical records) in the buffer pool. There is a break-even point in this process, however, where too many buffers are specified, and it takes longer to search and maintain the buffers than it takes to read a CI into the buffer.

The number of buffers is specified in the control statements for the DFSVSAMP or DFSVSMnn data sets.

NUMBER OF VSAM WRITES INITIATED BY IMS

This field tells you the number of times DL/I issued a write request to write data to the database. Write operations are issued when:

- A data set is closed. Database buffers containing data that has been altered by the data set being closed are written to the database.
- Abnormal termination occurs during application program processing. Database buffers containing data that has been altered are written to the database.
- The background write function is invoked. Selected database buffers containing data that has been altered are written to the database.
- A checkpoint call is issued. All altered database buffers are written to the database.

NUMBER OF VSAM WRITES TO MAKE SPACE IN THE POOL

This field tells you how many times a CI had to be read into a buffer containing a logical record with altered data. When this happens, the buffer (because it contains altered data) has to be written back to the database before the new CI can be read into it. This means the application program has to wait while the write operation takes place.

For best performance, ensure that the number in this field is close to zero. This can generally be achieved by turning on the background write function during batch processing and adjusting the number of buffers allocated.

NUMBER OF VSAM READS FROM HIPERSPACE BUFFERS
This field tells you the total number of successful VSAM reads (MOVEPAGE and NON-MOVEPAGE) from the hiperspace buffers.

NUMBER OF VSAM WRITES FROM HIPERSPACE BUFFERS
This field tells you the total number of successful VSAM writes (MOVEPAGE and NON-MOVEPAGE) to the hiperspace buffers.

NUMBER OF FAILED VSAM READS FROM HIPERSPACE BUFFERS
This field tells you the number of times that a VSAM read request from hiperspace failed, resulting in a read from DASD.

NUMBER OF FAILED VSAM WRITES FROM HIPERSPACE BUFFERS
This field tells you the number of times that a VSAM WRITE request to hiperspace failed, resulting in a write to DASD.

Using the Report

The primary usefulness of the VSAM-Buffer-Pool report is to calculate how many I/O operations were required to read to or write from the subpool. You might want to increase buffer pool size to see if you can decrease the number of I/O operations. You might also want to turn on background write. Or, if the number of I/O operations is increasing over time, you might need to reorganize your database.

To calculate the number of I/O operations required to read to or write from the buffer pool, use the following formula:

Total I/O equals the sum of:

1. The number of times a CI had to be read into the buffer from the database (NUMBER of CONTR INT READ FROM EXTERNAL STORAGE field in the report).
2. The number of times a buffer had to be written to the database (NUMBER OF VSAM WRITES INITIATED BY IMS field in the report).
3. The number of times a buffer had to be written to the database so a new CI could be read into the buffer (NUMBER OF VSAM WRITES TO MAKE SPACE IN THE POOL field in the report).

VSAM-Statistics Report

VSAM-Statistics reports are based on:

- A specific application program PCB
- A data set the application program is using
- The type of DL/I call the application program issued

The VSAM-Statistics report has no meaning for HSAM or SHSAM databases because neither of these databases can use VSAM as the access method.

Fields in the Report

Figure 44 is an example of a VSAM-Statistics report.

```
     IMS MONITOR   ****VSAM STATISTICS****           TRACE START 1989 076  12:42:54     TRACE STOP  1989 076  12:43:07  PAGE  0012
                  DL/I  VSAM      RET    RET    ISRT   ISRT    BFR    BKG    SYN                                  USR    NUR
PCBNAME  DDNAME   FUNC  IWAITS    RBA    KEY    ESDS   KSDS    ALT    WTS    PTS   GETS  SCHBFR  FOUND   READS   WTS    WTS
________ ________ ____  ______   ____   ____    ____   ____   ____   ____   ____   ____  ______  _____   _____   ____   ____
DLVNTZ02 DBHVSAM2 STAT   1       1.00   0.00   0.00   0.00   1.00   0.00   1.00   1.00   0.00   1.00   0.00   1.00   0.00
         DD TOTAL
         __ _____        1       1.00   0.00   0.00   0.00   1.00   0.00   1.00   1.00   0.00   1.00   0.00   1.00   0.00
         HIDAM    STAT   4     307.25  36.50   0.50   0.00  35.00   0.00   0.25   120.25 0.00 157.00   0.25   0.75   0.00
         DD TOTAL
         __ _____        4     307.25  36.50   0.50   0.00  35.00   0.00   0.25 120.25   0.00 157.00   0.25   0.75   0.00
         XDLBT04I GU     2      15.00   4.50   0.00   0.00   0.00   0.00   0.00  17.50   0.00  21.50   1.00   0.00   0.00
         DD TOTAL
         __ _____        2      15.00   4.50   0.00   0.00   0.00   0.00   0.00  17.50   0.00  21.50   1.00   0.00   0.00
PCB TOTAL
___ _____                7     180.00  22.14   0.28   0.00  20.14   0.00   0.28  73.85   0.00  96.00   0.42   0.57   0.00
DLVNTZX2 HIDAM    GN     1      15.00   3.00   0.00   0.00   0.00   0.00   0.00  18.00   0.00  20.00   1.00   0.00   0.00
                  GU     1       1.00   0.00   0.00   0.00   0.00   0.00   0.00   1.00   0.00   0.00   1.00   0.00   0.00
         DD TOTAL
         __ _____        2       8.00   1.50   0.00   0.00   0.00   0.00   0.00   9.50   0.00  10.00   1.00   0.00   0.00
         DBHVSAM1 GU     8       0.00   0.12  12.75 520.00   7.50   0.00   0.00   0.12   0.00   0.00   0.87   0.00   0.00
         DD TOTAL
         __ _____        8       0.00   0.12  12.75 520.00   7.50   0.00   0.00   0.12   0.00   0.00   0.87   0.00   0.00
         XDLBT04I GU     6       0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   1.00   0.00   0.00
         DD TOTAL
         __ _____        6       0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   1.00   0.00   0.00
         DBHVSAM2 GU     3       0.66   0.00   0.00   0.00   0.00   0.00   0.00   0.66   0.00   0.00   1.00   0.00   0.00
         DD TOTAL
         __ _____        3       0.66   0.00   0.00   0.00   0.00   0.00   0.00   0.66   0.00   0.00   1.00   0.00   0.00
PCB TOTAL
___ _____               19       0.94   0.21   5.36 482.10   3.15   0.00   0.00   1.15   0.00   1.05   0.94   0.00   0.00
BATCH TOTAL
_____ _____             26      49.15   6.11   3.84  83.07   3.11   0.00   0.07  20.73   0.00  26.61   0.80   0.15   0.00
```

Figure 44. VSAM-Statistics Report

The meaning of the various fields in the report is as follows:

TRACE START and TRACE STOP
The trace start and trace stop fields tell you the time when the DB Monitor program was *last* started and stopped. The time is generated by the time-of-day clock. Clock times are read as follows:

```
Clock time = hh.mm.ss
```

where:

hh	Hours 0 through 23
mm	Minutes
ss	Seconds

If the DB Monitor was on during an entire batch run, the trace start and trace stop time is when the batch run started and stopped. If the DB Monitor was turned on and off more than once in the same batch run, the trace start and trace stop time is when the monitor was *last* started and stopped.

PCBNAME
This field tells you the name of the PCB the report is providing information about. Remember that each application program has one or more PCBs. This field can only be used to identify which application program the report is providing information about if PCB names are unique to application programs.

DDNAME
This field tells you the name of the data set the application program is using. Within one report, an application program (PCB) can access more than one data set. The statistics compiled are listed separately for each data set.

DL/I FUNC
This field tells you the type of DL/I calls the application program issued.

VSAM IWAITS

This field tells you, by type of DL/I call against a specific data set, the number of times IMS had to wait before processing could proceed. When IMS has to wait, it is almost always waiting for an I/O operation to take place, that is, data is being either read from the database to the buffer or written from the buffer back to the database.

The numbers in each column under all remaining fields in the report are averages. They tell the average number of times an activity occurred rather than the specific number of times. Averages are for waits. These numbers are truncated. For example, a value of 0.019 is printed as 0.01.

RET RBA

This field tells you how many retrieve by relative byte address (RBA) calls were issued for this subpool. Retrieve by RBA calls are calls issued internally by DL/I. One retrieve by RBA call is issued for each direct-address pointer that must be followed in searching for a segment. For example, a GN call for a dependent segment in an HDAM database uses a series of RBA calls to search for the dependent segment, one call for each direct-address pointer it follows.

If you want to know the exact sequence of a search when a retrieve by RBA call is used, you can record the sequence by turning on the buffer handler trace and using a SNAP call to see the trace records. You can turn on the buffer handler trace using the BHTRACE= operand on the OPTIONS statement for the DFSVSAMP or DFSVSMnn data set. The SNAP call can be issued from the application program or by using the DFSDDLT0 test utility.

One call from an application program can generate more than one retrieve by RBA call. The retrieve by RBA call might or might not require an I/O operation. Because the number in this field does not reflect the number of I/O operations to access a segment, do not use it to judge VSAM performance.

RET KEY

This field tells you how many retrieve by key calls were issued for this subpool. Retrieve by key calls are calls issued internally by DL/I. The calls are issued to search a KSDS using a key as a qualification (where key is equal to or greater than X). For example, a GU call for a root segment in a HIDAM database causes DL/I to issue a retrieve by key call to access the index segment pointing to the requested root segment.

If you want to know the exact sequence of a search when key calls are used, you can record the sequence by turning on the buffer handler trace and using a SNAP call to see the trace records. You can turn on the buffer handler trace using the BHTRACE= operand on the OPTIONS statement for the DFSVSAMP or DFSVSMnn data set. The SNAP call can be issued from the application program or by using the DFSDDLT0 test utility.

One call from an application program can generate more than one retrieve by key calls. The retrieve by key calls might or might not require an I/O operation. Because the number in this field does not reflect the number of I/O operations to access a segment, do not use it to judge VSAM performance.

ISRT ESDS

This field tells you how many of the logical records in your ESDS that were previously empty now contain segments. When a dependent segment is inserted into an ESDS in a HISAM or HIDAM database, the segment might not fit into a logical record that already contains other segments. In this case, the segment is put into a new ESDS logical record. When a dependent segment is inserted into a logical record in an ESDS in a HISAM database, other segments in the same logical record might need to be shifted into a new ESDS logical record to make room for the segment being inserted.

Look at this field from one report to the next. It helps you determine when you are running out of logical records in the primary space you have allocated. It is best to avoid using logical records from secondary space because this space is probably not close to the primary space.

ISRT KSDS
This field tells you how many of the logical records in your KSDS that were previously empty now contain segments. HISAM databases use a new logical record when a root segment is inserted. HIDAM index databases use a new logical record for the index segment created when a root segment is inserted.

Look at this field from one report to the next. It helps you determine when you are running out of logical records in the primary space you have allocated. It is best to avoid using logical records from secondary space because this space is probably not close to the primary space. The distance between the two areas of space might cause extra seek time and therefore poor performance. In general, it is best to reorganize your database before you need to use secondary space.

BFR ALT
This field tells you how many logical records, while in the buffer pool, were marked as altered. When a segment is inserted or replaced in a logical record, the logical record in the buffer is marked as altered until it is written back to the database.

BKG WTS
If you have specified use of the background write function, this field tells how many times the function was used. Background write, at intervals, writes buffers containing modified data back to the database. It does this so buffers are available for use when an application program needs them. Without background write, if an application program wants to read data into a buffer that already contains modified data, the application program has to wait while the contents of the buffer are written back to the database. The number of times background write was invoked is the same on each subpool report produced during a given execution of the monitor. This is because, once involved, background write writes buffers from all subpools.

Background write is specified in the BGWRT= operand of the OPTIONS statement for the DFSVSAMP or DFSVSMnn data set.

SYN PTS
This field tells you how many times checkpoint calls were issued in DL/I programs while the monitor was on.

GETS
This field tells you how many times VSAM GET calls were issued. VSAM GET calls are calls issued internally by DL/I. The GET call might be satisfied by data in the buffer pool or it might require that data be read into the buffer pool. Because the number in this field does not reflect the number of I/O operations required to access a segment, do not use it to judge VSAM performance.

SCHBFR
This field tells you how many times the HD space management routine issued calls to search for space in which to insert segments.

If, from one monitor report to the next, the number in this field is increasing, it means that space for storing new segments is not available in the most desirable location. Eventually, you must reorganize your database to improve performance. In reorganizing, pay special attention to the operands affecting database space (the BYTES operand in the RMNAME= keyword in the DBD statement and the fbff and fspf operands in the FRSPC= keyword in the DATASET statement).

FOUND

This field tells you how many times a logical record was found in a CI that was already in the buffers. When this occurs, no I/O operations are required to access the desired segments.

If you are trying to improve performance, increase the number of buffers you have allocated. If you increase the number of buffers, you can monitor this field to see if the number in it increases, which indicates improved performance.

READS

This field tells you how many times a logical record was *not* found in a CI that was already in the buffers. When this occurs, an I/O operation is required to read the CI containing the logical record into the buffer pool. Because performance is always better when fewer I/O operations are performed, you might want to increase the number of buffers you have specified to see how that affects the number in this field. Specifying more buffers keeps more CIs (and therefore logical records) in the buffer pool. There is a break-even point in this process, however, where too many buffers are specified, and it takes longer to search and maintain the buffers than it takes to read a CI into the buffer.

The number of buffers is specified in the control statements for the DFSVSAMP or DFSVSMnn data sets.

USR WTS

This field, which indicates user writes, tells you the number of times DL/I issued a write request to write data to the database. Write operations are issued when:

- A data set is closed. Database buffers containing data that has been altered by the data set being closed are written to the database.
- Abnormal termination occurs during application program processing. Database buffers containing data that has been altered are written to the database.
- The background write function is invoked. Selected database buffers containing data that has been altered are written to the database.
- A checkpoint call is issued. All altered database buffers are written to the database.

NUR WTS

This field, which indicates nonuser writes, tells how many times a CI had to be read into a buffer containing a logical record with altered data. When this happens, the buffer (because it contains altered data) has to be written back to the database before the new CI can be read into it. This means the application program has to wait while the write operation takes place.

For best performance, ensure that the number in this field is close to zero. This can generally be achieved by turning on the background write function during batch processing and adjusting the number of buffers allocated.

SCRS
This field tells you the total number of successful VSAM reads (MOVEPAGE and NON-MOVEPAGE) from hiperspace buffers.

SCWS
This field tells you the total number of successful VSAM writes (MOVEPAGE and NON-MOVEPAGE) to hiperspace buffers.

SCRF
This field tells you the number of times that a VSAM read request from hiperspace failed, resulting in a read from DASD.

SCWF
This field tells you the number of times that a VSAM write request to hiperspace failed, resulting in a write to DASD.

DD TOTAL
This field tells you, for a given data set, the overall average number of times an activity occurred (except for the VSAM CALLS field, which tells *total* number of times).

PCB TOTAL
This field tells you, for a given PCB, the overall average number of times an activity occurred (except for the VSAM CALLS field, which tells *total* number of times).

BATCH TOTAL
This field tells you, for the time the monitor was running, the overall average number of times an activity occurred (except for the VSAM CALLS field, which tells *total* number of times).

Using the Report

From a VSAM-Statistics report, you can determine which calls in an application program require a great many I/O operations. After you know this, you can improve performance by tuning either the database or the application program to reduce I/O operations. The following fields in the report tell actual I/O activity and are therefore the most important ones to monitor:

- READS
- USR WTS
- NUR WTS

If you can reduce the averages in these fields, performance can be improved.

In tuning to reduce I/O operations, pay most attention to calls issued a large number of times. It is more profitable to save 1 second on a call executed 2000 times than to save 5 seconds on a call executed ten times (2000 versus 50 seconds). The DL/I-Call-Summary report (discussed under “DL/I-Call-Summary Report” on page 255) tells you how many times each DL/I application program call is issued.

Database-Buffer-Pool Report

The Database-Buffer-Pool report gives you information about OSAM subpools during processing. One report is produced for *all* subpools in the buffer pool. (This report and the VSAM-Buffer-Pool report differ in that the VSAM report was produced for *each* subpool in the buffer pool.)

The Database-Buffer-Pool report has no meaning for HSAM, SHSAM, SHISAM, or GSAM databases because none of these databases can use OSAM as the access method.

Fields in the Report

Figure 45 is an example of a Database Buffer Pool report.

```
                   D A T A   B A S E   B U F F E R   P O O L
                                                                        FIX PREFIX/BUFFERS          Y/Y
                                                                        SUBPOOL ID                 004K
                                                                        SUBPOOL BUFFER SIZE        4096
                                                                        TOTAL BUFFERS IN SUBPOOL   1000

                                                              17:08:15           17:10:16

NUMBER OF LOCATE-TYPE CALLS                                    1117674            1676213            558539
NUMBER OF REQUESTS TO CREATE NEW BLOCKS                              0                  0                 0
NUMBER OF BUFFER ALTER CALLS                                    215874             322936            107062
NUMBER OF PURGE CALLS                                            25077              37454             12377
NUMBER OF LOCATE-TYPE CALLS, DATA ALREADY IN OSAM POOL          870306            1301187            430881
NUMBER OF BUFFERS SEARCHED BY ALL LOCATE-TYPE CALLS            1258247            1886843            628596
NUMBER OF READ I/O REQUESTS                                     238165             360260            122095
NUMBER OF SINGLE BLOCK WRITES BY BUFFER STEAL ROUTINE                0                  0                 0
NUMBER OF BLOCKS WRITTEN BY PURGE                                95057             142413             47356
NUMBER OF LOCATE CALLS WAITED DUE TO BUSY ID                       780               1297               517
NUMBER OF LOCATE CALLS WAITED DUE TO BUFFER BUSY WRT                 0                  0                 0
NUMBER OF LOCATE CALLS WAITED DUE TO BUFFER BUSY READ                0                  0                 0
NUMBER OF BUFFER STEAL/PURGE WAITED FOR OWNERSHIP RLSE             178                261                83
NUMBER OF BUFFER STEAL REQUESTS WAITED FOR BUFFERS                   0                  0                 0
TOTAL NUMBER OF I/O ERRORS FOR THIS SUBPOOL                          0                  0                 0
NUMBER OF BUFFERS LOCKED DUE TO WRITE ERRORS                         0                  0                 0
```

Figure 45. Database-Buffer-Pool Report

The meaning of the various fields in the report is as follows:

FIX PREFIX/BUFFERS
This field indicates the fix options for the buffer prefix/data buffers for this subpool.

SUBPOOL ID
This field is a 4 character pool ID provided at subpool definition time.

SUBPOOL BUFFER SIZE
This field indicates the size, in bytes, of the buffers in this subpool.

TOTAL BUFFERS IN SUBPOOL
This field indicates the total number of buffers in this subpool.

On the following line are time entries that indicate the start trace and end trace times. The start trace and end trace fields tell you the time when the DB Monitor program was last started and stopped.

NUMBER OF LOCATE-TYPE CALLS
This field indicates the number of locate-type calls for this subpool.

NUMBER OF REQUESTS TO CREATE NEW BLOCKS
This field indicates the number of times a block had a segment inserted for the first time. When this happens, the block is marked as modified and must eventually be written back to the database.

NUMBER OF BUFFER ALTER CALLS
This field indicates the number of buffer alter calls for this subpool. This count includes NEW BLOCK and BYTALT calls.

NUMBER OF PURGE CALLS
This field indicates the number of purge requests for this subpool.

NUMBER OF LOCATE-TYPE CALLS, DATA ALREADY IN SUBPOOL
This field indicates the number of locate-type calls for this subpool where the data was already in an OSAM pool.

NUMBER OF BUFFERS SEARCHED BY ALL LOCATE-TYPE CALLS
This field indicates the number of buffers searched by all locate-type calls for this subpool.

NUMBER OF READ I/O REQUESTS
This field indicates the number of read I/O requests for this subpool.

NUMBER OF SINGLE BLOCK WRITES BY BUFFER STEAL ROUTINE
This field indicates the number of single block writes initiated by buffer steal routine for this subpool.

NUMBER OF BLOCKS WRITTEN BY PURGE
This field indicates the number of blocks for this subpool written by purge.

TOTAL NUMBER OF I/O ERRORS FOR THIS SUBPOOL
This field indicates the total number of I/O errors for this subpool.

NUMBER OF BUFFERS LOCKED DUE TO WRITE ERRORS
This field indicates how many buffers are currently “frozen” in storage because a permanent I/O error occurred when writing them to the database. When a write operation results in a permanent I/O error, the affected buffers are frozen in storage until the data set is closed or, in an online system, until the system is shut down. Once the data set is closed, or the online system shut down, the buffers are written to the log, and this number returns to 0.

NUMBER OF LOCATE CALLS WAITED DUE TO BUSY ID
This field indicates the number of locate calls for this subpool which waited due to busy ID.

NUMBER OF LOCATE CALLS WAITED DUE TO BUFFER BUSY WRT
This field indicates the number of locate calls for this subpool which waited due to buffer busy writing.

NUMBER OF LOCATE CALLS WAITED DUE TO BUFFER BUSY READ
This field indicates the number of locate calls for this subpool which waited due to buffer busy reading.

NUMBER OF BUFFER STEAL/PURGE WAITED DUE TO BUFFER BUSY READ
This field indicates the number of locate calls for this subpool which waited for ownership to be released.

NUMBER OF BUFFER STEAL REQUESTS WAITED FOR BUFFERS
This field indicates the number of buffer steal requests for this subpool which waited because no buffers were available to be stolen.

Using the Report

The primary usefulness of the Database-Buffer-Pool report is to calculate how many I/O operations were required to read to or write from the OSAM buffer pool. You might want to increase buffer pool size to see if you can decrease the number of I/O operations. Or, if the number is increasing over time, you might need to reorganize your database.

To calculate the number of I/O operations required to read to or write from the buffer pool, use the following formula:

Total I/O equals the sum of:

1. Blocks read from the database

   Number of blocks read for OSAM specific requests (Number of Read Requests Issued)

2. Blocks written to the database

Number of blocks written because of buffer steal processing and purge processing (Number of Blocks Written)

Program-I/O Report

The Program-I/O report tells how long IMS was inactive (no IMS code was being executed) because IMS was waiting for use of a resource or for completion of an event. This time is called IWAIT time and, for all practical purposes, can be considered the time IMS had to wait while an I/O operation took place. One report is produced each time the DB Monitor is run, and the report describes IWAIT time by PCB and data set name.

All times in the Program-I/O report are in microseconds. A microsecond is one millionth of a second (in other words, 7050 microseconds equals 0.007050 seconds).

Fields in the Report

Figure 46 is an example of a Program-I/O report.

```
    IMS MONITOR   ****PROGRAM I/O****                TRACE START 1989 076  12:42:54     TRACE STOP  1989 076  12:43:07  PAGE  0008
                                  .........IWAIT TIME..........                         DISTR.
        PCB NAME      IWAITS       TOTAL        MEAN     MAXIMUM   DDNAME    MODULE       NO.

        ___ ____      ______       _____       ____     _______   ______    ______       ___
        DLVNTZ02           1       35853       35853       35853   DBHVSAM2   VBH          127
                           4      257649       64412      196028   HIDAM      VBH          128
                           2       79222       39611       62452   XDLBT04I   VBH          129
       PCB TOTAL
       ___ _____           7      372724       53246
        DLVNTZX2           2       57645       28822       40686   HIDAM      VBH          130
                           8      176622       22077       46141   DBHVSAM1   VBH          131
                           6      105340       17556       27843   XDLBT04I   VBH          132
                           3       65296       21765       23458   DBHVSAM2   VBH          133
       PCB TOTAL
       ___ _____          19      404903       21310
BATCH TOTAL
_____ _____               26      777627       29908
```

Figure 46. Program-I/O Report

The meaning of the various fields in the report is as follows:

TRACE START and TRACE STOP
The trace start and trace stop fields tell you the time when the DB Monitor program was *last* started and stopped. The time is generated by the time-of-day clock. Clock times are read as follows:

```
Clock time = hh.mm.ss
```

where:

hh	Hours 0 through 23
mm	Minutes
ss	Seconds

If the DB Monitor program was on during an entire batch run, the trace start and trace stop times is when the batch run started and stopped. If the DB Monitor program was turned on and off more than once in the same batch run, the trace start and trace stop times are the times at which the monitor was *last* started and stopped.

PCBNAME
This field tells you the name of the PCB the report is providing information about. Remember that each application program has one or more PCBs. If PCB

names are unique to application programs, this field can only be used to identify which application program the report is providing information about.

IWAITS
This field tells you the number of times IMS was inactive.

IWAIT Time
This field tells you the elapsed time during which IMS was inactive.

TOTAL	The total time IMS waited
MEAN	The average time IWAIT IMS waited
MAXIMUM	The longest single time IMS waited

DDNAME
This field tells you the name of the data set the application program is using. Within one report, an application program (PCB) can access more than one data set. The statistics compiled are listed separately for each data set.

MODULE
This field tells you the modules that issued the internal call (in response to a DL/I call) that caused IMS to wait.

DBH	Module DFSDBHR0
DLE	Module DFSDDLE0
VBH	Module DFSDVSM0

DISTR.NUMBER
The distribution number is a reference number to be used in conjunction with the Distribution-Appendix report. For a description of what it means, see “Distribution-Appendix Report” on page 257.

PCB Total
For a given PCB this field tells you:

IWAITS	The total number of times IMS was inactive
TOTAL	The total time IMS waited
MEAN	The average time per IWAIT

BATCH Total
For this execution of the DB Monitor this field tells you:

IWAITS	The total number of times IMS was inactive
TOTAL	The total time IMS waited
MEAN	The average time per IWAIT

Using the Report

Using the Program-I/O report, you can correlate IWAIT time with a specific PCB and data set. This allows you to identify databases that are causing a relatively large number of IWAITs. Because IWAITS are identified by PCB and data set names, you might be able to trace the large number of IWAITs back to a particular application program. If you can, give the application program special attention when tuning for performance. The DL/I-Call-Summary report, described in the following section, helps you identify specific DL/I calls in an application program that are causing a large number of IWAITs.

DL/I-Call-Summary Report

The DL/I-Call-Summary report tells two things:

- How long IMS was inactive (no IMS code was being executed) because IMS was waiting for use of a resource or for completion of an event. This time is called IWAIT time and, for all practical purposes, can be considered the time IMS had to wait while an I/O operation took place.
- Of the elapsed time during which IMS was inactive, how much of it was actually IWAIT time.

One report is produced each time the DB Monitor is run, and the report describes times by PCB name and type of DL/I call.

All times in the DL/I-Call-Summary report are in microseconds. A microsecond is one millionth of a second (in other words, 7050 microseconds equals 0.007050 seconds).

Fields in the Report

Figure 47 is an example of a DL/I-Call-Summary report.

```
IMS MONITOR   ****DL/I CALL SUMMARY****          TRACE START 1989 076  12:42:54     TRACE STOP  1989 076  12:43:07  PAGE  0009
                                                               (C)          (A)                    (B)
             CALL LEV           STAT  DL/I                 IWAITS/     ..ELAPSED TIME...      .NOT IWAIT TIME..   DISTRIB.
    PCB NAME FUNC NO.SEGMENT    CODE CALLS       IWAITS    CALL        MEAN    MAXIMUM        MEAN    MAXIMUM   NUMBER

    ________ ____ __________    ____ _____       ______    ____        ____    _______        ____    _______   ______
    DLVNTZ02 STAT (00)          GA       1          1   1.00      968281     968281      932428     932428   1 A,B,C
             GN   (00)          GB       7          0   0.00        5703      29519        5703      29519   4 A,B,C
             GN   (03)K1Z                6          0   0.00         165        253         165        253   5 A,B,C
             GN   (02)K8K5      GA       8          0   0.00         263        888         263        888   6 A,B,C
             GN   (03)K1Z       GK       3          0   0.00        6844      20272        6844      20272   7 A,B,C
             GN   (03)K6Y                3          0   0.00         140        173         140        173   8 A,B,C
             GN   (03)K5YK1     GA       3          0   0.00         197        219         197        219   9 A,B,C
             GN   (04)K1Y       GK       5          0   0.00         306        892         306        892   10 A,B,C
             GN   (04)K42                5          0   0.00         121        173         121        173   11 A,B,C
             GN   (03)K5XK2     GK       5          0   0.00        9951      41900        9951      41900   12 A,B,C
             GN   (03)K6                 7          0   0.00        1292       7219        1292       7219   13 A,B,C
             GN   (02)K5        GA       9          0   0.00         160        220         160        220   14 A,B,C
             GN   (04)K1Y                7          0   0.00       12008      45991       12008      45991   15 A,B,C
             GN   (03)K5XK2              4          0   0.00        6333      20917        6333      20917   16 A,B,C
             GN   (02)K5                 3          0   0.00         126        138         126        138   17 A,B,C
             GN   (01)K1        GA       5          0   0.00        6773      23625        6773      23625   18 A,B,C
             GN   (03)K5YK1             10          0   0.00        9444      30320        9444      30320   19 A,B,C
             GN   (04)K6X       GK       4          0   0.00         141        165         141        165   20 A,B,C
             GN   (04)K1X       GK       4          0   0.00        3278      12578        3278      12578   21 A,B,C
             GN   (04)K4                 4          0   0.00         423       1342         423       1342   22 A,B,C
             GN   (03)K3K5               6          0   0.00        1360       6982        1360       6982   23 A,B,C
             GN   (02)K2                 4          0   0.00         325        910         325        910   24 A,B,C
             GN   (03)K3K5      GA       2          0   0.00         189        196         189        196   25 A,B,C
             GN   (04)K1X                2          0   0.00         134        142         134        142   26 A,B,C
             GU   (01)K1                 9          0   0.00        3387      26518        3387      26518   27 A,B,C
             GN   (02)K8K5               3          0   0.00         179        207         179        207   48 A,B,C
             GN   (02)K5        GE       1          0   0.00         116        116         116        116   49 A,B,C
             GU   (03)K5YK1              8          0   0.00        7752      26664        7752      26664   50 A,B,C
             GNP  (02)K5        GE       2          0   0.00         108        120         108        120   51 A,B,C
             GNP  (03)K5YK1              5          0   0.00        7873      37362        7873      37362   52 A,B,C
             GU   (04)K1Y                1          0   0.00         669        669         669        669   56 A,B,C
             GU   (04)K4                 4          0   0.00       11572      44675       11572      44675   59 A,B,C
             GU   (02)K5                 6          0   0.00        9025      45363        9025      45363   61 A,B,C
             GNP  (02)K2        GE       1          0   0.00          87         87          87         87   62 A,B,C
             GNP  (03)K3K5               2          0   0.00         165        167         165        167   63 A,B,C
             GU   (02)K2                 3          0   0.00         431        566         431        566   64 A,B,C
             GU   (03)K5XK2              5          0   0.00       13570      52617       13570      52617   65 A,B,C
             GU   (04)K6X                3          0   0.00        4892      13242        4892      13242   66 A,B,C
             GU   (03)K6                 2          0   0.00        1247       1810        1247       1810   67 A,B,C
             GU   (04)K42                3          0   0.00         998       1390         998       1390   68 A,B,C
             DLET (04)K42                2          0   0.00       46374      49018       46374      49018   72 A,B,C
             DLET (04)K6X                2          0   0.00      108321     118802      108321     118802   73 A,B,C
             DLET (02)K2                 1          0   0.00       44789      44789       44789      44789   74 A,B,C
             DLET (03)K3K5               3          0   0.00       87567     121071       87567     121071   75 A,B,C
             GU   (02)J5        GE       1          0   0.00        1644       1644        1644       1644   111 A,B,C
             DLET (03)J6                 1          0   0.00       35633      35633       35633      35633   112 A,B,C
             GU   (02)J9        GE       1          0   0.00         705        705         705        705   113 A,B,C
             DLET (04)J7J9               1          0   0.00      245921     245921      245921     245921   114 A,B,C
             REPL (03)J7PJ6     DA       1          0   0.00         225        225         225        225   115 A,B,C
             BATCH TOTAL
             _____ _____               392         22   0.05       22310                   20983
```

Figure 47. DL/I-Call-Summary Report

The meaning of the various fields in the report is as follows:

TRACE START and TRACE STOP
The trace start and trace stop fields tell you the time when the DB Monitor program was *last* started and stopped. The time is generated by the time-of-day-clock. Clock times are read as follows:

```
Clock time = hh.mm.ss
```

where:

hh	Hours 0 through 23
mm	Minutes
ss	Seconds

If the DB Monitor was on during an entire batch run, the trace start and trace stop time is when the batch run started and stopped. If the DB Monitor was turned on and off more than once in the same batch run, the trace start and trace stop times are when the monitor was *last* started and stopped.

PCBNAME
This field tells you the name of the PCB the report is providing information about. Remember that each application program has one or more PCBs. If PCB names are unique to application programs, this field can only be used to identify which application program the report is providing information about.

CALL FUNC
This field tells you the type of DL/I call the application program issued.

LEV NO.
This field tells you the level that was accessed in the hierarchy of the database record to perform the DL/I call. If this field contain zeros, it means position was not established and therefore no level was set. This generally happens when a DL/I call cannot be processed for some reason.

SEGMENT
This field tells you the 8-character name of the segment accessed by the DL/I call.

STAT CODE
This field tells you the status code returned after the call (if the status code was not blank).

DL/I Calls
This field tells you how many times this particular DL/I call was issued and had the five unique characteristics listed in the previous five columns.

IWAITS
This field tells you the number of times IMS was inactive.

IWAITS/CALL
This field tells you, by DL/I call, the average number of times IMS was inactive.

ELAPSED TIME
This field tells you, by DL/I call, the elapsed time for calls.

MEAN	The average elapsed time per DL/I call
MAXIMUM	The longest elapsed time for a single DL/I call

NOT IWAIT TIME
This field tells you, by DL/I call, the elapsed time minus the IWAIT time.

MEAN	The average NOT IWAIT TIME

MAXIMUM	The longest single NOT IWAIT TIME

NOT IWAIT TIME includes any time spent by higher priority tasks running in the IMS region. NOT IWAIT TIME might be about equal to total processor time if the IMS database region is the high priority task and no low priority tasks are causing interrupts.

DISTRIB.NUMBER
The distribution number is a reference number to be used in conjunction with the Distribution-Appendix report. For a description of what it means, see “Distribution-Appendix Report”.

C, A, and B
These letters over the IWAITS/CALL, ELAPSED TIME, and NOT IWAIT TIME columns are reference letters to be used in conjunction with the Distribution-Appendix report. For a description, see “Distribution-Appendix Report”.

BATCH Total
For this execution of the DB Monitor this field tells you:

DL/I Calls	The total number of DL/I calls
IWAITS	The total number of times IMS was inactive
IWAITS/CALL	The average number of IWAITS per DL/I call

ELAPSED TIME MEAN
The average time per IWAIT

NOT IWAIT TIME MEAN
The average elapsed time, minus IWAIT time

Using the Report

The primary usefulness of the DL/I-Call-Summary report is to track down DL/I calls that are causing a large number of IWAITs. If the number in the IWAITS/CALL field is relatively high, you want to know why. Because the number in the report is related to a specific DL/I call, segment, and PCB, you can trace the IWAITs back to a specific part of an application program. In addition, by using the Distribution-Appendix report, you can see how many of the DL/I calls being issued are distorting the average in the IWAITS/CALL field. See “Distribution-Appendix Report” for additional ways in which information from the DL/I-Call-Summary report can be used.

Remember, in tuning to decrease I/O operations, pay most attention to calls issued a large number of times. It is more profitable to save 1 second on a call executed 2000 times than to save 5 seconds on a call executed ten times (2000 versus 50 seconds).

Distribution-Appendix Report

The Distribution-Appendix report takes specific events for which a time or a total has been generated and distributes the events across ranges. It does this for specific events from the Program-I/O report and the DL/I-Call-Summary report. Use a Distribution-Appendix report when you suspect some unusual combination of events has occurred, and the times or totals on the Program-I/O or DL/I-Call-Summary report do not give you enough information to highlight the problem.

To get an idea of when the Distribution-Appendix report is useful, use the following example. Suppose in a DL/I-Call-Summary report, the following row of information appears:

```
                                            (C)
PCB       CALL      LEV NO.       DL/I   IWAITS         DISTRIB.
NAME      FUNC      SEGMENT       CALLS  CALL    ...    NUMBER
----      ----      -------       -----  ------  ----   --------
DHVBT203  DLET      (01)A1111111  11     8.63           11C
```

The (C) over the IWAITS/CALL field means that this event is detailed on the Distribution-Appendix report. The DISTRIB. NUMBER field says this event is broken down on the Distribution-Appendix report specifically on line 11 C. In this example, the IWAITS/CALL field value of 8.63 is a very high number relative to the other IWAITS/CALL numbers on the DL/I-Call-Summary report. For more information, check line 11 C in the Distribution-Appendix report. It looks like this:

```
11 C ....0....0....1....2....3....4....5....6....7....8....INF
           0    0    0    0    0    2    4    1    3    1
```

The numbers next to line 11 C (0 to INF) are predefined ranges. The numbers beneath line 11 C are the distribution of the event. Look for more information about why the IWAITS/CALL total is high (for DLET calls issued under PCB DHVBT203 against segment A1111111). The distribution of numbers beneath line 11 C says that of the 11 DLET calls issued:

- Two calls caused 5 IWAITs
- Four caused 6 IWAITs
- One caused 7 IWAITs
- Three caused 8 IWAITs
- One caused more than 8 IWAITs

Because one call caused more than 8 IWAITs, this call might be distorting the average in the IWAITS/CALL field in the report. Tune your database to eliminate relatively high IWAITs per DL/I call, investigate the call that required more than 8 IWAITS. In this example, the interval between distributions is 1. For other entries (for example, line 4B in the report in Figure 48 on page 259), the intervals are much larger than 1. In these cases, interpret the data as (using line 4B): 1 call fell in the range of 16000 to 32000.

Figure 48 on page 259 is an example of a Distribution-Appendix report. To read it, remember:

- The lines in the Distribution-Appendix report are always identified by a number (55, 56, etc.) or a number and a character (3A, 3B, etc.). The numbers are the numbers used in the DISTRIBUTION NUMBER fields in the Program-I/O or DL/I-Call-Summary report. The characters are used in the DL/I-Call-Summary report to identify which event is being distributed.
- The numbers next to the line are predefined ranges across which an event can be distributed. These so-called “buckets” are made appropriate to the event. For example, buckets 0, 1, 2, 3, etc. are appropriate for distributing IWAIT *totals* for DL/I calls. (In the example used, we wanted to know how many calls required 0, 1, 2, 3, etc. IWAITs.) Buckets with ranges such as 0, 1000, 2000, etc. are appropriate for all other events in the Program-I/O and DL/I-Call Summary reports because all other events are *times*, not totals. These 0, 1000, 2000, etc. numbers are in microseconds. A microsecond is one millionth of a second (in

other words, 2000 microseconds equals 0.002000 second). The buckets have default values, but you can redefine the buckets if you wish (more about this later).

- The number below the line is always the number for the event being distributed.

```
     IMS MONITOR   ****DISTRIBUTION APPENDIX****       TRACE START 1989 076  12:42:54     TRACE STOP  1989 076  12:43:07  PAGE  0014

#    1A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      0          0          0          0          0          0          0          0          0          1
     1B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      0          0          0          0          0          0          0          0          0          1
     1C.........0..........0..........1..........2..........3..........4..........5..........6..........7..........8....INF
                      0          1          0          0          0          0          0          0          0          0
#    4A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      5          0          0          1          0          1          0          0          0          0
     4B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      5          0          0          1          0          1          0          0          0          0
     4C.........0..........0..........1..........2..........3..........4..........5..........6..........7..........8....INF
                      7          0          0          0          0          0          0          0          0          0
#    5A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      6          0          0          0          0          0          0          0          0          0
     5B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      6          0          0          0          0          0          0          0          0          0
     5C.........0..........0..........1..........2..........3..........4..........5..........6..........7..........8....INF
                      6          0          0          0          0          0          0          0          0          0
#    6A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      8          0          0          0          0          0          0          0          0          0
     6B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      8          0          0          0          0          0          0          0          0          0
     6C.........0..........0..........1..........2..........3..........4..........5..........6..........7..........8....INF
                      8          0          0          0          0          0          0          0          0          0
#    7A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      2          0          0          0          0          1          0          0          0          0
     7B.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      2          0          0          0          0          1          0          0          0          0
     7C.........0..........0..........1..........2..........3..........4..........5..........6..........7..........8....INF
                      3          0          0          0          0          0          0          0          0          0
#    8A.........0.......1000.......2000.......4000.......8000......16000......32000......64000.....128000.....256000....INF
                      3          0          0          0          0          0          0          0          0          0
⋮
#  131..........0.......2000.......8000......24000......50000.....100000.....150000.....200000.....250000.....300000....INF
                      0          1          5          2          0          0          0          0          0          0
#  132..........0.......2000.......8000......24000......50000.....100000.....150000.....200000.....250000.....300000....INF
                      0          0          5          1          0          0          0          0          0          0
#  133..........0.......2000.......8000......24000......50000.....100000.....150000.....200000.....250000.....300000....INF
                      0          0          3          0          0          0          0          0          0          0
```

Figure 48. Distribution-Appendix Report

The TRACE START and TRACE STOP fields at the top of the report tell you the time when the DB Monitor was *last* started and stopped. The time is generated by the time-of-day clock. Clock times are read as follows:

```
Clock time = hh.mm.ss
```

where:

hh	Hours 0 through 23
mm	Minutes
ss	Seconds

If the DB Monitor was on during an entire batch run, the trace start and trace stop times are the times at which the batch run started and stopped. If the DB Monitor was turned on and off more than once in the same batch run, the trace start and trace stop times are the times at which the monitor was *last* started and stopped.

How to Generate the Distribution-Appendix Report

The Distribution-Appendix report is not generated automatically when the DB Monitor is run. To get the Distribution-Appendix report, you have to include a DIS input control statement in the analysis control data set.

Events That Can Be Distributed and Their Default Ranges

Table 9 shows the events that can be distributed in the Program-I/O and DL/I-Call-Summary reports. Each event has an ID.

Table 9. Events That Can Be Distributed and Their IDs

Events That Can Be Distributed	ID
Program-I/O report	
IWAIT time for OSAM	D23
IWAIT time for VSAM	D24
IWAIT time for HSAM	D34
Elapsed time per DL/I call	D11
Not IWAIT time per DL/I call	D12
IWAITs per DL/I call	D13

Table 10 shows, using this ID, what each predefined set of ranges consists of when the default ranges are used.

Table 10. Predefined Ranges When the Default Is Used

ID	Default Ranges
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

Notice that the first number in a range always defaults to zero, and the last number always defaults to infinity (INF).

Redefining the Default Ranges

You can redefine the default ranges. To do this, you have to include an input control statement in the analysis control data set for each default range you want to override.

Related Reading: The analysis control data set is explained in *IMS/ESA Utilities Reference: System*.

To override default ranges, you specify control statements in the form Dn n1,n2,...where:

- Dn is the ID of the event to be distributed (D23, for example, is the ID for IWAIT time for the OSAM event in the Program-I/O report)
- n1 is a specific default value. Each of the 10 default values or buckets can be redefined. For example, the Program-I/O report IWAIT time for OSAM could be redefined as follows:

```
D23    0,500,1000,1500,2000,4000,,,100000,500000
```

This would result in the 7th and 8th ranges remaining at their default values (150000 and 200000) and the final range (INF) remaining at its default value.

Monitor-Overhead Report

This report tells you about the overhead required to run the DB Monitor. Because the monitor runs while IMS is executing, the monitor’s use of resources causes IMS’s performance to be slightly less than it is when the monitor is not on.

Fields in the Report

Figure 49 is an example of a Monitor-Overhead report.

```
     IMS MONITOR   ****MONITOR OVERHEAD****            TRACE START 1989 076  12:42:54     TRACE STOP  1989 076  12:43:07  PAGE  0013
MONITOR OVERHEAD DATA                                                                                                            M
_____________________                                                                                                            M
      13144 MILLISECONDS, TRACE INTERVAL
        550 MILLISECONDS, MONITOR MODULE TIME
        853 MONITOR RECORDS WERE PRODUCED
        645 MICROSECONDS PER MONITOR ENTRY
```

Figure 49. Monitor-Overhead Report

The meaning of the various fields in the report is as follows:

TRACE START and TRACE STOP
The trace start and trace stop fields tell you the time when the DB Monitor program was *last* started and stopped. The time is generated by the time-of-day clock. Clock times are read as follows:

```
Clock time = hh.mm.ss
```

where:

hh	Hours 0 through 23
mm	Minutes
ss	Seconds

If the DB Monitor was on during an entire batch run, the trace start and trace stop times are the times at which the batch run started and stopped. If the DB Monitor was turned on and off more than once in the same batch run, the trace start and trace stop times are the times at which the monitor was *last* started and stopped.

MILLISECONDS, TRACE INTERVAL
This field tells you how long the monitor was turned on.

MILLISECONDS, MONITOR MODULE TIME
This field tells you, during the time the monitor was turned on, how long code in the monitor was actually being executed. (During this time, no IMS code can be executed, so this field tells the real overhead for using the monitor.)

MONITOR RECORDS WERE PRODUCED
This field tells you how many records the monitor wrote.

MICROSECONDS PER MONITOR ENTRY
This field tells you the average length of time it took the monitor to write a record.

Chapter 28. Program-Isolation-Trace Report Utility (DFSPIRP0)

The Program Isolation Trace logs all PI enqueue and dequeue requests in X'67FA' PI trace log records when PI tracing is active. Use the PI-Trace Report utility to print a report from these X'67FA' log records that shows only those enqueue requests that required a wait (the resource was not immediately available). The report can be restricted to a time period by specifying a PRINT control statement.

Input and Output

The report produced by the Program-Isolation-Trace Report utility shows:

- Requested resource (DMB name, DCB number, and 4-byte hexadecimal ID (RBA)).
- Time of the enqueue request (time of call).
- Elapsed time of the wait (how long the requesting task had to wait for the resource to become available); if PI trace timing is in effect, use the `/TRACE ALL` command.

  Exception: No elapsed wait time for Fast Path is recorded.
- Names of requesting and holding PSBs.
- Total number of waits by ID, DCB, and DMB.

In a Fast Path environment, the requested resource varies according to the first byte of the 4-byte hexadecimal ID (RBA). The first byte of the ID equates to EPSTLKID, the lock sub ID. The following list contains the hexadecimal IDs for EPSTLKID and names within the DMB name for the requested resource:

EPSTLKID ID	**Name**
(EPSTCILK)X'00'	DEDB area name
X'F0'	NOTAVAIL
(EPSTMDLK)X'F1'	MSDB symbolic name
X'F2'	NOTAVAIL
X'F3'	NOTAVAIL
(EPSTARLK)X'F8'	DEDB area name
X'FF'	NOTAVAIL

The remainder of the ID contains the high 3 bytes of the CI, regardless of what EPSTLKID contains.

If PI trace timing is in effect, the PI trace log records (X'67FA') of enqueue requests for any ID appear in the log data set in the same order in which the ID was acquired. Thus, the requesting transaction of an enqueue request is considered to be the holding transaction of the next enqueue request for the same ID, if the latter required a wait. If timing is not in effect during PI tracing, it is possible, although not likely, that the PI trace log records might not be in the same order in which the ID was acquired. This can occur if an enqueue request acquires an ID and, just before it is added to the PI trace logger buffer, a higher priority task interrupts with an enqueue request for the same ID. This second task is be required to wait, but its PI trace log record is be ahead of the record corresponding to the holding task.

Restriction: This cannot occur if timing is in effect because the log records used by the trace report utility are added to the log buffer only after the resource has been acquired.

JCL Requirements

The following are required:

- A JOBLIB DD statement
- An EXEC statement
- DD statements specifying input and output data sets

JOBLIB DD Statement

The JOBLIB DD statement describes the program library containing the utility program. The format of this statement is:

```
//JOBLIB DD DSNAME=IMS.RESLIB,DISP=SHR
```

EXEC Statement

The EXEC statement invokes the utility program. The format of this statement is:

```
//PITRACE EXEC PGM=DFSPIRP0
```

DD Statements

LOGTAPE DD
Describes the log input data sets. Multiple data sets are concatenated if the log extends over more than one data set. The format is:

```
//LOGTAPE DD DSNAME=nnnn,DISP=OLD,VOL=SER=xxxxxx,
//           UNIT=YYYY
```

where nnnn is the data set name, xxxxxx is volume serial, and YYYY is the input type of the log data set.

PRINT DD
Describes the report data set. The format is:

```
//PRINT DD SYSOUT=A
```

SORTLIB DD
Describes the sort program library. The format is:

```
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
```

SYSOUT DD
Describes the message output data set for sort. The format is:

```
//SYSOUT DD SYSOUT=A
```

SORTWK01-32 DD
Describes the sort program's work data sets. The space defined can vary. You must have at least three data sets. 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)
```

The SYSDA must equal one type of disk storage because the sort program does not allow work areas across mixed device types.

SYSPRINT DD
Describes the system messages data set. The format is:

```
//SYSPRINT DD SYSOUT=A
```

SYSIN DD
Describes the data set containing the optional utility control statement (described in next section). If it is included in the input stream, this DD statement is normally DD *. This statement can be omitted if the optional utility control statement is also omitted.

Utility Control Statement

Use the optional utility control statement to specify the starting and stopping times desired for the report. The control statement is printed in the program output. The format is:

```
►►──┬───────┬──┬────────────┬──┬───────────┬──┬────────────┬──────────────────►◄
    └─PRINT─┘  └─START=HHMM,┘  └─STOP=HHMM,┘  └─DATE=MM/DD─┘
```

For START and STOP, *HH/MM* specify hours (00-99) and minutes (00-59)

For DATE, *MM/DD* specify the month (01-12) and the day (01-maximum number of days for the specified month)

PRINT must be coded in the operation field. Keywords can be entered in any order, separated by a comma.

Restriction: Keywords cannot be repeated.

Attention: Because the times are all relative to the beginning of the trace period, the control statements for this utility are not changed. However, problems will occur if this utility is used to create a report for a trace period that includes a change in the local time.

Data can be entered from columns 1 through 71 with one or more spaces before and after PRINT. A space following a parameter indicates the end of data; anything after the space is considered a comment.

If only PRINT is specified, or if a control statement is omitted (SYSIN data set not provided), or if a blank control statement is supplied, the entire log data set is searched. If only PRINT and START are specified, the log data set is searched to the end from the start time. If only PRINT and STOP are specified, the log data set is searched from the beginning until the stop time.

START and STOP times are relative to the date specified or, if DATE is omitted, relative to the date on which PI tracing started, as recorded in the PI trace log records. If only PRINT and DATE are specified, the log data set is searched for records beginning at 00:00:00 on that date. If DATE and STOP are specified without START, the log data set is searched from 00:00:00 on that date until the stop time relative to that date is encountered.

The DATE specified must be within 12 days of the date that PI tracing was started.

Example

The following example requests a report of all enqueues that required a wait and were requested between 0900 and 0930 on the date that PI tracing was started.

```
//PITSTATS JOB MSGLEVEL=1
//JOBLIB    DD DSNAME=IMS.RESLIB,DISP=SHR
//*
//        EXEC PGM=DFSPIRP0,REGION=150K
//LOGTAPE   DD DSNAME=IMSLOG,DISP=OLD,
//             UNIT=TAPE,VOL=SER=XXXXXX
//PRINT     DD SYSOUT=A
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SYSOUT    DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//SORTLIB   DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SYSIN     DD *
           PRINT START=0900,STOP=0930
/*
```

If the report is required to be from 11:30 p.m. until 1:10 a.m. the control statement is:

```
PRINT START=2330,STOP=2510
```

The following examples are all identical if PI tracing was first started on 11/25.

```
PRINT START=1000,STOP=1530,DATE=11/27
PRINT START=3400,STOP=3930,DATE=11/26
PRINT START=5800,STOP=6330
```

The report heading date is the date from the log when PI tracing was first started. The time of the calls and the start and stop times in the report are relative to this date.

Chapter 29. SB Test Utility (DFSSBHD0)

Use the SB Test utility for tuning and problem determination of sequential buffering (SB). This utility allows you to reprocess the SB buffer handler call sequence issued during previous execution of other programs.

If you provided a SBIC control statement in the //DFSCTL file of the JCL of an IMS program, all internal IMS calls to the SB buffer handler are captured on the IMS log during execution of that program. These "SB image capture log records" can then be used as input to the SB Test utility.

Related Reading: For more information about the SBIC control statement, see "SB Image Capture (SBIC) Control Statement" in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The SB Test utility can only process the SB image capture log records of one program execution which used the same PSB as the utility. If your input data set for this utility contains SB image capture log records from multiple program executions using the same PSB, you need to specify which execution of the program you want this utility to process SB image capture log records for. Otherwise, this utility automatically processes the SB image capture log records of the first execution of the program. See "SELECT Statement" on page 272.

The SB Test utility causes SB buffer handler calls to be processed as if they had been issued during the execution of a normal program. However, the SB buffer handler only simulates DB Read I/Os, unless you specify that it issue DB Read I/Os. See "DBIO Statement" on page 271.

Because the SB Test utility can be used to recreate and reexecute the SB buffer handler call sequence generated by an application, it can be useful for the documentation and fix-testing of problems related to algorithms of the SB buffer handler that analyze the I/O reference pattern, as well as other SB buffer handler-related problems.

The SB Test utility can be used during tuning to experiment with different SB parameter values (different numbers of buffer sets) of the SB algorithm and study the impact of the changes in the //DFSSTAT report.

Exception:A reexecution of an SB buffer handler call sequence by the SB test utility does not always result in exactly the same number of sequential reads and random reads as the original execution, even though the same SB buffer handler parameters (BUFSETS values) are used. This is because the SB buffers invalidated during both executions are not always identical.

Related Reading: For more information about invalid SB buffers, see *IMS/ESA Utilities Reference: System*.

Execution of the SB Test utility does not update database data sets. This utility is not sensitive to database changes, such as ISRT, REPL, and DLET, performed between the original SB image capture and execution of this utility.

Figure 50 on page 268 depicts the data set requirements for the SB Test utility.

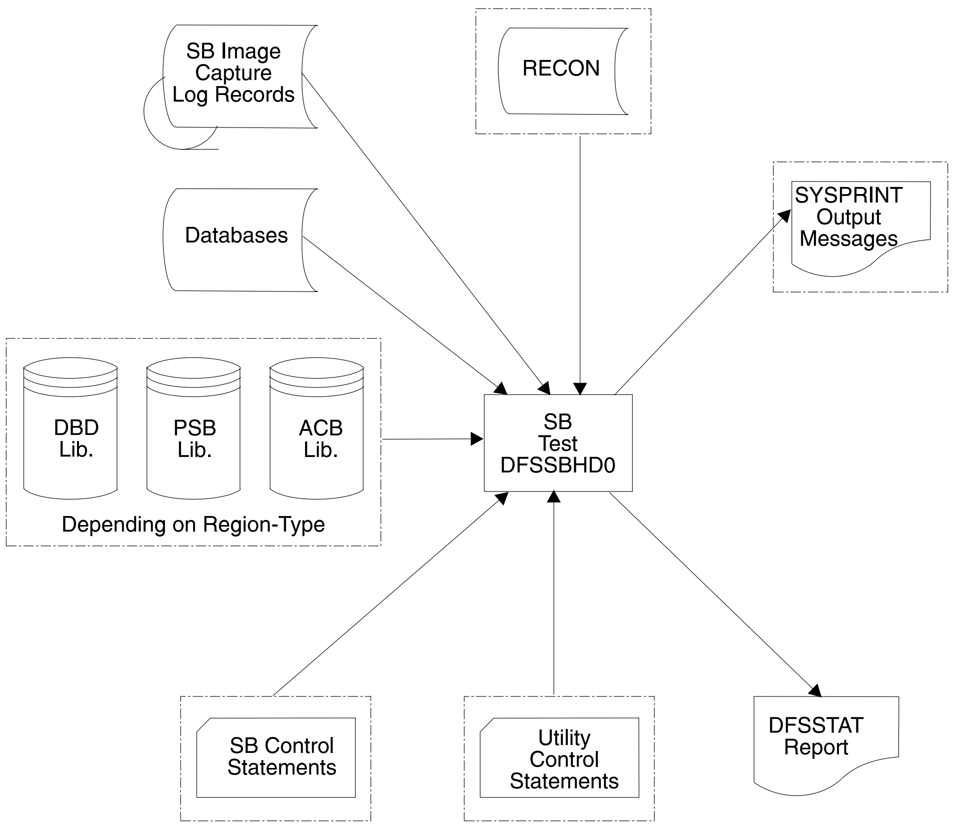


Figure 50. Data Set Requirements for the SB Test Utility

Restrictions

The following restrictions apply to the SB Test utility:

- The SB Test utility can process the SB image capture log records of only one program execution each time it is run.
- This utility can only be executed in batch regions.
- In order to execute, this utility must access the PSB and all referenced DBDs used by the application that generated the captured SB buffer handler calls, as well as the DB data sets of the application.
- This utility does not process SB image capture log records for DB PCBs with a Load processing option.
- This utility can only be used to simulate the sequential buffering behavior of an individual program, not the behavior of the entire system. Do not confuse global system optimization with the local SB optimization of a single program.
- Using the IMS Monitor is not recommended during execution of this utility, because it does not provide meaningful reports for this utility.

Input and Output

The SB Test utility uses the following input:

- A SYSUT1 data set containing the SB image capture log records.

- An optional SYSIN file containing DBIO and SELECT statements. See "Utility Control Statements" on page 271.
- The PSBs, DBDs and databases used by the application which generated the SB image capture log records.

Execution of the SB Test utility produces the following output:

- A SYSPRINT file listing the utility control statements that were read from SYSIN, along with messages written by the utility.
- A DFSSTAT file containing SB summary and SB detail reports.

Related Reading: For a detailed description of DFSSTAT reports, see *IMS/ESA Utilities Reference: System*.

JCL Requirements

The SB Test utility is executed as a standard MVS job. The following are required:

- A JOB statement that you define
- An EXEC statement
- DD statements defining inputs and outputs

EXEC Statement

This statement must be in one of the following forms:

- If the program which generated the SB image capture log records was running with a PSB from the PSBLIB:
  ```
  PGM=DFSRRC00,PARM='DLI,DFSSBHD0,psbname'
  ```
- If the program which generated the SB image capture log records was running with a PSB from the ACBLIB:
  ```
  PGM=DFSRRC00,PARM='DBB,DFSSBHD0,psbname'
  ```
- If the program which generated the SB image capture log records was a utility program running without a PSB:
  ```
  PGM=DFSRRC00,PARM='ULU,DFSSBHD0,dbdname'
  ```

psbname is the name of the PSB used by the application which generated the SB image capture log records. dbdname is the name of the DBD which was used by the utility that generated the SB image capture log records.

The normal IMS positional parameters such as SPIE, BUF and DBRC can follow psbname or dbdname.

Related Reading: See DBBBATCH or DLIBATCH in *IMS/ESA Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the PSB and/or DBDs to be used when running this utility in a DLI or ULU region.

IMSACB DD
Defines the library containing the ACBs for the PSB and DBDs to be used when running this utility in a DBB region.

DFSVSAMP DD
Describes the data set that contains the buffer information required by the DL/I Buffer Handler. This DD statement is required.

Related Reading: For additional information on control statement format and buffer pool structure, see “Specifying the IMS Buffer Pools” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DFSCTL DD
Points to the card image file containing the SB control statements. This can be either a sequential data set or a member of a PDS. The record format must be F, FB, or FBS and the record length must be 80.

The SBIC control statement must be provided in the //DFSCTL input data set of the executed program in order to create the SB image capture log records necessary as input to this utility.

If SB Test is being used for problem determination, SBSNAP, SBESNAP and SNAPDEST control statements can be provided as required in the //DFSCTL card-image input data set.

Related Reading: For more information, see the section about SB control statements in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

SYSABEND DD or SYSUDUMP DD
Defines a dump data set. These DD statements are optional. If both statements are present, the last occurrence is used for the dump.

IEFRDER DD
Describes the system log created during sequential buffer testing. The data set resides on a tape or DASD.

IEFRDER2 DD
Describes the secondary system log created during sequential buffer testing. The data set resides on a tape or DASD. Include this statement only when you want dual log output.

RECON1 DD
Defines the first DBRC (Database Recovery Control) RECON data set.

If you are using dynamic allocation, do not use these RECON data set ddnames.

RECON2 DD
Defines the second DBRC RECON data set.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set used by the control region.

database DD
Defines the database data sets used by the application or utility that generated the SB image capture log records.

SYSUT1 DD
Defines the data set containing the SB image capture log records. This DD statement must be supplied. Typically, this data set is either an IMS log data set or an extract from an IMS log data set. The data set can reside on either a tape or DASD.

SYSIN DD
Points to a card image data set containing utility control statements. The data set can reside on a tape, DASD, or be routed through the input stream. The record format of the data set must be F, FB, or FBS and the record size must be 80. This statement is optional.

SYSPRINT DD
Defines the message output data set. The data set can reside on a tape, DASD, or a printer, or be routed through the output stream. The following DCB parameters are provided by DFSSBHD0 and should not be specified on this DD statement:

- RECFM=FBA
- LRECL=121
- BLKSIZE=605

DFSSTAT DD
Points to the //DFSSTAT file, which is used to write sequential buffering statistics. The following DCB attributes for this statement are set by IMS modules:

- RECFM=FBA
- LRECL=133
- BLKSIZE=1330

Utility Control Statements

The SB Test utility uses two control statements:

- DBIO
- SELECT

DBIO Statement

Use the DBIO statement to specify whether the SB buffer handler issues real DB Read I/Os or only simulates DB Read I/Os. Issuing real DB Read I/Os substantially increases the elapsed time of the job step. However, you can measure and compare the DB Read I/O times while you are executing the utility multiple times with different numbers of buffer sets during tuning. The DBIO statement has the following format:

```
          ┌─NO──┐
►►──DBIO──┴─YES─┴──────────────────────────────────────────────────►◄
```

YES
Specifies that the SB buffer handler issues DB Read I/Os.

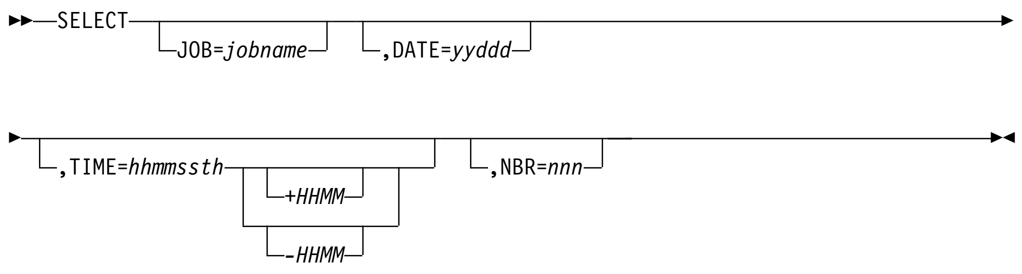
When YES is specified, only those OSAM Read I/O operations issued by the OSAM buffer handler are issued. I/O operations to VSAM database data sets and output DB I/O operations are not issued during execution of this utility.

Recommendation: Using DBIO YES is not recommended if the DB has been reorganized or reloaded between the SB image capture and execution of this utility, and the new DB version has a smaller High Used relative byte address (RBA) than the original DB version. This is because SB image capture log records that describe calls to the SB buffer handler for RBAs higher than the actual High Used RBA are not processed if DBIO YES is specified. Therefore, the SB buffer handler call sequence is not the same during execution of this utility and execution of the original program, so no comparison can be made.

NO
Specifies that the SB buffer handler only simulates DB Read I/Os. NO is the default. Even if DBIO NO is specified, this utility must access the DB data sets.

SELECT Statement

Use the SELECT statement when your input data set contains SB image capture log records that were generated by multiple executions of programs using the same PSB. Use this statement to select only the SB image capture log records of a specific application execution. You can specify only one SELECT statement. If you do not specify a SELECT statement, the first program execution is selected. The format of the SELECT statement is:



The word SELECT must start in column 1, must be followed by exactly one blank and then by one or multiple keyword parameters. Multiple keyword parameters must be separated by one comma (,) and the last keyword parameter must be followed by at least one blank. The keyword parameters can appear in any order.

Restriction: The SELECT statement cannot continue on another line.

JOB=
Specifies the jobname of the application whose SB image capture log records should be selected.

DATE=
Specifies the date when the application was started.

TIME=
Specifies the approximate time when the application was started. This value must be coded as an 8-digit number including all leading and trailing zeros: hh (hours), mm (minutes), ss (seconds), t (tenths of a second), h (hundredths of a second). This keyword can be coded with or without the DATE= keyword.

If you specify the TIME= keyword, this utility ignores any program start records of program executions started prior to the specified time. The following offset from UTC is optional. It is only needed if the current UTC offset is different from that which was in effect when the image capture log records were created due to an intervening entry or exit from Daylight Saving Time.

+|- Specifies the sign of the time zone offset to UTC.

HH
Specifies the number of whole hours of offset to UTC.

MM
Specifies the minutes of offset. This can be 00, 15, 30, 45.

NBR=
Specifies that the SB image capture log records of the *n*th execution of a given program within a given region and within a hundredth of a second are selected. This keyword is useful when other keywords are not sufficient for a unique identification of the application start record to be selected.

Example

The following example shows JCL for execution of the SB Test utility.

```
//    EXEC  DLIBATCH,MBR=DFSSBHD0,PSB=xxxxxxxx
//SKILLDB   DD DSN=user.db.data.dsname,DISP=SHR
//SKILLIX   DD DSN=user.db.indx.dsname,DISP=SHR
//SYSUT1    DD DSN=imslog,DISP=SHR
//DFSVSAMP  DD *
4096,6
//DFSCTL    DD *
SBPARM ACTIV=COND,BUFSETS=10
//SYSIN     DD *
DBIO YES
//DFSSTAT   DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
/*
```

Part 6. Utility Control Facility

Chapter 30. Utility Control Facility (DFSUCF00)

This chapter describes the Utility Control Facility (UCF). This program controls the execution of the utilities.

Use the UCF to implement the functions of the reorganization utilities described in this manual. The correct execution of the UCF depends on your knowledge of the requirements for these utilities.

The UCF is a utility that acts as a controller for the execution of other utilities. The UCF is driven by control statements coded in a freeform manner. You can supply multiple control statements that cause the UCF to execute multiple utilities (such as execution of the Reorganization and batch Image Copy utilities) on the same or multiple databases 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 then restarted at your convenience.
- The outstanding Write to Operator with Reply (WTOR) function 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 you from certain operational problems. (See “Normal Processing” on page 279.)

Restrictions

The following restrictions apply to the Utility Control Facility:

- For restart of the user’s Initial Load program, the HD Reorganization Reload utility, or both under the UCF, the following restrictions apply:
  - The restart applies only to a VSAM data set.
  - Multiple load PCBs can be included in the same PSB. However, during the initial load, or the restart of the initial load, the initial load program must use only one load PCB per execution of a FUNCTION=IL statement.
  - Root segments must be keyed such that the user’s HDAM randomizing module can locate the root.
  - Restart must begin with an ISRT for the next root segment following the root with the key equal to the key feedback area.
  - The user-written initial load program is responsible for repositioning the input file.
- Observe the following precautions for the restart of the user’s Initial Load program or if restart is being done for an HDAM database that was being reloaded by the HD Reorganization Reload utility:
  - The checkpoint value for this restart must be the same value as specified in the failing UCF function.
  - If the root sequence field is nonunique, any root segments that were inserted after the checkpoint at which the restart was made remain in the database. The only valid condition for restart is a controlled termination (application program returns with a nonzero return code).

  - If the root sequence field is nonunique and a root segment insert is done for a segment that already exists in the database because of segments inserted after the checkpoint, the data is compared. If the segment data is the same, the old segment is overlaid with its replacement and the dependent segments are dropped because they are reinserted by subsequent user/reload insert. This occurs only until a unique root is found. After a segment with a new key or with different data is encountered, LB status codes are returned for any subsequent duplicates.

    On a restart, if a path call is used to insert a root and its dependent when the root already exists in the database, any insert of the dependent segment in that root in the path call fails.

- Explicitly close any data sets opened in any user-written routine; otherwise, the UCF abends.
- When reorganizing a VSAM database using one execution of the UCF, specify EXEC=STOP on the utility control statement requesting the unload function. When the unload operation has been completed, the UCF stops execution. The database can then be deleted and reallocated using the Access Method Services utility, and the UCF can then be restarted.

  If multiple databases are unloaded, you can specify 'STOP' in each unload function or in only the last unload function that causes UCF to stop after all databases are unloaded. UCF functions are performed on databases in the collating sequence order of the database names (DBX before DBY, etc.). Place the STOP in the statement referring to the last database name.
- Do not use the UCF for database backout or restart; the Database Backout utility does not require any of the functions provided by the UCF.
- During reorganization of a database 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.
- Do not use the UCF to initially load a GSAM database.
- The UCF cannot be used to execute the Online Database Image Copy utility, the Database Surveyor utility, or the Partial Database Reorganization utility.
- The FUNCTION=OP card must have the same checkpoint value and request parameters as the FUNCTION=OP card in the failing UCF step.
- In case of an abrupt system termination, such as a loss of power, you must terminate the unclosed output data set before restart is begun.
- If your database has logical relationships, run utilities either completely with or without control of UCF. Mixing UCF and non-UCF utilities can cause unpredictable results.
- If, during restart of an initial load or reload program, UCF determines there are no valid checkpoints from which to restart, UCF restarts from the beginning of the load or reload. If any segments had already been written to the database, message DFS730I, reason code I,30 is issued. When this occurs, you must scratch and reallocate database data sets before restarting again.
- Database recovery is not supported for UCF. If you try to run the DB Recovery utility with the UCF, you receive error message DFS3142.

- FUNCTION=ZB does not update RECON when DBRC is active.
- The UCF does not support the Change Accumulation utility, the DB Recovery utility, or the Database Image Copy 2 utility.

Normal Processing

Normal processing requires control statements for the direction of all utility functions except Database Scan (DFSURGS0), Prefix Resolution (DFSURG10), and Prefix Update (DFSURGP0). The UCF automatically generates all required statements for these three functions except:

- If an unload or reload only is requested, Database Scan, Prefix Resolution, and Prefix Update processing is not automatic; you must provide control statements to request execution of these utilities.
- If only an initial load is requested, processing of these utilities is automatic, unless keywords on their control statements indicate no processing is to take place. One control card is necessary to request the execution of Database Scan. Subsequent scan requests are ignored.

The control statements are read at one time and a control data set is built. All entries in the control data set are cross-referenced to verify that no conflicting requests are made and that all logical relationship functions are either requested by you or generated by the UCF.

The UCF executes the utilities in a particular order, regardless of the order in which you submit the control statements. The order of execution is as follows:

1. HISAM Reorganization Unload
2. HISAM Reorganization Reload
3. Database Scan
4. HD Reorganization Unload
5. HD Reorganization Reload
6. Initial Load
7. Prefix Resolution
8. Prefix Update
9. Reorganization Unload for Secondary Indexes
10. Reorganization Reload for Secondary Indexes
11. Image Copy (batch)
12. Database Zap
13. Module Zap

When the same function is requested on several control statements, the UCF executes the functions in ascending order of database name. If two or more control statements for the same function also specify the same database name, then these statements are executed in the order they were read in by the UCF.

In summary, the control statements are executed in the order of functions shown above. Within a function they are executed in database name order; within the same function and database name, they are executed in the order submitted.

Processing proceeds by determining the function to be started next, recording the event in the journal data set, and attaching the appropriate utility.

All functions are organized and executed in a manner that protects against certain operational difficulties that might otherwise occur. Consider, for example, the case of reorganizing a database containing a logical parent when the logical child has a direct pointer to it. If the logical parent database is unloaded and *reloaded* before database scan is executed, the logical relationship is destroyed. The UCF, however, scans the logical child database first, unloads, and then reloads the logical parent database.

The stand-alone 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.

The HD Reorganization Reload utility step internally generates a Batch Database Image Copy utility request upon successful completion. This step does not take place immediately if there are other higher priority steps yet to process. The Secondary Index Reload step does not initiate a Batch Database Image Copy utility step.

Initial Load Application Program Considerations

General-use programming interface

Initial load programs can be run under control of the UCF and 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 ────►  DBPCB 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 ───────► DBPCB list
```

When restart processing of a user's initial load program is in progress, a status code of UR (this is a restart) is returned in the load PCB upon entry to the program. Another parameter, the fully concatenated key contained in the PCB key feedback area, is also passed. The information in this key feedback area determines the nature of the restart, and two conditions apply:

- If the information is other than zeros, restart processing begins from the point of termination.
- If the information is zeros, restart processing is f rom the beginning of the program.

Because the user's program needs to respond with the next root or dependent segment to be loaded, the user I/O files might have to be adjusted by the application program.

Because further inspection might be necessary to determine the restart point on the input files, the actual segment data is also provided. (This would be important, for example, if there are nonunique or unsequenced fields.)

Additionally, the user’s program needs to 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 database is checkpointed at the logical record preceding the root that was just loaded.
UR	This is a restart.
US	The operator has requested initial load program to stop processing.
UX	A combination of UC and US.

For both the US and UX conditions, processing can be stopped at the next checkpoint.

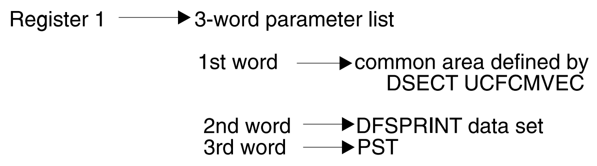
Be aware of certain restrictions that apply to Initial Load application programs. See “Restrictions” on page 277.

End of General-use programming interface

Initial Load Exit Routine

General-use programming interface

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 you to change checkpoint intervals, to checkpoint work data sets, or both. The interface to the exit routine is:



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.

You can synchronize (in whatever manner desired) checkpointing of individual data sets with checkpoints currently in effect for the Initial Load program by doing one of the following:

- Monitoring the count and checkpointing data sets when the counter (U7CKPTNK) reaches zero
- Forcing a checkpoint by clearing the counter to hexadecimal zeros and checkpointing data sets

Upon return to DFSUCP90, if the U7CKPTNK field is zero, a checkpoint record is written to the journal data set, and the field is reinitialized to the value contained in the U7CURCKP field.

End of General-use programming interface

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, restart messages are generated and the job ends with a return code of 4. If no stop-processing request was made, the next function is determined and processing continues as described for normal processing.

If errors are discovered during the error-checking phase of execution, the completion of the event is recorded as an error and processing ends to allow for restart of the function. When the entire job is completed, 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 termination requests from the user. The checkpoint function requires a journal data set, and you must allocate this for the UCF. The IMS 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 MVS checkpoints. The frequency of checkpoints can be specified as a user option or can be defaulted to every 2000 database related records.

For examples of the JCL for executing restart of the UCF, see “Examples” on page 324.

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, completed, or both is determined by matching the journal records to the functions in the control data set, and processing continues as in normal processing.

Depending on the type of checkpoint records that were last written to the journal data set, the restart function occurs in one of the following ways:

- If the records indicate the start of a functional utility (that is, one of the database 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 data sets are positioned as necessary.

Restart processing requires the availability of all data sets that were in use at the time the checkpoint was taken.

During restart, when repositioning any multiple volume output data set, the DD statement must be changed to remove those volume serial numbers not used in the original execution. If this is not done, the output data set might not be correctly positioned.

Restart of the Prefix Resolution utility must be from the beginning of execution. For this utility, all data sets created up to the point of termination must be scratched and the utility must be restarted as if for the first time.

User-Supplied Exit Routine Processing

General-use programming interface

UCF allows user-supplied exit routines in all utilities to examine records or to compile statistics. While no IMS control data can be altered during the user exit routine processing, the user data can be altered as long as the segment record length requirements are observed. The user exit routine is required to maintain all IMS data unchanged during the routine's processing. The HD Reorganization Unload utility (DFSURGU0) and the HD Reorganization Reload utility (DFSURGL0) user exit routines can delete or insert segments and view blocks after they have been loaded.

The user exit routines are specified on the utility control statements associated with a given function by coding the EXITRTN keyword and specifying the name of the exit routine. The user exit routine must reside in the LINKLIB or be defined in a STEPLIB or JOBLIB data set.

On entry to the user exit routine, register 1 points to a parameter list that contains three entries:

Address of the data

Address of the DFSPRINT DCB

Address of the partition specification table (PST) at successful (nonabend) completion

The user exit routine is entered a final time with the address of the data equal to 0.

The HD Reorganization Reload utility user exit is entered at one of two locations:

At offset 0, when the reload record has been read from the unload tape

At offset 4, when the record has been loaded into the database

The user exit routine can write on the DFSPRINT data set by issuing a PUT macro statement for a MOVE MODE operation. The DFSPRINT data set is opened before the exit routine is entered. Any non-utility data set used by the exit routine must be opened and closed by the routine.

The user exit routines used with the HD Reorganization utilities can pass return codes in register 15 that inform the utility of segment disposition. The return codes recognized by the HD Reorganization utilities are:

Code	Meaning
0	Process normally
4	Delete the segment passed to the exit routine
8	Insert the segment pointed to by register 1 before the current segment. Return to the exit routine with the same segment that was originally passed to it.

The segment name and the segment level can be found in the PCB pointed to by the PST. Any logical children of the segment being inserted must be inserted separately into their own databases.

End of General-use programming interface

Service Aids

There are two types of service aids available with the UCF—error-point abends, and database zaps and module zaps.

Restriction: The zap aids should be used only by an IBM Field Engineering Program Support Representative or by someone authorized and under proper direction.

Error-Point Abends

The UCF allows selective abend requests. If a diagnostic message is generated during processing, a control statement can be used to select points at which to invoke an abend to the program. Specifying REQUEST=MSGALL indicates that all A or W type diagnostic messages are to be set as abend requests. The MSGNUM keyword indicates the exact message at which to abend if that message is issued during processing. (See “The FUNCTION=OP Statement” on page 291 for additional information.)

Database Zaps or Module Zaps

You can use the UCF to zap database 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 in 2-byte hexadecimal form. The data must not be separated by commas or spaces.

Related Reading: For further information on SPZAP, refer to *MVS/ESA Service Aids*.

Zaps on modules are performed in storage, while database zaps are performed on disk. If a zap statement contains a RELATE keyword, the module zap is performed before execution of the related module. Database 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. Database zaps are performed before unload utility functions, after load utility functions, and after all requested functional utilities have been executed. Refer toTable 11.

Table 11. Database Zaps

Before Unload Utility Functions:	After Load Utility Functions:
IM	SR
SU	DR
RU	RR
DU	RV

All other database zaps are executed after the last requested functional utility has executed.

Write-to-Operator-with-Reply Function

The Write-to-Operator-with-Reply function (WTOR) issues messages to, and processes replies from, the UCF user. The outstanding WTOR provided by the UCF allows you to query 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.

Restriction: The restart cannot be initiated by a WTOR.

The WTOR is processed between executions of the utilities and at checkpoint time.

In the following example, you receive message DFS367I at the console, request the status of the UCF's execution, and learn that it is executing the HD Reorganization Reload utility with the database "DBNAME". (The DBNAME is not included in the message for function PR.)

```
@09   *DFS367I UTILITY CONTROL FACILITY RUNNING,
        ENTER REQUESTS AS NEEDED
r 9, status
IEE600I  REPLY TO 09 IS 'STATUS'
DFS369I  FUNCTION IS DR FOR DATABASE DBDNAME
@10   *DFS367I UTILITY CONTROL FACILITY RUNNING,
        ENTER REQUESTS AS NEEDED
```

In response to the last DFS367I message, you might, for example, request that a checkpoint value be changed immediately as follows:

```
r 10,ckpnt=100
```

Or you 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, you can enter certain control replies and option replies in response to message DFS367I.

Control Replies	Option Replies	
END	Identifiers	
		FUNCTION=
FUNCTION xx END		SEQ=
		EXEC=
STATUS	Values	
		REQUEST=
		CKPNT=

Only one of the control replies can be entered on one response.

END
Stops processing at the next checkpoint in, or after, the current function (whichever checkpoint occurs first).

FUNCTION xx END
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" on page 325 shows the JCL for executing restart of the UCF.

STATUS
Causes a WTO message that explains which function, and, if possible, which database or data set are being processed. Status requests are processed between executions of the utilities or at a checkpoint, whichever occurs first.

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=PR,SEQ=004, the control data set with SEQ=004 must be associated with FUNCTION=PR.

If the FUNCTION= identifier is not entered on an option reply, FUNCTION=OP is assumed.

Restriction: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 Utility Control Facility is executed as a standard MVS job. The following are required:

- A JOB statement you define
- An EXEC statement
- DD statements defining input and output

For additional information on JCL requirements, see the UCF examples in“Examples” on page 324.

EXEC statement

The EXEC 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.

Requirement:A region size of 600KB is the minimum required for execution.

For restart processing, specify PARM='ULU,DFSUCF00,,,0001'.

Related Reading: For further information on specifying PARM options, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DD Statements

STEPLIB DD
Points to IMS.RESLIB, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.RESLIB, a DFSRESLB DD statement must be included.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules.

IMS DD
Defines the library containing the DBDs and PSBs that describe the databases and their processing blocks (that is, DSN=IMS.DBDLIB and IMS.PSBLIB).

DFSPRINT DD
Defines the output message data set. The data set can reside on a tape, direct-access device, 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 a direct-access device. It contains control records and checkpoint records for use in restart processing. This data set must always be specified. It is specified in the program with DCB parameters RECFM=VB, BLKSIZE=4008, and LRECL=4000. DISP=(,KEEP) must be specified by the user.

To use a multivolume data set, you must preform at all volumes except the first one. A volume sequence number of 1 must also be specified in the VOLUME parameter.

DFSOJRNL DD
Defines the old UCF journal data set created in a prior run that requires restart processing. This data set can reside on a tape or a direct-access device. It

must always be specified. When restarting, it is defined in the program with the following DCB values: RECFM=VB, LRECL=4000, and BLKSIZE=4008. When not restarting, it must be specified as DD DUMMY.

DFSNCDS DD
Defines the new control data set for this program. This data set can reside on a tape or a direct-access device. DISP=(,KEEP) must always 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 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
Can be omitted or specified as DD DUMMY. It defines the IMS 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.

Restriction: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 databases used are VSAM, ISAM, or OSAM data sets.

Related Reading: For additional information on control statement format and buffer pool structure, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*. 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 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.

SORTWKnn DD
Defines 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 database with logical relationships, initial loads, the Prefix Resolution utility, or the Prefix Update utility executions.

DFSURWF1 DD
Defines the data set used to resolve logical or secondary index relationships.

The data set is used as input to the Prefix Resolution utility. This data set can reside on a tape or a direct-access device. Specify DISP=KEEP since this data set might be involved in restart processing. You must specify RECFM=VB, LRECL=900, 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.)

DFSURWF2 DD
Defines the intermediate sort work data set. This data set can reside on a tape or a direct-access device. Its size is approximately the same as that of the data set defined by the DFSURWF1 DD statement. The DCB parameters specified in the program are RECFM=VB and LRECL=900. BLKSIZE must be specified on this DD statement.

DFSURWF3 DD
Defines the output work data set that contains all update data for segments involved in logical relationships for that particular execution. This data set is created by the Prefix Resolution utility and is used by the Prefix Update utility. It can reside on a tape or a direct-access device. The DCB parameters are defined in the program are RECFM=VB and LRECL=900. BLKSIZE must be specified on this DD statement. Specify DISP=KEEP for restart purposes if the UCF terminates while running the Prefix Update utility.

DFSURIDX DD
Defines an output work data set that is used if secondary indexes are present in the DBDs being processed. The requirements for this data set are the same as those described above for DFSURWF3. This DD statement is required only if secondary indexes are present. DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on the DFSURIDX DD statement.

The following DD statement is required if REQUEST=EXTRACT was specified for a secondary index reorganization unload.

DFSEXTDS DD
Used to create an unloaded version of those records extracted from a shared secondary index as specified in control statements. This optional DD statement is required only if E (REQUEST=EXTRACT) is specified on the FUNCTION=RU control statement. The DCB attributes are determined dynamically, depending on the type of output device and the VSAM LRECLs used. Standard labels must be used.

DD statements are also required to define all databases referenced by the functional utilities during this execution, and all output data files created during this execution (for example, Reorganization Unload and batch Image Copy). Input files used by the Reorganization Reload utility must also be refined. Table 12 summarizes the use of the FUNCTION keyword on the various DD statements.

Table 12. JCL DD Statements Summary Table

DD	FUNCTION= Keywords Used on Utility Control Statements																
Statements	UCF*	OP	DR	DU	DX	IL	IM	PR	PU	RR	RU	SN	SR	SU	SX	ZB	ZM
IMS	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
DFSYSIN	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
DFSOJRNL	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

Table 12. JCL DD Statements Summary Table (continued)

DD	FUNCTION= Keywords Used on Utility Control Statements																
Statements	**UCF***	**OP**	**DR**	**DU**	**DX**	**IL**	**IM**	**PR**	**PU**	**RR**	**RU**	**SN**	**SR**	**SU**	**SX**	**ZB**	**ZM**
DFSOCDS	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
DFSCNTRL	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	
SYSOUT					R			R									
SORTLIB					R			R									
SORTWKnn					R			R									
DFSURWF1			R		R	R		O				R					
DFSURWF2					R			R									
DFSURWF3					R			R	R								
DFSURIDX											R						
DFSEXTDS											E						
Data Set DDs			R	R	R	R	R		R	R	R	R	R	R	R	R	

Key:
- R—Required
- D—Specify as DD DUMMY when not restricting
- V—Required for VSAM organized files
- 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.

When a parameter can have several values to be enclosed in parentheses, the values are not positional and can appear in any order. For example:

```
    ,REQUEST=(SUMM,MSGALL)
    ,REQUEST=(MSGALL,SUMM,STATS)
```

Each utility control statement must contain the FUNCTION keyword. This keyword is defined as follows:

FUNCTION=xx

where xx is:

OP	Option statement
DR	HD Reorganization Reload utility
DU	HD Reorganization Unload utility

DX	Combined HD Reorganization Unload and Reload utilities
IL	User’s Initial Load Program
IM	Batch Database Image Copy utility
PR	Prefix Resolution utility
PU	Prefix Update utility
RR	Secondary Index Reload
RU	Secondary Index Unload
SN	Database Scan utility
SR	HISAM Reorganization Reload utility
SU	HISAM Reorganization Unload utility
SX	Combined HISAM Reorganization Unload and Reload utilities
ZB	Database Zaps
ZM	Module Zaps

The functions are requests for invocation of the functional utilities, the user’s initial load of a database, or both.

The FUNCTION=OP Statement

Because UCF takes the place of Prereorganization, the REQUEST= options from this statement is used by Prefix Resolution to control the report writing. Therefore, code the REQUEST keyword primarily for Prereorganization/Prefix Resolution, and secondarily for UCF defaults.

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

```
►►──FUNCTION=OP──┬─────────────────────────┬──────────────────────────►
                 │         ┌─NORM────┐     │
                 └─,COND=──┼─CDS─────┼─────┘
                           └─RESTART─┘

►──┬──────────────────────────────────────────────────────────┬───────►
   │            ┌─STATS────┐                                   │
   └─,REQUEST=(─┴─NOSTATS──┴───┬───────┬────┬─────────┬────)──┘
                               └─,SUMM─┘    └─,MSGALL─┘

►──┬──────────────────────┬──┬─────────────────────────┬──────────────►◄
   │          ┌─2000───┐  │  │              ┌─,──────┐  │
   └─,CKPNT=──┼────────┼──┘  └─,MSGNUM=(───▼──ccc───┴─)─┘
              ├─0──────┤
              └─n──────┘
```

COND=
Specifies the type of processing for this execution of the UCF. COND= can only be used once for each UCF run. 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 indicates 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. If REQUEST=NOSTATS is specified for the initial execution of UCF, then all subsequent restart steps must also specify REQUEST=NOSTATS. This parameter is also used by Prefix Resolution to control its output report.

REQUEST=SUMM
Specifies that a summary of the statistics is to be printed after the functional utility completes processing. This parameter is also used by Prefix Resolution to control its output report.

REQUEST=MSCALL
Specifies that all A and W type messages are to be set as abend requests. This parameter is used only as a diagnostic aid.

CKPNT=
Specifies the checkpoint interval to be used as the default specification during the UCF run. The value specified must be between 0 and 999999. All function control statements that do not have the CKPNT keyword default to this value. If CKPNT is not specified for FUNCTION=OP, 2000 is used as the default CKPNT value on all UCF control statements. CKPNT=0 means that checkpoints are not to be taken for this function. See the appropriate control statement for the default values.

MSGNUM=
Is used to set the error point abend flags. The value *ccc* is the message number extracted from the message identifier 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=DR Statement

This control statement causes the UCF to execute the HD Reorganization Reload utility (DFSURGL0) to reload an HDAM or HIDAM database from a data set created by the HD Reorganization Unload utility (DFSURGU0).

The HD Reorganization Reload utility step internally generates a Batch Database Image Copy utility REQUEST upon successful completion.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=DR control statement is:

```
►►──FUNCTION=DR,DBNAME=dbname──┬─────────────────────────────────────┬──────────────►
                               │         ┌─DFSUINPT──────────┐       │
                               └─,INDDS=─┴─reload file ddname─┴──────┘
```

```
►──┬───────────┬──┬──────────────────┬──┬─────────────────────────┬──────────────►
   └─,SEQ=nnn──┘  │        ┌─YES──┐  │  │               (1)       │
                  └─,EXEC=─┼─NO───┼──┘  │         ┌─2000───────┐  │
                           └─STOP─┘     └─,CKPNT=─┼────────────┼──┘
                                                  ├─0──────────┤
                                                  └─n──────────┘

►──┬──────────────────────────────┬──┬──────────────────────────┬─────────────────►
   └─,EXITRTN=exit module name────┘  └─,DBDDDS=(─ddname───)─────┘

►──┬────────────────────────────────┬────────────────────────────────────────────►◄
   │             ┌─DFSURWF1────┐    │
   └─,WF1DDS=────┴─ddname──────┴────┘
```

Notes:

1 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

INDDS=
Specifies a ddname for the input data set containing the data to be reloaded. This is the data set specified by the OUTDDS keyword of the FUNCTION=DU control statement (HD Reorganization Unload utility). The INDDS keyword must not specify a database name and can be specified only once for this control statement. If a ddname is not specified, a default of DFSUINPT is assumed.

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

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during the execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there

is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

WF1DDS=
Defines the output work data set that is supplied as an input to the Prefix Resolution utility. The data set can reside on a 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 or HIDAM database to a QSAM-formatted data set. When 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.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=DU control statement is:

The FUNCTION=DU Statement

```
►►──FUNCTION=DU,DBNAME=dbname,,OUTDDS=──┬──────────────────────────────────────────┬──────►
                                        │   ┌─DFSURGU1──┐   ┌─DFSURGU2──┐          │
                                        └─(─┴─ddname1───┴───┴─ddname2───┴───)──────┘

►─┬────────────┬──┬─────────────────────────────┬──┬─────────────────────┬────────────────►
  └─,SEQ=nnn───┘  │                (1)          │  │       ┌─YES──┐      │
                  │           ┌─STATS──────────┐│  └─,EXEC=─┼─NO───┼──────┘
                  └─,REQUEST=─┴─NOSTATS────────┴┘          └─STOP─┘

►─┬──────────────────────┬──┬───────────────────────────────┬────────────────────────────►
  │         ┌─2000─────┐ │  └─,EXITRTN=exit module name─────┘
  └─,CKPNT=─┤          ├─┘
            │   (2)    │
            ├─0────────┤
            └─n────────┘

►─┬──────────────────────────────┬───────────────────────────────────────────────────────►◄
  │            ┌─,──────────┐    │
  │            ▼            │    │
  └─,DBDDDS=(───ddname──────┴─)──┘
```

Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

OUTDDS= (DFSURGU1 ddname-1)|(DFSURGU2 ddname-2)
Specifies up to 2 ddnames that define the primary and secondary data sets for the HD Reorganization Unload utility. These data sets can reside on either tape or direct-access devices. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.) If multiple DU functions are included in one execution of UCF, each must specify a unique output data set. If ddname-1 is not specified, a default name of DFSURGU1 is assumed. If ddname-2 is not specified, a default name of DFSURGU2 is assumed. Temporary data sets cannot be used for Image Copy output data sets.

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

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

EXEC=STOP must be specified when reorganizing a VSAM database. This is further explained under "Restrictions" on page 277.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow the user to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

DBDDDS=
Allows you to verify that all database 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 database reorganization, as described for FUNCTION=DR and FUNCTION=DU.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=DX control statement is:

```
►►──FUNCTION=DX,DBNAME=dbname──┬──────────────────────────────────────────────────┬──────►
                               │                ┌─DFSURGU1─┐     ┌─DFSURGU2─┐     │
                               └─,OUTDDS=(──┴─ddname1──┴─────┴─ddname2──┴──)─┘

►──┬─────────────────────────────┬──┬─────────────┬──┬────────────────────┬─────────────►
   │          ┌─DFSUINPT─┐       │  └─,SEQ=nnn─┘    │        ┌─YES──┐      │
   └─,INDDS=──┴─ddname───┴───┘                     └─,EXEC=─┼─NO───┼──────┘
                                                             └─STOP─┘

►──┬───────────────────────────┬──┬──────────────────────────────────┬──────────────────►
   │              (1)          │  └─,EXITRTN=unload exit module name─┘
   │          ┌─2000──────┐    │
   └─,CKPNT=──┼───────────┼────┘
              ├─0─────────┤
              └─n─────────┘

►──┬──────────────────────────────────┬──┬──────────────────────────────┬───────────────►
   └─,EXITRLD=reload exit module name─┘  │            ┌─,────────┐      │
                                         └─,DBDDDS=(──▼─ddname───┴──)──┘

►──┬─────────────────────────────┬────────────────────────────────────────────────────►◄
   │           ┌─DFSURWF1─┐      │
   └─,WF1DDS=──┴─ddname───┴──────┘
```

Notes:

1 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

OUTDDS=
Specifies up to 2 ddnames that define the primary and secondary data sets for the HD Reorganization Unload utility. These data sets can reside on either tape or direct-access devices. (Specifying multiple output copies can be

advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.) If multiple DX functions are included in one execution of UCF, each must specify a unique output data set. If ddname-1 is not specified, a default name of DFSURGU1 is assumed. If ddname-2 is not specified, a default name of DFSURGU2 is assumed.

INDDS=
Specifies the input data set containing the data to be reloaded. This ddname must correspond to the ddname specified for the OUTDDS keyword during the unload portion of the database reorganization. The data set must reside on a tape or a direct-access device. If a ddname is not specified, a default name of DFSUINPT is assumed.

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

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

The EXEC=STOP parameter remains in effect for the entire control statement. A STOP is performed for both the unload and reload functions, requiring a restart to continue executing subsequent functions.

EXEC=STOP must be specified when reorganizing a VSAM database. This is further explained under “Restrictions” on page 277.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified for function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control at the unload phase of this reorganization to allow you to examine records or compile statistics. The unload-exit-module-name must reside in a library known to this function.

EXITRLD=
Specifies an exit routine to be given control at the reload phase of this reorganization to allow you to examine records or compile statistics. The reload-exit-module-name must reside in a library known to this function.

DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

WF1DDS=
Defines the output work data set that is supplied as an input to the Prefix Resolution utility. The data set can reside on a tape or a direct-access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

The FUNCTION=IL Statement

This control statement attaches a user-supplied Initial Load program. Restart capabilities are available for these programs under the UCF. In most cases, only minor changes are required to the programs to provide the proper interface to the UCF. For additional information, see “Initial Load Application Program Considerations” on page 280.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=IL control statement for your Initial Load program is:

```
►►──FUNCTION=IL,DBNAME=dbname,ILPGM=loadpgm,ILPSBNAM=psbname──┬───────────┬────────►
                                                              └─,SEQ=nnn──┘

►──┬─────────────────────┬──┬─────────────────────────────┬─────────────────────►
   │         ┌─YES──┐    │  │                (1)          │
   └─,EXEC=──┼─NO───┼────┘  │          ┌─2000──────────┐  │
             └─STOP─┘       └─,CKPNT=──┼───────────────┼──┘
                                       ├─0─────────────┤
                                       └─n─────────────┘

►──┬─────────────────────────────────┬──┬─────────────────────────────┬─────────►
   │              ┌─,─────────┐      │  │           ┌─DFSURWF1─┐      │
   │              ▼           │      │  └─,OUTDDS=──┤          ├──────┘
   └─,DSDDNAM=(───ddname──────┴──)───┘              └─ddname───┘

►──┬────────────────────────────────┬──┬───────────────────────────┬────────────►◄
   │             ┌─,─────────┐      │  └─,EXITRTN=exit module name─┘
   │             ▼           │      │
   └─,DBDDDS=(───ddname──────┴──)───┘
```

Notes:

1 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

ILPGM=
Defines the name of your initial load program to be attached. This module must reside in the LINKLIB or in a JOBLIB or STEPLIB data set or must reside in a LINKPACK area because it is located by `BLDL` and `ATTACH` commands.

ILPSBNAM=
Defines the name of the PSB that is to be used by your initial load program. This PSB must reside in the data set defined by the IMS DD statement. This keyword must be specified once on each complete IL control statement; it cannot be specified more than once for each control statement.

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

EXEC=
: Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
: Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of roots loaded in the database and must be between 1 and 999999.

  If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default for this function. CKPNT=0 means that checkpoints are not to be taken for this function.

DSDDNAM=
: Directs your 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 preparation begins to terminate the UCF.

OUTDDS=
: Defines the input work data set that is supplied as an input to the Prefix Resolution utility. The data set can reside on a tape or a direct-access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

DBDDDS=
: Verifies that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

EXITRTN=
: Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=IM Statement

This control statement causes the UCF to execute the batch Database Image Copy utility (DFSUDMP0) to create an output copy of the data sets within a database.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=IM control statement is:

```
►►──FUNCTION=IM,DBNAME=dbname──────────────────────────────────────────►
```

```
►─,OUTDDS=(ddname1─┬───────────┬─),DBDDDS=image copy input ddname──────────►
                   └─,ddname2──┘

►──┬──────────┬──┬─────────────────────────────────────────────┬──────────►
   └─,SEQ=nnn─┘  │                     (1)                     │
                 │                ┌─STATS────┐                 │
                 └─,REQUEST=(─────┼──────────┼──┬────┬─)───────┘
                                  └─NOSTATS──┘  └─,I─┘

►──┬───────────────────┬──┬─────────────────────────┬─────────────────────►
   │         ┌─YES──┐  │  │                (2)      │
   └─,EXEC=──┼─NO───┼──┘  │           ┌─2000─────┐  │
             └─STOP─┘     └─,CKPNT=───┼──────────┼──┘
                                      ├─0────────┤
                                      └─n────────┘

►──┬─────────────────────────────┬────────────────────────────────────────►◄
   └─,EXITRTN=exit module name───┘
```

Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

OUTDDS=
Specifies up to 2 copies for the Image Copy output data set. The ddname-1 defines the primary output data set; ddname-2 defines the secondary output data set. They can reside on either tape or direct-access devices. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to the normal end of job.) When multiple FUNCTION=IM statements are coded, the Image Copies are done in order based on the collating sequence of the OUTDDS ddname. If the OUTDDS data sets are allocated or a new generation data set group (GDG) is created, then the data sets must be put in collating ddname sequence in the JCL to avoid an open error.

DBDDDS=
Specifies the Image Copy input data set that is to be dumped. This data set must reside on a direct-access device and must be specified once per control statement. If REQUEST=I is specified, the data set must be a KSDS.

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

REQUEST=
STATS specifies that statistics are to be printed upon completion of this

functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

REQUEST=I specifies an image copy of an index of a KSDS. if specified, the DBDDDS keyword value must refer to a KSDS. The only recovery that can be used with this Image Copy is a track recovery.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow you 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 Database Prefix Resolution utility (DFSURG10). This utility accumulates the information generated on work data sets during the load, reorganization, or both, of one or more databases. It produces an output data set, DFSURWF3, that includes one or more updated records to be applied to each segment that contains logical relationship information.

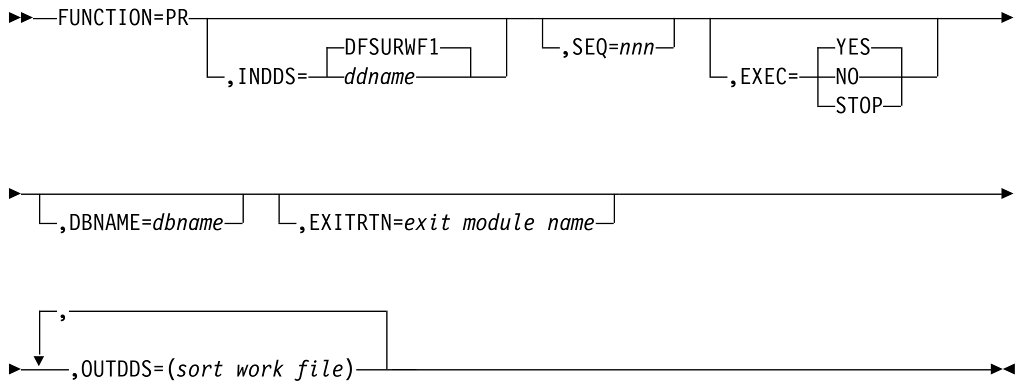
The content of this data set may be different when compared to the same data set generated by the Prefix Resolution utility run without UCF. This can occur if you have logically related databases and database records which are logical parents without logical children. In this case, some pointers will not require resolution and those records will be discarded. Since the non-UCF controlled utility run may require the coding of the DBIL control statement for the Prereorganization utility (DFSURPR0), the number and type of records discarded may differ between the two utility runs. The final database content is not affected by this difference in the utility runs.

The updated records have been sorted in physical-location order by database 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 or update secondary index databases.

The STATS and SUMM reports are controlled by the REQUEST= keyword on the FUNCTION=OP statement.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=PR control statement is:



INDDS=
: Specifies that the DFSURWF1 or user-defined data set generated by the Database Scan utility is to serve as one of the inputs to the Database Prefix Resolution utility.

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

EXEC
: Specifies whether this control statement is to be executed. If STOP is specified, UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

DBNAME=
: Specifies the database 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 database with this name. The function is performed for all databases on the work file whether the DBNAME parameter is coded or not.

EXITRTN=
: Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

OUTDDS=
: Allows you 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 Database Prefix Update utility (DFSURGP0). This utility uses the output generated by the Prefix Resolution utility to update the prefix of each segment whose prefix information was affected by a database load or reorganization.

This statement is optional, because it is automatically generated by the UCF for normal processing.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=PU control statement is:

```
►►──FUNCTION=PU──┬───────────┬──┬─────────────────────┬───────────────────────────►
                 └─,SEQ=nnn──┘  │          ┌─YES──┐   │
                                └─,EXEC=───┼─NO───┼───┘
                                           └─STOP─┘

►──┬─────────────────────────────┬──┬──────────────────────────────┬──────────────►
   │                 (1)         │  │             ┌─,────────┐     │
   │         ┌─2000──────────┐   │  │             ▼          │     │
   └─,CKPNT=─┼───────────────┼───┘  └─,DBNAME=(───dbname─────┴──)──┘
             ├─0─────────────┤
             └─n─────────────┘

►──┬─────────────────────────────┬────────────────────────────────────────────────►
   └─,EXITRTN=exit module name───┘

►──┬──────────────────────────────────────────────────┬───────────────────────────►
   │         ┌─DFSURWF3─────────────────────────────┐ │
   └─,INDDS=─┴─ddname from Prefix Resolution run────┴─┘

►──┬─────────────────────────────┬───────────────────────────────────────────────►◄
   │            ┌─,────────┐     │
   │            ▼          │     │
   └─,DBDDDS=(──ddname─────┴──)──┘
```

Notes:

1 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

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

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of input work records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

DBNAME=
Specifies the database 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 database with this name. This function is performed for all databases on the work file whether the DBNAME parameter is coded or not.

EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

INDDS=
Specifies 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 you specified).

DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart processing begins to terminate the UCF.

The FUNCTION=RR Statement

This control statement causes the UCF to execute the HISAM Reorganization Reload utility (DFSURRL0) to create, merge, or replace a member in a secondary index.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=RR control statement is:

```
►►──FUNCTION=RR,DBNAME=dbname──┬──────────────────┬──────────────────────────────►
                               └─,KDSDD=ddname─┘


►──┬─────────────────────────────────────────────────┬──┬───────────────────┬───►
   │          ┌─DFSUIN01────────────────────────────┐ │  └─,DBDDDS=ddname─┘
   │          │      ┌─,────────────────────┐       │ │
   │          │      ▼                      │       │ │
   └─,INDDS=──┴──(───reload─file─ddname─┴───)────┴─┘


►──┬───────────┬──┬────────────────────────────┬──┬──────────────────┬───────────►
   └─,SEQ=nnn─┘  │                  (1)        │  │         ┌─YES──┐ │
                 │           ┌─STATS────────┐  │  └─,EXEC=──┼─NO───┼─┘
                 └─,REQUEST=─┴─NOSTATS──────┴──┘            └─STOP─┘


►──┬─────────────────────────┬──┬─────────────────────────────┬──────────────────►◄
   │           (2)           │  └─,EXITRTN=exit module name─┘
   │         ┌─2000────────┐ │
   └─,CKPNT=─┼─────────────┼─┘
             ├─0───────────┤
             └─n───────────┘
```

Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

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 database with this name.

KDSDD=
Specifies a keyed data set ddname that is to be operated on. This keyword is used to verify the presence of the DD statement for the keyed data set.

INDDS=
Specifies the input data set containing the data to be reloaded. This ddname must correspond to the ddname specified for the OUTDDS keyword during the unload portion of the database reorganization. If this keyword is omitted, the default is DFSUIN01. The data set can reside on either a tape or a direct-access device.

IBDDS=
Defines the ddname of the overflow (ESDS) database data set required by this function. Only one DBDDDS can be specified for each control statement.

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

REQUEST=
STATS specifies statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CHKPNT=0 means that checkpoints are not to be taken for this function.

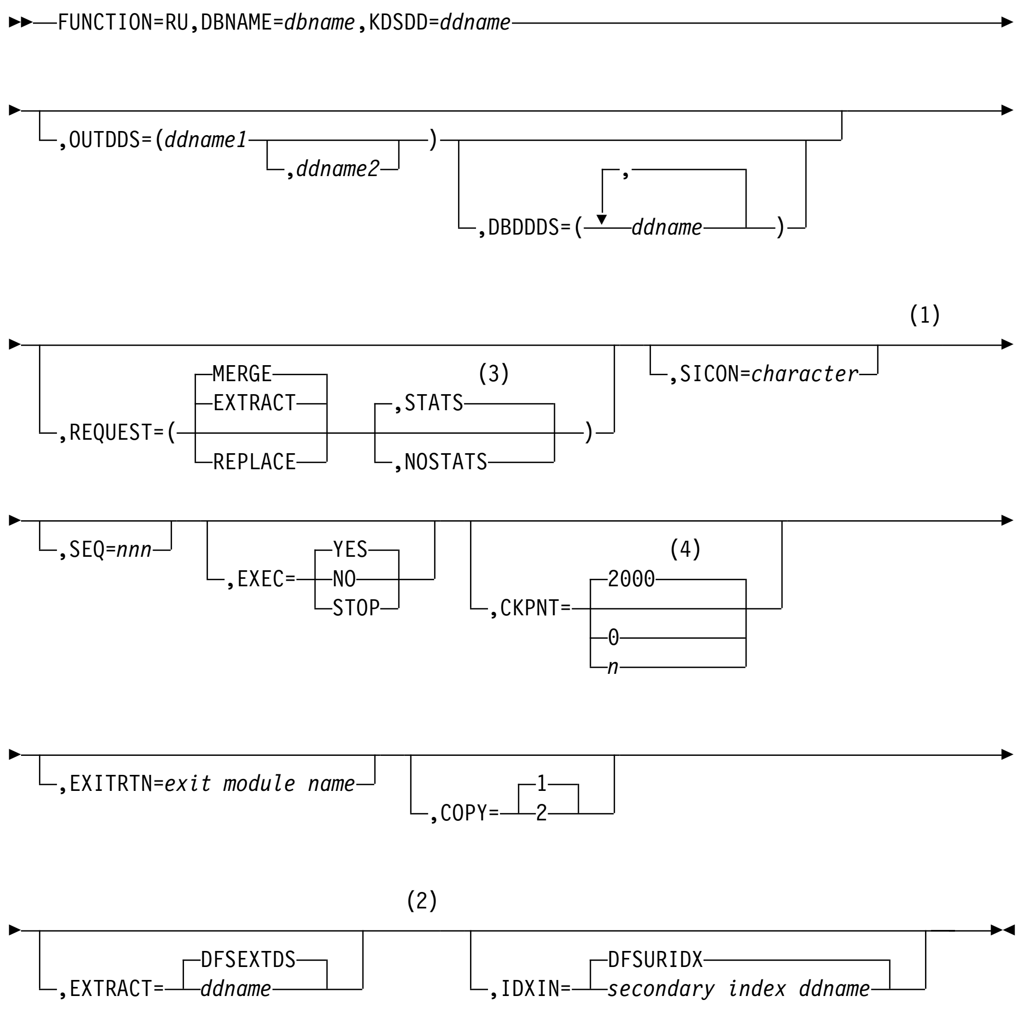
EXITRTN=
Specifies an exit routine to be given control in order to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=RU Statement

This control statement causes the UCF to execute the HISAM Reorganization Unload utility (DFSURUL0) to create an input work data set to the HISAM Reorganization Reload utility.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=RU control statement is:



Notes:

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

2 Can only be specified with REQUEST=EXTRACT.

3 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

4 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

KDSDD=
Defines the VSAM KSDS of the database 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 database.

OUTDDS=
Specifies up to two copies for the secondary index output data set. The ddname-1 defines the primary output data set. If ddname-2 is specified, the COPY keyword must have a value of 2. These copy data sets can reside on either tape or direct-access devices.

DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

REQUEST=EXTRACT or MERGE or REPLACE
EXTRACT extracts a secondary index (as defined by the SICON keyword) from either a shared index database or from the index work data set from Prefix Resolution. If REQUEST=EXTRACT is specified, the SICON and EXTRACT keywords are required.

MERGE merges the index work data set created during either database 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 replaces 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 must be performed prior to the replace. The SICON keyword is required for a replace operation.

REQUEST=STATS or NOSTATS or specified-on-option-statement
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

SICON=
Specifies a 1-byte constant defined in the DBD generation of the shared secondary index. This keyword is required if REQUEST=EXTRACT or REQUEST=REPLACE is defined on this control statement.

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

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that a checkpoint is not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

COPY=
Provides for multiple copies of output data sets. COPY can be specified only once per control statement. COPY=1 produces only 1 copy and is the default. COPY=2 produces an additional output copy. Specify COPY=2 if a ddname-2 is specified for the OUTDDS keyword. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.)

EXTRACT=
Defines the ddname of the EXTRACT database data set and must be specified if REQUEST=EXTRACT is used. DFSEXTDS is the default if no ddname is specified.

IDXIN=
Describes the output data set (DFSURIDX) from the Prefix Resolution utility that contains secondary index information. This keyword is required and can be specified only once for each control statement. This keyword corresponds to the index DD statement in the HISAM Reorganization Unload utility. DFSURIDX is the default if no ddname is specified.

The FUNCTION=SN Statement

This control statement causes the UCF to execute the Database Scan utility (DFSURGS0). This utility scans non-reorganized databases that contain logical relationships affected by loading or reorganizing other databases. It also generates output work data sets that is used by the Database 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.

Do not specify more than 20 ddnames for this function.

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

```
►►──FUNCTION=SN──┬──────────────────────────┬──────────────────────────────►◄
                 └─,DBNAME=dbname─┤ A ├─────┘
```

A:

```
    ┌─,──────────────────────────────┐
    ▼                                │
├───┬────────────────────────────────┬──┬──────────────────────────────┬───────►
    └─,DBDDDS=(database data sets)───┘  │             ┌─DFSURWF1─┐     │
                                        └─,OUTDDS=────┴─ddname───┴─────┘
```

```
►─┬──────────────────────────┬─┬──────────────────────────┬─┬──────────────┬──────────────►
  └─,SEGNAME=segment name────┘ │            ┌─SEG──┐      │ └─,SEQ=nnn─────┘
                               └─,SCANTYP=──┴─SEQ──┴──────┘

►─┬───────────────────────────────────────────────────┬─┬───────────────────┬─────────────►
  │           ┌─,STATS──────────────────────────────┐ │ │        ┌─YES──┐   │
  └─,REQUEST=─┼─────────────────────────────────────┼─┘ └─,EXEC=─┼─NO───┼───┘
              ├─,NOSTATS────────────────────────────┤            └─STOP─┘
              └─,specified──on──option──stmt────────┘

►─┬───────────────────────────────────────────────┬─┬────────────────────────────────┬───┤
  │         ┌─2000──────────────────────────────┐ │ └─,EXITRTN=exit module name──────┘
  └─,CKPNT=─┼───────────────────────────────────┼─┘
            ├─0─────────────────────────────────┤
            ├─1 9999999─────────────────────────┤
            └─specified──on──option──stmt───────┘
```

Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

DBDDDS=
Verifies that all data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames. If no DD statement exists for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

OUTDDS=
Defines the output data set that is used to resolve logical relationships. This data set is used as an input to the Prefix Resolution utility. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

SEGNAME=
Defines the segment name to be scanned for. This keyword is used in conjunction with a DBNAME= keyword in this same control statement. It can be specified only once for each complete control statement.

SCANTYPE=
Defines the order of scan to be performed on the DBNAME associated with this same control statement. This keyword can be specified only once for each complete control statement. The value SEQ means that the order of search is with unqualified GN calls from the beginning of the database. The value SEG means that the order of search is with GN calls qualified on the segment name from the beginning of the database.

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

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed.

If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of calls that equals the number of root segments read and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

The FUNCTION=SR Statement

This control statement causes the UCF to execute the HISAM Reorganization Reload utility (DFSURRL0). This utility can be used to reload a HISAM or HIDAM primary index database from a QSAM-formatted data set created by the HISAM Reorganization Unload utility.

Reorganization of HISAM databases 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 database. The HISAM Reorganization Unload/Reload utilities cannot be used to reorganize HISAM databases 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.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=SR control statement is:

```
►►──FUNCTION=SR,DBNAME=dbname──────────────────────────────────────────►
```

```
►──┬─────────────────────────────────────────────────┬──────────────────────────►
   │              ┌─DFSUIN01───────────────────────┐ │
   │              │    ┌─,──────────────────┐      │ │
   │              │    ▼                    │      │ │
   └─,INDDS=──────┴─(──────reload file ddname─┴──)──┴─┘

►──┬──────────────────────────────┬──┬───────────────┬──┬───────────┬─────────────►
   │          ┌─,──────────┐      │  └─,KDSDD=ddname─┘  └─,SEQ=nnn──┘
   │          ▼            │      │
   └─,DBDDDS=(────ddname───┴──)───┘

►──┬────────────────────────────────────────────────────┬──┬──────────────────┬───►
   │                (1)                                 │  │        ┌─YES──┐  │
   │            ┌─STATS────────┐                        │  └─,EXEC=─┼─NO───┼──┘
   └─,REQUEST=(─┼──────────────┼──┬───────┬──)──────────┘           └─STOP─┘
                └─NOSTATS──────┘  └─,VSAM─┘

►──┬───────────────────────────┬──┬──────────────────────────────┬─────────────►◄
   │              (2)          │  └─,EXITRTN=exit module name────┘
   │          ┌─2000────────┐  │
   └─,CKPNT=──┼─────────────┼──┘
              ├─0───────────┤
              └─n───────────┘
```

Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

INDDS=
Specifies the input data set containing the data to be reloaded. This is the data set created from the FUNCTION=SU control statement (HISAM Reorganization Unload utility). If this keyword is omitted, the default is DFSUIN01.

DBDDDS=
Use to verify the existence of a database data set that is to be reloaded.

KDSDD=
Defines the 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).

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

REQUEST=
STATS specifies that statistics are to be printed upon completion of this

functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

VSAM
Specifies that the 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 OSAM database is unloaded and the REQUEST=VSAM keyword is specified, the output is in VSAM format. The MVS 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 OSAM to VSAM.
- If a VSAM database is unloaded, the reloaded database is VSAM regardless of what is specified for the REQUEST= keyword statement. The original data set must be scratched and reallocated with the MVS Access Method Services utility, or a new data set name must be created by the Access Method Services utility before the HISAM Reload utility can be run.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of logical records and must be between 1 and 999999. If a CKPNT is not specified, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow you 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 database and creates a reorganized output that can be used as input to either the Database Recovery utility or the HISAM Reorganization Reload utility.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=SU control statement is:

```
►►──FUNCTION=SU,DBNAME=dbname,KDSDD=ddname──────────────────────────►
```

```
►─,OUTDDS=(ddname1─┬──────────┬─)─┬──────────────────────────────┬──────────►
                   └─,ddname2─┘   │           ┌─,──────────┐     │
                                  │           ▼            │     │
                                  └─,DBDDDS=(───ddname─────┴─)───┘

►─┬─────────┬─┬──────────────────────────────────┬─┬──────────────────┬─────►
  └─,SEQ=nnn┘ │                  (1)             │ │         ┌─YES──┐ │
              │            ┌─,STATS──────┐       │ └─,EXEC=──┼─NO───┼─┘
              └─,REQUEST=──┼─────────────┼───────┘           └─STOP─┘
                           └─,NOSTATS────┘

►─┬──────────────────────────┬─┬──────────────────────────────┬─────────────►
  │              (2)         │ └─,EXITRTN=exit module name─────┘
  │          ┌─2000──────┐   │
  └─,CKPNT=──┼───────────┼───┘
             ├─0─────────┤
             └─n─────────┘

►─┬────────────────┬────────────────────────────────────────────────────────►◄
  │        ┌─1─┐   │
  └─,COPY=─┴─2─┴───┘
```

Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

KDSDD=
Defines the VSAM KSDS of the database 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 database.

OUTDDS=
Specifies up to two copies for the unload data set. The ddname-1 defines the primary output data set; ddname-2 defines the secondary output data set. If ddname-2 is specified, the COPY keyword must have a value of 2. These copy data sets can reside on either tape or direct-access devices.

DBDDDS=
Use to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames. If no DD statement exists for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

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

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed.

If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

EXEC=
Specifies whether this control statement is to be executed. If EXEC=STOP is specified, the UCF terminates processing upon completion of this function. If EXEC=YES is specified, this function is to be processed. If EXEC=NO is specified, this control statement is not executed. EXEC=YES is the default.

EXEC=STOP must be specified when reorganizing a VSAM database. This is further explained under “Restrictions” on page 277.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. This value is equal to the number of root segments and must be between 1 and 999999. If a CKPNT is not specified for a function, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

COPY=
Provides for multiple copies of output data sets. COPY can be specified only once for each control statement. COPY=1 produces only one copy and is the default. COPY=2 produces an additional output copy. Specify it 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.)

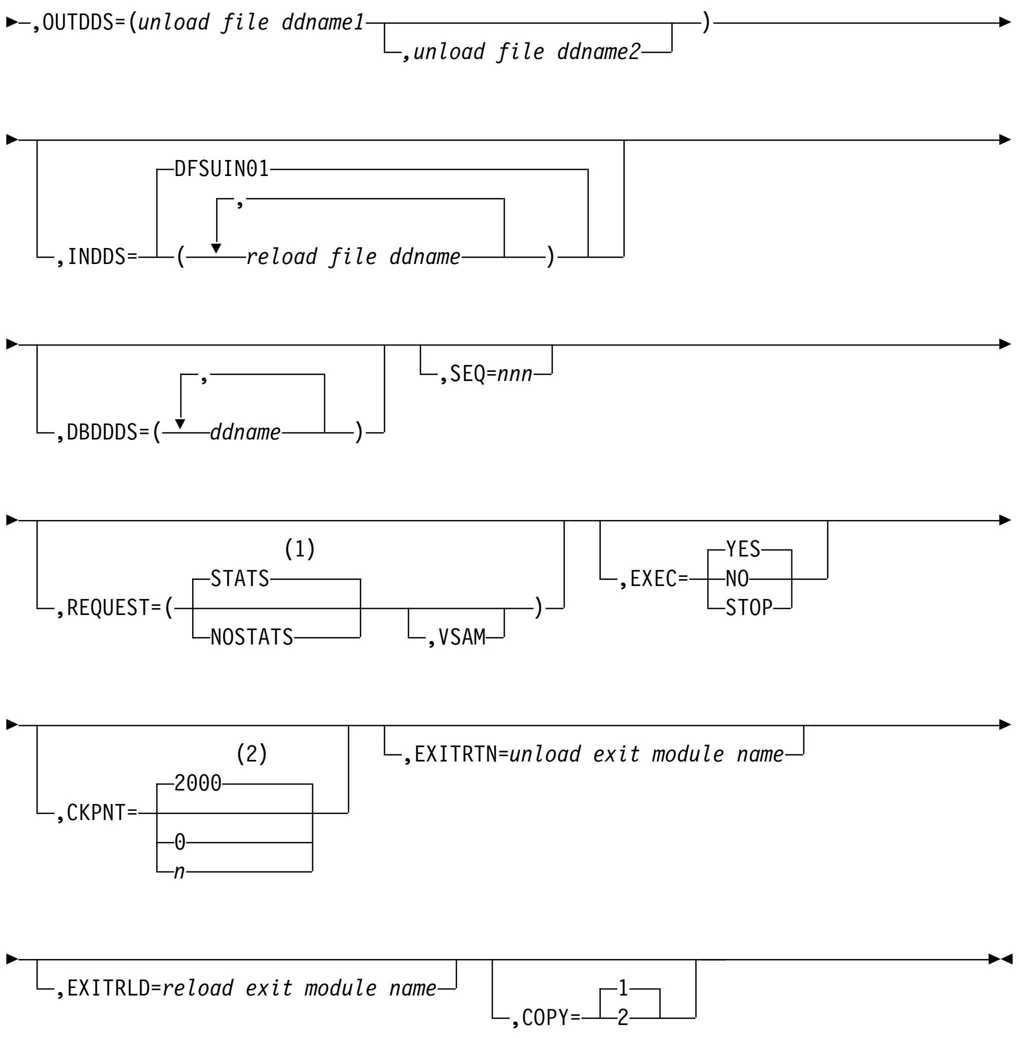
The FUNCTION=SX Statement

This control statement causes the UCF to execute the HISAM Reorganization Unload and Reload utilities to reorganize a HISAM database. The FUNCTION=SX combines the FUNCTION=SU and FUNCTION=SR specifications.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=SX control statement is:

```
►►──FUNCTION=SX,DBNAME=dbname,KDSDD=ddname──────────────────────────────►
```



Notes:

1 If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS is not specified on the FUNCTION=OP control statement, the default is STATS.

2 If a CKPNT is not specified for a function, the default is the CKPNT value specified on the FUNCTION=OP control statement. If CKPNT is not specified on the FUNCTION=OP control statement, the default is 2000.

DBNAME=
Specifies the database 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 database with this name.

KDSDD=
Defines the VSAM KSDS of the database 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 database.

OUTDDS=
Specifies up to two copies for the unload data set. The ddname-1 defines the primary output data set; ddname-2 defines the secondary output data set. If

ddname-2 is specified, the COPY keyword must have a value of 2. These copy data sets can reside on either tape or direct-access devices.

INDDS=
Specifies the input data set containing the data to be reloaded. This is the data set created from the FUNCTION=SU control statement (HISAM Reorganization Unload utility). If this keyword is omitted, the default is DFSUIN01.

DBDDDS=
Use to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

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

REQUEST=
STATS specifies that statistics are to be printed upon completion of this functional utility. NOSTATS specifies that statistics are not to be printed. If neither STATS nor NOSTATS is specified, the default is the value specified for the FUNCTION=OP control statement. If STATS or NOSTATS was not specified for the FUNCTION=OP control statement, STATS is the default.

VSAM
Specifies that the OSAM input reload copy is to be reloaded to a VSAM data set.

The following restrictions apply when using the REQUEST=VSAM keyword for a FUNCTION=SX control statement:

- If an OSAM database is unloaded and the REQUEST=VSAM keyword is specified, the output is in VSAM format. Run the MVS Access Method Services utility to create the required data sets, and make all of the necessary DBD changes before you run the HISAM Reload utility.
- REQUEST=VSAM can only be used when making a conversion from OSAM to VSAM.
- If a VSAM database is unloaded, the reloaded database is VSAM regardless of what is specified for the REQUEST= keyword statement. The original data set must be scratched and reallocated with the MVS Access Method Services utility, or a new DSNAME must be created by the Access Method Services utility before the HISAM Reload utility can be run.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

The EXEC=STOP parameter stays in effect for the entire control statement. A STOP is performed for both the unload and reload functions, requiring a restart to continue executing subsequent functions.

EXEC=STOP must be specified when reorganizing a VSAM database. This is further explained under “Restrictions” on page 277.

CKPNT=
Specifies a user-supplied checkpoint interval used during execution of this function. The value is equal to the number of root segments and logical records

and must be between 1 and 999999. If a CKPNT is not specified, the default is to the CKPNT specification on the FUNCTION=OP control statement. If CKPNT was not specified on a FUNCTION=OP control statement, 2000 is the default. CKPNT=0 means that checkpoints are not to be taken for this function.

EXITRTN=
Specifies an exit routine to be given control at the unload phase of this reorganization to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

EXITRLD=
Specifies an exit routine to be given control at the reload phase of this reorganization to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

COPY=
Provides for multiple copies of output data sets. COPY can be specified only once for each control statement. COPY=1 produces only one copy and is the default. COPY=2 produces an additional output copy. Specify it 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 database blocks to correct errors or force abends. Only the VERIFY and REP control statements of the MVS2 service aids SPZAP program are supported.

Restriction: This utility function should be used only by an IBM Field Engineering Program Support Representative or by someone under that person’s direction.

FUNCTION=ZB does not update the RECON when DBRC is active.

Recommendation: It is recommended that you take an image copy after using this function.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=ZB control statement is:

```
►►──FUNCTION=ZB,DBNAME=dbname,DBDDS=ddname──────────────────────────────────►


                                            (1)
►──┬─,VERIFY=nnnnnnnn,VALUE=compare value─┬──────,RBNID=dbblock──────────────►
   └─,REP=nnnnnnnn,VALUE=replace value────┘
```

```
                                                                               (2)
►──┬──────────────────────┬──┬──────────────────────────────────────────────┬──────────────►
   │          ┌─YES──┐    │  └─,RELATE=─┬─DU─┬──────────────────────┬──────┘
   └─,EXEC=───┼─NO───┼────┘             ├─DR─┤  └─,SEQ=nnn─┘
              └─STOP─┘                  ├─IM─┤
                                        ├─SN─┤
                                        ├─SR─┤
                                        └─SU─┘

►──┬──────────────────────────────┬─────────────────────────────────────────────────────►◄
   └─,EXITRTN=exit module name─┘
```

Notes:

1 For each complete control statement, you can specify either the VERIFY or REP keyword; they cannot be specified on the same statement. When specifying the VERIFY keyword on one complete statement and the REP keyword on a separate complete statement, the order of execution is critical; because statements are not automatically sorted by the UCF, make sure that the VERIFY precedes the REP.

The VERIFY and REP control statements must be included in the same job step. If the REP statement comes in a later step than the VERIFY statement, the REP statement is no longer associated with the VERIFY statement.

2 Use of the RELATE keyword without the SEQ keyword causes the UCF to performs zaps on all of the databases 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.

DBNAME=
Specifies the database 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 database with this name.

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

VERIFY=
Defines the hexadecimal offset (from the address of the RBA in the RBNID) that the associated VALUE= field is to be compared against. If any *verify* fails, the entire zap terminates. VERIFY must be coded as 8 hexadecimal digits.

REP=
Defines the hexadecimal offset (relative to zero) that is to be replaced by the associated VALUE= data. REP must be coded as 8 hexadecimal digits.

VALUE=
Defines the data (in hexadecimal representation; for example, C“A” is C1 in the statement) that is to be operated on as a compare field if associated with a VERIFY keyword or as replacement data if associated with a REP keyword. Exactly 8 hexadecimal characters must be specified. This means that 4 bytes of data must be changed or compared for each control statement. This keyword must be specified once for each control statement.

RBNID=
Defines the database block that is to be operated on by the associated VERIFY= or REP= data in hexadecimal representation (for example, RBNID= 00001000). For OSAM-HISAM data sets, this is a relative record number (RRN). For OSAM non-HISAM data sets, this is a relative block number (RBN). For VSAM data sets, this is a relative byte address (RBA) of the first byte in the block. When specified, this keyword must be specified as exactly 8 hexadecimal characters. This value is packed into a 4-byte field.

EXEC=
Specifies whether this control statement is to be executed. If STOP is specified, the UCF terminates processing upon completion of this function. If YES is specified, this function is to be processed. If NO is specified, this control statement is not executed. EXEC=YES is the default.

RELATE=
Relates one functional control statement to another. The value *xx* is defined on the related functional control statement and is described under the FUNCTION keyword description. The use of the SEQ keyword defines the related control statement. If it is 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
IM	for DFSUDMP0
SU	for DFSURUL0
SR	for DFSURRL0

Sequence=nnn links this zap function to another UCF functional control statement with an identical sequence number. This keyword is not valid if the RELATE keyword is not specified.

EXITRTN=
Use 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 MVS2 service aids SPZAP program are supported.

Restriction: This utility function should be used only by an IBM Field Engineering Program Support Representative or by someone under skilled direction.

The format of the FUNCTION=ZM control statement is:

```
►►──FUNCTION=ZM──,MODULE=──┬─DU──────────────┬───────────────────────────────────►
                           ├─DR──────────────┤
                           ├─IM──────────────┤
                           ├─PR──────────────┤
                           ├─PU──────────────┤
                           ├─SN──────────────┤
                           ├─SU──────────────┤
                           ├─SR──────────────┤
                           ├─RU──────────────┤
                           ├─RR──────────────┤
                           └─CF,CSECT=csectname─┘

►──┬─,VERIFY=nnnnnnnn,VALUE=compare value─┬──────────────────────────────────────►
   └─,REP=nnnnnnnn,VALUE=replace value────┘

►──┬───────────────────────────────────────┬──┬──────────────────────────────┬──►◄
   └─,RELATE=──┬─DU─┬──┬──────────────┬────┘  └─,EXITRTN=exit module name─┘
               ├─DR─┤  └─,SEQ=nnn─────┘
               ├─IM─┤
               ├─PR─┤
               ├─PU─┤
               ├─SN─┤
               ├─SU─┤
               ├─SR─┤
               ├─CF─┤
               ├─RU─┤
               └─RR─┘
```

Notes:

1 Use of the RELATE keyword without the SEQ keyword causes the UCF to performs zaps on all of the databases 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.

2 For each complete control statement, you can specify either the VERIFY or REP keyword; they cannot be specified on the same statement When specifying the VERIFY keyword on one complete statement and the REP keyword on a separate complete statement, the order of execution is critical; because statements are not automatically sorted by the UCF, make sure that the VERIFY precedes the REP.

The VERIFY and REP control statements must be included in the same job step. If the REP statement comes in a later step than the VERIFY statement, the REP statement is no longer associated with the VERIFY statement.

MODULE=
Defines the load module to be zapped. This keyword can be specified only once for each control statement and must specify one of the following modules:

DU	for DFSURGU0
SN	for DFSURGS0
DR	for DFSURGL0
SU	for DFSURUL0

IM	for DFSUDMP0
SR	for DFSURRL0
PR	for DFSURG10
CF	for DFSUCF00
PU	for DFSURGP0
RU	for DFSURUL0
RR	for DFSURRL0

CSECT=
Defines the CSECT within the load module that is to be zapped. If this keyword is omitted, the load module is changed in the CSECT containing the entry point and is relative to address zero as the entry point offset. (Refer to the IMS SYSGEN listings for the valid CSECT names of the particular load module.) This keyword is valid only with MODULE=CF and is ignored in all other instances.

VERIFY=
Defines the hexadecimal offset (from the address of the RBA in the RBNID) that the associated VALUE= field is to be compared with. If any verify fails, the entire zap terminates. VERIFY must be coded as 8-hexadecimal digits.

REP=
Defines the hexadecimal offset (relative to zero) that is to be replaced by the associated VALUE= data. REP must be coded as 8-hexadecimal digits.

VALUE=
Defines the data (in hexadecimal representation; for example, C“A” is C1 in the statement) that is to be operated on as a compare field if associated with a VERIFY keyword or as replacement data if associated with a REP keyword. Exactly 8 hexadecimal digits must be specified. This means that 4 bytes of data must be changed or compared for each control statement. The VALUE keyword must be specified once for each control statement.

RELATE=
Relates this zap statement to another UCF control statement. This keyword can be specified only on a zap statement.

The value *xx* is defined on the related control statement and is described under the FUNCTION keyword description. The use of the SEQ keyword defines the related functional control statement. If it is 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:

DU	for DFSURGU0
SN	for DFSURGS0
DR	for DFSURGL0
SU	for DFSURUL0
IM	for DFSUDMP0
SR	for DFSURRL0
PR	for DFSURG10

CF for DFSUCF00

PU for DFSURGP0

RU for DFSURUL0

RR for DFSURRL0

SEQ=
Links this zap function to another UCF functional control statement with an identical sequence number. This keyword is not valid if the RELATE keyword is not specified.

EXITRTN=
Use to obtain control at the time the zap has been executed.

Keywords Summary Tables

\

Table 13 summarizes the use of the keywords with the different functional utilities. The “Times Used” 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.

Table 13. UCF FUNCTION Summary Table

	Times	Function															
Keyword	**Used**	**OP**	**DR**	**DU**	**DX**	**IL**	**IM**	**PR**	**PU**	**RR**	**RU**	**SN**	**SR**	**SU**	**SX**	**ZB**	**ZM**
FUNCTION	N	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
COND	1	1															
REQUEST	N	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	
CKPNT	N	1	1	1	1	1	1		1	1	1	1	1	1	1		
IDLEN	1																
RECID	N																
DBNAME	N		1	1	1	1	1	1	1	1	1	1	1	1	1	1	
RELATE	N															1	1
PURGDT	N																
KDSDD	N									1	1		1	1	1		
CUMOUT	1																
CUMIN	N																
ILPGM	N					1											
DSDDNAM	N					D											
OUTDDS	N			2	2	1	2	D			2	D		2	2		
INDDS	N		1		1			1	1	1			D		D		
LOGIN	N																
LOGOUT	1																
EXITRTN	N		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SEGNAME	N											1					
SCANTYP	N											1					
DBDDDS	N		D	D	D	D	1		D	1	1	D	1	1	1	1	

Table 13. UCF FUNCTION Summary Table (continued)

Keyword	Times Used	Function															
		OP	DR	DU	DX	IL	IM	PR	PU	RR	RU	SN	SR	SU	SX	ZB	ZM
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																
RBNID	N															1	
EXITRLD	N				1										1		
WF1DDS	N		1		1												

Key:

blank	A blank space means this keyword cannot be used in this function.
N	This keyword can be used any number of times.
D	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 function request.
E	This keyword can be used only *one* time and only if no other keyword with an E designation in this chart has *not* also been specified on the same control statement. (It is not permissible to specify the VERIFY and REP keywords on the same control statement).
1,2	This keyword can be used that number of times as a maximum.

Table 14 shows the minimum keywords that you must specify when constructing each type of control statement.

Table 14. UCF-Required Keywords by Function

Keyword	Function															
	OP	DR	DU	DX	IL	IM	PR	PU	RR	RU	SN	SR	SU	SX	ZB	ZM
FUNCTION	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	
KDSDD										R			R	R		
ILPGM					R									R		
OUTDDS						R				R			R			
INDDS														R		
LOGIN																
SEGNAME											R					
DBDDDS						R									R	
ILPSENAM					R											
MODULE																R
VERIFY															R	R
REP															R	R

Table 14. UCF-Required Keywords by Function (continued)

	Function															
Keyword	**OP**	**DR**	**DU**	**DX**	**IL**	**IM**	**PR**	**PU**	**RR**	**RU**	**SN**	**SR**	**SU**	**SX**	**ZB**	**ZM**
VALUE															R	R
RBNID															R	

Key:

R Required

Return Codes

Upon termination of the UCF, the following return codes are passed to register 15.

Code	Meaning	Action
0	Processing was normal.	DFSNJRNL and DFSNCDS data sets can be scratched.
4	Warning messages have been issued, or termination was caused because EXEC=STOP was specified.	Verify that warnings are not errors; if no errors exist take same action as for return code 0. If restart is desired because of errors or EXEC=STOP, the DFSNJRNL and DFSNCDS must have been saved.
8	Serious errors have occurred. UCF terminated.	Correct errors and begin restart processing. The DFSNJRNL and DFSNCDS data sets must be available as DFSOJRNL and DFSOCDS on restart.
12	UCF failed and terminated with an unrecoverable error.	Correct the error and start the UCF from the beginning. Do not attempt restart processing. The DFSNJRNL and DFSNCDS data sets can be scratched because restart is not possible.

If the UCF terminates with a nonzero return code, all data sets that were in use at the time of termination might be required for restart. The following data sets must be saved under certain conditions:

DFSURWF1 Created during Initial Load, HD Reorganization Reload, and Database Scan. These data sets are required if the UCF terminates while running one of these functions.

DFSURWF3 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

The following examples show the use of the Utility Control Facility.

Example 1

The following example shows the control statement input data set and the minimum JCL required to execute the UCF.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUCF00'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSPRINT  DD SYSOUT=A,DCB=(BLKSIZE=605,LRECL=121,
//             RECFM=FBA)
//IMS       DD DSNAME=IMS.PSBLIB,DISP=SHR
//          DD DSNAME=IMS.DBDLIB,DISP=SHR
//DFSNJRNL  DD DSNAME=NJRNL,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2)),DCB=BLKSIZE=1600
//             DCB=(RECFM=VB,BLKSIZE=1008,LRECL=300)
//DFSOJRNL  DD DUMMY
//DFSNCDS   DD DSNAME=NCDS,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2))
//DFSOCDS   DD DUMMY
//DFSRDER   DD DUMMY
//DFSYSIN   DD *
      UCF CONTROL STATEMENT INPUT
/*
//DFSCNTRL  DD DSNAME=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 statement. The DFSYSIN DD statement is specified as DD DUMMY in this run.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUCF00,,,0001'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSPRINT  DD SYSOUT=A
//IMS       DD DSNAME=IMS.PSBLIB,DISP=SHR
//          DD DSNAME=IMS.DBDLIB,DISP=SHR
//DFSNJRNL  DD DSNAME=NJRNL2,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2))
//DFSOJRNL  DD DSNAME=NJRNL,DISP=(OLD,KEEP)
//DFSNCDS   DD DSNAME=NCDS2,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2)),DCB=BLKSIZE=1600
//DFSOCDS   DD DSNAME=NCDS,DISP=(OLD,KEEP)
//DFSRDER   DD DUMMY
//DFSYSIN   DD DUMMY,DCB=BLKSIZE=80
//DFSCNTRL  DD DSNAME=CNTRL,UNIT=SYSDA,SPACE=(CYL,(2,2)),
//             DCB=BLKSIZE=80
//*
//*OTHER JCL AS REQUIRED TO
//*IDENTIFY DATA SETS REQUIRED
//*FOR THE ACCESS METHOD BEING RESTARTED.
```

Example 3

The following example shows the JCL used to execute the UCF in a restart situation through use of a control statement.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUCF00'
//STEPLIB   DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.RESLIB,DISP=SHR
//DFSPRINT  DD SYSOUT=A
//IMS       DD DSNAME=IMS.PSBLIB,DISP=SHR
//          DD DSNAME=IMS.DBDLIB,DISP=SHR
//DFSNJRNL  DD DSNAME=NJRNL1,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2))
//DFSOJRNL  DD DSNAME=NJRNL,DISP=(OLD,KEEP)
//DFSNCDS   DD DSNAME=NCDS1,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2)),DCB=BLKSIZE=1600
```

```
//DFSOCDS   DD DSNAME=NCDS,DISP=(OLD,KEEP)
//DFSRDER   DD DUMMY
//DFSYSIN   DD *
      FUNCTION=OP,COND=RESTART
/*
//DFSCNTRL  DD DSNAME=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.
```

Part 7. Appendixes

Appendix A. Summary of DEDB Utility Commands

Table 15 is a summary of the DEDB utility commands, operands, and their synonyms.

Use these commands with the DEDB utilities that run online. In the figure, O=optional, R=required, and N/A=not applicable. More detailed information on commands follows the figure.

Table 15. Summary of DEDB Utility Commands

COMMAND	OPERATOR	SCAN UTILITY	DELETE UTILITY	REORG UTILITY	CREATE UTILITY	COMPARE UTILITY
ALLFMNEW		0	0	N/A	N/A	N
AREA		R	R	R	R	R
BUFNO		N/A	O	O	O	O
DDNAME		N/A	N/A	N/A	R	O
ERRORACTION		O	O	O	N/A	N/A
	STOP	O	O	O	N/A	N/A
	SCAN	O	O	O	N/A	N/A
	SCANRUN	O	O	O	N/A	N/A
EXIT		O	N/A	N/A	N/A	N/A
GO		O	O	O	O	O
INDOUBT		O	N/A	N/A	N/A	N/A
NOSORT		O	N/A	N	N	N
NSQCI		O	O	N	N	N
QUITCI		O	O	N	N	N
SORTSETUP		O	N/A	N	N	N
STARTIME		O	N/A	N	N	N
STARTRBA		O	N/A	N/A	N/A	N/A
STARTROOT		O	N/A	N/A	N/A	N/A
STARTSEQ		O	N/A	N/A	N/A	N/A
	FIELD	O	N/A	N/A	N/A	N/A
	OP	O	N/A	N/A	N/A	N/A
	VALUE	O	N/A	N/A	N/A	N/A
STARTUOW		N/A	N/A	O	N/A	N/A
STOPRBA		O	O	N/A	N/A	N/A
	EXCLUDE	O	O	N/A	N/A	N/A
STOPROOT		O	O	N/A	N/A	N/A
STOPSEQ		O	O	N/A	N/A	N/A
	FIELD	O	O	N/A	N/A	N/A
	OP	O	O	N/A	N/A	N/A
	VALUE	O	O	N/A	N/A	N/A
STOPTIME		O	O	N/A	N/A	N/A
STOPUOW		N/A	N/A	O	N/A	N/A

Table 15. Summary of DEDB Utility Commands (continued)

COMMAND	OPERATOR	SCAN UTILITY	DELETE UTILITY	REORG UTILITY	CREATE UTILITY	COMPARE UTILITY
TYPE		R	R	R	R	R
	COMPARE	N/A	N/A	N/A	N/A	R
	CREATE	N/A	N/A	N/A	R	N/A
	DLET	N/A	R	N/A	N/A	N/A
	REORG	N/A	N/A	R	N/A	N/A
	SCAN	R	N/A	N/A	N/A	N/A

Command Format

Specify parameters in free-form.

Exception: Except for the 120-character maximum, fields are not restricted to any particular columns.

You must begin each statement on a new line. Begin the command name with the first nonblank character and end it with a blank or an equal sign. An asterisk as the first character indicates a comment.

If the command requires operands, the operand field begins with the next character that is either nonblank or not an equal sign. If the command allows multiple operands, the operands are separated by commas. The operand field is ended by a blank or by the end of the line. However, characters that are part of an EBCDIC value (specified as a character string within quotation marks) do not count as ending commas or as ending blanks.

Characters following the operand field on a line are treated as comments.

Command Continuation

You can continue the operand field between operands by using a dangling comma as follows:

```
STARTSEQ  OP='FIELD=FLD1',     (first line)
          VALUE=X'C4C5C2'      (continuation line)
```

Continue a quoted string by using a closing quotation mark, a comma, and a reopening quotation mark as follows:

```
STARTROOT X'C1C2C3C4C',        (first line)
          '5C6C7',             (second line)
          'C8C9'               (last line)
```

Comments can be included on each line, with a blank between the dangling comma and the comments.

Command Descriptions

The input parameters to the DEDB online utilities are supplied by the following commands.

ALLFMNEW
Indicates that all CIs in the area use formats from Version 6 and later versions. If IMS detects any IMS/ESA Version 5 or earlier format CIs, the utility abends. This parameter can be used to identify the existence of any old format CIs.

Note: If SORTSETUP is used, ALLFMNEW should be specified only when it is certain that there are no old-format segments left in the SDEP section

AREA

Identifies the area to be processed by the name that is on the control region DD statement. This command must be repeated after each GO or RUN command.

The name must be 1 to 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

BUFNO

Specifies the number of buffers to use to read and/or write the DEDB.

For the utilities other than the Reorganization utility, a minimum of 7 buffers is required. If BUFNO is not specified, the default is taken. The default number of buffers is the number of control intervals (CIs) per unit of work (UOW) plus 7.

For the High-Speed DEDB Direct Reorganization utility, a minimum number of buffers is calculated as follows:

```
Minimum number of buffers = number of
  hierarchic levels of DEDB + 2 + 10
```

If BUFNO is not specified, the default is taken. The default number of buffers is the number of CIs per UOW plus the number of hierarchic levels of the DEDB + 2 + 10.

See “How the Reorganization Utility Uses the BUFNO Command” on page 137 for information on how BUFNO is used for the High-Speed DEDB Direct Reorganization utility.

DDNAME

Specifies the area data set to be created or compared.

For the create utility, the area data set specified on a DDNAME statement must be in an unavailable status in the DBRC RECON data set. The maximum allowable number of the DDNAME statements is 6.

For the compare utility, the area data set specified on a DDNAME statement must be in an available status. The maximum allowable number of the DDNAME statements is 7.

ERRORACTION

Specifies the action for the specified utility to take after detecting an error and printing an error message. This command is optional and can be specified as many times as desired. If ERRORACTION is not included, the STOP operand is the default.

STOP

Specifies that the utility stop immediately.

ABEND

Specifies that the utility produce a U1039 abend dump.

SCAN

Specifies that the utility continue scanning the input for errors.

SCANRUN

Specifies the same processing as the SCAN operand, except that the utility ignores a detected error after the next GO command is encountered.

EXIT
Identifies the user exit routine by load module name. This command is optional and is used only with the scan utility.

The name must be 1 to 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

GO
Is used to separate a series of requests. It must be specified to process more than one area, to use more than one exit routine, or to process more than one range of data within a single job step.

INDOUBT
Specifies that RBAs of in-doubt segments are to be written to to the SYSPRINT output data set.

NOSORT
Specifies that the sequential dependent segments are not sorted. These segments are passed directly to the user segment. For areas that currently have a SHARELVL of 0 or 1, SORT is not invoked. If the area has previously been SHARELVL 2 or 3, there may be islands of SDEPs that will be returned out of sequence. NOSORT can be abbreviated with NS or NOS.

NSQCI
Specifies that IMS partners will give up their current SDEP CI and any preallocated CI RBAs during the next commit interval. This option is suitable only if all sharing IMS systems are inserting at a similar rate.

QUITCI
Specifies that all IMS partners will give up their current SDEP CI and any preallocated CI RBAs immediately. This option is useful if there is an IMS partner that processes a low volume of transactions and therefore takes an extended period of time to fill its preallocated CIs. QUITCI forces all in-use but only partially-filled CIs to be written to the ADS and then released, which in turn enables the delete utility to process the entire CI and advance the logical beginning (DMACXVAL) as far as possible, ensuring that sufficient space is available when the SDEPs cycle around. Using the QUITCI parameter guarantees that no new SDEPs will be inserted in the QUITCIs. The next utility run can therefore start from the CI following the last UHWM, assuming that the default STOP is used.

SORTSETUP
Identifies the name of a user-written routine to be called before SORT is invoked, allowing different SORT parameters to be passed instead of the segment time stamp. The DBFUMSC0 source code contains the default sort setup, as well as the sort input and output exits.

STARTIME
Specifies a specific start time for selecting valid SDEP segments. Used for SCAN.

STARTUOW
Specifies the first unit of work (UOW) to reorganize. This command is optional and is used with the reorganization utility.

STARTRBA
Specifies up to 8 bytes (16 digits) of sequential dependent address information. The low-order 4 bytes specify the relative byte address within the area to start

processing sequential dependents. The high-order 4 bytes are optional and are used to supply a cycle number. `STARTRBA` is an optional command and is used with the scan utility.

A hexadecimal value is a value of the form X'hex digits'.

STARTROOT
Specifies the root key field value for the root used to find the start RBA. The `STARTROOT` command is optional and is used with the scan utility.

The value must be a hexadecimal value, a character value, or a packed value. The value begins with a letter (X, C, or P) and continues with a quoted string, such as X'2A1B'', C'AIN''T', or P'-00199'. Values are evaluated for both length and content. For example, X'00' (1 byte) is not the same as X'0000' (2 bytes).

STARTSEQ
Specifies the sequential dependent segment used to find the start RBA. The `STARTSEQ` command is optional and is used with the scan utility.

FIELD=(name2)
Is a field name as is specified in an SSA.

The name must be 1 to 8 characters. The characters must be alphabetic or numeric. The alphabetic characters include @, #, and $.

OP=(operator)
Is a comparison operator as is specified in an SSA.

VALUE=(value)
Is a field value as is specified in an SSA.

The FIELD, OP, and VALUE fields are used, together with the STARTROOT data, to set up a POS call to find the start RBA.

STOPRBA
Specifies the stop RBA. The rules are the same as described for specifying the starting RBA using the `STARTRBA` command. This optional command is used with either the scan or delete utilities.

The hexadecimal value is a value of the form X'hex digits'. The address specified for the STOPRBA command must fall on an SDEP boundary; otherwise, the utility will abend.

EXCLUDE
Specifies that the DEDB scan utility extracts SDEPS up to, but excluding the SDEP segment at STOPRBA, while the DEDB delete utility sets field DMACLBTS to STOPRBA rather than to STOPRBA+1.

STOPROOT (value)
Specifies the root key field value. This optional command is used with either the scan or delete utilities.

The value must be a hexadecimal value, character value, or a packed value. The value begins with a letter (X, C, or P) and continues with a quoted string, such as X'2A1B', C'AIN''T', or P'-00199'. Values are evaluated for both length and content.

STOPSEQ
Specifies the sequential dependent segment used to find the stop RBA. This optional command is used with either the scan or delete utilities.

FIELD=(name2)
Is a field name as is specified in a segment search argument (SSA).

The name must be 1 to 8 characters. The characters must be alphabetic or numeric. The alphabetic characters include @, #, and $.

OP=(operator)
Is a comparison operator as is specified in an SSA.

VALUE=(value)
Is a field value as is specified in an SSA.

The FIELD, OP, and VALUE fields are used, together with the STOPROOT data, to find the STOP RBA in the same way STARTSEQ data is used to find the START RBA.

STOPTIME
Specifes a stop time to use for selecting valid SDEP segments. Used in DELETE and SCAN.

STOPUOW
Specifies the last unit of work (UOW) to reorganize. This command is optional and is used with the reorganization utility.

TYPE
Specifies either the scan, delete, or reorganization utility as the type of run.

Requirement: This command is required and must be included before the first `GO` command or before the end of the SYSIN file.

Specify the `TYPE` command only once within a job step because the program cannot switch from one utility to another within the same job step.

The DEDB online utilities recognize alternative spellings that can be used as abbreviations or synonyms for the commands and keywords. Table 16 and Table 17 on page 335 show these abbreviations and synonyms.

Table 16. DEDB Online Utility Command Abbreviations and Synonyms

Commands	Abbreviations/Synonyms
ALLFMNEW	AFN
AREA	A, AREANAME
BUFNO	BUFFERNUMBER, BUFFNO, BUFN
DDNAME	DD, DDN
ERRORACTION	ERR, ERROR
EXIT	USEREXIT
GO	
NOSORT	NOS,NS
QUITCI	QCI
SORTSETUP	SSE
STARTIME	
STARTRBA	F, FA, FIRST, FIRSTA, FIRSTRBA, FRBA, STARTA
STARTROOT	FIRSTR, FIRSTROOT, FR, FROOT, STARTR
STARTSEQ	FIRSTS, FIRSTSEQ, FS, FSEQ, STARTS

Table 16. DEDB Online Utility Command Abbreviations and Synonyms (continued)

Commands	Abbreviations/Synonyms
STOPTIME	
STARTUOW	FIRSTUOW, FIRSTW, FUOW, FW, STARTW
STOPRBA	K, KEEP, KEEPA, KEEPRBA, KRBA, L, LAST, LASTA, LASTRBA, LRBA, STOPA
STOPROOT	KEEPR, KEEPROOT, KR, KROOT, LASTR, LASTROOT, LR, LROOT, STOPR
STOPSEQ	KEEPS, KEEPSEQ, KS, KSEQ, LASTS, LASTSEQ, LS, LSEQ, STOPS
STOPUOW	LASTUOW, LASTW, LUOW, LW, STOPW
TYPE	T

Table 17. DEDB Online Utility Keyword Abbreviations and Synonyms

Keywords	Abbreviations/Synonyms
COMPARE	CM, COM, COMP
CREATE	C, CR
DLET	D, DELETE, DL
EXCLUDE	E, EX
FIELD	F
NO	N, OFF
OP	O, OPERATOR
REORG	DR, HSR, HSREORG, R
SCAN	SKIP (as ERRORACTION operand)
SCAN	S, SC (as TYPE operand)
SCANRUN	SKIPRUN
STOP	END, HALT, QUIT
VALUE	V, VAL
YES	ON, Y

Appendix B. Database Utilities in an RSR Environment

In a Remote Site Recovery (RSR) environment, certain information related to the running of utilities at the active site must be made available to the tracking site. Most of this information is handled automatically by RSR. However, you are responsible for the following:

- Whenever tracked databases are involved, DBRC must be active when the following utilities are run: the Batch Backout utility (DFSBBO00), the Database Change Accumulation utility (DFSUCUM0), the Database Recovery utility (DFSURDB0), and the database reorganization utilities.

  Whenever tracked databases are involved, DBRC and IMS logging must be active when the Partial Database Reorganization utility (DFSPRCT2) is run.
- It is your responsibility to send image copies from the active site to the tracking site and to record them in the tracking site RECON data set.

Only these database utilities can be run at the tracking site:

- Database Change Accumulation utility (DFSUCUM0)
- Database Image Copy utility (DFSUDMP0)
- Database Recovery utility (DFSURDB0)
- Database Image Copy 2 utility (DFSUDMT0)

Database Reorganization Utilities in an RSR Environment

In a Remote Site Recovery (RSR) environment, when tracked databases are involved in reorganizations that cause new image copies to be required at the active site, these image copies must be made available at the tracking site. An information record is written to the log at the next allocation of the database to let the tracking site know that a new image copy is needed, but it is your responsibility to send the image copy to the tracking site.

You must send image copies from the active site to the tracking site if:

- A database is reloaded with the HISAM Reload utility (DFSURRL0). The unload data set that was reloaded can be used as the image copy.
- A database is processed by the Prefix Update utility (DFSURGP0), the updates are logged at the active site and you must apply these updates at the tracking site during normal tracking. If the Prefix Update utility creates log data, an image copy taken any time after the reload step is needed at the tracking site. If the Prefix Update utility does not create a log, an image copy taken after the prefix update step is needed at the tracking site.
- A database is reloaded with the HDAM Reload utility (DFSURGL0) and does not require prefix update.

Use the following procedure after reorganizing a database at the active site to send a new image copy to the tracking site and install it:

- At the active site:
  1. Reorganize the database
  2. Make an image copy of the database
  3. Send the image copy to the tracking site
  4. RSR sends an information log record about the database reorganization to the tracking site
- At the tracking site:

Database Utilities in an RSR Environment

If the reorganized database is tracked at the recovery readiness level, all you need to do is to record the image copy in RECON with the `NOTIFY.IC` command. The following applies to databases that are tracked at the database readiness level.

1. RSR uses the reorganization information log record to update the RECON data set and issues a message indicating that recovery is needed.

   This step may occur at any time during the following process. If the database reorganization information record is processed before the image copy has been recorded in the RECON data set, RSR automatically stops the database. If the record is processed after the image copy has been applied, no message is issued.
2. Make sure the database is not being used when the image copy arrives.
   - Issue the `/DBRECOVERY DATABASE|AREA` command for the database if it is not currently stopped.
   - If a database recovery job is currently running for the database, cancel it.
3. Record the image copy in the RECON data set using the `NOTIFY.IC` command.
4. Run the Database Recovery utility to apply the image copy. You can use the `GENJCL.RECOV` command to generate the necessary JCL.
5. Issue the `/START DATABASE|AREA|DATAGRP` command to resume tracking of the database.

Database Utility Verification

The Active Site

RSR support requires additional verification for IMS database utilities at the active site. For a tracked database or area, the following restrictions apply:

- Time-stamp recovery of one DBDS of a tracked database requires a corresponding time-stamp recovery of any other DBDSs of the database. Subsequent authorization requests for the database (except those from the Database Recovery utility) will be rejected until all DBDSs have been recovered to the same point in time.
- The recovery utility supports recovery to USID boundaries rather than to log volume boundaries. Hence, DBRC is able to relax its requirements for time-stamp recovery. Specifically, the log volume boundary requirements no longer apply. Thus, you can use the `NOFEOV` keyword on the `/DBRECOVERY` command when preparing for time-stamp recovery.

The Tracking Site

RSR support requires additional verification for IMS database utilities at a tracking site. For a covered database or area, the following restrictions apply:

- Database Reorganization Utilities

  Database reorganization is not allowed at the tracking site. If a database reorganization utility requests authorization to a tracked database while signed on to the tracking service group (SG), the authorization is rejected.
- Database Image Copy Utility

  Image copying of a tracking DBDS or area is allowed, as long as the database or area is not authorized to the tracking subsystem when the utility is executed. You can create a batch image copy by specifying `NOCIC` in the image copy parameters, but you cannot create a concurrent image copy (`CIC`) at a tracking site.

An image copy data set created at the tracking site is very similar to a concurrent image copy in that the master database is generally still being updated while the shadow database is being copied. However, like batch image copy, the image copy is, in effect, an instantaneous copy of the DBDS or area because tracking is suspended while the utility is running.

The time stamp (`RUNTIME`) of an image copy data set created at a tracking site is not the time that it was created but the time recorded for the database or area in the active site's RECON. Thus a tracking site image copy has an "effective time" relating it to the active site database or area, and that time can be earlier than the last update applied to the database or area. So recovery using the tracking site image copy might involve some reprocessing of data.

- Database Image Copy 2 Utility

  Image copying of a tracking DBDS or area is allowed, as long as the database or area is not authorized to the tracking subsystem when the utility is executed. You can create a batch image copy by specifying `NOCIC` in the image copy parameters, but you cannot create a concurrent image copy (`CIC`) at a tracking site.

  An image copy data set created at the tracking site is very similar to a concurrent image copy in that the master database is generally still being updated while the shadow database is being copied. However, like batch image copy, the image copy is, in effect, an instantaneous copy of the DBDS or area because tracking is suspended while the utility is running.

  The time stamp (`RUNTIME`) of an image copy data set created at a tracking site is not the time that it was created but the time recorded for the database or area in the active site's RECON. Thus a tracking site image copy has an "effective time" relating it to the active site database or area, and that time can be earlier than the last update applied to the database or area. So recovery using the tracking site image copy might involve some reprocessing of data.

- Database Recovery Utility

  Database recovery of a tracking DBDS or area can be performed as long as the database or area is not authorized to the tracking subsystem when the utility is executed.

  For block-level data sharing subsystems tracked at the recovery readiness level (RLT), time-stamp recovery is performed at the active site, and an image copy of each data set of the database must be sent to the tracking site following the time-stamp recovery.

  For block-level data sharing subsystems tracked at the database readiness level (DLT), time-stamp recovery can be performed at the tracking site by way of online forward recovery.

- Batch Backout Utility

  Batch backout is not allowed at the tracking site. Batch backout is not allowed to sign on to a tracking SG.

Database Reorganization Utilities

In RSR environment, when covered (tracked) databases are involved in reorganizations which cause new image copies to be required at the active site, then these image copies must be made available at the tracking site. An information record is written to the log at the next allocation of the database to let the tracking site know that a new image copy is needed, but it is your responsibility to send the image copy to the tracking site.

Database Utilities in an RSR Environment

Image copies must be sent from the active site to the tracking site in the following cases:

- If a database is reloaded with the HISAM Reload utility (DFSURRL0), an image copy must be provided. The unload data set that was reloaded may be used as the image copy.
- If a database is processed by the Prefix Update utility (DFSURGP0), the updates are logged at the active site and these updates are applied at the tracking site during normal tracking. If the Prefix Update utility creates log data, an image copy taken any time after the reload step is needed at the tracking site. If the Prefix Update utility does not create a log, an image copy taken after the prefix update step is needed at the tracking site.
- If a database is reloaded with the HDAM Reload utility (DFSURGL0) and does not require prefix update.

Use the following procedure after reorganizing a database at the active site to send a new image copy to the tracking site and install it:

- At the active site:
  1. Perform the database reorganization
  2. Make an image copy of the database
  3. Send the image copy to the tracking site
  4. RSR sends an information log record about the database reorganization to the tracking site
- At the tracking site:

  If the reorganized database is tracked at the recovery readiness level, all you need to do is to record the image copy in RECON with the `NOTIFY.IC` command. The following applies to databases that are tracked at the database readiness level.

  1. RSR uses the reorganization information log record to update the RECON data set and issues a message indicating that recovery is needed.

     This step may occur at any time during the following process. If the database reorganization information record is processed before the image copy has been recorded in the RECON data set, RSR automatically stops the database. If the record is processed after the image copy has been applied, no message is issued.
  2. Make sure the database is not being used when the image copy arrives.
     – Issue the `/DBRECOVERY DATABASE|AREA` command for the database if it is not currently stopped
     – If a database recovery job is currently running for the database, cancel it
  3. Record the image copy in the RECON data set using the `NOTIFY.IC` command.
  4. Run the Database Recovery utility to apply the image copy. You can use the `GENJCL.RECOV` command to generate the necessary JCL.
  5. Issue the `/START DATABASE|AREA|DATAGRP` command to resume tracking of the database.

Bibliography

This bibliography includes all the publications cited in this book, including the publications in the IMS library.

DFSMS/MVS V1R4 Access Method Services, SC26-4906

DFSORT Application Programming Guide, SC33-4035

IBM 3990 Storage Control Reference (Models 1, 2, and 3), GA32-0099.

IMS SU/DBT DEDB Unload/Reload User's Guide, SH21–0571–03

MVS/ESA Service Aids, GC28-1669

S/360 OS Sort/Merge: Programmer's Guide, SC33-4007

IMS/ESA Version 6 Library

SC26-8725	ADB	Administration Guide: Database Manager
SC26-8730	AS	Administration Guide: System
SC26-8731	ATM	Administration Guide: Transaction Manager
SC26-8727	APDB	Application Programming: Database Manager
SC26-8728	APDG	Application Programming: Design Guide
SC26-8726	APCICS	Application Programming: EXEC DLI Commands for CICS and IMS
SC26-8729	APTM	Application Programming: Transaction Manager
SC26-8732	CG	Customization Guide
SC26-9517	CQS	Common Queue Server Reference
SC26-8733	DBRC	Database Recovery Control Guide and Reference
LY37-3731	DGR	Diagnosis Guide and Reference
LY37-3732	FAST	Failure Analysis Structure Tables (FAST) for Dump Analysis
GC26-8736	IIV	Installation Volume 1: Installation and Verification
GC26-8737	ISDT	Installation Volume 2: System Definition and Tailoring
SC26-8740	MIG	Master Index and Glossary
GC26-8739	MC	Messages and Codes
SC26-8743	OTMA	Open Transaction Manager Access Guide
SC26-8741	OG	Operations Guide
SC26-8742	OR	Operator's Reference
GC26-8744	RPG	Release Planning Guide
SC26-8767	SOP	Sample Operating Procedures
SC26-8769	URDB	Utilities Reference: Database Manager
SC26-8770	URS	Utilities Reference: System
SC26-8771	URTM	Utilities Reference: Transaction Manager

Supplementary Publications

GC26-8738	LPS	Licensed Program Specifications
SC26-8766	SOC	Summary of Operator Commmands

Online Softcopy Publications

LK3T-2326	CDROM	IMS/ESA Version 6 Softcopy Library
SK2T-0730	CDROM	IBM Online Library: Transaction Processing and Data
SK2T-0710	CDROM	MVS Collection
SK2T-6700	CDROM	OS/390 Collection

Index

Readers' Comments — We'd Like to Hear from You

IMS/ESA
Utilities Reference: Database Manager
Version 6

Publication No. SC26-8769-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC26-8769-05

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International
Business Machines Corporation
Department BWE/H3
P.O. Box 49023
San Jose, CA
95161-9945

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Program Number: 5655-158

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-8769-05

Spine information:

IMS/ESA IMS/ESA V6 Utilities Ref: DB Version 6