IMS

Utilities Reference: Database and Transaction Manager

Version 9

SC18-7833-00

IMS

Utilities Reference: Database and Transaction Manager

Version 9

SC18-7833-00

Note:
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 555.

First Edition (October 2004)

This edition applies to Version 9 of IMS (product number 5655–J38) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About This Book

This information is available as part of the DB2® Information Management Software Information Center for z/OS® Solutions. To view the information within the DB2 Information Management Software Information Center for z/OS Solutions, go to http://publib.boulder.ibm.com/infocenter/dzichelp. This information is also available in PDF and BookManager® formats. To get the most current versions of the PDF and BookManager formats, go to the IMS™ Library page at www.ibm.com/software/data/ims/library.html.

This book is a reference manual for database and transaction administrators and system programmers who use the IMS Version 9 utilities for the IMS Database Manager (IMS DB) and the IMS Transaction Manager (IMS TM) 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 two utilities reference manuals in the IMS library. The scope of the two books is as follows:

- *IMS Version 9: Utilities Reference: System* describes utilities that apply to IMS at a system level or that affect both database and transaction manager operations.
- *IMS Version 9: Utilities Reference: Database and Transaction Manager* describes utilities that affect database operations and transaction operations separately.

With IMS Version 9, you can reorganize HALDB partitions online, either by using the integrated HALDB Online Reorganization function or by using an external product. In this information, the term *HALDB Online Reorganization* refers to the integrated HALDB Online Reorganization function that is part of IMS Version 9, unless otherwise indicated.

Organization of This Book

This book has eight parts and three appendixes:

- Parts 1 through 6 apply to the IMS Database Manager
  - Part 1, “Definition, Migration, Initialization, and Reorganization Utilities,” on page 1 describes the utilities you use to define, migrate, and initialize databases.
  - Part 2, “Reorganization and Conversion Utilities,” on page 97 describes the utilities you use to reorganize databases and explains the MSDB-to-DEDB conversion utility.
  - Part 3, “Backup Utilities,” on page 193 describes the utilities you use to make backup copies of a database.
  - Part 4, “Recovery Utilities,” on page 231 explains the utilities you use to recover databases.
  - Part 5, “Report and Test Utilities,” on page 297 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 339 describes how to use the Utility Control Facility as a controller for the execution of the other utilities.
- Parts 7 through 8 apply to the IMS Transaction Manager:

  - Part 7, “Generation Utilities,” on page 391 describes the transaction manager utilities used during the system definition process.
  - Part 8, “Service Utilities,” on page 501 describes the transaction manager service utilities.
- Appendix A, “Summary of DEDB Utility Commands,” on page 539 summarizes the utility commands you use with the DEDB utilities.
- Appendix B, “Database Utilities in an RSR Environment,” on page 547 explains considerations about database utilities in a Remote Site Recovery (RSR) environment.
- Appendix C, “Summary of HALDB Utilities,” on page 553 lists the utilities that can be used on HALDBs.

For a complete list of all books this manual cites, see the “Bibliography” on page 559.

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: www.ibm.com/ims

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

- *IMS Version 9: Administration Guide: System*
- *IMS Version 9: Administration Guide: Database Manager*
- *IMS Version 9: 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 that the utility produces
- Error processing, with return codes and their meanings
- Examples of how to use the utility.

IBM Product Names Used in This Information

In this information, the licensed programs shown in Table 1 are referred to by their short names.

Table 1. Licensed Program Full Names and Short Names

Licensed program full name	Licensed program short name
IBM Application Recovery Tool for IMS and DB2	Application Recovery Tool

Table 1. Licensed Program Full Names and Short Names (continued)

Licensed program full name	Licensed program short name
IBM CICS® Transaction Server for OS/390®	CICS
IBM CICS Transaction Server for z/OS	CICS
IBM DB2 Universal Database™	DB2 Universal Database
IBM DB2 Universal Database for z/OS	DB2 UDB for z/OS
IBM Enterprise COBOL for z/OS and OS/390	Enterprise COBOL
IBM Enterprise PL/I for z/OS and OS/390	Enterprise PL/I
IBM High Level Assembler for MVS™ & VM & VSE	High Level Assembler
IBM IMS Advanced ACB Generator	IMS Advanced ACB Generator
IBM IMS Batch Backout Manager	IMS Batch Backout Manager
IBM IMS Batch Terminal Simulator	IMS Batch Terminal Simulator
IBM IMS Buffer Pool Analyzer	IMS Buffer Pool Analyzer
IBM IMS Command Control Facility for z/OS	IMS Command Control Facility
IBM IMS Connect for z/OS	IMS Connect
IBM IMS Connector for Java™	IMS Connector for Java
IBM IMS Database Control Suite	IMS Database Control Suite
IBM IMS Database Recovery Facility for z/OS	IMS Database Recovery Facility
IBM IMS Database Repair Facility	IMS Database Repair Facility
IBM IMS DataPropagator™ for z/OS	IMS DataPropagator
IBM IMS DEDB Fast Recovery	IMS DEDB Fast Recovery
IBM IMS Extended Terminal Option Support	IMS ETO Support
IBM IMS Fast Path Basic Tools	IMS Fast Path Basic Tools
IBM IMS Fast Path Online Tools	IMS Fast Path Online Tools
IBM IMS Hardware Data Compression-Extended	IMS Hardware Data Compression-Extended
IBM IMS High Availability Large Database (HALDB) Conversion Aid for z/OS	IBM IMS HALDB Conversion Aid
IBM IMS High Performance Change Accumulation Utility for z/OS	IMS High Performance Change Accumulation Utility
IBM IMS High Performance Load for z/OS	IMS HP Load
IBM IMS High Performance Pointer Checker for OS/390	IMS HP Pointer Checker
IBM IMS High Performance Prefix Resolution for z/OS	IMS HP Prefix Resolution
IBM Tivoli® NetView® for z/OS	Tivoli NetView for z/OS
IBM WebSphere® Application Server for z/OS and OS/390	WebSphere Application Server for z/OS
IBM WebSphere MQ for z/OS	WebSphere MQ
IBM WebSphere Studio Application Developer Integration Edition	WebSphere Studio
IBM z/OS	z/OS

How to Read Syntax Diagrams

The following rules apply to the syntax diagrams that are used in this information:

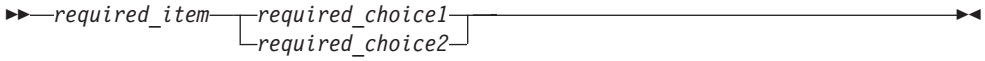
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

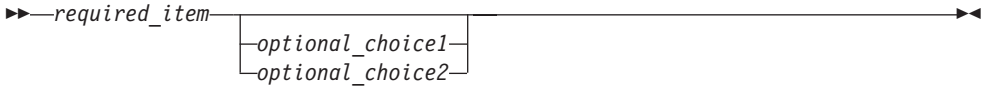
  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

- If you can choose from two or more items, they appear vertically, in a stack.

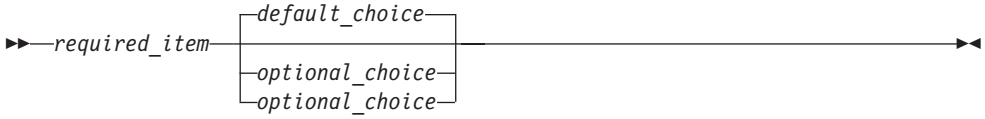
  If you *must* choose one of the items, one item of the stack appears on the main path.



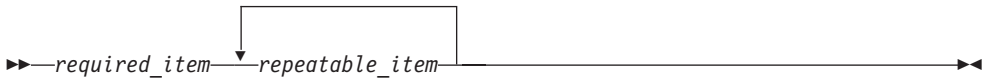
  If choosing one of the items is optional, the entire stack appears below the main path.



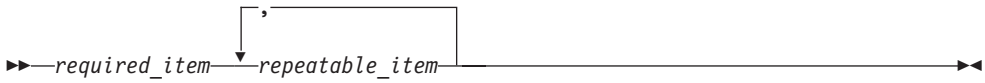
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



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



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 repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.

fragment-name:

- In IMS, a b symbol indicates one blank position.
- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

How to Send Your Comments

Your feedback is important in helping us provide the most accurate and highest quality information. If you have any comments about this or any other IMS information, you can take one of the following actions:

- Go to the IMS Library page at www.ibm.com/software/data/ims/library.html and click the Library Feedback link, where you can enter and submit comments.
- Send your comments by e-mail to imspubs@us.ibm.com. Be sure to include the title, the part number of the title, the version of IMS, and, if applicable, the specific location of the text on which you are commenting (for example, a page number in the PDF or a heading in the Information Center).

Summary of Changes

Changes to This Book for IMS Version 9

This book contains new technical information for IMS Version 9, changed technical information, and editorial changes.

New information on the following enhancements is included:

- HALDB Specific Partition Initialization: Chapter 4, “HALDB Partition Data Set Initialization Utility (DFSUPNT0),” on page 33
- DBRC Enhancements: Chapter 20, “Database Image Copy Utility (DFSUDMP0),” on page 195
- HALDB Online Reorganization Support:

  Chapter 20, “Database Image Copy Utility (DFSUDMP0),” on page 195

  Chapter 21, “Database Image Copy 2 Utility (DFSUDMT0),” on page 207

  Chapter 22, “Online Database Image Copy Utility (DFSUICP0),” on page 223

  Chapter 23, “Database Change Accumulation Utility (DFSUCUM0),” on page 233

  Chapter 25, “Database Recovery Utility (DFSURDB0),” on page 253

  Chapter 26, “Batch Backout Utility (DFSBBO00),” on page 267
- FP Serviceability/Usability: Appendix A, Table 31 on page 539

The following information has changed significantly:

- Product-sensitive Programming Interface Information: “Notices” on page 555
- Specifying DSFMSdss SET PATCH Commands: “Specifying DFSMSdss SET PATCH Commands” on page 211
- Updated Examples:

  “Examples of DFSURGU0” on page 135

  “Examples for DFSUPNT0” on page 36

  “Examples of DFSURPR0” on page 173

The following organizational changes have been made to this information:

- Part 1 has been retitled Part 1, “Definition, Migration, Initialization, and Reorganization Utilities,” on page 1.
- Part 2 has been retitled Part 2, “Reorganization and Conversion Utilities,” on page 97.
- The following chapters have been moved from Part 1 to Part 2:
  - Chapter 12, “HISAM Reorganization Unload Utility (DFSURUL0),” on page 101
  - Chapter 13, “HISAM Reorganization Reload Utility (DFSURRL0),” on page 117
  - Chapter 14, “HD Reorganization Unload Utility (DFSURGU0),” on page 125
  - Chapter 15, “HD Reorganization Reload Utility (DFSURGL0),” on page 139
  - Chapter 16, “Partial Database Reorganization Utility (DFSPRCT1 and DFSPRCT2),” on page 149
  - Chapter 17, “Database Prereorganization Utility (DFSURPR0),” on page 167
  - Chapter 18, “High-Speed DEDB Direct Reorganization Utility (DBFUHDR0),” on page 175

- Part 4 entitled *Conversion Utilities* has been deleted, and the information included in this part has been moved to Part 2, "Reorganization and Conversion Utilities," on page 97.

For detailed information about technical enhancements for IMS Version 9, see the *IMS Version 9: Release Planning Guide*.

Library Changes for IMS Version 9

Changes to the IMS Library for IMS Version 9 include the addition of one title, a change of one title, organizational changes, and a major terminology change. Changes are indicated by a vertical bar (|) to the left of the changed text.

The IMS Version 9 information is now available in the DB2 Information Management Software Information Center for z/OS Solutions, which is available at http://publib.boulder.ibm.com/infocenter/dzichelp. The DB2 Information Management Software Information Center for z/OS Solutions provides a graphical user interface for centralized access to the product information for IMS, IMS Tools, DB2 Universal Database (UDB) for z/OS, DB2 Tools, and DB2 Query Management Facility (QMF™).

New and Revised Titles

The following list details the major changes to the IMS Version 9 library:

- *IMS Version 9: IMS Connect Guide and Reference*

  The library includes new information: *IMS Version 9: IMS Connect Guide and Reference*. This information is available in softcopy format only, as part of the DB2 Information Management Software Information Center for z/OS Solutions, and in PDF and BookManager formats.

  IMS Version 9 provides an integrated IMS Connect function, which offers a functional replacement for the IMS Connect tool (program number 5655-K52). In this information, the term *IMS Connect* refers to the integrated IMS Connect function that is part of IMS Version 9, unless otherwise indicated.
- The information formerly titled *IMS Version 8: IMS Java User's Guide* is now titled *IMS Version 9: IMS Java Guide and Reference*. This information is available in softcopy format only, as part of the DB2 Information Management Software Information Center for z/OS Solutions, and in PDF and BookManager formats.
- To complement the IMS Version 9 library, a new book, *An Introduction to IMS* by Dean H. Meltz, Rick Long, Mark Harrington, Robert Hain, and Geoff Nicholls (ISBN # 0-13-185671-5), is available starting February 2005 from IBM Press. Go to the IMS Web site at www.ibm.com/ims for details.

Organizational Changes

Organization changes to the IMS Version 9 library include changes to:

- *IMS Version 9: IMS Java Guide and Reference*
- *IMS Version 9: Messages and Codes, Volume 1*
- *IMS Version 9: Utilities Reference: System*

The chapter titled ″DLIModel Utility″ has moved from *IMS Version 9: IMS Java Guide and Reference* to *IMS Version 9: Utilities Reference: System*.

The DLIModel utility messages that were in *IMS Version 9: IMS Java Guide and Reference* have moved to *IMS Version 9: Messages and Codes, Volume 1*.

Terminology Changes

IMS Version 9 introduces new terminology for IMS commands:

type-1 command
A command, generally preceded by a leading slash character, that can be entered from any valid IMS command source. In IMS Version 8, these commands were called *classic* commands.

type-2 command
A command that is entered only through the OM API. Type-2 commands are more flexible than type-2 commands and can have a broader scope. In IMS Version 8, these commands were called *IMSplex* commands or *enhanced* commands.

Accessibility Enhancements

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products. The major accessibility features in z/OS products, including IMS, enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

User Assistive Technologies

Assistive technology products, such as screen readers, function with the IMS user interfaces. Consult the documentation of the assistive technology products for specific information when you use assistive technology to access these interfaces.

Accessible Information

Online information for IMS Version 9 is available in BookManager format, which is an accessible format. All BookManager functions can be accessed by using a keyboard or keyboard shortcut keys. BookManager also allows you to use screen readers and other assistive technologies. The BookManager READ/MVS product is included with the z/OS base product, and the BookManager Softcopy Reader (for workstations) is available on the IMS Licensed Product Kit (CD), which you can download from the Web at www.ibm.com.

Keyboard Navigation of the User Interface

Users can access IMS user interfaces using TSO/E or ISPF. Refer to the *z/OS V1R1.0 TSO/E Primer*, the *z/OS V1R5.0 TSO/E User's Guide*, and the *z/OS V1R5.0 ISPF User's Guide, Volume 1*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Part 1. Definition, Migration, Initialization, and Reorganization Utilities

Chapter 1. HALDB Migration Aid Utility (DFSMAID0)

The High Availability Large Database (HALDB) Migration Aid utility will scan an existing database, collect data, perform analysis, and provide statistics and recommendations for HALDB partition boundaries for migrating a full-function database to a HALDB.

You can choose one of the following parameters to analyze the partitioning of the database:

- Key Range (KR)
- Maximum partition size (MAX)
- Number of partitions (NBR)

If KR is chosen, the entire database is traversed to read the root keys. If MAX or NBR is chosen, you can select a sampling technique to analyze the distribution statistics. The sampling technique is selected through the generation of random numbers. A seed number can be used to select random numbers. The sampling technique reduces by orders of magnitude the amount of data that is to be analyzed and therefore, decreases the time required significantly.

The following topics provide additional information:

Input and Output for DFSMAID0

The HALDB Migration Aid utility uses the following input:

- Key ranges per HALDB partition.
- Maximum number of bytes per HALDB partition (leaving space for growth).
- Number of HALDB partitions desired.
- Sample size, when estimation based on a random sample is desired.

The HALDB Migration Aid utility produces the following output displayed in a generated report:

- The input control statements supplied to the utility
- Total bytes — prefix + data lengths that exist for the current database expressed in KB
- Number of database records
- Number of segments by type
- Increase in prefix size (in bytes) that is created in a new HALDB
- Increase due to physical pairing (in bytes) that is created in a new HALDB

Note: The total prefix size of the entire database increases because an EPS (extended pointer set) is used to point to logical parents and because the addition of a physical logical child is provided to replace the virtual logical child.

JCL Requirements for DFSMAID0

The following JCL statements are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that specify the input and output

EXEC Statement

The EXEC statement can be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSMAID0,dbname,,0,,,,,,,,Y,N'
```

The variables in the EXEC Statement are:

dbname
The target database name to be analyzed.

0 Not a ULU restart

Y DBRC to be activated

N IRLM not activated

DD Statements

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

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

IMS DD
Defines the libraries containing the DBD that describe the database to be analyzed. This data set must reside on a direct-access device. This statement is required and must always define the DBD library.

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.

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.

SYSUDUMP DD
Defines an optional dump data set.

database DD
Identifies the database that is to be scanned as indicated by the Database Prereorganization utility. These DD statements are not necessary if the user has provided the DFSMDA member to allow the database to be dynamically allocated. The ddnames must match the ddnames indicated in the DBD. The data set must reside on a direct-access device.

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

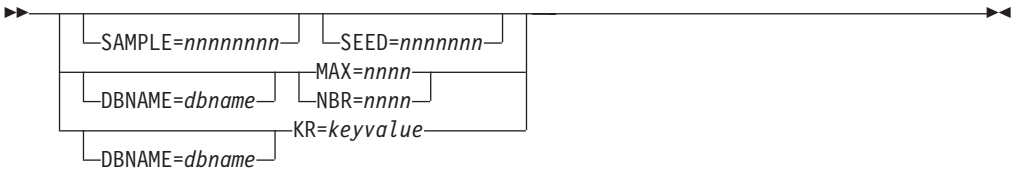
Utility Control Statement for DFSMAID0

Input statements are used to describe processing options for the HALDB Migration Aid utility.

The SYSIN input file has the following syntax rule and restrictions:

- Comments start with '*' in column one
- Mixed case is allowed
- Values in one statement can not be split into two lines. However, multiple statements are allowed.
- No column restrictions
- Only one analysis per run: KR, NBR, or MAX

The format of the HALDB Migration Aid utility control statement is:



DBNAME=
Specifies optionally the database to be scanned.

MAX=
Specifies the maximum number of bytes desired for each HALDB partition in the new database.

NBR=
Specifies the number of HALDB partitions desired in the new database. The range of NBR is one to 1001.

KR=
Specifies the high key values desired for each HALDB partition. A KR parameter is required for each partition except one with “high values.” The analysis always includes a “high values” partition. When KR parameters are specified, MAX, NBR, and SAMPLE can not be used.

SAMPLE
This specification is optional. Specifies the size of a random sample to partition. If a sample size is specified, it can be followed by the seed keyword in the same statement. If sampling is selected by specifying a sample size, then it must be done in the first control statement. Sampling is recommended for large databases. SAMPLE can be used with NBR or MAX. It can not be used with KR. If used, it must precede any other specifications.

SEED=
This specification is optional. It specifies the starting random seed. If SAMPLE is used and SEED is not specified, then SEED defaults to one. SAMPLE and SEED are coded on the same control statement.

Notes:

1. Only one statement with SAMPLE, MAX, or NBR is allowed for each input set.
2. MAX, NBR, and KR are mutually exclusive for an input set.
3. As many KR statements can be entered as the maximum number of HALDB partitions.

Output Messages and Statistics for DFSMAID0

The HALDB Migration Aid utility (DFSMAID0) provides output messages and statistics.

Figure 1 is an example of the statistics obtained from this utility. Partition keys are written in dump format for the length of the key.

```
partition 1:
minimum key=
  +0000 d2c1c1f1f1                |KAA11      |
maximum key=
  +0000 d2f2f3f9 f9               |K2399      |

segment name     segments       bytes          pref-incr      pair-inc
1. 'K1    '      263            14728      2104               0
2. 'K2    '      37             1036       296                0
3. 'K3    '      68             3808       2176               0
4. 'K4    '      35             560        420                0
5. 'K5    '      46             1656       368                0
6. 'K6    '      40             640        480                0
```

Figure 1. DFSMAID0 Output Statistics

The above report is shown for each HALDB partition, followed by the overall totals. Shown is the total increase in prefix size for the entire database. This total increase is due to:

- The use of an EPS to point to logical parents
- A total increase in bytes for the addition of a physical logical child to replace the virtual logical child

Note: When a partition is empty, the minimum key will be set to 0xFFFF..FF and the maximum key set to 0x0000..00.

Return Codes for DFSMAID0

The following return codes are provided at program termination:

Code	Meaning
0	Successful completion
12	Utility terminated unsuccessfully
0100	Too few Access List Entry Table (ALET) slots are preallocated for keys
0101	Too few ALET slots are preallocated for recStats
0102	Too few ALET slots are preallocated for indices
0103	Encountered database records larger than 16 MB

Related Reading: See *IMS Version 9: Messages and Codes, Volume 2* for explanations of the messages accompanying all nonzero return codes.

Examples of DFSMAID0

These examples show how the input parameters are used.

Example 1

The input in Figure 2 defines three partitions. The first two partitions are limited by the value shown in the example and the third partition ends with the highest possible value of the key.

```
//SYSIN DD *
KR=C'1050'
KR=X'F4F5F6F7'
```

Figure 2. Defining Partitions

Example 2

The example in Figure 3 shows the use of the MAX keyword and indicates that a maximum of 750,000,000 bytes is to be allocated to each partition.

```
//SYSIN DD *
MAX=750000000
```

Figure 3. Using Max Keyword

Example 3

The example in Figure 4 shows that sampling will be performed while doing the partition analysis with a sample size of 10000. The SEED defaults to 0. The number of partitions is 100.

```
//SYSIN DD *
SAMPLE=10000
NBR=100
```

Figure 4. Partition Analysis

Chapter 2. HALDB Partition Definition Utility (%DFSHALDB)

Use the HALDB Partition Definition utility to create HALDBs, and to add, modify, and delete HALDB partitions.

You can use both the HALDB Partition Definition utility and DBRC commands to manage your HALDBs.

The HALDB Partition Definition utility is called from within ISPF using the following startup command:

```
TSO %DFSHALDB
```

You can perform the following tasks on the HALDB master and the HALDB partitions by navigating through panels in the HALDB Partition Definition utility:

- Create HALDBs and add HALDB partitions to an existing HALDB.
- Find, view, sort, copy, modify, delete, and print HALDB partition data.
- Create and modify data set groups.
- Edit HALDB information.
- Export and import HALDB definitions.
- View IMS ddname concatenations.
- Select IMS RECON/DBDLIB libraries.

Note: The HALDB Partition Definition utility does not impact RECON data set contention of online IMS subsystems. The RECON data set is reserved only for the time it takes to process a DBRC request. It is not held for the duration of the utility execution.

Related Reading:

- For detailed information on using the HALDB Partition Definition utility, see the information on implementing database design in *IMS Version 9: Administration Guide: Database Manager*.
- For information on using the HALDB Partition Definition utility with HALDB Online Reorganization, see the *IMS Version 9: Administration Guide: Database Manager*.
- You can start the HALDB Partition Definition utility using the IMS Application Menu. For more information, see the information on partitioning sample applications in *IMS Version 9: Installation Volume 1: Installation Verification*.

The following topics provide additional information:

Restrictions for %DFSHALDB

The following restrictions apply when using the HALDB Partition Definition utility:

- The HALDB Partition Definition utility will only operate on databases that are identified as a HALDB in the DBDLIB member. If the DBD is changed after the HALDB definitions are in place, you might not be able to manipulate the definitions. For example, you can not change the number of data set groups without deleting and redefining the HALDB partition definitions.
- Only one person can update the HALDB definition of a particular database at a time. The serialization is on the name of the RECON1 data set and the name of the database.
- This utility does not support Sequential Buffering (SB).

Input and Output for %DFSHALDB

The HALDB Partition Definition utility receives input from the following sources:

- DBD generation information is read from DBDLIB.
- Saved definitions are retrieved from the RECON.
- User input is solicited from interactive panels.
- The result of an export is used as input to a subsequent import operation.
- Configuration information is retrieved from the ISPF profile data set.
- A data set with key strings can be used as input when HALDB partitions are defined.
- Batch import parameters are specified as TSO command parameters.

The HALDB Partition Definition utility produces the following output:

- HALDB definitions are stored in the RECON data set.
- Interactive messages are displayed on the ISPF panel.
- Exported HALDB definitions are saved to a user specified data set.
- Some messages are written to a SYSOUT file.
- HALDB definition information can be printed to the ISPF list file.

Foreground JCL Requirements for %DFSHALDB

The HALDB Partition Definition utility is primarily executed as an interactive application. ISPF is used as the dialog manager. In order to use the HALDB Partition Definition utility, you must log on to TSO and have the HALDB dialog components available to you. A sample TSO logon procedure is shown in Figure 5 on page 13.

```
//HALDB01  EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR /* IMS SDFSRESL */
//SYSPROC  DD DSN=IMS.SDFSEXEC,DISP=SHR /* IMS rexx execs */
//         DD DSN=ISP.SISPCLIB,DISP=SHR
//IMS      DD DSN=your.local.DBDLIB,DISP=SHR
//ISPPLIB  DD DSN=IMS.SDFSPLIB,DISP=SHR /* IMS ISPF panels*/
//         DD DSN=ISP.SISPPLIB,DISP=SHR
//ISPSLIB  DD DSN=IMS.SDFSSLIB,DISP=SHR /* IMS ISPF skeletons*/
//         DD DSN=ISP.SISPSLIB,DISP=SHR
//ISPMLIB  DD DSN=IMS.SDFSMLIB,DISP=SHR /* IMS ISPF messages */
//         DD DSN=ISP.SISPMLIB,DISP=SHR
//ISPTLIB  DD DSN=IMS.SDFSTLIB,DISP=SHR /* IMS ISPF tables*/
//         DD DSN=ISP.SISPTLIB,DISP=SHR
//ISPLOG   DD SYSOUT=*,DCB=(RECFM=VA,LRECL=125,BLKSIZE=129)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VA,LRECL=125,BLKSIZE=129)
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*,DCB=(RECFM=F,LRECL=255,BLKSIZE=255)
//SYSPRINT DD TERM=TS,SYSOUT=A
//ISPCTL0  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//ISPCTL1  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//ISPCTL2  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//ISPLST1  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=121,RECFM=FBA,BLKSIZE=1210)
//ISPLST2  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=121,RECFM=FBA,BLKSIZE=1210)
//ISPWRK1  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=256,RECFM=FB,BLKSIZE=2560)
//ISPWRK2  DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
//         DCB=(LRECL=256,RECFM=FB,BLKSIZE=2560)
//SYSTERM  DD TERM=TS,SYSOUT=A
//SYSIN    DD TERM=TS
```

Figure 5. TSO Logon Procedure

The TSO logon procedure must include:

- DD statements from the production TSO / ISPF logon procedure
- IMS SDFSRESL data set in the STEPLIB DD statement
- IMS dialog components in the appropriate ISPF DD statements

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS load modules.

IMS DD
Defines the library containing the DBD for the HALDB databases.

SYSPROC DD
Defines the data sets that contain CLISTS and REXX execs. The HALDB Partition Definition utility requires its execs to be in this set of data sets.

ISPPLIB DD
Defines the data sets that contain ISPF panels. The HALDB Partition Definition utility requires its panels to be in this set of data sets.

ISPMLIB DD
Defines the data sets that contain ISPF message members. The HALDB Partition Definition utility requires its message members to be in this set of data sets.

ISPTLIB DD
Defines the data sets that contain ISPF tables. The HALDB Partition Definition utility requires its tables to be in this set of data sets.

RECON1 DD
This optional DD statement defines the first DBRC RECON data set. If it is not provided, there must be a RECON1 member in the IMS.SDFSRESL data set that identifies the RECON1 data set.

RECON2 DD
This optional DD statement defines the second DBRC RECON data set. If it is not provided, there must be a RECON2 member in the IMS.SDFSRESL data set that identifies the RECON2 data set.

RECON3 DD
This optional DD statement defines the third DBRC RECON data set. If it is not provided, there must be a RECON3 member in the IMS.SDFSRESL data set that identifies the RECON3 data set.

Starting the HALDB Partition Definition Utility

The HALDB Partition Definition utility is started from within ISPF. From the ISPF command line, issue:

```
TSO %DFSHALDB
```

The HALDB Partition Definition utility can also be started from the IMS Application Menu. For more information, see the information on partitioning sample applications in *IMS Version 9: Installation Volume 1: Installation Verification*.

Batch JCL Requirements for %DFSHALDB

The import function of the HALDB Partition Definition utility can be performed in a batch job. Even though it does not execute interactively, it still uses ISPF for internal functions. The JCL must include normal ISPF DD statements and start ISPF.

The JCL must include:

- A JOB statement within which you define DD statements that are used by ISPF
- The IMS.SDFSRESL data set in the STEPLIB DD statement
- The IMS dialog components in the appropriate ISPF DD statements

Figure 6 on page 15 is a sample Batch Import JCL.

```
//*
//DSPXRUN  EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M
//STEPLIB  DD DISP=SHR,DSN=IMS.SDFSRESL
//         DD DISP=SHR,DSN=ISP.SISPLOAD
//         DD DISP=SHR,DSN=ISP.SISPLPA
//SYSPROC  DD DISP=SHR,DSN=IMS.SDFSEXEC
//         DD DISP=SHR,DSN=ISP.SISPCLIB
//RECON1   DD DISP=SHR,DSN=IMS.RECON1
//RECON2   DD DISP=SHR,DSN=IMS.RECON2
//RECON3   DD DISP=SHR,DSN=IMS.RECON3
//IMS      DD DISP=SHR,DSN=IMS.DBDLIB
//ISPPROF  DD DSN=&&PROFILE,UNIT=SYSDA,DISP=(NEW,DELETE),
//         SPACE=(3200,(30,30,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//ISPPLIB  DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSLIB
//         DD DISP=SHR,DSN=ISP.SISPSLIB
//ISPMLIB  DD DISP=SHR,DSN=IMS.SDFSMLIB
//         DD DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB  DD DISP=SHR,DSN=ISP.SISPTENU
//ISPLOG   DD SYSOUT=*,DCB=(RECFM=VA,LRECL=125,BLKSIZE=129)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VA,LRECL=125,BLKSIZE=129)
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*,DCB=(RECFM=F,LRECL=255,BLKSIZE=255)
//SYSTSIN  DD *
   ISPSTART CMD( +
   DSPXRUN +
   IMPORT +
   DSN('USRT002.ISPF.PROFILE') +
   DBN(PARTDBA) +
   MEM(PARTDBA))
```

Figure 6. %DFSHALDB Batch Import JCL

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS load modules.

IMS DD
Defines the library containing the DBD for the HALDB.

SYSPROC DD
Defines the data sets that contain CLISTS and REXX execs. The HALDB Partition Definition utility requires its execs to be in this set of data sets.

ISPMLIB DD
Defines the data sets that contain ISPF message members. The HALDB Partition Definition utility requires its message members to be in this set of data sets.

ISPTLIB DD
Defines the data sets that contain ISPF tables.

RECON1 DD
This optional DD statement defines the first DBRC RECON data set. If it is not provided, there must be a RECON1 member in the IMS.SDFSRESL data set that identifies the RECON1 data set.

RECON2 DD
This optional DD statement defines the second DBRC RECON data set. If it is not provided, there must be a RECON2 member in the IMS.SDFSRESL data set that identifies the RECON2 data set.

RECON3 DD
This optional DD statement defines the third DBRC RECON data set. If it is not provided, there must be a RECON3 member in the IMS.SDFSRESL data set that identifies the RECON3 data set.

SYSTSIN DD
The SYSTSIN DD statement contains the TSO command that starts both ISPF and the HALDB Partition Definition utility. When it is necessary to continue to the next line, use a plus or minus sign as the last character of the line you wish to continue.

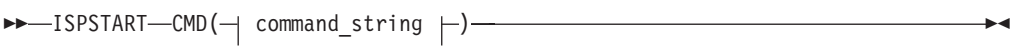
Utility Control Statement for %DFSHALDB

You can use utility control statements to describe processing options for the HALDB Partition Definition utility. DSPXRUN supports the import and export of HALDB definitions. ISPSTART supports the execution if ISPF code in general.

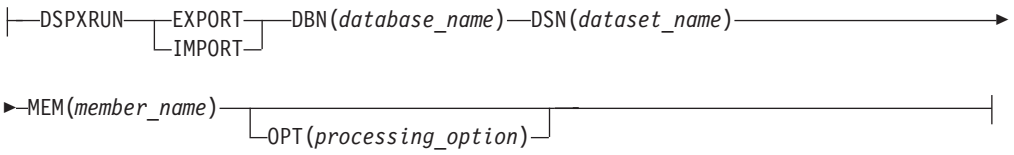
The utility control statements follow normal TSO command syntax and continuation rules:

- Continue the line by placing a plus sign as the last character on the line.
- Separate parameters with blanks.

The command syntax for the DSPXRUN command for a HALDB follows:



command_string:



database_name
The database name that was specified in the primary panel when the Export file was created by the PDU.

data set_name
The input data set name is the name of the data set that contains the partition information. The data set must be a partitioned data set.

member_name
The input member name is the name of a member within the input data set. The member must have been exported using the HALDB Partition Definition utility.

processing_option
Each partition in the imported table can be defined in RECON. If there are errors, you can choose to try the remaining partitions or to stop the process. The valid values are:

1. Stop on first error.
2. Try all partitions.

Output Messages for %DFSHALDB

The HALDB Partition Definition utility provides various output messages. Figure 7 is an example of the messages obtained from this utility.

```
DSPM083I Start Import to DBN=PARTDBA from MEM=PARTDBA in DSN='IBMUSER.HALDB.EXPORT' Options=2
DSPM085I Imports start at 99/07/09 11:50
DSPM084I Import successful for partition PAAA
DSPM084I Import successful for partition PAAB
DSPM084I Import successful for partition PAAC
DSPM084I Import successful for partition PAAD
DSPM082I 4 of a total 4 partitions from table PARTDBA were imported to database successfully.
```

Figure 7. %DFSHALDB Output Message

Return Codes for %DFSHALDB

These return codes are provided at program termination for batch import. Termination return codes are not used for the interactive dialog.

Code	Meaning
0	Successful completion
4	Some HALDB partitions could not be imported
8	No HALDB partitions were imported

Some messages displayed by the HALDB Partition Definition utility are standard ISPF message prompts when the field values are not correct. ISPF messages begin with a prefix of ISR or ISP.

Related Reading: See *IMS Version 9: Messages and Codes, Volume 2* for more information on return codes generated by this utility.

Examples of %DFSHALDB

The examples assume that JCL similar to Figure 6 on page 15 is provided. The examples include the SYSTSIN statement as a point of reference.

Example 1

This example exports a database definition.

```
//SYSTSIN DD *
   ISPSTART CMD( +
                 DSPXRUN EXPORT   +
                   DSN('PROD.RSR.PARTS') +
                   DBN(IVPDB1) +
                   MEM(IVPDB1) +
                 )
 /*
```

Example 2

This example imports a database definition and stops if there is an error.

```
//SYSTSIN DD *
   ISPSTART CMD( +
                 DSPXRUN IMPORT   +
                   DSN('PROD.RSR.PARTS') +
                   DBN(IVPDB1) +
                   MEM(IVPDB1) +
                   OPT(1)   +
                 )
 /*
```

Example 3

This example imports two HALDBs and continues with other HALDB partitions even if an error occurs. Each of the imports is independent of the other.

```
//SYSTSIN DD *
   ISPSTART CMD(DSPXRUN IMPORT DSN('PROD.RSR.HALDB') +
                DBN(IVPDB1) MEM(IVPDB1) OPT(2)  )

   ISPSTART CMD(DSPXRUN IMPORT DSN('PROD.RSR.HALDB') +
                DBN(IVPDB2) MEM(IVPDB2) OPT(2)  )
 /*
```

Chapter 3. 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 16, “Partial Database Reorganization Utility (DFSPRCT1 and DFSPRCT2),” on page 149 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.

The following topics provide additional information:

Input and Output for DFSPRSUR

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 for DFSPRSUR” on page 20 and “Utility Control Statement for DFSPRSUR” on page 22 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 of DFSPRSUR” on page 24 for an annotated sample Surveyor report.

JCL Requirements for DFSPRSUR

The Database Surveyor utility is executed as a standard z/OS 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

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 the macro information in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for information for member names DBBBATCH, IMSBATCH, and DLIBATCH, and 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.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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: See the information on tailoring the IMS system to your environment in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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

Related Reading: See the information on specifying sequential buffering control statements in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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.

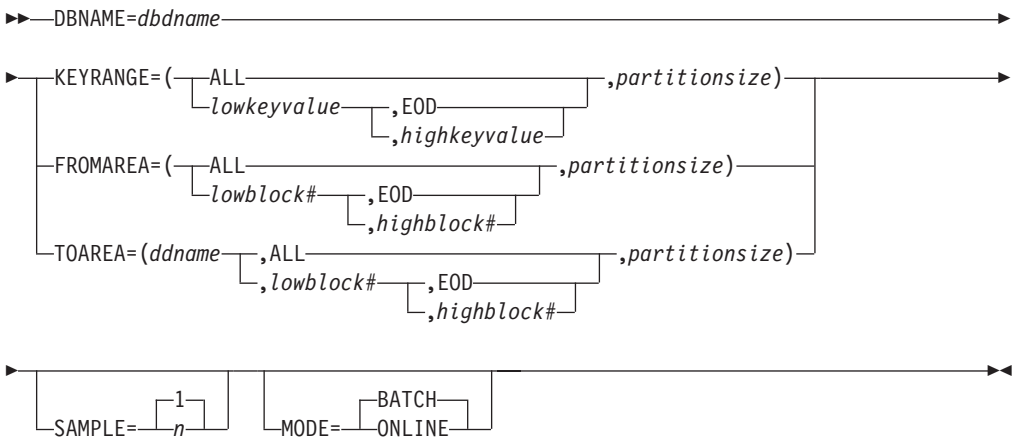
Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statement for DFSPRSUR

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 for DFSPRSUR

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 Version 9: Messages and Codes, Volume 1* for explanations of all messages produced by the Database Surveyor utility.

Examples of DFSPRSUR

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 DD statements shown in Figure 8 to the sample JCL:

```
//RECON1    DD DSN=IMS.RECON1,DISP=SHR
//RECON2    DD DSN=IMS.RECON2,DISP=SHR
//RECON3    DD DSN=IMS.RECON3,DISP=SHR
```

Figure 8. DD Statements for Using DBRC without Dynamic Allocation

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'
//IMS      DD DSN=IMS.PSBLIB,DISP=SHR
//         DD DSN=IMS.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  input for VSAM and OSAM buffers and options
```

Figure 9 on page 25 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

NO. BLK
Heading line of number of blocks

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

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 the partition being reported) which physically reside in this portion of this DSG

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

PERCENT OF AREA OCCUPIED
Percentage of area occupied by segments (found in the 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

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'
//IMS      DD DSN=IMS.PSBLIB,DISP=SHR
//         DD DSN=IMS.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  input for VSAM and OSAM buffers and options
```

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'
//IMS      DD DSN=IMS.PSBLIB,DISP=SHR
//         DD DSN=IMS.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   input for VSAM and OSAM buffers and options
```

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

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

Chapter 4. HALDB Partition Data Set Initialization Utility (DFSUPNT0)

The HALDB Partition Data Set Initialization utility (DFSUPNT0) initializes HALDB partitions in two ways. It can pass a HALDB master database name on the execute parameter list to the IMS Region Controller. It can also pass a HALDB master database name directly with a list of HALDB master database names or HALDB partition names that are specified as SYSIN statements. If a HALDB master name is specified, then partitions identified in the RECON that require partition initialization (PINIT=Y) are initialized. If a HALDB partition name is specified as a SYSIN statement, then initialization for that partition is done unconditionally, and PINIT=N is set in the RECON.

The following topics provide additional information:

- "Restrictions for DFSUPNT0"
- "JCL Requirements for DFSUPNT0"
- "Utility Control Statement for DFSUPNT0" on page 35
- "Return Codes for DFSUPNT0" on page 36
- "Examples for DFSUPNT0" on page 36

Restrictions for DFSUPNT0

The following restrictions apply when using the HALDB Partition Data Set Initialization utility:

- Data sets are dynamically allocated and must have been previously defined.
- DBRC is required.
- RECON DD statements are required if not dynamically allocated.
- DFSUPNT0 produces no error messages.
- Do not code JCL DD statements for partition data sets. Partition data sets must be dynamically allocated by IMS based on information from the RECON.
- DFSUPNT0 may be executed as multiple job tasks. Each task will attempt to initialize all required partitions found in RECON and will serialize around the initialization function using a system enqueue. After a successful enqueue, DBRC is queried for a list of partitions requiring initialization. Partitions requiring initialization different from the ones at job batch initialization time are ignored. Partitions that required initialization at batch initialization time but no longer require it are ignored. Only partitions that still require initialization after the enqueue are initialized.

Use the HALDB Partition Initialization utility after migrating to HALDB. HALDB Partition Initialization provides complete checking and validation of each partition by IMS.

JCL Requirements for DFSUPNT0

The HALDB Partition Data Set Initialization utility is executed as a standard z/OS 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

Two EXEC statements are possible; one is without a SYSIN, and one is with a SYSIN. Only a master database name can be passed on the exec statement as follows:

```
PARM=(ULU,DFSUPNT0,HDODB1,,,,,,,,SYS3,,Y,N)
```

where HDODB1 is always a master database name. Partition names are only allowed as SYSIN statements.

Without a SYSIN, the EXEC statements must be in this form:

```
//PINIT01  EXEC PGM=DFSRRC00,REGION=2048K,
//             PARM=(ULU,DFSUPNT0,HDODB1,,,,,,,,SYS3,,Y,N)
//STEPLIB   DD DSN=IMS.SDFSRESL
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//RECON1    DD DSN=IMS.RECON1,DISP=SHR
//RECON2    DD DSN=IMS.RECON2,DISP=SHR
//RECON3    DD DSN=IMS.RECON3,DISP=SHR
//DFSVSAMP  DD input for VSAM and OSAM buffers and options
//SYSUDUMP  DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//SYSIN DD  DUMMY
/*
```

With a SYSIN, the EXEC statements must be in this form:

```
//PINIT02  EXEC PGM=DFSUPNT0,REGION=2048K
//STEPLIB  DD DSN=IMS.SDFSRESL
//STEPCAT  DD DSN=VCATSHR,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD DSN=IMS.DBDLIB,DISP=SHR
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DBVHDJ05
DB0HIDK5
/*
```

Note: This JCL initialized all database partitions recorded in RECON as PINIT for HALDB master databases DBVHDJ05 and DB0HIDK5. Partition PARTSK1 in HALDB master database DD41SK01 is initialized unconditionally and is recorded in RECON with PINIT off.

DD Statements

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

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

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.

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.

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.

The SYSPRINT data set is also used to return error messages to the user.

Figure 15 is an example of DFS3911 output to SYSPRINT:

```
DFS391I PARTITION INITIALIZATION UTILITY
DFS391I SYSIN CONTROL CARDS
DFS391I POHIDJB
DFS391I PHVNTKA
DFS391I POHIDJC
DFS391I POHIDJA
DFS391I END OF SYSIN CONTROL CARDS
```

Figure 15. DFS3911 Output to SYSPRINT

SYSUDUMP DD
Defines a dump data set.

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

DFSOVRDS DD
Defines the option for unconditional partition initialization. A HALDB with all of its partitions can be initialized, without the PARTITION INIT NEEDED flag in the RECON being set to Y, by using this override option. To use the option, the following DD statement must be used:

```
//DFSOVRDS DD *
INITALL
```

Figure 16. DD Statement for Unconditional Partition Initialization

Note: All previously existing partition data for all the HALDB partitions is lost.

Utility Control Statement for DFSUPNT0

Input statements are used to describe processing options for the HALDB Partition Data Set Initialization utility. The input statement must conform to the following:

- To execute with a SYSIN, specify HALDB master in column one.
- To execute without a SYSIN, specify the database name in region controller.

Return Codes for DFSUPNT0

The following return codes are provided at program termination:

Code	Meaning
0	Successful completion
8	Processing error
12	Invalid input or IMS control block
16	Environment or user error
32	Abend returned from attach of IMS
99	Internal logic error

Examples for DFSUPNT0

The examples in this section provide sample JCL to execute DFSUPNT0.

Example 1

This sample JCL executes DFSUPNT0 for HALDB HDODB1 to initialize all partitions recorded in RECON as PINIT or partition initialization needed. The parameters to the batch program controller are for utility region type `ULU`. Partition data sets must previously exist and any containing data will be reset to empty. Empty data sets for OSAM must have been defined in a previous job step. The partition data sets are dynamically allocated using information from RECON. RECON data sets are dynamically allocated using MDA members found in the IMSVS.

```
//STEP1    EXEC PGM=IEFBR14       /* Define OSAM data sets here */
//HDOSAM   DD ... DISP=(,CATLG)
//PINIT    EXEC PGM=DFSRRC00,REGION=2048K,
//             PARM={ULU,DFSUPNT0,HDODB1,,,,,,,,SYS3,,Y,N}
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.PSBLIB,DISP=SHR
//         DD  DSN=IMSVS.DBDLIB,DISP=SHR
//DFSVSAMP DD  DSN=IMSTESTG.DFSVSAMP.DATA{VSM885FP},DISP=SH
//SYSUDUMP DD  SYSOUT=A
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  DUMMY
//HDOSAM   DD ... DISP=OLD
/*
```

Example 2

This sample JCL executes DFSUPNT0 to initialize all partitions recorded in RECON as PINIT for a list of HALDB master database names specified in the SYSIN. IMS is attached as a utility region type `ULU` using a static parameter list.

```
//STEP1    EXEC PGM=IEFBR14       /* Define OSAM data sets here */
//HDOSAM   DD ... DISP=(,CATLG)
//STEP2    EXEC PGM=IDCAMS        /* Define VSAM data sets here */
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DEFINE ...
//PINIT    EXEC PGM=DFSUPNT0,REGION=2048K
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//DFSRESLB DD  DSN=IMSVS.RESLIB,DISP=SHR
//IMS      DD  DSN=IMSVS.PSBLIB,DISP=SHR
//         DD  DSN=IMSVS.DBDLIB,DISP=SHR
//DFSVSAMP DD  DSN=IMSTESTG.DFSVSAMP.DATA{VSM885FP},DISP=SH
```

```
//SYSUDUMP DD  SYSOUT=A
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  DBVHDJ05
  DBOHIDK5
/*
```

Chapter 5. Database Scan Utility (DFSURGS0)

Use the Database Scan utility to scan non-HALDB 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.

The information in Table 2 identifies inputs and outputs for the Database Scan utility.

Table 2. Inputs and Outputs used by the Database Scan Utility

Input	Output
RECON	Work data set
DBD library	Output messages
Input control statements	
Control data set	
Databases to be scanned	

Figure 17 on page 40 shows a flow diagram of the Database Scan utility.

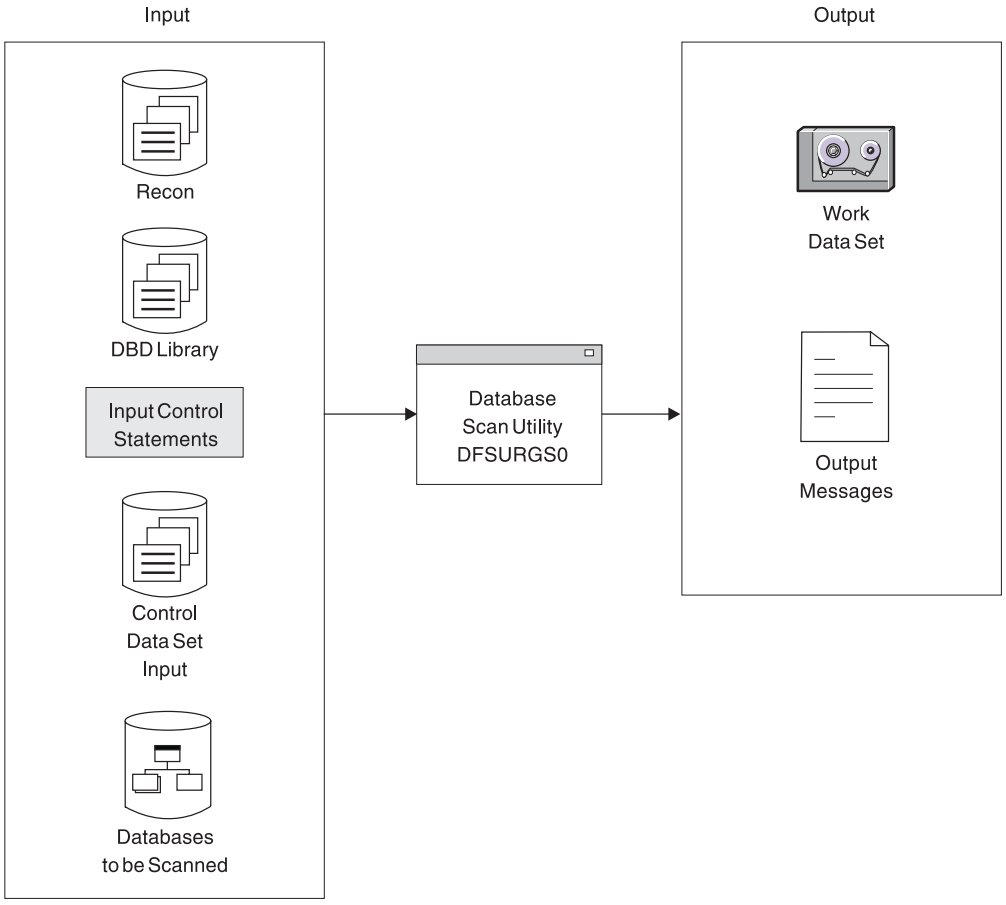


Figure 17. Database Scan Utility

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

Related Reading: Refer to Chapter 34, "Utility Control Facility (DFSUCF00)," on page 341 for a description of its operation.

The following topics provide additional information:

- "Recovery and Restart for DFSURGS0"
- "JCL Requirements for DFSURGS0" on page 41
- "Utility Control Statements for DFSURGS0" on page 43
- "Output Messages and Statistics for DFSURGS0" on page 45
- "Return Codes for DFSURGS0" on page 46
- "Example of DFSURGS0" on page 46

Recovery and Restart for DFSURGS0

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 for DFSURGS0

The Database Scan utility is executed as a standard z/OS job. You must provide JCL statements:

- 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 Version 9: 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 7 KB 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.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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 contains output from this database scan utility and is included in the concatenation which forms the SORTIN data set for the Prefix Resolution utility. 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: See “Specifying the IMS Buffer Pools” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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: See "SB Parameters (SBPARM) Control Statement" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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 for DFSURGS0

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

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, or 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 or 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

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 z/OS 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

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 z/OS 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 z/OS 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

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 for DFSURGS0

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 Version 9: Messages and Codes, Volume 1*.

Return Codes for DFSURGS0

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 of DFSURGS0

For the example in this section, if you are using DBRC without dynamic allocation, you must add the DD statements shown in Figure 18 to the sample JCL:

```
//RECON1    DD DSN=IMS.RECON1,DISP=SHR
//RECON2    DD DSN=IMS.RECON2,DISP=SHR
//RECON3    DD DSN=IMS.RECON3,DISP=SHR
```

Figure 18. DD Statements for Using DBRC without Dynamic Allocation

Figure 19 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 the DBIL or the DBR control statement 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.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGS0'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1200
//DFSURWF1  DD DSN=IMS.URWF1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL),
//             DCB=(LRECL=300,BLKSIZE=1008,RECFM=VB)
//HDRELTD   DD DSN=DATABASE.DBRELATD,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0003,
//DFSURCDS  DD DSN=IMS.RLCDS,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSMSC
//DFSVSAMP  DD input for VSAM and OSAM buffers and options
```

Figure 19. Scanning a Database Defined by the HDRELTD DD Statement

DD statements must be included for all logically related databases.

Chapter 6. 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.

Note: Prefix resolution is not required for HALDB databases.

The information in Table 3 identifies inputs and outputs used by the Database Prefix Resolution utility.

Table 3. Data sets used by the Database Prefix Resolution Utility

Input	Output
Control data set	Work data set
Sort input	Messages
Sort merge library	
Sort work data set	
Work data set	
Index work data set	

Figure 20 on page 48 is a flow diagram of the Database Prefix Resolution utility.

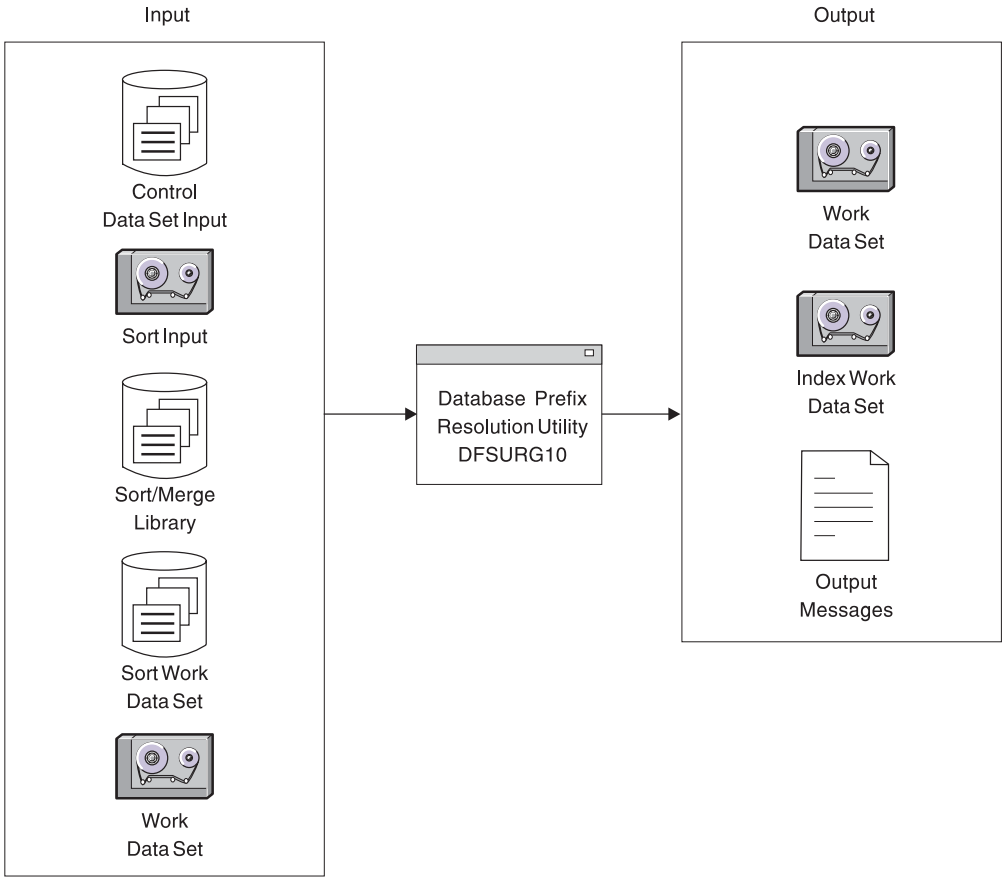


Figure 20. Database Prefix Resolution Utility

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

Related Reading: Refer to Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for a description of its operation.

The following topics provide additional information:

- “Restrictions for DFSURG10”
- “JCL Requirements for DFSURG10” on page 49
- “Output Messages and Statistics for DFSURG10” on page 53
- “Return Codes for DFSURG10” on page 53
- “Example of DFSURG10” on page 53

Restrictions for DFSURG10

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

- For any given logical parent and logical child pair, the sum of items 1 and 2 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 these quantities can be omitted from the summations as follows:
  - 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 limit check is not satisfied for either a logical parent or a logical child, you can omit loading 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 limit check for logical parent and 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 and logical child combination, message DFS885 is issued.

Related Reading: See *IMS Version 9: Messages and Codes, Volume 1* 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 for DFSURG10

The Database Prefix Resolution utility is executed as a standard z/OS 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.

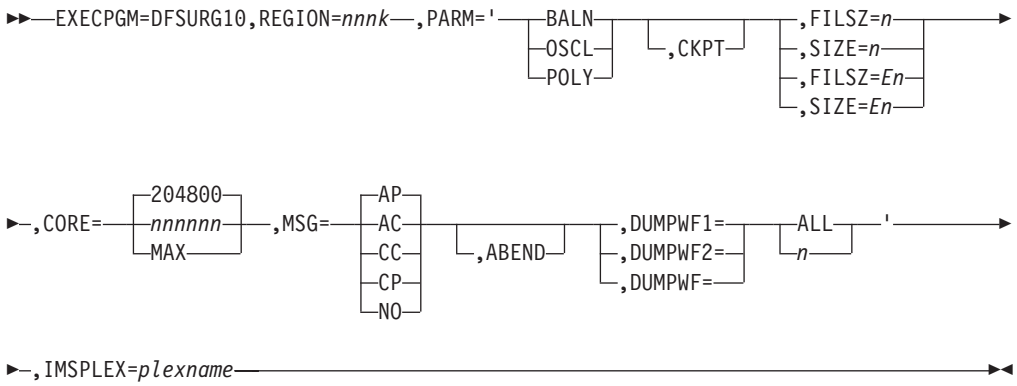
EXEC Statement

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:



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

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

Only the keywords shown in this example 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).

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 work-file 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.SDFSRESL, 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.

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 DFSURWF1 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 generated for this program. It must be the output control data set 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 for DFSURG10

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 for DFSURG10

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 z/OS 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: See *IMS Version 9: Messages and Codes, Volume 2* for an explanation of the DFS878 and DFS885 cautionary messages.

Example of DFSURG10

Figure 21 on page 54 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
/*
```

Figure 21. Resolving Logical Relationships and Secondary Indexes with the Database Prefix Resolution Utility

Chapter 7. 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, reorganization, or both. The updating is done using the output generated by the Prefix Resolution utility.

Note: Prefix update is not required for HALDB databases.

The information in Table 4 identifies inputs and outputs used by the Database Prefix Update utility.

Table 4. Data sets used by the Database Prefix Update Utility

Input	Output
RECON	Output messages and statistics
Unloaded database	Reload output OSAM data set
DBD library	
Input control statements	

Figure 22 is a flow diagram of the Database Prefix Update utility.

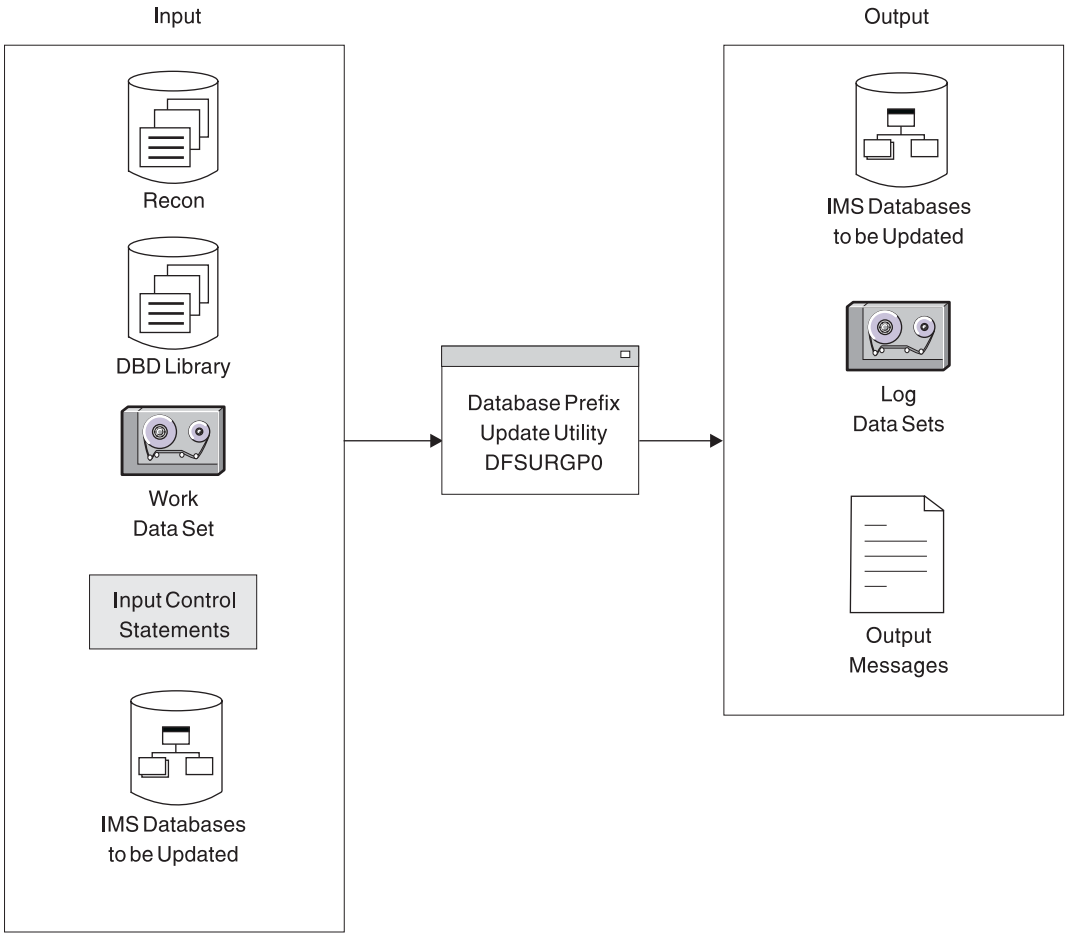


Figure 22. Database Prefix Update Utility

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

Related Reading: Refer to Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for a description of its operation.

The following topics provide additional information:

- “Output for DFSURGP0”
- “Recovery and Restart for DFSURGP0”
- “JCL Requirements for DFSURGP0”
- “Utility Control Statements for DFSURGP0” on page 59
- “Output Messages and Statistics for DFSURGP0” on page 60
- “Return Codes for DFSURGP0” on page 60
- “Examples of DFSURGP0” on page 60

Output for DFSURGP0

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 for DFSURGP0

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 re-execute the program, using the original input work data set.

JCL Requirements for DFSURGP0

The Database Prefix Update utility is executed as a standard z/OS 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 Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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, reorganized, or both. 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, reorganized, 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: See “Specifying the IMS Buffer Pools” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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: See “SB Parameters (SBPARM) Control Statement” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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 for DFSURGP0

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

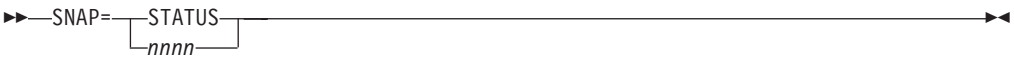
This utility control statement indicates that checkpoint operations are to be performed during operation of this program. The Prefix Resolution utility automatically writes records on the DFSURWF3 data set as a service for the Prefix Update utility, and these records are used as checkpoints. As each checkpoint record is encountered, a checkpoint message (DFS867) is issued to the z/OS 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

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 z/OS 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



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

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 for DFSURGP0

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 for DFSURGP0

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 of DFSURGP0

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=IMS.RECON1,DISP=SHR
//RECON2  DD  DSN=IMS.RECON2,DISP=SHR
//RECON3  DD  DSN=IMS.RECON3,DISP=SHR
```

Example 1

Figure 23shows 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=IMS.DBDLIB,DISP=SHR
//         DD DSN=IMS.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)
/*
```

Figure 23. Example of Updating Five Databases

Example 2

Figure 24 on page 62 is an example of reorganizing two logically related databases. The two databases are DIVNTZ02 (a HISAM VSAM database) and DHVNTZ02 (a HIDAM or VSAM database) with an index VSAM database (DXVNTZ02).

```
//JOBLIB    DD DSN=IMS.SDFSRESL,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,
//         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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,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 DSN=IMS.DBDLIB, DISP=SHR
//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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//*******************************************************************
//*      UNLOAD THE HISAM DATABASE - DIVNTZ02
//*******************************************************************
//*
//UNLOAD2  EXEC PGM=DFSRRC00,COND=(1,LT),
//         PARM='ULU,DFSURGU0,DIVNTZ02,,1,,,,,,,,Y,N'
```

Figure 24. 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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,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 24. 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)
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
//*
//*******************************************************************
//*      RELOAD THE HIDAM DATABASE - DHVNTZ02
//*******************************************************************
//*
//RELOAD1  EXEC PGM=DFSRRC00,
//         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
```

Figure 24. Example of Reorganizing Two Logically Related Databases (Part 3 of 5)

```
//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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//*
//*****************************************************************
//*      RELOAD THE HISAM DATABASE - DIVNTZ02
//*****************************************************************
//*
//RELOAD2  EXEC PGM=DFSRRC00,
//         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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
//*
//*****************************************************************
//*     PREFIX RESOLUTION
//*****************************************************************
//*
//PREFRES  EXEC PGM=DFSURG10
//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 24. 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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,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=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSVSAMP DD *
512,10
1024,10
2048,10
4096,10
IOBF=(4096,5)
/*
//DFSCTL DD *
SBPARM ACTIV=COND,DB=SKILLDB,BUFSETS=6
/*
SB CONTROL STATEMENT
//SYSIN DD *
SNAP=STATUS
/*
//
```

Figure 24. Example of Reorganizing Two Logically Related Databases (Part 5 of 5)

Chapter 8. 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 a z/OS 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.

The following topics provide additional information:

Restrictions for DBFUMIN0

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 Fast Path 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 for DBFUMIN0

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
- 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 for DBFUMIN0

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,PARM=(IMSPLEX=plexname)
```

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 system definition 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.

The IMSPLEX parameter specifies which IMSplex DBRC should join. See initializing and maintaining RECONs in *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference* for detail information about the IMSPLEX parameter.

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.

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.

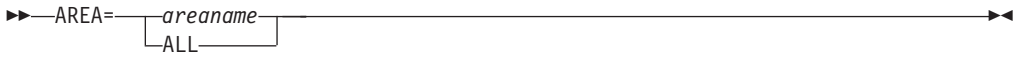
DISP=OLD, UNIT, and VOL parameters must be specified if the DD statement describes a multivolume data set.

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 DBFUMIN0” for an explanation of the required format.

Utility Control Statement for DBFUMIN0

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.



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 for DBFUMIN0

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 Version 9: Messages and Codes, Volume 1* for explanations of the messages accompanying all nonzero return codes.

Example of DBFUMIN0

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 DD statements shown in Figure 25 to the sample JCL:

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

Figure 25. DD Statements for Using DBRC without Dynamic Allocation

This example shows sample JCL to initialize a DEDB.

```
//*
//*       INITIALIZE DEDB DATABASE       (DATAB01)
//*
//TESTIT  EXEC PGM=DBFUMIN0
//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 9. MSDB Maintenance Utility (DBFDBMA0)

Use the MSDB Maintenance utility to create a z/OS sequential data set for MSDB initial load and to maintain the MSDBs. This utility is an offline utility that runs in a z/OS 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.

The information in Table 5 identifies the inputs and outputs for the MSDB Maintenance utility.

Table 5. Data sets used by the MSDB Maintenance Utility

Input	Output
Change	New MSDBINIT
Control	Messages
Old MSDBINIT	

Figure 26 shows the input and output data sets used by the MSDB Maintenance utility for initializing and altering MSDBs.

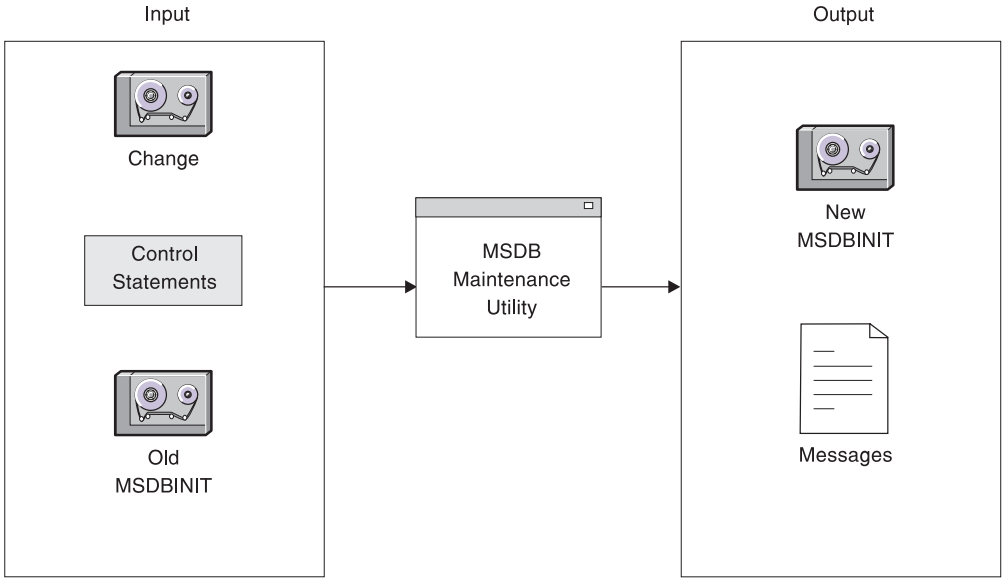


Figure 26. MSDB Maintenance Utility Input and Output Data Sets

The following topics provide additional information:

- "Using the MSDB Maintenance Utility" on page 74
- "Restrictions for DBFDBMA0" on page 75
- "Input and Output for DBFDBMA0" on page 75
- "JCL Requirements for DBFDBMA0" on page 76
- "Utility Control Statements for DBFDBMA0" on page 77
- "Return Codes for DBFDBMA0" on page 80
- "Examples of DBFDBMA0" on page 80

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.

Insert, replace, and delete functions of the MSDB Maintenance utility relate to 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.

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.

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 for DBFDBMA0

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 for DBFDBMA0

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 79, 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 for DBFDBMA0” on page 77.
- 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 for DBFDBMA0

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:

```
//  EXEC  PGM=DBFDBMA0
```

Requirement: The utility requires at least 512 KB of virtual storage.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, 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 for DBFDBMA0

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 of DBFDBMA0” on page 80.

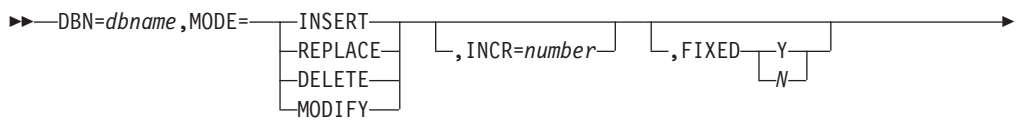
Run Statement

One run statement is used for the entire utility execution. The format of the run statement is:

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*
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 Version 9: 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:

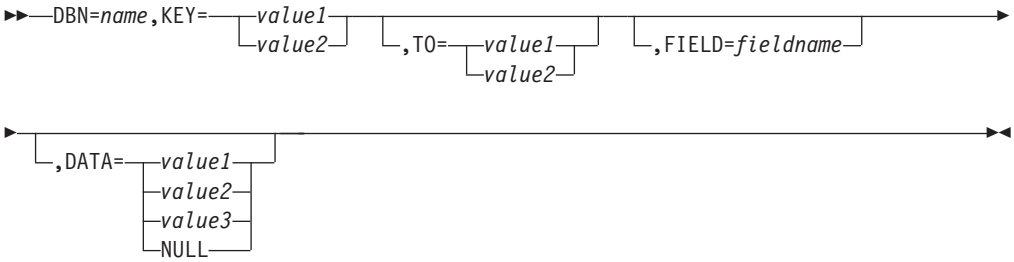
- To replace a segment, supply a record with an existent key in the KEY field and the desired segment content in the MSDB segment field.
- To insert a segment, supply a record with a nonexistent key in the KEY field and the desired segment content in the MSDB segment field.
- To delete a segment, supply 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=

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 under the DATA keyword.

FIELD=
Is an optional keyword that specifies which fields are to be modified. FIELD= is valid only for INSERT, REPLACE, and MODIFY. Each FIELD= keyword must be followed by its associated DATA= keyword.

DATA=
Specifies the contents of a field or segment. If a FIELD= keyword precedes DATA=, it specifies the contents of a field. If FIELD= does not precede DATA=, it specifies the entire contents of a segment.

value1
Is a character field within quotation marks. This field can be continued by breaking the field at column 71, inserting a nonblank character in column 72, and continuing the field in column 16 of the next statement.

value2
Is the hexadecimal representation of the character field within quotation marks and is preceded by an 'x'. This field can be continued by breaking the field at column 71, inserting a nonblank character in column 72, and continuing the field in column 16 of the next statement.

value3
Is an arithmetic field supplied as a signed or unsigned decimal number within parentheses instead of quotation marks. The utility converts this number to the format matching the field type. Leading zeros 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 for DBFDBMA0

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 of DBFDBMA0

“Example 1” on page 81, “Example 2” on page 81, and “Example 3” on page 82 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.SDFSRESL,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.SDFSRESL,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.SDFSRESL,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 10. 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 for DBFUMSC0” on page 84.
- Specify an exit routine, other than the one provided by IMS, which can change the segment content and length.
- Specify a load module which can:
  - Change the sort criteria.
  - Sort for a specific record number and average record size.
  - Use a specific type of work space for sorting.
  - Specify that sort criteria not be used at all. The version 7.0 format CI subset will be processed without sorting.

The following topics provide additional information:

Restrictions for DBFUMSC0

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 “Guidelines for Writing IMS Exit Routines” and “Data Entry Database Sequential Dependent Scan Utility Exit Routine” in *IMS Version 9: 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.

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 DBFUMSC0

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 539 for more detail on these commands).

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, default end processing is invoked. In the high-water-mark (HWM) CI, a permanent default end timestamp segment is built. The scan starts with the oldest sequential dependent CI boundary in the area. The scan ends with— but does not include—the timestamp from the HWM CI default end segment.

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.

When a STARTRBA CI boundary is specified:

- The first segment timestamp is the selected starting point .
- For V5COMP, all segments in this CI are included.

When a STARTRBA segment boundary is specified:

- The specified segment acts like a committed marker segment for timestamp and location attributes.
- For V5COMP, the specified segment is the start point and the timestamp is ignored.

When default start is used:

- All segments with a timestamp not less than the logical begin timestamp (LBTS) will be selected.
- For V5COMP, all non-aborted segments in the first CI are selected.

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. Only one STARTSEQ command is valid per STARTROOT command. If more than one STARTSEQ command is specified, the last command is used. The same applies to STOPROOT and STOPSEQ. 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.

When a STOPRBA CI boundary is specified:

- The first segment timestamp in the CI is used as a stop timestamp.
- For V5COMP, the CI is excluded and the timestamp is ignored.

When a STOPRBA segment boundary is specified:

- The specified segment acts like a committed marker segment for timestamp and location attributes.
- For V5COMP, the specified segment is the stop point and the timestamp is ignored.

When default end is used:

- All segments with a timestamp less than the high-water-mark (HWM) CI default end segment timestamp are included.
- For V5COMP, all non-aborted segments up to, but excluding, the HWM CI are selected.

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 for DBFUMSC0

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 for DBFUMSC0

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 "FPUTIL Procedures" in *IMS Version 9: 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.

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 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.

EXPANDSEG
Specifies that segment expansion is to be performed when in a STEPLIB concatenation, for example.

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 might 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.

STARTHEXT
Specifies a start time in hex format.

You must supply 16 hex digits matching the storeclock value desired. The syntax for STARTHEXT is:

```
STARTHEXT x '0123456789abcdef'   /* 16 hex digits required.  */
```

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. For example, only one blank space can exist between the SS=second parameter and the hh:mm=offset parameter as follows:

```
TYPE   SCAN
AREA   <area name>
STARTIME=C'1998.181 12:30:25 +02:00'
```

Note: Each z/OS partner must set its clock to the same time as the other partner z/OSs in the complex or the conversion to stored clock will not be the same from one z/OS partner to another.

STOPHEXT
Specifies a stop time in hex format.

You must supply 16 hex digits matching the storeclock value desired. The syntax for STOPHEXT is:

```
STOPHEXT x'0123456789abcdef'   /* 16 hex digits required.  */
```

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 z/OS offset.

V5COMP
Indicates that all scan and delete operations follow the rules of IMS 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 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 timestamps across the CIs.

Related Reading: See Appendix A, “Summary of DEDB Utility Commands,” on page 539 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 of DBFUMSC0

This example shows the JCL and utility control statements for executing the DEDB Scan utility.

```
//SCAN2   EXEC  FPUTIL,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 11. 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.

The following topics provide additional information:

- "Restrictions for DBFUMDL0"
- "Input and Output for DBFUMDL0"
- "Recovery and Restart for DBFUMDL0" on page 93
- "JCL Requirements for DBFUMDL0" on page 93
- "Example of DBFUMDL0" on page 94

Restrictions for DBFUMDL0

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 for DBFUMDL0

The input to the DEDB Sequential Dependent Delete utility consists of a data set that contains the input parameters supplied by commands.

Related Reading: See Appendix A, "Summary of DEDB Utility Commands," on page 539 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, default end processing is invoked. In the high-water-mark (HWM) CI, a permanent default end timestamp segment is built. The delete starts with the oldest sequential dependent CI boundary in the area. The delete ends with the timestamp from the HWM CI default end segment, which is excluded from the delete.

When V5COMP is specified for default end:

- The delete range starts with the oldest sequential dependent CI boundary in the area.
- The Logical Begin is advanced to DMACNXTS (the next SDEP CI to be allocated) and all preallocated and current SDEP CIs are discarded. The use of this combination will empty the SDEP portion of the DEDB.

`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.

When STOPRBA CI boundary is specified:

- The first segment timestamp in the CI is used as a stop timestamp.
- For V5COMP, the CI is excluded and the timestamp is ignored.

Note: When STOPRBA is a CI boundary equal to HWM CI or a preallocated SDEP CI (within the HWM set), and the other user also specifies QUITCI, this will signal the SDEP DELETE utility to move Logical Begin past the HWM CI.

When a STOPRBA segment boundary is specified:

- The specified segment acts like a committed marker segment for timestamp and location attributes.
- For V5COMP, the specified segment is the stop point and the timestamp is ignored.

When default end is used:

- All segments with a timestamp less than the HWM CI default end segment timestamp are included.
- For V5COMP, Logical Begin will be set to DMACNXTS (next SDEP CI to be allocated) and message DFS2637I is issued.

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.

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 for DBFUMDL0

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 for DBFUMDL0

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 “FPUTIL Procedures” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for information on FPUTIL.

DD Statements

STEPLIB DD
Describes the library that contains the DEDB Sequential Dependent Delete utility.

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 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.

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.

STOPHEXT
Specifies a stop time in hex format.

You must supply 16 hex digits matching the storeclock value desired. The syntax for STOPHEXT is:

```
STOPHEXT x '0123456789abcdef'   /* 16 hex digits required.  */
```

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. If the offset field is omitted, it is assumed to be the current, local z/OS offset.

Note: Each z/OS must set its clock to the same time as the other partner z/OSs in the complex or the conversion to stored clock format will not be the same from one z/OS partner to another.

Related Reading: See Appendix A, “Summary of DEDB Utility Commands,” on page 539 for a description of how to write the control commands and operands.

SYSPRINT DD
Describes the output data set that contains messages and statistics.

V5COMP
Indicates that all scan and delete operations follow the rules of IMS Version 5, including:

- START and STOP on a segment boundary, without regard to time stamps.
- 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.

Using V5COMP allows you to produce results from the utilities that are identical to those of IMS 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.

Example of DBFUMDL0

Figure 27 on page 95 shows the JCL for the DEDB Scan utility. Figure 28 on page 95 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,
//             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 27. Sample JCL for the DEDB Scan Utility

```
//DELETE EXEC  FPUTIL,
//       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 28. Sample JCL for the DEDB Delete Utility

Part 2. Reorganization and Conversion Utilities

Chapter 12. 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 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 sets with a shared secondary index, if one exists.

The information in Table 6 identifies inputs and outputs for the HISAM Reorganization Unload utility.

Table 6. Data Sets Used by the HISAM Reorganization Unload Utility

Input	Output
RECON	Output messages and statistics
DBD library	Extracted index
VSAM KSDS data set	Reorg output
VSAM ESDS data set	
Index work data set	
Input control statements	

Figure 29 on page 102 is a flow diagram of this utility.

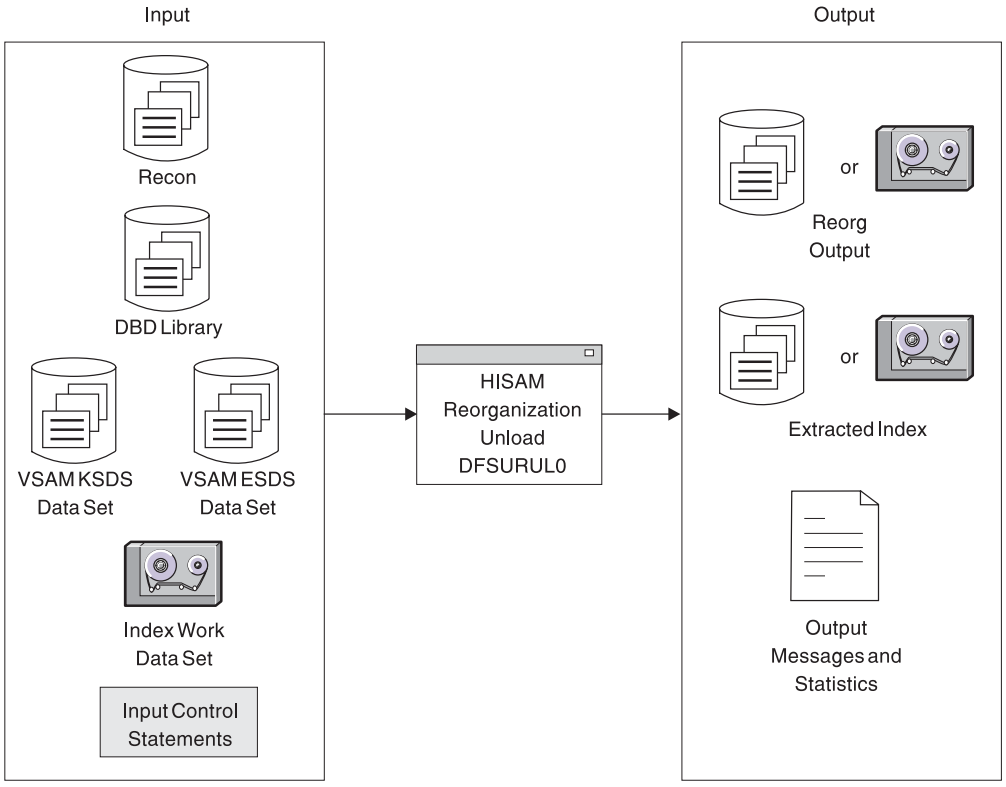


Figure 29. HISAM Reorganization Unload Utility

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: See Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for more information on the Utility Control Facility and a description of its operation.

The following topics provide additional information:

- “Restrictions for DFSURUL0”
- “JCL Requirements for DFSURUL0” on page 103
- “Utility Control Statement for DFSURUL0” on page 105
- “Output Messages and Statistics for DFSURUL0” on page 108
- “Return Codes for DFSURUL0” on page 112
- “Examples of DFSURUL0” on page 112

Restrictions for DFSURUL0

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

- This utility cannot unload a PSINDEX database. The HD Reorganization Unload utility is used to unload the PSINDEX database. For more information on the HD Reorganization Unload utility see Chapter 14, “HD Reorganization Unload Utility (DFSURGU0),” on page 125.
- 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 recovery impossible.
- 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 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 control statement to specify the new sizes. Otherwise, use CHNG=DBD in this field.

  Related Reading: Refer to *DFSMS Access Method Services for Catalogs* for more information.
- 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 are not registered to DBRC. Message DFS194W is issued if this is tried.
- Sequential buffering does not support this utility.

JCL Requirements for DFSURUL0

The HISAM Reorganization Unload utility is executed as a standard z/OS 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 the IMSGEN macro information in *IMS Version 9: Installation Volume 2: System Definition and Tailoring*.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized (because libraries that are not APF authorized are concatenated to IMS.SDFSRESL), 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 can be routed to the output stream. DCB parameters specified for this data set are RECFM=FBA and LRECL=133. BLKSIZE must be provided on the SYSPRINT DD statement and must be a multiple of 133.

SYSIN DD
Defines the input control statement data set. This data set can reside on a tape, or a direct-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 match the name in the DBD that describes this data set. It must also appear on the utility control statement in the SYSIN data set of this job step. One DD statement of this type must be present for each VSAM KSDS that is to be reorganized.

This DD statement represents the primary data set of the HISAM database. In case of secondary index creation, you must provide 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 match the 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, and the second volume continues to the normal 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 for DFSURUL0"). 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: See "Specifying the IMS Buffer Pools" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

The data set can reside on a tape or direct-access device, or it can be 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 that the control region is using.

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

Utility Control Statement for DFSURUL0

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.

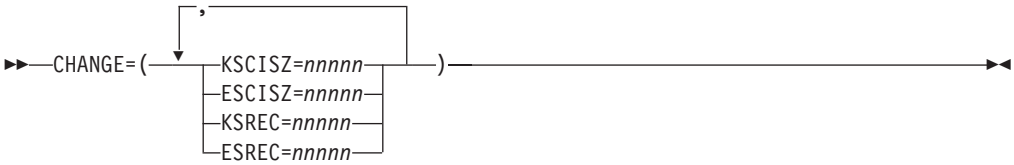
40-48 Must contain the ddname of the secondary index work data set if this control statement is type 'X'.

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: The CHANGE statement is used to specify a new KSDS CI size and a new ESDS CI size in this example:

```
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 that no statistics output be generated. If this parameter is used, the HISAM Reorganization Reload utility also ignores statistics output.

Output Messages and Statistics for DFSURUL0

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 30 on page 109 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 30. 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: Refer to *IMS Version 9: Messages and Codes, Volume 1* for explanations of all numbered messages.

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 that 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 for DFSURUL0

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 of DFSURUL0

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 DD statements shown in Figure 31 to the sample JCL:

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

Figure 31. DD Statements for Using DBRC without Dynamic Allocation

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 DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1330
```

```
//DBIDX1    DD DSN=IMS.INDX1,DISP=SHR
//DXOUT1    DD DSN=IMS.DBXOUT1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=SER=DXOUT1,LABEL=(,SL)
//DBIDX2    DD DSN=IMS.INDX2,DISP=SHR
//DXOUT2    DD DSN=IMS.DBXOUT2,DISP=(,KEEP),
//             UNIT=TAPE,VOL=SER=DXOUT2,LABEL=(,SL)
//NDXDS     DD DSN=IMS.NDXWDS,DISP=(OLD,DELETE)
//DFSVSAMP  DD input for VSAM and OSAM buffers and options
//* +----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 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 DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1330
//DBIDX2    DD DSN=IMS.INDX2,DISP=SHR
//DBIDX1    DD DSN=IMS.INDX1,DISP=SHR
//DBXOUT1   DD DSN=IMS.DBXOUT1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=SER=DXOUT1,LABEL(,SL)
//DFSEXTDS  DD DSN=IMS.EXTDS1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=SER=DEXTDS,LABEL=(,SL)
//NDXWDS1   DD DSN=IMS.XWDS1,DISP=(OLD,PASS)
//NDXWDS2   DD DSN=IMS.XWDS2,DISP=(OLD,PASS)
//NDXWDS3   DD DSN=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 DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1330
//DBIX1A    DD DSN=IMS.DBIX1A,DISP=OLD
//DBIX2B    DD DSN=IMS.DBIX2B,DISP=OLD
//DBOUT1    DD DSN=IMS.DBOUT1A,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,1))
//DBOUT2    DD DSN=IMS.DBOUT2A,DISP=(,KEEP),
//             VOL=SER=DBOUT2,LABEL=(,SL)
//INDXWDS1  DD DSN=IMS.INDXWDS1,DISP=(OLD,DELETE)
//INDXWDS2  DD DSN=IMS.INDXWDS2,DISP=(OLD,DELETE)
//DFSEXTDS  DD DSN=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 13. HISAM Reorganization Reload Utility (DFSURRL0)

Use the HISAM Reorganization Reload utility to:

- Reload a 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.

The information in Table 7 identifies inputs and outputs for the HISAM Reorganization Reload utility.

Table 7. Data Sets Used by the HISAM Reorganization Reload Utility

Input	Output
RECON	Reload Output OSAM data set
Unloaded database	Output messages and statistics
DBD library	
Input control statements	

Figure 32 is a diagram of the HISAM Reorganization Reload utility.

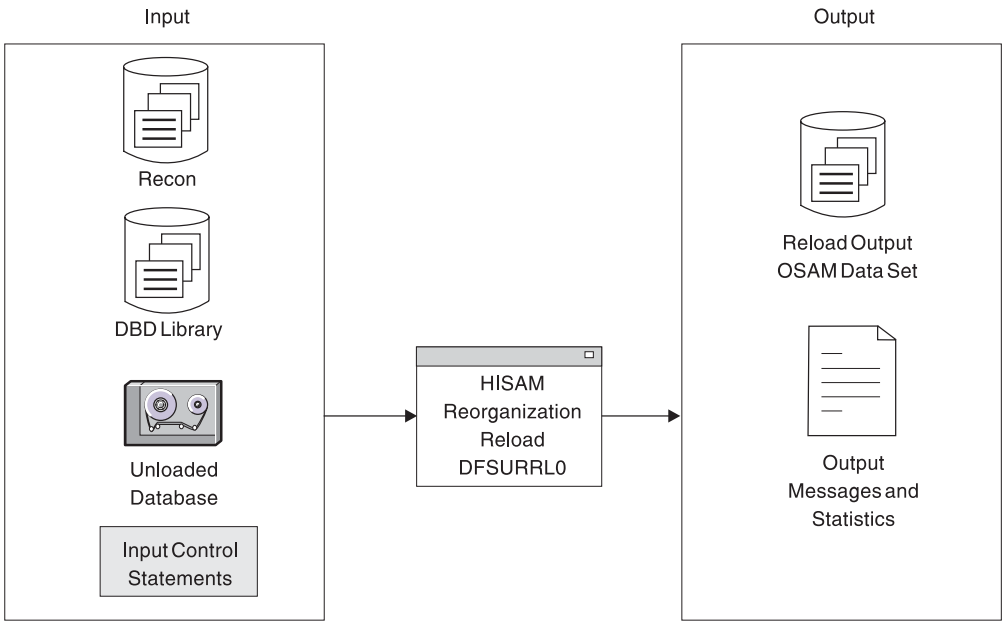


Figure 32. HISAM Reorganization Reload Utility

Related Reading: The functions of this utility can also be performed by the Utility Control Facility. Refer to Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for a description of its operation.

The following topics provide additional information:

- “Restrictions for DFSURRL0” on page 118
- “JCL Requirements for DFSURRL0” on page 118

Restrictions for DFSURRL0

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

- You cannot use this utility to reload a PSINDEX database.
- 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.
- 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 for DFSURRL0

The HISAM Reorganization Reload utility is executed as a standard z/OS 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,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 Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having libraries that are not APF authorized concatenated to IMS.SDFSRESL, 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: See “Specifying the IMS Buffer Pools” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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.

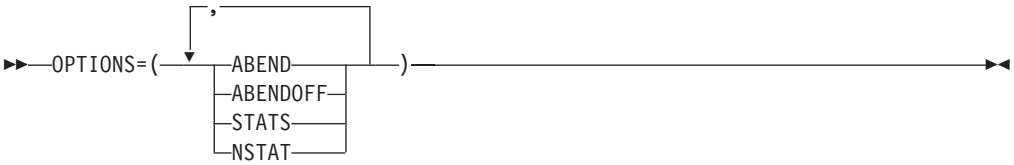
If you are using dynamic allocation, do not use these RECON data set ddnames.

Utility Control Statement for DFSURRL0

The HISAM Reorganization Reload utility has one optional control statement.

OPTIONS Statement

Options Statement



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 for DFSURRL0

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 33 on page 121.

```
                    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 33. 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 Version 9: Messages and Codes, Volume 1*.

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

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 for DFSURRL0

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 14. HD Reorganization Unload Utility (DFSURGU0)

Use the HD Reorganization Unload utility to:

- Unload an HDAM, PHDAM, HIDAM, PHIDAM, PSINDEX, or HISAM database to a sequential data set.
- Generate a data set with prefix information (if logical relationships exist).
- Make structural changes to a HDAM, PHDAM, HIDAM, PHIDAM, or HISAM database.
- Record logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents in a database.
- Migrate non-HALDB to HALDB and fall back from HALDB.

Related Reading:
See the information on tuning databases in *IMS Version 9: Administration Guide: Database Manager* for detailed information on performing HD Reorganization Unload utility functions.

Migration unload requires running HD Unload of each secondary index database. Additionally, the primary database is read to resolve source and target segments needed to construct the ILK (Indirect List Entry Key) and EPS (Extended Pointer Set) information for the migration unload record destined for the new PSINDEX. Sequential processing of secondary index segments causes many random reads of the primary database.

These random accesses of the primary database across multiple secondary index unloads causes poor overall performance of the migration unload of DL/I secondary index databases to HALDB PSINDEX.

For example, if a single primary database has seven secondary index databases and each migration unload job takes thirteen hours, then the total time spent unloading all of the secondary index databases would be 7 x 13, or 91 hours. A significant portion of this time is spent reading the prime database multiple times.

You must use a utility control statement when running the HD Reorganization Unload utility against a PHDAM and PHIDAM database partition or range of partitions. You must also use a utility control statement when migrating a non-HALDB to a HALDB.

When unloading HALDBs, the HD Reorganization Unload utility creates a prefix for unloaded segments from HALDBs to support reorganization. The prefix is mapped by the macro DFSURGUP. DFSURGUP contains the ILK (Indirect List Entry Key) and the EPS (Extended Pointer Set) if the unloaded segment is a logical child. The alternate prefix also contains the HALDB partition ID and the reorganization number. For an example of a format for the ILK prefix, see Figure 34 on page 126.

However, when unloading non-HALDBs, the HD Reorganization Unload utility creates a prefix for unloaded segments from non-HALDBs to support reorganization as well. The prefix is mapped by the macro DFSURGUF. For an example, see Figure 35 on page 133.

The information in Table 8 identifies inputs and outputs for the HD Reorganization Unload utility.

Table 8. Data Sets Used by the HD Reorganization Unload Utility

Input	Output
RECON	Checkpoint data set
DBD library	Output messages and statistics
Database data set	Unloaded data set
Checkpoint data set	

Figure 34 is a flow diagram of the HD Reorganization Unload utility.

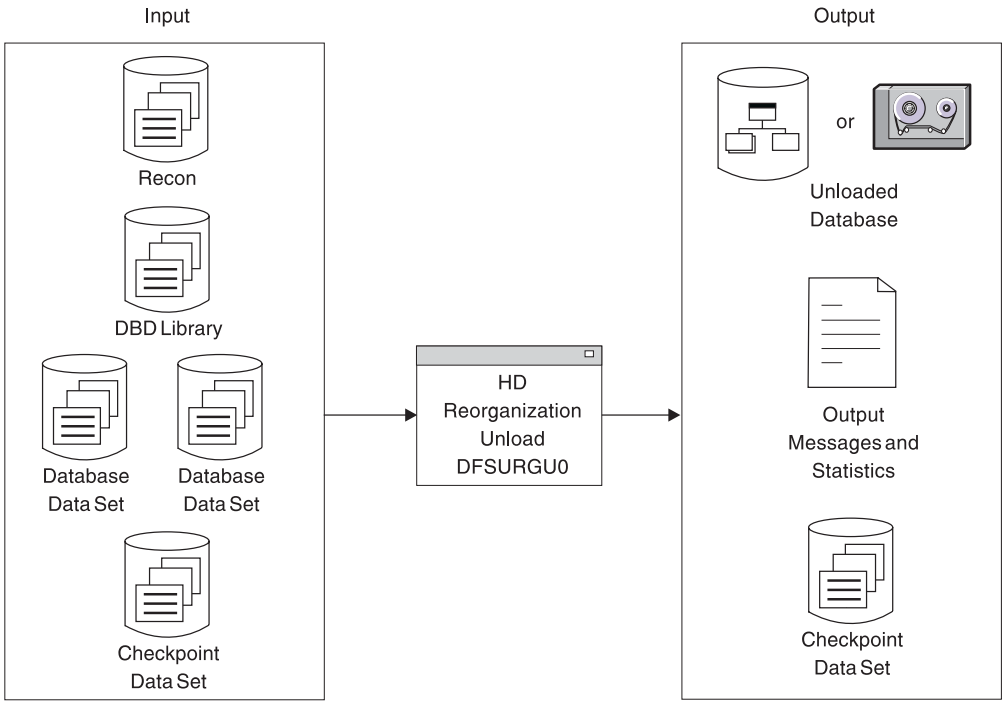


Figure 34. HD Reorganization Unload Utility

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

Related Reading: Refer to Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for a description of its operation.

The following topics provide additional information:

• “Examples of DFSURGU0” on page 135

Rules and Restrictions for DFSURGU0

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:

- The Utility Control Facility is not supported for HALDBs.
- 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.
- To change from OSAM format to VSAM format or VSAM format to OSAM format for HALDBs, the HALDB definition must be deleted and redefined in RECON.
- When unloading an HDAM or PHDAM database, the randomizing module and any IMS user exit routines, such as compression and sparse index, 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 DL/I access method can be changed with the exception of HDAM or PHDAM, which cannot be changed to either indexed method. HISAM, HIDAM, and PHIDAM can be changed to HDAM or PHDAM.
- Segment pointer options for HDAM, PHDAM, HIDAM, and PHIDAM databases 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 that will become virtual must be deleted. Virtual pairing is not supported for HALDB.

  Note: This restriction does not apply to HALDB.
- If you do not furnish a utility control statement when running this utility against a HALDB, the entire database is unloaded. Unloading either a single HALDB partition or a range of HALDB partitions is supported.
- The migrate unload command can be performed only on non-HALDBs.
- 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, with exception for PSINDEX.
  - 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.
- Migration of secondary index databases with non-unique keys requires separate JCL steps to sort and merge the unload records and create new /SX values prior to inputting them into HDRELOAD. When /SX values are generated in the unload record for non-unique keys the HALDB DBD must be changed to accommodate the /SX prior to performing the reload step.

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: See the information on tuning databases in *IMS Version 9: Administration Guide: Database Manager* for further information on the use of the HD Reorganization Unload utility.

JCL Requirements for DFSURGU0

The HD Reorganization Unload utility is executed as a standard z/OS job. The following JCL statements 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.

Coding JCL to Sort HDUNLOAD Records for Secondary Index Databases with Symbolic Pointing When Migrating to HALDB

Follow these steps for writing code to sort HDUNLOAD records for secondary index databases with symbolic pointing:

1. Get the offset to the key and the key length of the concatenated key from DBD to use in the SORT FIELDS control statement.
   a. Determine that the offset to the start of the secondary index concatenated key is always 63 bytes from the RDW of the unload record. Use the number of bytes as the offset value in the SORT FIELDS parameter for DFSORT™. This value is defined in the DFSURGUP macro used to map the HALDB unload records.

b. Calculate the sortkey length by adding the length of the source, the length of the target, and the length of the /SX value (8). For example, if the length of the source is 10 bytes and the length of the target is 10 bytes, then the length of the sortkey is:

```
source(10) + target(10) + /SX(8) = 28
```

Use this value as the length value in the SORT FIELDS parameter for DFSORT.

2. Code a JCL step that writes the header, trailer, and unload records to separate files, as in this example:

```
//SORTIN  DD DSN=UNLOAD.OUTPUT,DISP=SHR
//HEADER  DD DSN=HEADER.FILE,DISP=(NEW,PASS)
//TRAILER DD DSN=TRAILER.FILE,DISP=(NEW,PASS)
//ULCOPY  DD DSN=UNLOAD.COPY,DISP=(NEW,PASS)
//SYSIN   DD *
OPTION COPY
OUTFIL INCLUDE=(5,2,CH,EQ,X'0080'),FNAMES=HEADER
OUTFIL INCLUDE=(5,2,CH,EQ,X'0090'),FNAMES=TRAILER
OUTFIL SAVE,FNAMES=ULCOPY
RECORD TYPE=V
END
```

3. Code a JCL step that sorts just the unload file, as in this example:

```
//SORTIN  DD DSN=IMSTESTS.HOSIX.UNLOAD.COPY,DISP=SHR
//SORTOUT DD DSN=IMSTESTS.HOSIX.UNLOAD.SORTED1,DIS=(,CATLG),
//           UNIT=SYSDA,VOL=SER=000000,SPACE=(CYL,(1,5))
//SYSIN   DD *
SORT FIELDS=(63,28,CH,A)FILSZ=E1000
RECORD TYPE=V
END
```

4. Code a JCL step that merges the header, sorted unload, and trailer file, as in this example:

```
//SORTIN DD DSN=IMSTESTS.HOSIX.UNLOAD.HEADER,DISP=(OLD,DELETE),
//          UNIT=SYSDA,VOL=SER=000000,SPACE=(CYL,(1,5))
//       DD DSN=IMSTESTS.HOSIX.UNLOAD.SORTED1,DISP=(OLD,DELETE),
//          UNIT=SYSDA,VOL=SER=000000,SPACE=(CYL,(1,5))
//       DD DSN=IMSTESTS.HOSIX.UNLOAD.TRAILER,DISP=(OLD,DELETE),
//SORTOUT DD DSN=IMSTESTS.HOSIX.UNLOAD.SORTED2,DSP=(NEW,KEEP),
            UNIT=SYSDA,VOL=SER=000000,SPACE=(CYL,(1,5))
//SYSOUT DD SYSOUT=A
//SYSIN  DD *
OPTION COPY
END
```

EXEC Statement

The EXEC statement must be in the form:

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

This statement is used for both HALDB and non-HALDB databases.

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 Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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.

SYSIN DD
Defines the input control statement data set only to HALDB and fallback. This data set can reside on a tape, or a direct-access device, or it can be routed through the input stream.

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 that 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: See the information on tailoring the IMS system to your environment in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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: See the information on specifying sequential buffering control statements in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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.

DFSWRKnn
Where nn is the number of secondary indexes (not less than 01). The DFSWRKnn data set is where the secondary index migration unload records are written.

DFSSRTnn
Where nn is the number of secondary indexes (not less than 01). The DFSSRTnn data set is the sort control statements that is used for a corresponding DFSWRKnn.

Note: The DFSSRTnn and DFSWRKnn data sets are allocated in the order of the secondary-index definition in the DBD. The utility does not attempt to relate a secondary index name to the suffix of the DFSSRTnn or DFSWRKnn DD statements or data sets. Secondary indexes use the next DFSSRTnn and DFSWRKnn data-set pair in ascending order.

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.

Utility Control Statements for DFSURGU0

Input statements are used to describe processing options for the HD Reorganization Unload utility. Input statements must conform to one of the following statements:

PARTITION=
Specifies the name of a HALDB partition (or the first of several sequential HALDB partitions) to unload.

NUMBER=
Specifies the number of sequential HALDB partitions to unload. The default setting is 1.

STAT=
Specifies whether to generate partition statistics. Partition statistics are generated if STAT=DET. Partition statistics are not generated if STAT=SUM. The default setting is STAT=DET.

This utility control statement is provided in a SYSIN stream.

MIGRATE=
Specifies whether to unload a non-HALDB database for migration. You must call a subroutine to build the alternate prefix when unloading a non-HALDB.

MIGRATX=

1. Eliminates the need to run multiple migration HDUNLOAD jobs for migration to PSINDEX by creating multiple work files from a single pass of the database for each indexed source segment. Corresponding sort control statements are generated for each work file. The sorted file is then usable as input to HDRELOAD to load the PSINDEX. DFSWRKnn and DFSSRTnn DD cards are required for each secondary index database referencing the primary database being unloaded.
2. A DD card is required for all of the related secondary index databases.

FALLBACK=
Specifies the unload of a HALDB database for fallback, creating fallback records.

Output Messages and Statistics for DFSURGU0

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 35.

```
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 35. Example of Output Messages and Statistics—HD Reorganization Unload Utility

| **Note:** For STAT=DET, there will be an output statistics report for each HALDB
| partition in addition to the database statistics shown above. Each partition statistics
| report shows the partition name and lists the same fields as the database statistics.

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 Version 9: Messages and Codes, Volume 1*.

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.

Return Codes for DFSURGU0

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 of DFSURGU0

All of the examples in this section use DBRC with dynamic allocation.

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 example demonstrated the unload of a single data set group HDAM or PHDAM database, no DD statements would be necessary for access to the database.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGU0,DI32DB02'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1210
//DFSUCKPT  DD DSN=IMS.CHKPT,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=222222,SPACE=(TRK,(50))
//DFSURGU1  DD DSN=IMS.UNLOAD1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSURGU2  DD DSN=IMS.UNLOAD2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=TAPE21,LABEL=(,SL)
//HDPAYROL  DD DSN=DATABASE.PAYROLL,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0001
//HDINDEXO  DD DSN=DATABASE.INDEXO,DISP=OLD,
//             UNIT=SYSDA,VOLUME=SER=DB0003
//DFSVSAMP  DD input for VSAM and OSAM buffers and options
//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.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSURGU0,DI32DB01',
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A DCB=BLKSIZE=1210
//DFSUCKPT  DD DSN=IMS.CHKPT,DISP=(OLD,KEEP),
//             UNIT=SYSDA,VOL=SER=222222
//DFSURSRT  DD DSN=IMS.CHKPT,DISP=(OLD,KEEP),
//             UNIT=SYSDA,VOL=SER=222222
//DFSURGU1  DD DSN=IMS.UNLOAD1,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=(,,,2,SER=(TAPE11,TAPE12)),
//             LABEL=(,SL)
//DFSURGU2  DD DSN=IMS.UNLOAD2,DISP=(MOD,KEEP),
//             UNIT=TAPE,VOL=(,,,2,SER=(TAPE21,TAPE22)),
//             LABEL=(,SL)
//HDPAYROL  DD DSN=DATABASE.PAYROLL,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=DB0001
//HDINDEXO  DD DSN=DATABASE.INDEXO,DISP=OLD,
//             UNIT=SYSDA,VOLUME=SER=DB0003
//DFSVSAMP  DD input for VSAM and OSAM buffers and options
//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.

Example 3

HDUNLOAD produces this report to aid in identifying and sorting the work files. Reference this report to determine which DFSWRKnn is the unload of a given secondary index when MIGRATX is used.

```
Work File Statistics
SINAME     WFNAME     SFNAME     RCDTOTAL   OFFSET        LENGTH
INDEX001   DFSWRK01   DFSSRT01   00000075       0069      0018
INDEX002   DFSWRK02   DFSSRT02   00000150   0069          0018
INDEX003   DFSWRK03   DFSSRT03   00000300       0069      0018
INDEX004   DFSWRK04   DFSSRT04   00000075   0069      0018
```

The JCL in Figure 36 on page 137 is used to unload a primary database and its two secondary index bases.

```
//HDUNLOAD EXEC PGM=DFSRRC00,
//         PARM=(ULU,DFSURGU0,DDPRIM01,9,0000,,0,,N,),,,,N,N,,N)
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSURGU1 DD DSN=HDAM2IX.UNLOAD,DISP=(SHR),
//         UNIT=SYSDA,SPACE=(CYL,(5,3)),DCB=BUFNO=5
//DDPRIM01 DD DSN=DDPRIM01,DISP=SHR
//DDINDEX1 DD DSN=INDEX001,DISP=SHR
//INDOVF01 DD DSN=INDOVF01,DISP=SHR
//DDINDEX2 DD DSN=INDEX002,DISP=SHR
//INDOVF02 DD DSN=INDOVF02,DISP=SHR
//DFSWRK01 DD DSN=DFSWRK01,DISP=(NEW,KEEP),VOL=SER=000000,
//         UNIT=SYSDA,SPACE=(CYL,(500,3)),DCB=BUFNO=5
//DFSSRT01 DD DSN=DFSSRT01,DISP=(NEW,KEEP),VOL=SER=000000,
//         UNIT=SYSDA,SPACE=(TRK,(1,0)),DCB=BUFNO=5
//DFSWRK02 DD DSN=DFSWRK02,DISP=(NEW,KEEP),VOL=SER=000000,
//         UNIT=SYSDA,SPACE=(CYL,(500,3)),DCB=BUFNO=5
//DFSSRT02 DD DSN=DFSSRT02,DISP=(NEW,KEEP),VOL=SER=000000,
//         UNIT=SYSDA,SPACE=(TRK,(1,0)),DCB=BUFNO=5
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//SYSIN DD *
MIGRATX=YES
```

Figure 36. JCL to Unload a Primary Database

The JCL in Figure 37 sorts the work file DFSWRK01 of the secondary index database INDEX01 using the sort control statements in DFSSRT01 from the HD Unload with the MIGRATX=YES option. The sorted output is passed to the HD Reload job step to load HALDB PSINDEX database PSNDX001.

```
//SORT01   EXEC PGM=SORT,PARM='CORE=MAX'
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTIN   DD DSN=DFSWRK01,DISP(OLD,PASS),VOL=SER=000000
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(100,5),,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE(CYL,(100,5),,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(100,5),,CONTIG)
//SYSIN    DD DSN=DFSSRT01,DISP=OLD,VOL=SER=000000
//SORTOUT  DD DSN=INDEX001.SORTED.UNLOAD,DISP=(,PASS),
//         UNIT=SYSDA,VOL=SER=000000,SPACE=(CYL,(500,3))
//HDRELOAD EXEC PGM=DFSRRC00,
//         PARM=(ULU,DFSURGL0,PSNDX001,9,0000,,0,,N,O,,,,N,N,,N)
//IMS      DD DSN=IMS.DBDLIB,DISP=SHR
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSUNIPT DD DSN=INDEX001.SORTED.UNLOAD,DISP=OLD
//DFSURWF1 DD DUMMY
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD input for VSAM and OSAM buffers and options
```

Figure 37. JCL to Sort the Work File of a Secondary Index Database

Chapter 15. HD Reorganization Reload Utility (DFSURGL0)

Use the HD Reorganization Reload utility to:

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

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

The information in Table 9 identifies inputs and outputs for the HD Reorganization Reload utility.

Table 9. Data Sets Used by the HD Reorganization Reload Utility

Input	Output
RECON	Output messages and statistics
Unloaded database	Index data sets
DBD library	Database data set
Control data set	Work data set

A flow diagram of the HD Reorganization Reload utility is shown in Figure 38 on page 140.

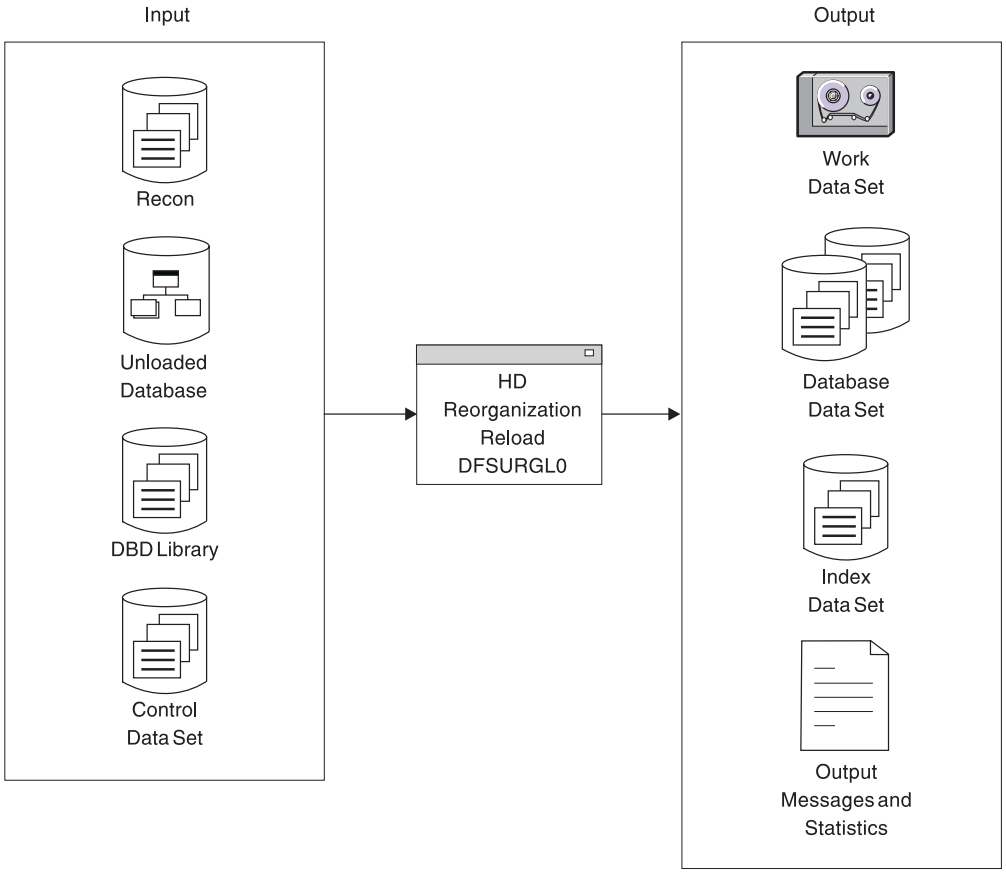


Figure 38. HD Reorganization Reload Utility

For HDAM and HIDAM databases, the functions of this utility can be performed by the Utility Control Facility (UCF). The UCF cannot be used for HALDBs

Related Reading: Refer to Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for a description of its operation.

The following topics provide additional information:

- “Restrictions for DFSURGL0”
- “JCL Requirements for DFSURGL0” on page 141
- “Output Messages and Statistics for DFSURGL0” on page 144
- “Return Codes for DFSURGL0” on page 145
- “Examples of DFSURGL0” on page 145

Restrictions for DFSURGL0

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

- If logical relationships or secondary indexes exist for non-HALDB databases, 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 Rules and Restrictions for DFSURGU0 in Chapter 14, “HD Reorganization Unload Utility (DFSURGU0),” on page 125.
- The Utility Control Facility is not supported for HALDB.
- If changing a DBD, you must do the following before executing the HD Reorganization Reload utility:

1. Assemble and link-edit the new DBD into the IMS 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 Version 9: Installation Volume 1: Installation 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.
- Concatenated unload files cannot be used as input files to the HD Reorganization Reload utility if they were created using MIGRATE, MIGRATX, or FALLBACK by the HD Reorganization Unload utility.

Related Reading: See the information on tuning databases in *IMS Version 9: Administration Guide: Database Manager* for further information on using the HD Reorganization Reload utility.

JCL Requirements for DFSURGL0

The HD Reorganization Reload utility is executed as a standard z/OS job. The following JCL statements 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 positional parameters such as SPIE, BUF, and DBRC can follow *dbdname*.

Related Reading: See the Member Name DBBBATCH or DLIBATCH in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD

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

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

IMS DD
Describes the library containing the DBD referenced in the EXEC statement PARM field. (Normally, this is IMS.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.

SYSIN DD
Defines the input control statement data set for HALDB Migration Reload statements NOILDS and ILDSMULTI. The data set can reside on a tape, or a direct-access device, or it can be routed through the input stream. This DD statement is not necessary if no utility control statements are provided as input to the utility.

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. For HALDB databases, two or more data sets can be concatenated to be reloaded by the HD Reorganization Reload utility. When using concatenated files, it is recommended that they be in partition selection order. Otherwise, unpredictable results could occur.

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. The data set ddname can be specified as DUMMY if the database being reloaded is not involved in a logical relationship or with a secondary index, or the database is PHIDAM or PHDAM. Prefix resolution is not required for PHIDAM or PHDAM databases.

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 that represent the index. The DD statements that 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: See the information on specifying the IMS buffer pool in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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: See the information on specifying sequential buffering control statements in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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. If either statement is supplied, any return code greater than 4 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.

Utility Control Statements for DFSURGL0

Input statements are used to describe processing options for the HD Reorganization Reload utility. Input statements must be one of the following two statements:

- NOILDS disables ILDS processing during a migration reload.
- ILDSMULTI enables multi-task ILDS processing.

Output Messages and Statistics for DFSURGL0

The HD Reorganization Reload utility provides output messages and statistics. Figure 39 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 39. Example of Output Statistics from the HD Reorganization Reload Utility

If the HD Reorganization Unload utility was run using STAT=DET in the SYSIN control statement, there will be one output statistics report for each HALDB partition as well as one for the entire database. In the case of concatenated input data sets to the HD Reorganization Reload utility, you will see output statistics from each input file. The detailed reports from the HD Reorganization Reload utility reflect the unloaded data, not the reloaded data. They do not specify partition names due to possible key range changes, consolidation, or expansion as a result of running 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 for DFSURGL0

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 or error encountered during ILDS processing
16	Database reload did not complete successfully

Examples of DFSURGL0

For the examples in this section, if you are using DBRC without dynamic allocation, you must add the DD statements shown in Figure 40 to the sample JCL:

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

Figure 40. DD Statements for using DBRC without Dynamic Allocation

Example 1

This example shows the JCL for an HDAM reorganization reload.

```
//STEP1    EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,DH32DB01',
//STEPLIB    DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB   DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS        DD DSN=IMS.DBDLIB,DISP=OLD
//SYSPRINT   DD SYSOUT=A
//DFSUINPT   DD DSN=IMS.UNLOAD1,DISP=OLD,
//              UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSURWF1   DD DSN=IMS.WRKTAPE1,DISP=(NEW,KEEP),
//              UNIT=TAPE,VOL=SER=WKTAPE,LABEL=(,SL),
//              DCB=(BLKSIZE=1008,LRECL=900,RECFM=VB)
//HDSKILLS   DD DSN=DATABASE.SKILLS,DISP=(NEW,KEEP),
//              UNIT=SYSDA,VOL=SER=DB0002,SPACE=(CYL,(10,10))
//DFSURCDS   DD DSN=IMS.RLCDS,DISP=(OLD,KEEP),
//              UNIT=SYSDA,VOL=SER=IMSMSC
//DFSVSAMP   DD input for VSAM and OSAM buffers and options
//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.

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

Example 3

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

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

```
//DFSUINPT  DD DSN=UNLOAD,DISP=(OLD,PASS),UNIT=SYSDA
//PRINTDD   DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//DHSK0301  DD DSN=VVDH0301,DISP=OLD
//DXSK0302  DD DSN=VVDX0302,DISP=OLD
//DXSK0301  DD DSN=VVDX0301,DISP=OLD
//DFSVSAMP  DD *
2048,10
//DFSCTL    DD *
SBPARM ACTIV=COND
/*
```

Example 4

This example shows the JCL for a HIDAM HALDB migration reload with the NOILDS option. The new HALDB has two partitions.

```
//STEP1  EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,PHIDMSTR',
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD DSN=IMS.DBDLIB,DISP=OLD
//SYSPRINT DD SYSOUT=A
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSUINPT DD DSN=IMS.UNLOAD1,DISP=OLD,
// UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//DFSCTL   DD *
SBPARM ACTIV=COND
//SYSIN    DD *
NOILDS
//STEP02 EXEC PGM=DFSRRC00,REGION=1300K,
//       PARM='ULU,DFSPREC0,PHIDMSTR,,,,,,,,,,Y,N'
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSNAME=IMS.SDFSRESL,DISP=SHR
//IMS      DD DSN=IMS.DBDLIB,DISP=OLD
//SYSPRINT DD SYSOUT=A
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//SYSIN    DD *
 PARTITION=PDHIDMA,RECOVTYP=ILE
//STEP03 EXEC PGM=DFSRRC00,REGION=1300K,
//       PARM='ULU,DFSPREC0,PHIDMSTR,,,,,,,,,,Y,N'
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD DSN=IMS.DBDLIB,DISP=OLD
//SYSPRINT DD SYSOUT=A
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//SYSIN    DD *
PARTITION=PDHIDMB,RECOVTYP=ILE
```

Example 5

This example shows the JCL for a HIDAM HALDB migration reload with the ILDSMULTI option.

```
//STEP1  EXEC PGM=DFSRRC00,PARM='ULU,DFSURGL0,PHIDMSTR',
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD DSN=IMS.DBDLIB,DISP=OLD
//SYSPRINT DD SYSOUT=A
```

```
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
//DFSUINPT DD DSN=IMS.UNLOAD1,DISP=OLD,
// UNIT=TAPE,VOL=SER=TAPE11,LABEL=(,SL)
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//DFSCTL   DD *
SBPARM ACTIV=COND
//SYSIN    DD *
ILDSMULTI
```

Chapter 16. 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: See Chapter 3, “Database Surveyor Utility (DFSPRSUR),” on page 19 for more information and samples of output.

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 for Partial Database Reorganization” on page 150.

The following topics provide additional information:

Restrictions for Partial Database Reorganization

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

- Structural changes to the database are not allowed.

- 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 for Partial Database Reorganization

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 of Partial Database Reorganization” on page 158.

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 160.) 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 z/OS 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: See Chapter 26, “Batch Backout Utility (DFSBBO00),” on page 267 for more information.

JCL Requirements for Partial Database Reorganization

The Partial Database Reorganization utility is executed as two standard z/OS jobs in an IMS batch processing region. The following JCL statements 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 152 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,PARM=(IMSPLEX=plexname)
```

The EXEC statement is standard except for the addition of the IMSPLEX parameter. For detailed information about the IMSPLEX parameter, see *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference* under initializing and maintaining RECONs.

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.SDFSRESL, 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 151 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 Version 9: 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.SDFSRESL, which contains the IMS nucleus and required action

modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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.

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: See “Specifying the IMS Buffer Pool” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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

Related Reading: See "SB Parameters (SBPARM) Control Statement" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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.

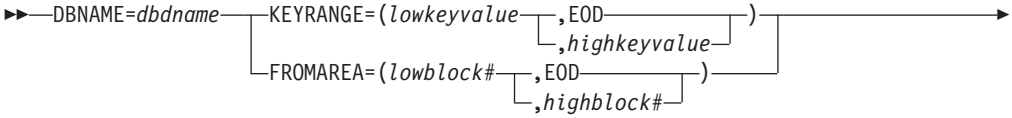
Do not use these RECON data set ddnames if you are using dynamic allocation.

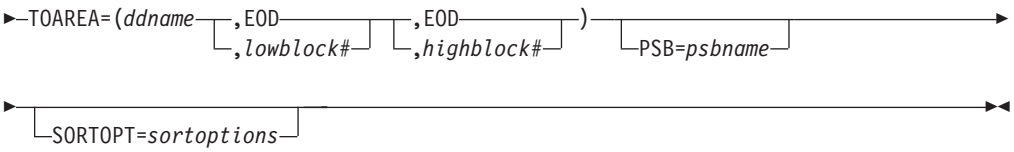
Utility Control Statements for Partial Database Reorganization

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





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 of 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.

KEYRANGE is invalid if the database is HDAM.

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

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

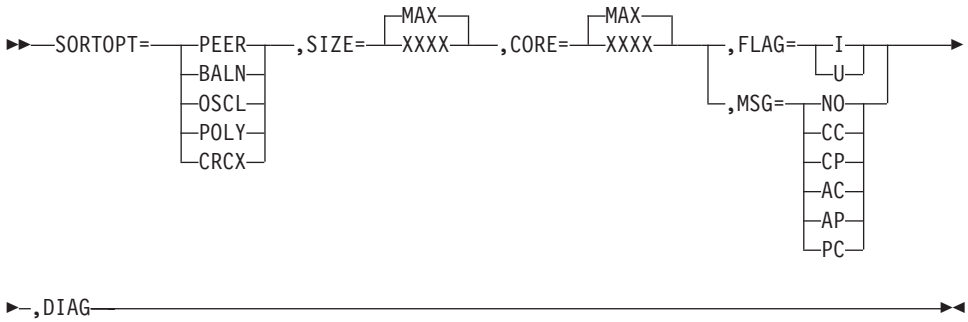
execution of Partial Database Reorganization, and a new PSB is not desired. This keyword can also be omitted if you provide a PSB that exactly follows the PSB example at the end of this section.

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.

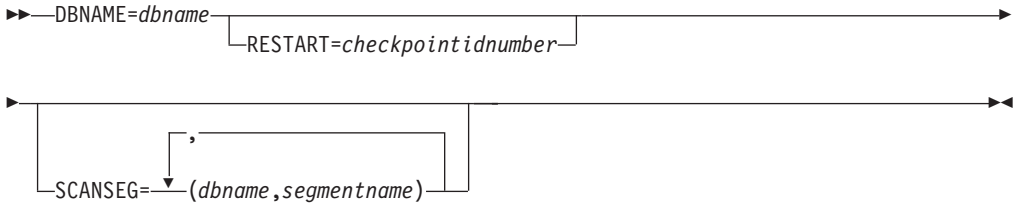
Only those sort options that you want to use are entered in the operand string. No extra commas are needed for omitted SORTOPT operands. Default options for sorts in Partial Database Reorganization are the normal z/OS sort option defaults.

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



Related Reading: See *DFSORT Application Programming Guide* for a description of all the sort options that can be used with this utility.

Step 2 Utility Control Statement



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 for Partial Database Reorganization

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: See *IMS Version 9: Messages and Codes, Volume 1* for a detailed explanation of the numbered messages.

Examples of Partial Database Reorganization

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 DD statements shown in Figure 41 to the sample JCL:

```
//RECON1   DD DSN=IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMS.RECON2,DISP=SHR
//RECON3   DD DSN=IMS.RECON3,DISP=SHR
```

Figure 41. DD Statements for Using DBRC without Dynamic Allocation

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'
//PRINTDD  DD  SYSOUT=A
//PR01DD   DD  DSN=PR01RW00,DISP=SHR
//PR01IDD  DD  DSN=PR01I,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//IEFRDER  DD  DUMMY
//DFSVSAMP DD  input for VSAM and OSAM buffers and options
//SYSIN    DD  DSN=IMS.DBTDATA(PR01DT),DISP=SHR
//PTSTE17  EXEC  PGM=DFSPRCT1,COND=EVEN
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.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 42 contains the input statements.

```
IMS PARTIAL REORGANIZATION - STEP 1 INPUT STATEMENTS          137/89     PAGE   1
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
     DBNAME=PR07RW12
     KEYRANGE=(000100,000200)
     TOAREA=(PR07DD,005,007)
     PSB=PTSTN07
```

Figure 42. Partial Database Reorganization Step 1 Input Statements

Figure 43 contains the range values.

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

Figure 43. Partial Database Reorganization Range Values

Figure 44 contains the names of segments for required scanning.

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

                NO SEGMENT FOUND FOR REQUIRED SCAN
```

Figure 44. Partial Database Reorganization Required Segment Scan

Figure 45 contains the names of segments for optional scanning.

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

              NO SEGMENT FOUND FOR OPTIONAL SCAN
```

Figure 45. Partial Database Reorganization Optional Segment Scan

PSB Example

Figure 46 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
```

Figure 46. PSB Source Statements

DBD Examples

Figure 47 shows the DBDs that 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
```

Figure 47. DBDs needed to execute Step 2

Step 2 Example

Figure 48 on page 161 shows the JCL and utility control statements required to run Step 2.

```
//STEP2   EXEC PGM=DFSRRC00
//            PARM=(DLI,DFSPRCT2,PTSTN07)
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,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 DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYC,1)
//IMS       DD DSN=IMS.PSBLIB,DISP=SHR
//          DD DSN=IMS.DBDLIB,DISP=SHR
//IEFRDER   DD DSN=IMS.LOG1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=222222
//SYSPRINT  DD SYSOUT=A,DCB=(BLKSIZE=1210)
//DFSPRCOM  DD DSN=*.PTSTN07.DFSPRCOM,DISP=(OLD,KEEP)
//SYSUDUMP  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//PR07DD    DD DSN=PR07RW12,DISP=OLD
//PR07RDD   DD DSN=PR07R,DISP=OLD
//PR07RIDD  DD DSN=PR07RI,DISP=OLD
//PR07IDD   DD DSN=PR07I,DISP=OLD
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN     DD *
   DBNAME=PR07RW12
//DFSVSAMP  DD *
1024,4
4096,8
//DFSCTL    DD *
SBPARM ACTIV=COND
/*
```

Figure 48. JCL and Utility Control Statements required for Step 2

The following are sample output reports from Step 2. Figure 49 contains the input statements.

```
IMS PARTIAL REORGANIZATION - STEP 2 INPUT STATEMENTS 137/89 PAGE 1
0........1.........2.........3.........4.........5.........6.........7.........8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
     DBNAME=PR07RW12
```

Figure 49. Partial Database Reorganization Step 2 Input Statements

Figure 50 on page 162, Figure 51 on page 163, and Figure 52 on page 163 contain unload statistics.

```
IMS 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 50. 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 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 51. 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 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 52. 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 53 and Figure 54 on page 165 show reload statistics.

```
          IMS 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 53. Partial Database Reorganization—Reload Statistics

The fields in Figure 53 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

Figure 54 on page 165 is a sample report for a range of reloaded segments.

```
        IMS 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 54. 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 55 shows scan statistics.

```
          IMS 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 55. 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 17. 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.

The information in Table 10 identifies inputs and outputs for the Database Prereorganization utility.

Table 10. Inputs and Outputs used by the Database Prereorganization Utility

Input	Output
RECON	Output messages
DBD library	Scan control list
Input control statements	Control data set

A flow diagram of the Database Prereorganization utility is shown in Figure 56.

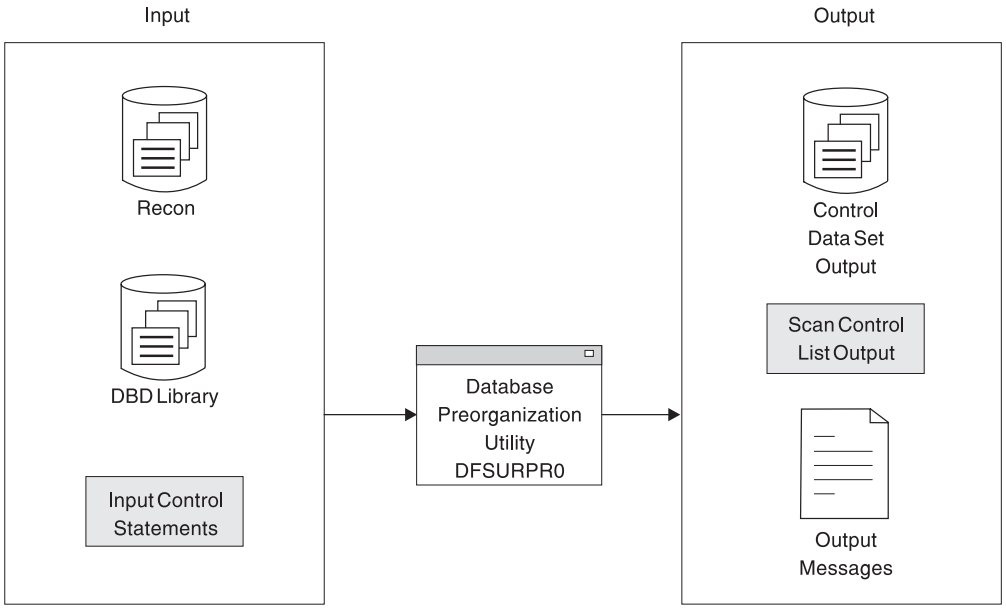


Figure 56. Database Prereorganization Utility

The following topics provide additional information:

- “Initializing One or More Partitions of a HALDB” on page 168
- “Restrictions for DFSURPR0” on page 168
- “Input and Output for DFSURPR0” on page 168
- “JCL Requirements for DFSURPR0” on page 169
- “Utility Control Statements for DFSURPR0” on page 170
- “Output Messages and Statistics for DFSURPR0” on page 173
- “Return Codes for DFSURPR0” on page 173
- “Examples of DFSURPR0” on page 173

Initializing One or More Partitions of a HALDB

The Database Prereorganization utility can be used to initialize one or more partitions of a HALDB that have been defined or changed using the HALDB Partition Definition utility. Once the HALDB Partition Definition utility has been used to create or change a HALDB partition definition, a record is placed in RECON indicating that the HALDB partitions require initialization. The Database Prereorganization utility is then run to create a Control Data Set (CDS) indicating that prefix resolution and update are not required, and initializes all data sets recorded in RECON as requiring initialization for HALDB DBD. If individual partitions are deleted and redefined, it is necessary to initialize them again. In order for initializations to take place for individual partitions, they must be recorded in RECON as PINIT (initialization required). In this case, it is necessary to use the CHANGE.DB command with the partition name for each partition prior to running prereorganization. For example `CHANGE.DB DBD` , where DBD is the partname.

Requirements

- Make RECON data sets available by one of two methods:
  - Coding JCL DD statements
  - Using dynamic allocation
- Ensure that buffer pool definitions in the DFSVSAMP DD card are large enough to permit DL/I to open all of the HALDB partition data sets for the HALDB.

To create or change a HALDB partition definition you must:

1. Define all HALDB partition data sets with the `REUSE` option on the `VSAM DEFINE CLUSTER` statement, as shown in Figure 57 on page 174.
2. Run the Database Prereorganization utility for an initial load, or reorganization for a reload. An example of a prereorganization is shown in Figure 59 on page 174.

Restrictions for DFSURPR0

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 are being loaded or reorganized, and when both databases contain logical relationships.

Note: This restriction does not apply to HALDB.

Input and Output for DFSURPR0

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. For HALDB, this utility's input must also include one or more un-initialized partitions. 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.

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. Additionally, the output can be one or more initialized HALDB partitions.

JCL Requirements for DFSURPR0

The Database Prereorganization utility is executed as a standard z/OS 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,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 Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

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

Related Reading: See the information on specifying the IMS buffer pool in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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 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.

For HALDBs these utilities are disabled in the CDS.

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.

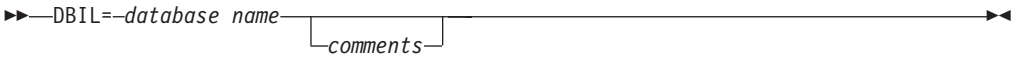
Do not use these RECON data set ddnames if you are using dynamic allocation.

Utility Control Statements for DFSURPR0

There are four 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
DBM	Disables DB scan bit in CDS for migration of non-HALDB to HALDB.
OPTIONS	Specifies any optional information to be provided during reorganization or initial load of the database

DBIL Statement



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 or PHIDAM database is to be initially loaded, list only its DBD name on a DBIL control statement. Do not list the HIDAM or PHIDAM primary index or any secondary index DBD names.

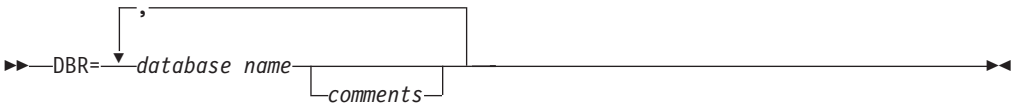
When structural changes affecting logical relationships are made to a database during a database reorganization process, the following must be performed prior to the HD reload step:

- 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.

  Related Reading: See the information on tuning and modifying databases in *IMS Version 9: Administration Guide: Database Manager* for information that can help you determine whether you need to specify DBIL for other logically related databases. *IMS Version 9: 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



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.

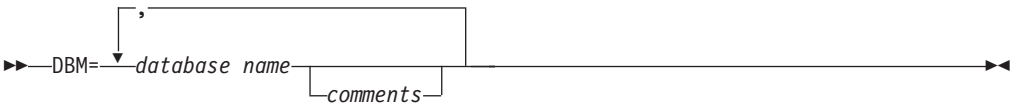
If a HISAM database is to be reorganized using the HISAM Reorganization Unload/Reload utilities, the HISAM DBD name must not be listed on a DBR control

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 statement.

If a HIDAM or PHIDAM 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. For HALDB prereorganization is not necessary unless it is required to initialize partitions.

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

DBM Statement



This utility control statement disables DB scan bit in CDS for migration of non-HALDB to HALDB.

OPTIONS Statement

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 using 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 for DFSURPR0

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.
- 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 for DFSURPR0

The following return codes are provided at program termination:

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

Examples of DFSURPR0

DBM is used for migration unload of non-HALDBs and DBIL is used to initialize HALDB partitions that have been recorded as partition initialization required either by %DFSHALDB or by the DBRC CHANGE.DB command.

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.

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

The example in Figure 57 on page 174 specifies `REUSE` for data sets containing user data.

```
//DBHDJ05  EXEC PGM=IDCAMS
//SYSIN    DD   *
  DEFINE   CLUSTER (NAME (IMSTESTS.DBVHDJ05.A00001) -
                 TRK(3,1) RECSZ (1017.1017) -
                 VOL (DSHR00) SHAREOPTIONS (3,3) -
                 CISZ (1024) NIXD) REUSE
/*
```

Figure 57. Defining a HALDB Partition Data Set

The example in Figure 58 notifies DBRC that a HALDB partition requires initialization. Use the command in the example before running the Database Prereorganzation utility. For more information about initialization of HALDB partition, see *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference*.

```
//UPDRECN1 EXEC PGM=DSPURX00
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD*
CHANGE.DB DBD(DBVHDJ05) PINT
```

Figure 58. Notifying DBRC About Initialization Requirement of a HALDB Partition

The example in Figure 59 shows the job control statements required to define a HALDB partition data set with the REUSE option on the VSAM CLUSTER Definition.

```
//PRERSTEP EXEC PGM=DFSRRC00,
//                PARM=(ULU,DFSURPR0,,,,,,,,,,,Y,N,,N)
//*               PREREORGANIZATION-UTILITY
//STEPLIB  DD   DSN=IMS.SDFSRESL,DISP=SHR
//DFSVSAMP DD   DSN=IMS.DFSVSAMP(VSM885FP),DISP=SHR
//DFSRESLB DD   DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD   DSN=IMS.DBDLIB,DISP=SHR
//RECON1   DD   DSN=IMS.RECON1,DISP=SHR
//RECON2   DD   DSN=IMS.RECON2,DISP=SHR
//RECON3   DD   DSN=IMS.RECON3,DISP=SHR
//SYSUDUMP DD   SYSOUT=A
//SYSPRINT DD   SYSOUT=A
//PRINTDD  DD   SYSOUT=A
//DFSURCDS DD   DSN=PINIT.CDS,UNIT=SYSDA,DISP=(,PASS),
//              SPACE=(TRK,1),DCB=BLKSIZE=1600
//SYSIN    DD   *
DBR=DBVHDJ05
/*
```

Figure 59. Prereorganization of a HALDB

Chapter 18. 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.

The following topics provide additional information:

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 might 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 might have to adjust for your environment.

Related Reading: See “Command Descriptions” on page 540 for more information on how to use the BUFNO command, and how to use it for the other DEDB utilities.

Restrictions for DBFUHDR0

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 for DBFUHDR0

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 539 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 for DBFUHDR0

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 z/OS 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.

For the Reorganization utility, no difference in processing exists between `REST=00` and `REST=01`.

Related Reading: See "FPUTIL Procedure" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of the `REST=` parameter.

Segment Shunting

Segment shunting is the capability to reorganize specified segments directly into DOVF or IOVF, bypassing copying these segments into the RAP CI even though space may currently exist in the RAP CI. This allows you to potentially reorganize the DEDB UOW and retain space in the RAP CI for new inserts.

You can specify which segments are to be shunted using input data set, INDD. This data set must be a fixed block with an logical record length (LRECL) of 80, a block length of 80, and a one input card per record. The input card must be left justified or the card will be ignored. This data set is only required for those interested in the segment shunting function.

The only segments that can be shunted are direct dependent segments. Root segments and SDEPs can not be shunted.

The INDD data set consists of three input card types:

ERRACTN=aaaa
Is an optional input card type. If included, it must be the first card in the INDD data set. The valid options for this card are ERRACTN=CONT and ERRACTN=EXIT.

ERRACTN=CONT is the default option. If an error is found in the remainder of the INDD data set, the invalid card is ignored and the remainder of the input cards are accepted.

When ERRACTN=EXIT is specified, if any cards in the INDD data set contain invalid data, the entire data set is ignored and the reorganization job proceeds as if the INDD data set was not specified.

AREA=areaname
Starts the input stream for a single areas. The input stream for this area terminates at the next AREA= card, or at the end of the file. All of the following cards in the input stream are considered segments to shunt for the associated AREA= card.

segmentn
Contain the segment names associated with the prior AREA= card.

The following example shows a DEDB containing 12 areas, a root segment ROO1, and direct dependents DD1, DD2, DD3, DD4, and DD5.

```
ERRACTN=CONT
AREA=AREA_1
DD1
DD2
DD3
AREA=AREA_10
DD4
DD2012
AREA=AREA_11
DD1
DD4
```

In this example:

- All invalid cards will be ignored because ERRACTN=CONT is specified.
- For AREA_1, segments DD1, DD2, and DD3 will not be reorganized into the RAP CI, but ROOT, DD4, and DD5 can be inserted into the RAP is space exists.
- For AREA_2 through AREA_9 and AREA_12, the areas will be reorganized as normal without segment shunting.
- For AREA_10, segment DD4 will be shunted and card DD2012 will be ignored.
- For AREA_11, segments DD1 and DD4 will both be shunted.

JCL Requirements for DBFUHDR0

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 "FPUTIL Procedure" in *IMS Version 9: 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.

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: See Appendix A, "Summary of DEDB Utility Commands," on page 539 for a description of how to write the control commands and operands.

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 might 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.

INDD DD
Points to the optional data set used for segment shunting. This data set must be fixed block, LRECL=BLOCKSIZE=80.

Error Processing for DBFUHDR0

Table 11 summarizes the I/O error handling by the Reorganization utility and the area status after an I/O error.

Table 11. Summary of I/O Error Handling by the 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 of UOWs
8	Error in parameter analysis or errors occurred; see utility output

Error messages accompany all nonzero return codes.

Example of DBFUHDR0

Figure 60 on page 181 shows the JCL and utility control statements for executing the Reorganization utility.

```
//ORGDB01 EXEC FPUTIL,
//        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
```

Figure 60. JCL and Utility Control Statements for Executing the High-Speed DEDB Direct Reorganization utility

Figure 61 on page 182 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
```

Figure 61. Output for the High-Speed DEDB Direct Reorganization utility

Chapter 19. MSDB-to-DEDB Conversion Utility (DBFUCDB0)

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 a z/OS batch region.

The following topics provide additional information:

Conversion for DBFUCDB0

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 unloaded format.

Conversion Process

Conversion requires the use of the MSDB Dump Recovery utility and either the Unload/Reload utilities in the IMS Fast Path Basic Tools for z/OS 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 unload format for each MSDB that is being converted.
5. Sort the data set in unload format for each MSDB by ascending order of the RAP RBA values.
6. Run the IMS Fast Path 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: See “Choosing Fast Path Database Types” in *IMS Version 9: Administration Guide: Database Manager* for more information on MSDB-to-DEDB conversion.

Fallback for DBFUCDB0

Fallback requires use of the MSDB Maintenance utility and a DEDB unload/reload utility (such as the unload/reload utility in the IMS Fast Path Basic Tools for z/OS).

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. Sort the MSDBLCHG data set by the root key. The format of the MSDBLCHG data set record format is identical to that of the MSDBINIT data set described in the main storage database section of the *IMS Version 9: Administration Guide: Database Manager*. The SORT control statements needed depend on the length and data type of the root key. Sort the data set in ascending order.
4. 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 for DBFUCDB0

EXEC Statement

Executes the MSDB-to-DEDB Conversion utility. This statement must be in the form:

```
//STEPNAME EXEC PGM=DBFUCDB0
```

DD Statements

STEPLIB DD

Defines IMS.SDFSRESL, 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 reload processor (part of the IMS Fast Path Basic Tools for z/OS).

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 DBFUCDB0" 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 for DBFUCDB0

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 for DBFUCDB0

This utility generates a summary report when it completes execution. The report is sent to the SYSPRINT data set.Figure 62 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 62. MSDB-to-DEDB-Conversion-Utility-Summary Report

Using Other Unload/Reload Utilities for DBFUCDB0

The MSDB-to-DEDB conversion utility uses the unload/reload utility in the IMS Fast Path Basic Tools for z/OS. 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 unload/reload 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 unload/reload 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 the unload/reload format.

Requirement: 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 the unload/reload format 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 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 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 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 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 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 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 for DBFUCDB0

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: See *IMS Version 9: Messages and Codes, Volume 1* for an explanation of the messages generated by this utility.

Examples of DBFUCDB0

The following examples show the JCL necessary to run this utility for both conversion and fallback.

Example 1 (Conversion)

Figure 63 on page 190 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 can be deleted.

The DBT.FORMAT.SAVE1 data set contains the output from the sort and is now usable by the Reload utility (part of the IMS Fast Path Basic Tools for z/OS).

The output data set DBT.FORMAT.SAVE1 will be used as input to the Reload utility (part of the IMS Fast Path Basic Tools for z/OS) 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.SDFSRESL,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.SDFSRESL,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
/*
//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 63. JCL for Converting an MSDB to a DEDB

Example 2 (Fallback)

Figure 64 on page 192 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 Unload utility (part of the IMS Fast Path Basic Tools for z/OS).

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.SDFSRESL,DISP=SHR
//OLDACB   DD    DSN=MSDB.ACBLIB,DISP=SHR
//NEWACB   DD    DSN=DEDB.ACBLIB,DISP=SHR
//SAVE1    DD    DSN=SAVE1,DISP=OLD
//DURD001O DD    DSN=DBT.SEGMENTS.SAVE1,DISP=(,CATLG,DELETE)
//DURDBDFN DD
//DURS001O 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.SDFSRESL,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)
//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.SDFSRESL,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 64. JCL for Converting a DEDB Back to an MSDB

Part 3. Backup Utilities

Chapter 20. 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: See Chapter 21, “Database Image Copy 2 Utility (DFSUDMT0),” on page 207 or Chapter 22, “Online Database Image Copy Utility (DFSUICP0),” on page 223 for other ways to create a copy of an online database.

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: See *IMS Version 9: Operations Guide* for more information on using the Database Image Copy utility.

The information in Table 12 identifies inputs and outputs for the Database Image Copy utility.

Table 12. Input to and output from the Database Image Copy Utility

Input	Output
RECON	SYSPRINT messages and statistics
DBD library	Copy output
IMS.DBHI3A	
IMS.DBHI3B	
Input control statements	

Figure 65 on page 196 is a flow diagram of the Database Image Copy utility.

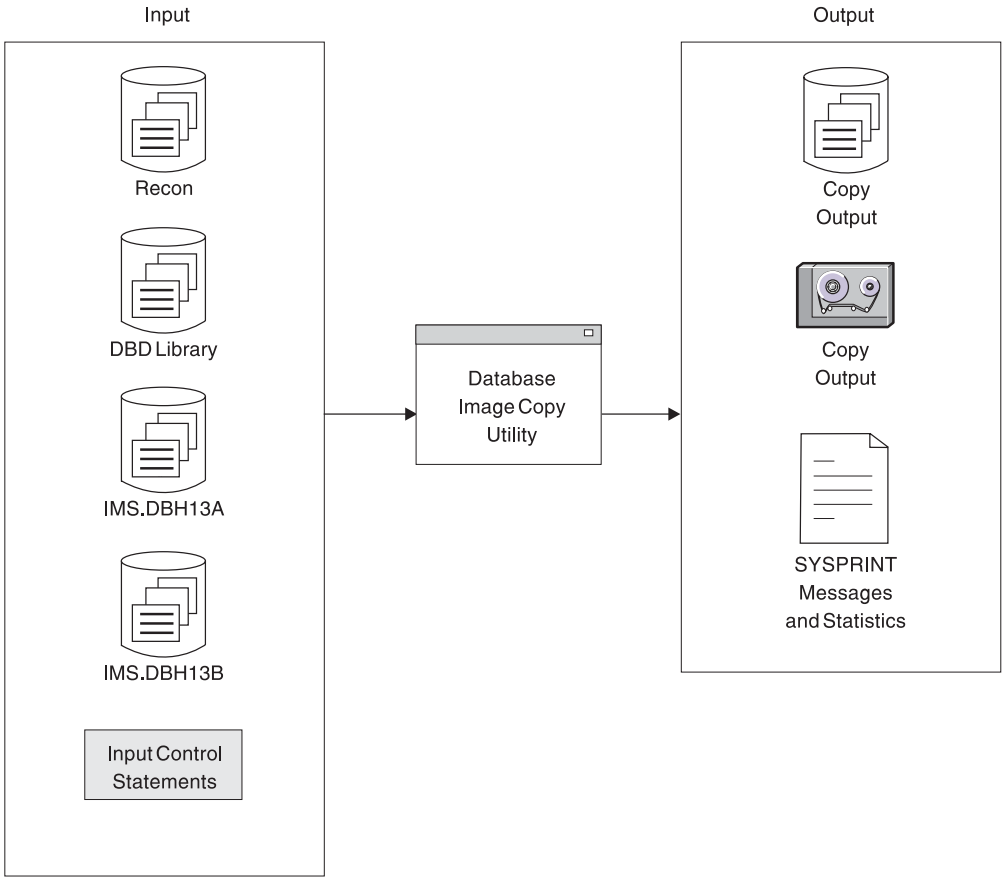


Figure 65. Database Image Copy Utility

The functions of this utility can be performed under control of the Utility control Facility except when creating image copies for HALDBs.

Related Reading: Refer to Chapter 34, “Utility Control Facility (DFSUCF00),” on page 341 for a description of its operation.

The following topics provide additional information:

- “Using the Database Image Copy Utility in an RSR Environment”
- “Restrictions for DFSUDMP0” on page 197
- “Input and Output for DFSUDMP0” on page 198
- “JCL Requirements for DFSUDMP0” on page 199
- “Utility Control Statement for DFSUDMP0” on page 202
- “Return Codes for DFSUDMP0” on page 203
- “Examples of DFSUDMP0” on page 203

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.

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 for DFSUDMP0

The following restrictions apply when using the Database Image Copy utility:

- HALDB requires DBRC to be active or copy request is rejected.
- HSAM, GSAM, and MSDB databases cannot be copied with 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.
- Concurrent image copy (fuzzy image copy) is not permitted for nonrecoverable DEDBs.
- 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 prohibit the Image Copy execution.
- The utility cannot be restarted.
- When running an image copy utility with HALDB Online Reorganization, additional restrictions apply. See *IMS Version 9: Administration Guide: Database Manager* for more information.
- The large block interface (LBI) is supported only for tape data sets on LBI capable tape devices. Database Image Copy supports system determined block size for output data sets on DASD.

Input and Output for DFSUDMP0

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 copies of the output image. 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 or delete one or more areas, image copies must be taken for all areas that follow the inserted 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.

If you want to copy block sizes greater than 32760 bytes, the large block interface is required on:

- 3480 magnetic tape
- 3490 and 3490E magnetic tape subsystems
- 3590 devices

Recommendation: Use system-determined block size.

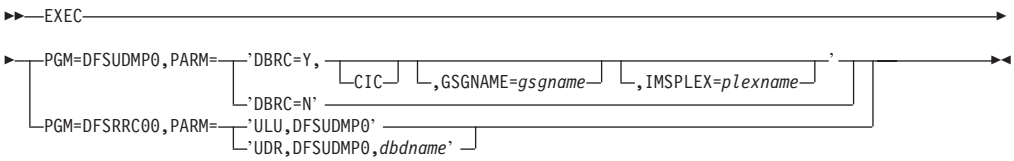
JCL Requirements for DFSUDMP0

The Database Image Copy utility is executed as a standard z/OS 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

This statement can either invoke a cataloged procedure containing the required statements or be in the following format:



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.

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 for Full Function databases. For Fast Path databases, CIC can be specified with DBRC=N. The resulting image copy data set can be used to run programs such as DEDB Pointer Checker (FABADA1). It is not to be used for database recovery.

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 definition using the DATABASE macro.

An EXEC parameter, IMSPLEX, can be specified on all job steps that use DBRC. The IMSPLEX= parameter specifies which IMSplex DBRC should join.

Related Reading: For more information on the IMSPLEX= parameter, see *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference* under initializing and maintaining RECONs.

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.
- CIC cannot be executed during the Online Reorganization of a HALDB partition.

Related Reading: See "Registering Databases and Database Data Sets" in *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference* for information on the `INIT.DB` command.

For information on changing the access level of a database, see *IMS Version 9: 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 196 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 Version 9: 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 re-linked 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.SDFSRESL. If executed as an IMS region, the old values within the IMS nucleus are used.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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. For a HALDB data set, this DD statement can be omitted. Otherwise, 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 z/OS I/O generation. For 3380s, the maximum BLKSIZE is 23 KB; if the logical record exceeds 23 KB, the maximum is 32 KB.

Exception: The following devices support block sizes greater than 32760 bytes:

- 3480 magnetic tape
- 3490 and 3490E magnetic tape subsystems
- 3590 devices

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 for DFSUDMP0

The utility control statement for the Database Image Copy utility has a fixed format using the positions described as follows:

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. For a HALDB data set, the DD statement can be omitted.

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 for DFSUDMP0

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 of DFSUDMP0

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 DD statements shown in Figure 66 on page 204 to the sample JCL:

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

Figure 66. DD Statements for Using DBRC without Dynamic Allocation

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.

```
//*
//STEP1   EXEC PGM=DFSUDMP0
//STEPLIB   DD DSNAME=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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.

```
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'
//STEPLIB   DD DSNAME=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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.

```
//*
//STEP1   EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'
//STEPLIB   DD DSNAME=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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 21. 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 *z/OS DFSMSdss™ Storage Administration Guide* or *z/OS DFSMSdss Storage Administration Reference*.

Once the concurrent copy session has been established, the copy is said to be logically complete. After the concurrent copy initialization, updates can 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.

When the image copy is physically complete, it is registered with DBRC as either a SMSNOCIC or SMSCIC image copy. Depending upon the size of the data set, the physical copy might 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: See “Making Database Backup Copies” in *IMS Version 9: Operations Guide* for more information on using the Database Image Copy 2 utility.

The following topics provide additional information:

Multiple Database Data Set Input

The Database Image Copy 2 utility can copy multiple DBDSs to be copied in one execution of the utility. Multiple control statements can be specified, one per database data set to be copied in a single execution.

With Database Image Copy 2 all of the DBDSs specified on the control statements are passed to DFSMSdss on a single invocation. This allows DFSMSdss to start multiple dump processes in parallel and logical completion can be achieved for all of the DBDSs in a very brief period of time.

Restriction: Copying multiple DBDSs in parallel to output data sets that are on the same tape volume is not supported. You can avoid this limitation by using the IBM Virtual Tape Server which implements virtual tape volumes. As an alternative, you can use the Same Data Set option.

The number of DFSMSdss dump tasks that can process in parallel depends on the availability of resources. Copying multiple data sets in a single execution requires more of certain resources (virtual storage, tape drives) than copying a single DBDS. If required resources are not available, some of the DFSMSdss tasks might be delayed until other tasks have ended and required resources have become available.

A group name can be specified to identify the set of DBDSs that are being copied as a group. When a group name is not specified, different processing options can be specified for each DBDS that is to be copied in the same utility execution. This allows clean, or consistent, image copies to be created for some DBDSs while fuzzy copies are created for others.

When a fuzzy copy of a KSDS is requested and other DBDSs are being copied in the same utility execution, the likelihood that the image copy process will fail with a DFS3145A message is greater than if the KSDS were being copied by itself. When the KSDS is the only data set being copied, the utility retries the DFSMSdss DUMP command several times before failing with the DFS3145A error. There is, however, no retry, if other DBDSs were being copied in the same execution.

Specifying Group Names

A group name can be specified to represent the collection of DBDSs that are to be copied in a single execution. The group name is specified on the group name control statement. If specified, the status for the group name is reported in the message that indicates logical completion or physical completion of clean image copies. (See “Image Copy Completion Notification” on page 209.) This capability allows the operator to monitor clean image copy processing on a group basis.

The group name can be the name of a DBRC group (a DB group, a DBDS group, or a CA group). The utility does not verify that a group with the name exists in the RECON or that the members of the group, if one exists, were specified to be copied by the utility. However, using a DB or DBDS group name on the group name statement for the utility can simplify operations.

The group name statement is followed by control statements identifying the DBDSs that are (or that are to be treated as) members of the group. All of the data sets in a DB or DBDS group can be included in a single named group and thus copied in a single Database Image Copy 2 execution. When clean image copies are taken, a /DBR DATAGROUP command can be used to stop the databases/areas in any DB

or DBDS group. A /START DATAGROUP command can then be used to restart the databases/areas when the logical or physical completion notification for the group has been given.

If a group statement is specified, processing options for the members of the group are specified at the group level. The options specified or defaulted for the group override the options specified individually on the DBDS statements in the group.

Single Output Data Set for Multiple Image Copies

DFSMSdss provides the capability to dump multiple data sets into one output data set. The multiple dumps are written one after another in the output data set, each preceded by DFSMSdss dump header records. By exploiting this capability, Database Image Copy 2 provides an option to concatenate the image copies that are created by a single execution of the utility. This form of stacking produces one output data set containing the image copy output for up to 255 DBDS instances; whereas, stacking achieved through JCL specifications produces multiple image copy instances in separate data sets on the same tape volume.

You can specify the Same Dataset option to indicate that the image copies for multiple database data sets are to be concatenated into the same output data set. In the RECON, the output data set is recorded in the image copy record for each of the DBDSs that were copied. If more than 255 DBDSs are specified, utility execution terminates.

The Same Dataset option is not available for DBDSs registered in the RECON with the REUSE attribute.

Note: Concatenating the image copies in this manner can increase the efficiency of tape media usage and decrease the number of tape volumes allocated to take the image copies, A disadvantage, however, is that recovery using the Database Recovery utility requires a separate read pass through the tape volume(s) to restore each DBDS from the stacked image copies.

Also, concatenating onto a single data set serializes the physical copy process and can extend the time of unavailability while clean copies are created with notification at physical completion.

Image Copy Completion Notification

When clean image copies are taken, the utility provides image copy completion notification to indicate when the database or group can be started. Completion notification is not provided when fuzzy image copies are taken. The completion of a fuzzy image copy requires no action to return the database to available status.

Logical Completion Notification

The DFS3121I message is issued when the image copy is logically complete. The image copy is logically complete when DFSMSdss has completed the concurrent copy initialization phase. When a clean image copy is taken and you want to resume updates to the database before the image copy is physically complete, this message indicates that the database can now be started on the online systems. The message is issued to the system console so that users can automate the starting of the database.

If a group statement is provided, then completion notification is given for the group name. Any arbitrary name can be used on the group statement, but it is more useful if the name chosen corresponds to the name of a pre-defined group, such as a CA group, a DB group, or a DBDS group.

- If a group name is specified to the utility, the DFS3121A message is issued just once for the group.
- If a group name is not specified, the message is issued once per database, HALDB partition, or area.

The DFS3121A message is issued at logical completion only if a clean image copy is being taken and updates are to be allowed once the copy is logically complete (XL was specified on the utility control statement). It is followed by DFS3121I messages identifying the DBDSs for the group or database that is being copied (for an area one DFS3121I message is issued identifying the area data set being copied). The DFS3121A message goes to the system console and to the SYSPRINT data set; the DFS3121I messages go only to the SYSPRINT data set.

This example shows the logical completion messages issued to SYSPRINT when a group statement is specified:

```
DFS3121A LOGICAL COPY COMPLETE FOR GROUP groupname; 0 OF 5 DATA SETS FAILED
DFS3121I COPIED DB/AREA dbnameA DDN ddnameA1 DSN dsnameA1
DFS3121I COPIED DB/AREA dbnameA DDN ddnameA2 DSN dsnameA2
DFS3121I COPIED DB/AREA dbnameA DDN ddnameA3 DSN dsnameA3
DFS3121I COPIED DB/AREA dbnameB DDN ddnameB1 DSN dsnameB1
DFS3121I COPIED DB/AREA dbnameC DDN ddnameC1 DSN dsnameC1
```

This example shows the logical completion messages issued when a group statement is not specified:

```
DFS3121A LOGICAL COPY COMPLETE FOR DB/AREA dbnameA; 0 OF 3 DATA SETS FAILED
DFS3121I COPIED DB/AREA dbnameA DDN ddnameA1 DSN dsnameA1
DFS3121I COPIED DB/AREA dbnameA DDN ddnameA2 DSN dsnameA2
DFS3121I COPIED DB/AREA dbnameA DDN ddnameA3 DSN dsnameA3
DFS3121A LOGICAL COPY COMPLETE FOR DB/AREA dbnameB; 0 OF 1 DATA SETS FAILED
DFS3121I COPIED DB/AREA dbnameB DDN ddnameB1 DSN dsnameB1
DFS3121A LOGICAL COPY COMPLETE FOR DB/AREA dbnameC; 0 OF 1 DATA SETS FAILED
DFS3121I COPIED DB/AREA dbnameC DDN ddnameC1 DSN dsnameC1
```

Database authorization is released by Database Image Copy 2 when logical copy is complete for all DBDS occurrences selected for the group, the database, the HALDB partition, or the area. This allows application processing to resume update activity.

Physical Completion Notification

The DFS3141A message is issued at physical completion when a clean image copy is being taken and updates are disallowed until the copy is physically complete (XP was specified). This message indicates that the image copies have been recorded in the RECON and the databases or areas can now be started. Like the DFS3121A message, the DFS3141A reports on the group name if a group name was specified. Otherwise, a DFS3141A message is issued for each database, HALDB partition, and area that was processed. The individual data sets that were successfully processed are identified by DFS3141 messages.

An example of the SYSPRINT output follows.

```
DFS3141A PHYSICAL COPY COMPLETE FOR GROUP groupname; 0 OF 4 DATA SETS FAILED
DFS3141I COPIED DB/AREA dbname1 DDN ddname1 DSN dsname1
DFS3141I COPIED DB/AREA dbname1 DDN ddname2 DSN dsname2
DFS3141I COPIED DB/AREA dbname3 DDN ddname3 DSN dsname3
DFS3141I COPIED DB/AREA dbname4 DDN ddname4 DSN dsname4
```

The DFS3141A message goes to the system console and to the SYSPRINT. The DFS3141I messages go only to the SYSPRINT. Authorization is released when physical copy is complete for all DBDS occurrences selected for the group, or for the database, HALDB partition, or area.

Completion Notification Summary

Image copy complete messages are issued to the system console as follows:

- When XL is specified for a group, database, or area, DFS3121A is issued at logical completion. DFS3141A is not issued.
- When XP is specified for a group, database, or area, DFS3141A is issued at physical completion. DFS3121A is not issued.
- When S is specified for a group, database, or area, neither DFS3121A nor DFS3141A is issued.

If a group name is not specified, different processing options can be specified for different DBDSs of the same database. In this case:

- If XP is specified for any DBDS, it applies to all DBDSs and DFS3141A is issued.
- Else, if XL is specified for any DBDS, it applies to all DBDSs and DFS3121A is issued.
- Else S is specified for all of the DBDSs and no message is issued.

The DFS3121A and DFS3141A messages also indicate if any DBDSs being copied for a group or database failed image copy processing. Providing this information in the message is helpful to users who might choose to delay starting the group or database until image copy for the failed DBDSs has been re-attempted. For each DBDS that failed, a DFS3122A (existing message) or DFS3144A message is issued.

Specifying DFSMSdss SET PATCH Commands

Product-sensitive programming interface

DFSMSdss provides a patch area that allows you to customize DFSMSdss processing. You can set specific patch bytes (using zap as needed) to customize the DFSMSdss processing that is invoked by the Database Image Copy 2 utility. However, when you set a patch byte, the patch is in effect for all DFSMSdss processing on the system, not just the processing that is done when the Database Image Copy 2 utility runs. Refer to the *z/OS DFSMSdss Diagnosis Guide* for more information.

DFSMSdss provides the SET PATCH command that can be used to set DFSMSdss patches. Patch bytes set by the command affect only the DFSMSdss processing for the task that specified the command. The Database Image Copy 2 utility defines generic SET PATCH commands (SET PATCH 00=00) in module DFSUDMT2. You can zap these commands to cause the utility to pass the modified SET PATCH commands to DFSMSdss along with the DUMP command. The patches are then in effect only for the Image Copy 2 job step and not for any other DFSMSdss

processing. Refer to *z/OS DFSMSdss Storage Administration Reference* for information on the SET PATCH command.

To have the utility pass a SET PATCH command to DFSMSdss, you can apply a zap in DFSUDMT2 to replace the patch byte and the patch value on one of the SET PATCH 00=00 character strings with the desired values (character strings of length 2), for example, SET PATCH 44=FF. One or more SET PATCH commands in the module can be changed. The utility passes DFSMSdss any SET PATCH command for which the patch byte character string is not 00. Note that the utility does not verify the command syntax or validate the patch byte or the patch value before passing the command to DFSMSdss.

Before using this capability, consult with and follow the guidance provided about DFSMSdss patches in the *z/OS DFSMSdss Diagnosis Guide*. If usage is restricted by the installation, you must ensure that authorization to issue the DFSMSdss SET command with the PATCH keyword is provided (see *z/OS DFSMSdss Storage Administration Reference* for more information).

End of Product-sensitive programming interface

Restrictions for DFSUDMT0

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 an Enterprise Storage Server® (ESS), a 3990 Storage Control, or a RAMAC® Virtual Array (RVA) with Snapshot capability) 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)*.
- Nonrecoverable databases and areas must be stopped by the Database Image Copy 2 utility. Because updates are not logged to be copied, nonrecoverable databases cannot be active during image copy processing. Utility control statements must specify X (for exclusive access).
- 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.
- Input 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 (That is, the high-level qualifier(s) of the 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.
- Routines coded for this utility must not call module DFSRRC00. Doing so results in an abend.

- When running an image copy utility with HALDB Online Reorganization, additional restrictions apply. See *IMS Version 9: Administration Guide: Database Manager* for more information.

Input and Output for DFSUDMT0

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.

The output from the Database Image Copy 2 utility can be used as input to the Database Recovery utility.

You can create up to four output image copies; however, only the first two are registered in the 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*, or the *DFSMS/MVS® V1 R3 DFSMSdss Storage Administration Guide*.

When creating an image copy of a shared secondary index, specify only the first DBD. This copies the entire data set.

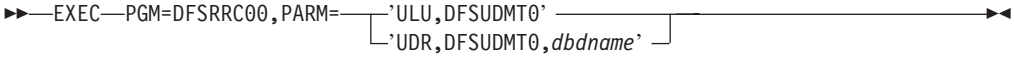
JCL Requirements for DFSUDMT0

The Database Image Copy 2 utility is executed as a standard z/OS 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 can either invoke a cataloged procedure containing the required statements or be in the following form:



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 Version 9: 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.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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. For a HALDB data set this DD statement can be omitted. Otherwise, one DD statement of this type must be specified for each data set to be copied. The data set must reside on hardware that supports the concurrent copy function.

For DEDB multiple 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 V1 R3 DFSMSdss Storage Administration Guide*.

Note: When multiple DBDSs are to be copied, multiple sets of output DD statements must be supplied.

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.

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.

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.

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 Statements for DFSUDMT0

There are two types of utility control statements for the Database Image Copy 2 utility: **DBDS Select Statement** and **Group Name Statement**. Each statement has a fixed format using column positions described in "DBDS Select Statement" and in "Group Name Statement" on page 218.

All of the database data sets specified in the utility control statements for an execution of the Database Image Copy 2 utility are specified to DFSMSdss to be processed in parallel or on a single dump command. There is no provision for serial processing of the control statements.

Copy Integrity options and Notify and Hold options for all DBDS occurrences with the same database must be the same. Authorization is done at the database/area level.

Syntax errors in either the Group or DBDS Select statements result in termination of the utility.

DBDS Select Statement

Each DBDS select statement identifies one DBDS that is to be copied. It contains the number of output copies and gives a DD name for each output copy data set. You can also choose data integrity and notification options as well as other options that are passed directly to DFSMSdss.

Parameters specified in positions 58-61 are ignored if a Group Name Statement is specified.

Position	Description
1	Statement type Blank or S. If blank, output data set(s) are specified on this control statement. If S, the copy is to be written to the same output data

set(s) specified on the preceding control statement that did not specify S. Only positions 4-20, 58 and 59 are operative with the S option.

2 Number of copies

This can be from 1 to 4. This number must be less than or equal to the number of output DD names supplied in positions 22 to 56. Ignored if S specified in position 1.

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.

12 Ignored

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. For a HALDB data set, the DD statement can be omitted.

21 Ignored

22-29 OUTPUT ddname

This must be the ddname of the primary output data set. A corresponding DD statement must have been provided.

This field is ignored if S is specified in column 1.

30 Ignored

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.

This field is ignored if S is specified in column 1.

39 Ignored

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.

This field is ignored if S is specified in column 1.

48 Ignored

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.

57 Ignored

58 Copy integrity control for DBDS

Specifies whether databases can be updated during the image copy. X (exclusive) indicates that the database cannot be updated during the logical copy phase or during both the logical and physical copy phases depending on the specification in position 59. 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 DBDS of a nonrecoverable database or for a nonrecoverable area. If X is not specified, image copy processing for the DBDS or area fails.

Note that an SMSNOCIC image copy can be used as input to the IMS High Performance Pointer Checker for z/OS while an SMCIC cannot.

This field is ignored if a group name statement is specified.

59 Notify and hold control for DBDS

If position 58 is X, this field specifies when the database is made available for update processing. This field is ignored when position 58 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.

This field is ignored if a group name statement is specified.

60 COMPRESS control for DBDS

C specified in this position indicates that the DFSMS COMPRESS option is used. If the position is left blank, the DFSMSdss COMPRESS option is not used.

This field is ignored if a group name statement is specified.

61 OPTIMIZE control for DBDS

OPTIMIZE is a DFSMS option that modifies performance. Use of OPTIMIZE provides trades between execution time and allocation of real and virtual storage, and also trades between utility performance and application/transaction processing.

OPTIMIZE control provides four levels of optimization that control the number of tracks of DASD that are transferred by one I/O command. The level of optimization is indicated here with a number, either 1, 2, 3, or 4. If this control option is omitted (blank), the utility defaults are used: OPTIMIZE(1) is the default when taking a fuzzy copy; OPTIMIZE(4) is the default when taking a clean copy.

This field is ignored if a group name statement is specified.

62–72 Unspecified in this IMS release.

73–80 Available for user comments.

If a Group Name statement is specified, the parameters specified in positions 58-61 of the Group Name statement override the corresponding parameters specified here on the DBDS Select statement.

Group Name Statement

To specify an optional group name to the utility, a group name control statement must be supplied as the first control statement. Only one such statement is allowed in the input stream for a single job step.

Note: The following attributes of the Group Name Statement should be noted.

- The group statement must be followed by control statements specifying the DBDSs that make up the group.
- The utility does not check for the existence of a group with the specified name.
- The utility does not check that the DBDSs specified following the group statement are actually members of such a group.

The parameters specified or defaulted in positions 58-61 override corresponding parameters specified in the DBDS Select Statement.

Position	Description
1	Statement type
	'G' indicates that a group name is supplied on this control statement.
2-3	Reserved
4-11	Group Name
	The DBDSs specified on subsequent control statements are members of this group.
12-57	Reserved
58	Copy integrity control for entire group
	Specifies whether databases can be updated during the image copy. X (exclusive) indicates that the databases cannot be updated during the logical copy phase or during both the logical and physical copy phases depending on the specification in position 59. The image copy is recorded in the RECON as an SMSNOCIC image copy. S (shared) indicates that the databases can be updated during the image copy phase. The image copies are recorded in the RECON as an SMSCIC image copy. The default is S.
	Note: X must be specified for a nonrecoverable database. If S is specified, the image copy processing will fail for any members of the group that are nonrecoverable.
	Note: An SMSNOCIC image copy can be used as input to the IMS High Performance Pointer Checker for z/OS while an SMCIC image copy cannot.
59	Notify and hold control for group
	If position 58 is X, this field specifies when the database is made available for update processing. This field is ignored when position

58 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 COMPRESS control for group

C specified in this position indicates that the DFSMSdss COMPRESS option is used. If the position is left blank, the DFSMS COMPRESS option is not used.

61 OPTIMIZE control for group

OPTIMIZE is a DFSMS option that modifies performance. Use of OPTIMIZE provides trades between execution time and allocation of real and virtual storage, and also trades between utility performance and application/transaction processing.

OPTIMIZE control provides four levels of optimization that control the number of tracks of DASD that are transferred by one I/O command. The level of optimization is indicated here with a number, either 1, 2, 3, or 4. If this control option is omitted, the utility defaults are used: OPTIMIZE(1) is the default when taking a fuzzy copy; OPTIMIZE(4) is the default when taking a clean copy.

62–72 Unspecified in this IMS release.

73–80 Available for user comments.

Return Codes for DFSUDMT0

The Database Image Copy 2 utility provides the following return codes:

Code	Meaning
0	Processing completed successfully
4	Warning messages were issued for one or more data sets
8	Processing was successful for some but not all data sets
12	Processing failed for all data sets
16	Severe errors caused the utility 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 of DFSUDMT0

Figure 67 on page 220 shows the utility control statement used to copy DBDSs for four different databases. The DBDSs to be copied are identified as members of group GROUPXYZ. All of the data sets will be copied using the same processing options (XL). When the image copies for all of the DBDSs are logically complete, one DFS3121A message will be issued for the group GROUPXYZ. This example is only valid if none of the output data sets resides on the same tape volume as any of the other output data sets.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
//SYSIN DD    *
G GROUPXYZ                                                    XL
  2 DBNAME1 DDNAME1A ICOUT1A1 ICOUT1A2
  2 DBNAME1 DDNAME1B ICOUT1B1 ICOUT1B2
  2 DBNAME1 DDNAME1C ICOUT1C1 ICOUT1C2
  2 DBNAME1 DDNAME1D ICOUT1D1 ICOUT1D2
  2 DBNAME1 DDNAME1E ICOUT1E1 ICOUT1E2
  2 DBNAME2 DDNAME2A ICOUT2A1 ICOUT2A2
  2 DBNAME2 DDNAME2B ICOUT2B1 ICOUT2B2
  2 DBNAME3 DDNAME3A ICOUT3A1 ICOUT3A2
  2 DBNAME4 DDNAME4A ICOUT4A1 ICOUT4A2
  2 DBNAME4 DDNAME4B ICOUT4B1 ICOUT4B2
  2 DBNAME4 DDNAME4C ICOUT4C1 ICOUT4C2
/*
```

Figure 67. Specifying a Data Group

Figure 68 shows utility control statement used to copy DBDSs for four different databases. The image copies for the five DBDSs for DBNAME1 are to be stacked into the ICOUT101 and ICOUT102 data sets. The image copies for the two DBDSs for DBNAME2 are to be stacked into the ICOUT201 and ICOUT202 data sets. The one DBDS for DBNAME3 and three DBDSs for DBNAME4 are each copied into their own image copy output data sets. None of the output data sets are on the same tape volume as any of the other output data sets.

In this case, because a group name was not specified, different processing options (S|XL|XP) can be specified for different DBDSs. When the image copies for the DBDSs for DBNAME1 are all logically complete, a DFS3121A message will be issued. When the image copies for the DBDSs for DBNAME4 are all physically complete, a DFS3141A message will be issued.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
//SYSIN DD    *
 2 DBNAME1 DDNAME1A ICOUT101 ICOUT102                    XL
S  DBNAME1 DDNAME1B                                      XL
S  DBNAME1 DDNAME1C                                      XL
S  DBNAME1 DDNAME1D                                      XL
S  DBNAME1 DDNAME1E                                      XL
 2 DBNAME2 DDNAME2A ICOUT201 ICOUT202                    S
S  DBNAME2 DDNAME2B                                      S
 2 DBNAME3 DDNAME3A ICOUT3A1 ICOUT3A2                    S C2
 2 DBNAME4 DDNAME4A ICOUT4A1 ICOUT4A2                    XP
 2 DBNAME4 DDNAME4B ICOUT4B1 ICOUT4B2                    XP
 2 DBNAME4 DDNAME4C ICOUT4C1 ICOUT4C2                    XP
/*
```

Figure 68. Stacking Image Copies in a Single Output Data Set

Figure 69 on page 221 shows the JCL for the Database Image Copy 2 utility.

```
//STEP1    EXEC   PGM=DFSRRC00,PARM='ULU,DFSUDMT0'
//STEPLIB  DD     DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD     DSN=IMS.SDFSRESL,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=DMPNO2,LABEL=(,SL)
//OUTPUTD3 DD     DSN=IMS.DBAOUT3,DISP=(NEW,KEEP),
//                UNIT=TAPE,VOL=SER=DMPNO3,LABEL=(,SL)
//OUTPUTD4 DD     DSN=IMS.DBAOUT4,DISP=(NEW,KEEP),
//                UNIT=TAPE,VOL=SER=DMPNO4,LABEL=(,SL)
//SYSIN    DD     *
 4 DBDNAMEX DBIN     OUTPUTD1 OUTPUTD2 OUTPUTD3 OUTPUTD4 XL
```

Figure 69. Database Image Copy 2 Utility JCL

Chapter 22. 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: See "Making Database Backup Copies" in *IMS Version 9: Operations Guide* for more information on using the Online Database Image Copy utility.

The following topics provide additional information:

Restrictions for DFSUICP0

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. See "Program Specification Block (PSB) Generation" in *IMS Version 9: 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 Version 9: 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 more than one IMS subsystem has the database open for update access. Use the Database Image Copy utility (DFSUDMP0 with the CIC parameter) instead of the Online Database Image Copy utility.
- When running an image copy utility with HALDB Online Reorganization, additional restrictions apply. See *IMS Version 9: Administration Guide: Database Manager* for more information.

Output for DFSUICP0

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 for DFSUICP0

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 for DFSUICP0” on page 227.

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 can be submitted.

JCL Requirements for DFSUICP0

The Online Database Image Copy utility is executed as a standard z/OS job. The following JCL statements 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 z/OS console).

The normal IMS positional parameters can follow.

Related Reading: See “Member Name IMSBATCH” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional parameters that can be specified in executing a batch message processing region.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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 double word. 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 23 KB is used unless the logical record length is larger than 23KB. 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.

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: See "SB Parameters (SBPARM) Control Statement" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for a description of SBPARM control statements.

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 for DFSUICP0

The utility control statement for the Online Database Image Copy utility is fixed format using the positions described as follows:

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 for DFSUICP0

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 of DFSUICP0

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 DD statements shown in Figure 70 on page 229 to the sample JCL:

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

Figure 70. DD Statements for Using DBRC without Dynamic Allocation

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.

```
//*
//OLIC    EXEC PGM=DFSRRC00,PARM='BMP,DFSUICP0,HHTASK41'
//STEPLIB   DD DSNAME=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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 4. Recovery Utilities

Chapter 23. 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 non database 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:

- See "Understanding IMS Logging" in *IMS Version 9: Operations Guide* for more information on using the Database Change Accumulation utility.
- See *IMS Version 9: Administration Guide: Database Manager* for more information on running the Database Change Accumulation utility with HALDB Online Reorganization.

The Database Change Accumulation utility can be run independently of IMS. When this utility is run with z/OS, the output is compressed. This compression is achieved using z/OS services.

The information in Table 13 depicts the sources of input to the Database Change Accumulation utility and the output created by this utility. These sources are also displayed in Figure 71.

Table 13. Input To and Output From the Database Change Accumulation Utility

Input	Output
RECON	New log data set
Input control statements	New change accumulation data set
SLDS and RLDS IMS system logs	
Previous change accumulation data set	
DBD library	

Figure 71 depicts the sources of input to the Database Change Accumulation utility and the output created by this utility.

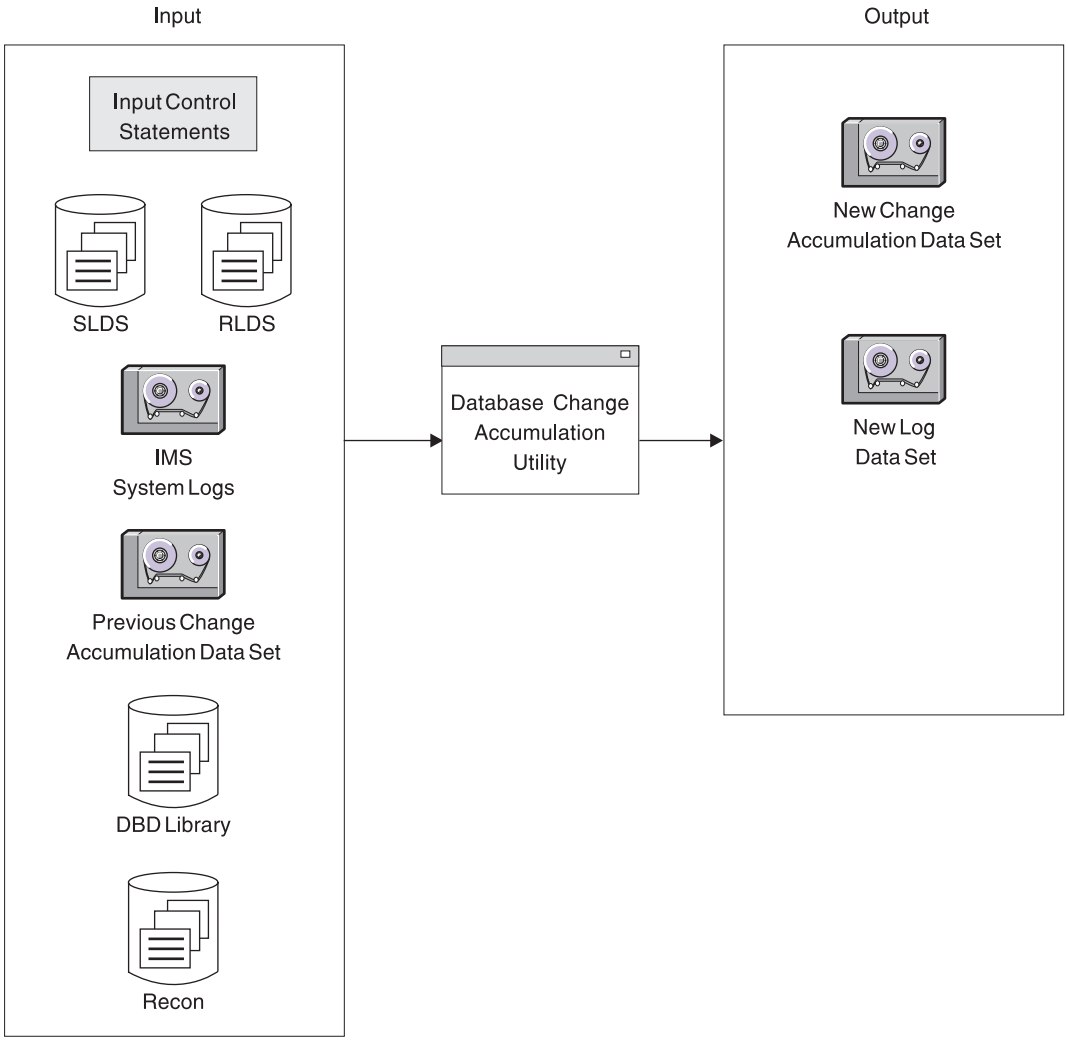


Figure 71. Database Change Accumulation Utility

The following topics provide additional information:

- “Restrictions for DFSUCUM0” on page 235

Restrictions for DFSUCUM0

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 for DFSUCUM0

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 71 on page 234 and "Utility Control Statements for DFSUCUM0" on page 238.

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, 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 for DFSUCUM0

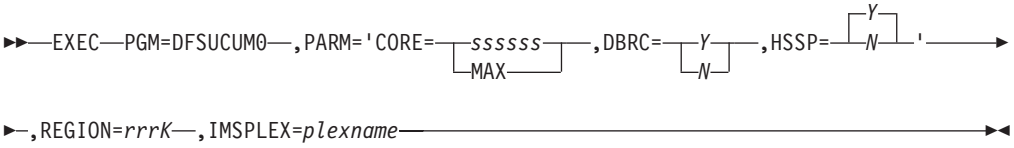
The Database Change Accumulation utility is executed as a standard z/OS job. The following JCL statements are required:

- 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:



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 100 KB, or 304 KB 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.

IMSPLEX=*plexname*
specifies which IMSplex DBRC should join. For more information about the IMSPLEX parameter, see *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference* under initializing and maintaining RECONs.

DD Statements

Lowercase ddnames on DD statements are supplied by you and can be any valid DD name.

STEPLIB DD
Points to IMS.SDFSRESL, 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.

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 DD statement shown in Figure 72 must be used:

```
//DFSUCUMO DD DUMMY,DCB=BLKSIZE=100
```

Figure 72. DD Statement for No Changes to be Merged

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 for DFSUCUM0

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:

- 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 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.
- If you manually code DB0 statements in the JCL for the Change Accumulation utility and do not specify an offset for the purge time, the utility assumes that the input time is local and uses the local z/OS setting of the offset from Universal Coordinated Time (UTC) to convert the time to UTC. If you explicitly code an offset of zeroes, the Change Accumulation utility interprets the input time stamps as being UTC with no offset.

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.

Recommendation: Use DBRC when running the Change Accumulation utility in order to ensure that the correct UTC purge time is applied for all change accumulation processing.

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.

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. 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. This utility control statement is a fixed format using the positions described as follows:

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 sequence identifier length (optional) In positions 31 through 33 you can enter the length of the maximum record identifier field for all database records referenced by the log records to be processed as a result of a DB0 control statement. For database records contained in an OSAM data set or ESDS, the identifier field is the RBN. For a KSDS, the record identifier is the key of the root segment. The maximum length value is used to calculate the length of padding with binary zeros required for record identifiers which are less than the maximum 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, or leave it blank and the default value of `4` will be used. If there are any VSAM KSDS occurrences to be processed, then the value must be in the range `1` through `256` and must be left-justified or supplied with leading zeros or omitted altogether. If the value is omitted and one or more KSDS occurrences are to be processed, the utility will determine the maximum key length for all listed KSDS occurrences. If a value is supplied in this statement but a larger value is found for any KSDS, the larger value will override the supplied value.
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.

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 as follows:

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 "*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.

Note: For HALDB databases, always code a ddname. It is highly recommended to use DBRC GENJCL commands to produce the change accumulation JCL instead of manually coding it.

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: For a detailed description of the types of standard timestamp formats that might be used in this field, see the command syntax parameters section of the *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference*.

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 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. For HALDB OLR capable databases, always code a ddname. Both the A-J and M-V ddnames are allowed as input.

Recommendation: Use DBRC GENJCL commands to produce the change accumulation JCL.

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.

The DB1 utility control statement is fixed format using the positions described as follows:

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, *ALL or *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, a DB1 Statement cannot be specified. 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:** For a detailed description of the types of standard timestamp formats that might be used in this field, see the command syntax parameter section of the *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference*.
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 written to the Sort output data set. Only one SO statement is allowed. 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 as follows:

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, only the 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 all positions are not blank, then all must be numeric digits using leading zeros as needed. If a database dataset ID is specified, only 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 block number) of a record or of many records. If a record contains the RBA or RBN specified, only records with that RBA or RBN are written. The RBA or RBN must be specified in hexadecimal.

Output Messages and Statistics for DFSUCUM0

Message IEC161I is a normal message during an IMS database recovery of a VSAM data set.

Return Codes for DFSUCUM0

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 of DFSUCUM0

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 DD statements shown in Figure 73 to the sample JCL:

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

Figure 73. DD Statements for Using DBRC without Dynamic Allocation

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.

```
//STEP1   EXEC PGM=DFSUCUM0,PARM='CORE=512000'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD DUMMY,DCB=BLKSIZE=100
//DFSUCUMN  DD DSN=IMS.CUM1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=CUMTAP
//DFSUDD1   DD DSN=IMS.NEWLOG,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=LOGTAP
//DFSULOG   DD DSN=IMS.LOG1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE1
//          DD DSN=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.

```
//STEP1   EXEC PGM=DFSUCUM0,PARM='CORE=512000'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD DSN=IMS.CUM1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=CUMTAP
//DFSUCUMN  DD DSN=IMS.CUM2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=CUMTP2
//DFSUDD1   DD DUMMY
//DFSULOG   DD DSN=IMS.LOG1,DISP=OLD.
//             UNIT=TAPE,VOL=SER=LTAPE3
//          DD DSN=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.

```
//STEP1   EXEC PGM=DFSUCUM0,PARM='CORE=512000'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(2),,CONTIG)
//DFSUCUMO  DD DUMMY,DCB=BLKSIZE=100
//DFSUCUMN  DD DSN=IMS.CUM1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=CUMTAP
//DFSUDD1   DD DSN=IMS.NEWLOG1,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=LOGTAP
//DFSUDD2   DD DSN=IMS.NEWLOG2,DISP=(NEW,KEEP),
//             UNIT=TAPE,VOL=SER=LOGTAP
//DFSULOG   DD DSN=IMS.LOG1,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LTAPE1
//          DD DSN=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 24. HALDB Index/ILDS Rebuild Utility (DFSPREC0)

Use the HALDB Index/ILDS Rebuild utility (DFSPREC0) to rebuild the prime index data set or Indirect List data set (ILDS) for a HALDB partition. This utility scans the HALDB partition and recreates the ILDS and Index KSDS from the HALDB partition data.

DFSPREC0 is designed based on the philosophy that if Index/ILDS rebuild is required for multiple HALDB partitions, time is of the essence. It is better to submit several jobs to rebuild multiple HALDB partitions concurrently than one job to rebuild them serially. The HALDB Index/ILDS Rebuild utility will rebuild both the prime index and ILDS for a HALDB partition during a single execution of the utility.

The HALDB Index/ILDS Rebuild run environment has these attributes:

- Runs as a batch utility in a Type=ULU region
- It is non-recursive so it can only process one partition in an execution

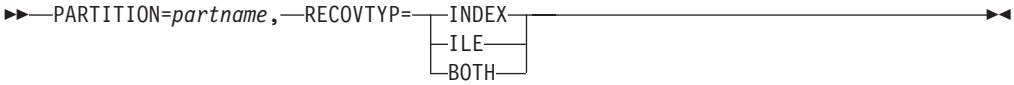
You can use DFSPREC0 to recover both the X and the Y data sets of OLR capable HALDB partitions. DFSPREC0 can be run while OLR is in a cursor-active state (between TERM OLR and (re)INIT OLR). It can be run to recover both the input and the output primary index data sets, meaning that one DFSPREC0 can be run for input and a second can be run for the output.

Related Reading: For additional information on using the HALDB Index/ILDS Rebuild utility with HALDB Online Reorganization, see *IMS Version 9: Administration Guide: Database Manager*.

The following topics provide additional information:

Input and Output for DFSPREC0

The HALDB Index/ILDS Rebuild utility uses the following control statement to rebuild the ILDS and Index after the rebuild of the partition. (The rebuild of the partition is optional. However, if rebuild of the partition is necessary, do it first, then rebuild the ILDS and Index.) The control statement input starts in column one.



- PARTITION=partition name
- Recovery type: index, ILE, or both
  - INDEX — rebuild the INDEX data set for the HALDB partition
  - ILE — rebuild the ILDS for the HALDB partition
  - BOTH — rebuild both the INDEX and ILDS for the HALDB partition

- *partname*— name of the HALDB partition to be rebuilt

In cases where both the INDEX and the ILDS need to be rebuilt, the INDEX data set must be rebuilt first. Specifying BOTH as the RECOVTYP assures this rebuild sequence.

The HALDB Index/ILDS Rebuild utility produces a rebuilt PHIDAM index, a rebuilt ILDS, or both. That is:

- The rebuilt index
- The rebuilt ILE
- Total number of lines sent to the SYSPRINT data set (indicating the number of index and ILE entries created)

JCL Requirements for DFSPREC0

The following JCL statements are required:

- A JOB statement that you define
- An EXEC statement
- DD statements that specify the input and output

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSRRC00,PARM='ULU,DFSPREC0,dbdname,,,,,,,,,,Y,N'    Y=DBRC REQUIRED
```

The dbdname on the EXEC card is the name of a HALDB master database. The ILDS and primary index of one of the partitions making up this HALDB will be rebuilt in the execution of DFSPREC0. The partition to be rebuilt will be selected by the utility control statement entered in the SYSIN DD.

The database is dynamically allocated.

DD Statements

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

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

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.

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.

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.

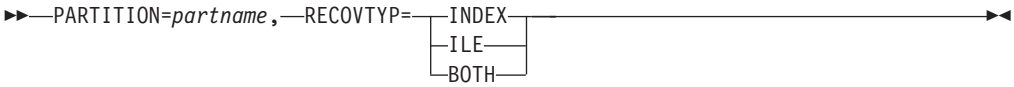
The SYSPRINT data set is also used to return error messages to the user.

SYSUDUMP DD
Defines a dump data set.

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

Utility Control Statement for DFSPREC0

Input statements are used to describe processing options for the HALDB Index/ILDS Rebuild utility. The format of the HALDB Index/ILDS Rebuild utility control statement is as follows with the control statement input starting in column one.



PARTITION=
Identifies the partition name of the HALDB with the bad INDEX, ILE, or both. It must be the first parameter and start in column one.

RECOVTYP=
Definition

- INDEX
  Specifies that the prime index is to be rebuilt.
- ILE
  Specifies that the ILDS is to be rebuilt.
- BOTH
  Specifies that both the prime index and the ILDS are to be rebuilt.

The utility control statement is a fixed format using the positions as follows:

Position	Description
1	Statement id This must be characters 'PARTITION='
11–17	Partname Name of the HALDB partition to be built
19	Statement id This must be characters 'RECOVTYP' or ″RECOVERY'
28	Recovery type INDEX-rebuild the index data set for the HALDB partition ILE - rebuild the ILDS for the HALDB partition BOTH - rebuild both the index and ILDS for the HALDB partition

Output Messages and Statistics for DFSPREC0

The HALDB Index/ILDS Rebuild utility provides output messages and statistics.

Output from the HALDB Index/ILDS Rebuild utility includes the rebuilt index, rebuilt ILDS, and total lines sent to the SYSPRINT data set (indicating the number of index and ILE entries created.) The SYSPRINT data set is also used to return error messages to the user.

Return Codes for DFSPREC0

For a complete description of the return codes for Index/ILDS Rebuild utility, refer to the section on DFS1982 in *IMS Version 9: Messages and Codes, Volume 2*.

Example of DFSPREC0

Sample JCL for rebuilding both ILDS and primary index. The part name is the partition of the HALDB master database named in the EXEC statement positional parameter MSTRDBNM.

```
 //STPPD    EXEC  PGM=DFSRRC00,
//         PARM='ULU,DFSPREC0,MSTRDBNM,,,,,,,,,,Y,N',
//STEPLIB  DD  DSN=IMS.SDFSREL,DISP=SHR
//IMS      DD  DSN=IMS.DBDLIB,DISP=SHR
//             DD  DSN=IMS.PSBLIB,DISP=SHR
//SYSUDUMP DD  SYSOUT=A
//SYSPRINT DD  SYSOUT=A,DCB=BLKSIZE=1200
//DFSVSAMP DD  input for VSAM and OSAM buffers and options
//SYSIN    DD  *
PARTITION=PARTNAM,RECOVTYP=BOTH
/*
```

Chapter 25. 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:

- See “Understanding IMS Logging” in *IMS Version 9: Operations Guide* for more information on the Database Recovery utility.
- For information on database recovery for databases using HALDB Online Reorganization, see *IMS Version 9: Administration Guide: Database Manager*.

The information in Table 14 identifies inputs and outputs for the Database Recovery utility.

Table 14. Input To and Output From the Database Recovery Utility

Input	Output
RECON	SYSPRINT messages
DBD library	Recovered data sets
Image copy	
Change accumulation input	
Log(s)	
Data set to be recovered	
Input control statements	

Figure 74 on page 255 is a flow diagram of the Database Recovery utility.

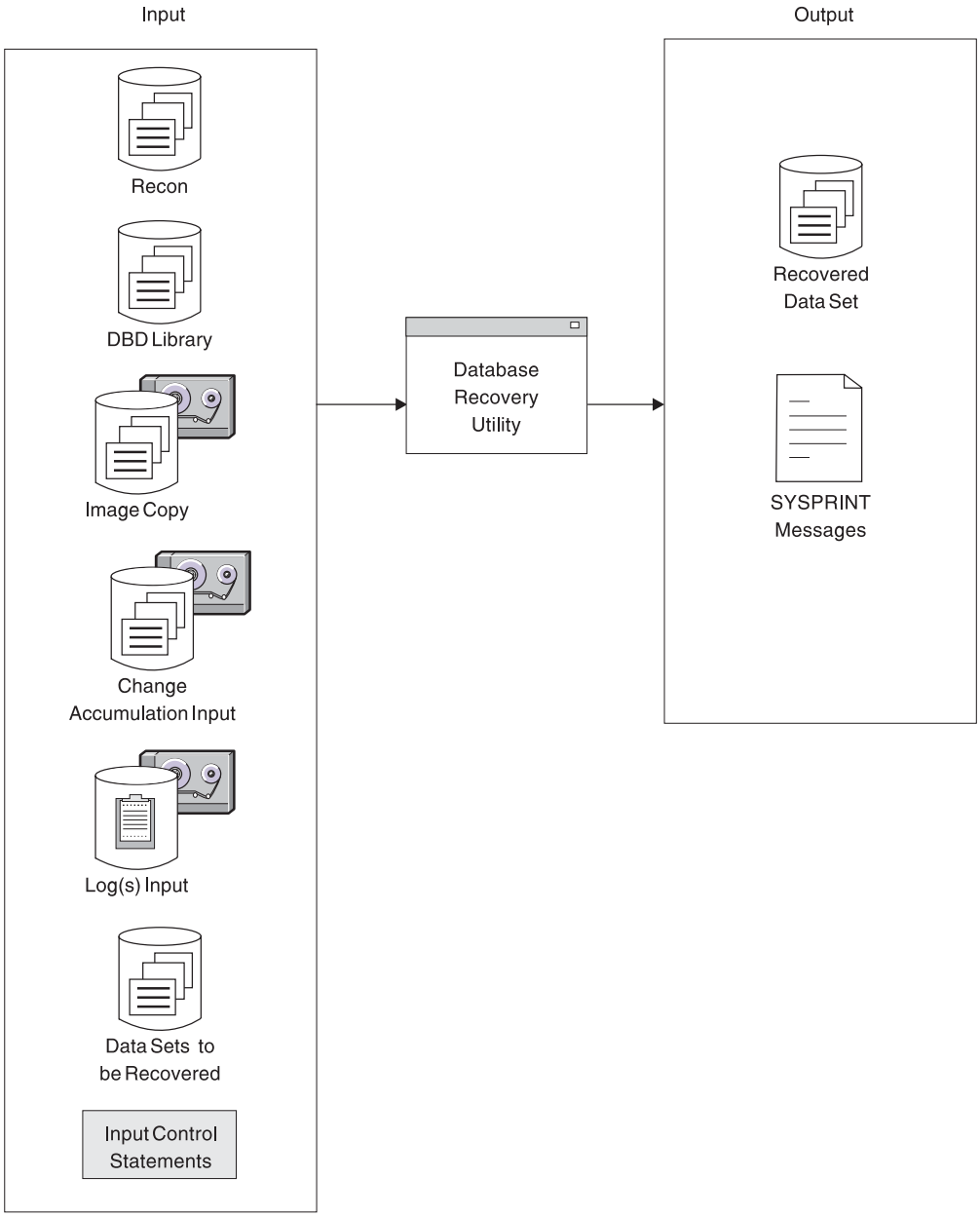


Figure 74. Database Recovery Utility

The following topics provide additional information:

- "Using the Database Recovery Utility in an RSR Environment" on page 256
- "Restrictions for DFSURDB0" on page 256
- "Input and Output for DFSURDB0" on page 257
- "JCL Requirements for DFSURDB0" on page 259
- "Utility Control Statement for DFSURDB0" on page 261
- "Output Messages and Statistics for DFSURDB0" on page 263
- "Return Codes for DFSURDB0" on page 263
- "Examples of DFSURDB0" on page 264

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 for DFSURDB0

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 deleted and reallocated. It is recommended that multi-volume OSAM data sets be deleted and allocated using IDCAMS in a jobstep prior to the recovery jobstep. If they are not deleted and allocated, the data set might be allocated with DISP=NEW in the recovery jobstep. Any attempt to perform a full recovery using an existing OSAM data set containing data will produce unpredictable results. A full recovery is a recovery using a valid image copy, change accumulation and log record data. This statement does not pertain to a USEDBDS or forward recovery without an image copy. An example of how to delete and reallocate an OSAM data set is shown here:

```
IF MAXCC = 0                                        -
 THEN DO
    DELETE (IMSVS.IMSIVP.DFSIVD1) NONVSAM
    IF LASTCC NE 0                                  -
    THEN SET MAXCC = 0
    ELSE
    ALLOCATE                                        -
       DSNAME('IMSTESTL.IMSIVP.DFSIVD1') -
       FILE(DFSIVD1)                -
       RECFM(F,B,S)                 -
       LRECL(2048)                  -
       BLKSIZE(2048)                -
       DSORG(PS)                    -
       NEW CATALOG                  -
       SPACE(15645 15645)           -
       MAXVOL(28)                   -
       UNIT(SYSDA)
    END
ELSE
```

- 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 other than one restored from a copy created by Image Copy 2, 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 “Initializing IMS System Data Sets” in *IMS Version 9: Administration Guide: System* for the levels of data sharing that the `DBRC INIT.DB` command issues.
- When an image copy created by the Image Copy 2 utility is used as input, DBRC must be active when the Database Recovery utility is run.

Input and Output for DFSURDB0

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 dual image copy data sets are listed in the RECON, then the Database Recovery utility will use the first image copy data set unless it is marked INVALID.

If you have already copied the database with a copy created by a z/OS 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 data set types, the physical sequence of the content of the data set must be preserved. Failure to copy the data set correctly can result in failure of the database recovery process. Be sure to use only utilities that copy the data set 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.

Recommendation: Use DBRC when running the Database Recovery utility to calculate the correct Universal Coordinated Time (UTC) offsets. If DBRC is not used, then the z/OS offset will be used as a default.

JCL Requirements for DFSURDB0

The Database Recovery utility is executed as a standard z/OS job. The following JCL statements 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 master DBD that includes the data set to be recovered

The normal IMS positional parameters such as BUF and SPIE can follow dbdname.

Related Reading: See “Member Name DLIBATCH” in *IMS Version 9: 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.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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 DSN=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 264 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 attributes shown in Figure 75:

```
DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
```

Figure 75. Attributes to define a SNAP Output Data Set

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. For HALDBs, this DD statement defines the partition data set of the partition to be recovered. The ddname must be the partition name, as defined with the Partition Definition utility.

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: See "Specifying the IMS Buffer Pools" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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 for DFSURDB0

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 Version 9: Database Recovery Control (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 as follows:

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. These columns of the control statement are treated as comments and are ignored when the utility is run with DBRC turned off.

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 . . .
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 z/OS 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 Version 9: Database Recovery Control (DBRC) Guide and Reference* for further information.

Output Messages and Statistics for DFSURDB0

It is normal to receive a VSAM information message, IEC161I 072-053, when recovering a VSAM data set.

Return Codes for DFSURDB0

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 of DFSURDB0

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 DD statements shown in Figure 76 to the sample JCL:

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

Figure 76. DD Statements for Using DBRC without Dynamic Allocation

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.

```
//STEP1   EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,DD32DB01'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//DUMPDS    DD DSN=IMS.DBBOUT1,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP3,LABEL=(,SL)
//DFSUCUM   DD DUMMY
//DFSULOG   DD DSN=IMSLOG.MONDAY,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=LOG1,LABEL=(,SL)
//          DD DSN=IMSLOG.TUESDAY,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=LOG2,LABEL=(,SL)
//DBHD3B    DD DSN=IMS.DBHD3b,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=DBASE2,
//             SPACE=(CYL,(20,10))
//DFSVSAMP  DD input for VSAM and OSAM buffers and options
//* +----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.

```
///STEP1   EXEC PGM=DFSRRC00,
// PARM='UDR,DFSURDB0,DI32DB01,,,,,,,,,,Y,,,,,,,,GSGNAME1'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=1210
//DFSUDUMP  DD DSN=IMS.DBAOUT1,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=DBDMP1,LABEL=(,SL)
//DFSUCUM   DD DSN=IMS.CUM1,DISP=(OLD,KEEP),
//             UNIT=TAPE,VOL=SER=DBCUM1,LABEL=(,SL)
//DFSULOG   DD DUMMY
//DDNAME1   DD DSN=DSNAME1,DISP=OLD
//DFSVSAMP  DD DSN=IMS.VSAM.PARM(OPTIONS),DISP=SHR
//RECON1    DD DSN=RECON1,DISP=SHR
//RECON2    DD DSN=RECON2,DISP=SHR
//RECON3    DD DSN=RECON3,DISP=SHR
//* +----1----+----2----+----3----+----4----+----5----+----6----+----7---
//SYSIN    DD  *
S  DI32DB01 AREANAM1
/*
```

Example 3

This example shows the JCL to recover the first partition data set in a database with more than one partition. MASTERDB is a master database name recorded in the RECON and contained in DBDLIB. PART001A is a partition ddname for the first part of partition PART001, which is also recorded in the RECON.

Related Reading: See Chapter 24, “HALDB Index/ILDS Rebuild Utility (DFSPREC0),” on page 249 for more information on recovery of HALDBs.

```
//RECOVDB  EXEC PGM=DFSRRC00,
//         PARM='UDR,DFSURDB0,MASTERDB,,,,,,,,,,Y,N'
//IMS      DD DSN=IMSTESTG.IMS910.DBDLIB,DISP=SHR
//         DD DSN=IMSTESTG.I11X.DBDLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSUDUMP DD DSN=IMSVS.PART001.IC.ICDSN1,DISP=OLD,
//         UNIT=SYSDA,
//         VOL=SER=222222
//DFSULOG  DD DSN=DBRC.RLDS000,DISP=SHR,
//         VOL=SER=000000,UNIT=SYSDA
//         DD DSN=DBRC.RLDS001,DISP=SHR,
//         VOL=SER=000000,UNIT=SYSDA
//DFSUCUM  DD DUMMY
//PART001A DD DSN=MASTERDB.PART.DATASET.A00001,DISP=SHR
//RECON1   DD DSN=IMSTESTL.IMS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSTESTL.IMS.RECON2,DISP=SHR
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//SYSIN DD *
S  MASTERDB PART001A
/*
```

Chapter 26. 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: For information on circumstances under which to run this utility, see “Backout” in *IMS Version 9: Operations Guide*.

The information in Table 15 identifies inputs and outputs for the Batch Backout utility.

Table 15. Data Set Requirements for the Batch Backout Utility

Input	Output
RECONs	
Input set of databases to be backed out (same as output set of databases)	Output set of databases to be backed out (same as input set of databases)
If parm=DLI, DBD/PSB libraries	IEFRDER (output log)
If parm=DBB, ACBLIB library	
IMSLOGR (input log)	
SYSIN control statements	

Figure 77 on page 268 is a flow diagram for the Batch Backout utility.

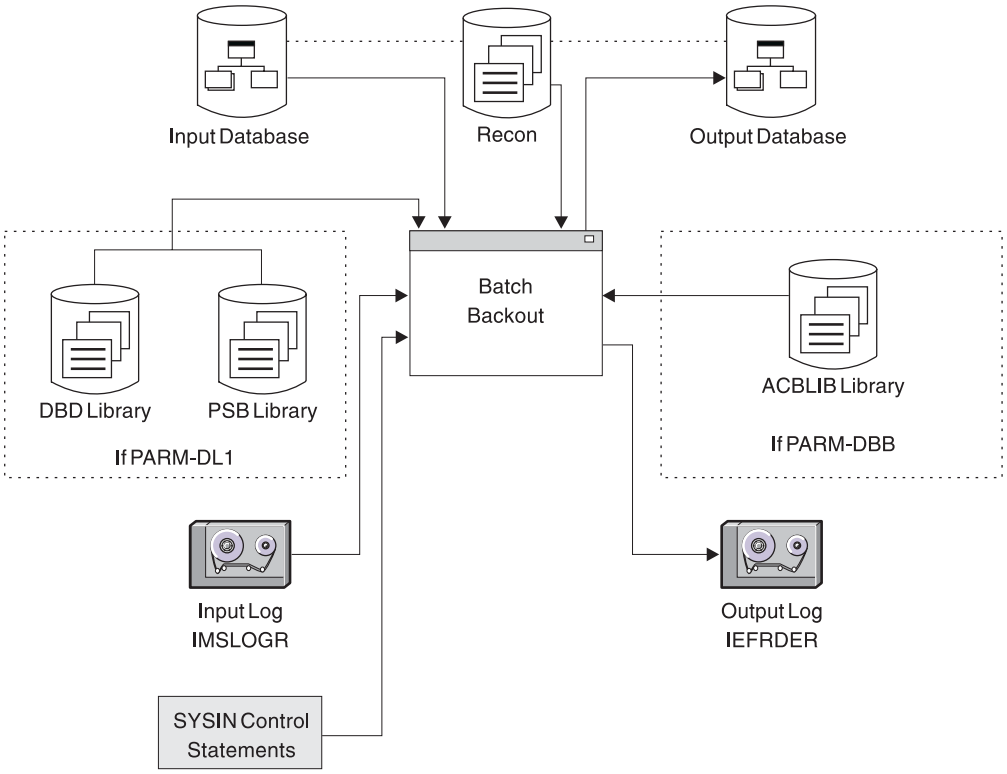


Figure 77. Data Set Requirements for the Batch Backout Utility

The following topics provide additional information:

- "Using the Batch Backout Utility in an RSR Environment"
- "Restrictions for DFSBBO00" on page 269
- "Input and Output for DFSBBO00" on page 270
- "JCL Requirements for DFSBBO00" on page 271
- "Utility Control Statements for DFSBBO00" on page 273
- "Return Codes for DFSBBO00" on page 277
- "Example of DFSBBO00" on page 279

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.

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 for DFSBBO00

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 these 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.
- You must run Batch Backout using a DLI region type to backout online reorganization PSBs. For more information on running the Batch Backout utility with HALDB Online Reorganization, see *IMS Version 9: Administration Guide: Database Manager*.

Input and Output for DFSBBO00

The input to the Batch Backout utility consists of:

- Log data sets (SLDS, OLDS, or both; tape, DASD, or both) 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 for DFSBBO00” on page 273 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 for DFSBBO00

The Batch Backout utility is executed as a standard z/OS job. The following JCL statements 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 Version 9: 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.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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.

Do not use FREE=CLOSE. After opening the input log file, it can be closed and re-opened. FREE=CLOSE causes the re-opening to fail.

IMSLOGxx
Describes additional input logs. The data sets can be OLDS, SLDS, or both. 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: See “Defining the IMS Buffer Pools” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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

This data set 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 Version 9: 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: See the section about SB control statements in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for more information.

Utility Control Statements for DFSBBO00

Eight optional utility control statements can be used by the Batch Backout utility:

- ABEND
- ABENDMSG
- ACTIVE
- BYPASS LOGVER
- BYPASS SEQVER
- CHKPT
- COLDSTART
- NOREADBACK
- READBACK

You can provide control statements to the utility by including them in the SYSIN data set. See the SYSIN DD description in “DD Statements” on page 271.

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 as follows:

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 as follows:

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 as follows:

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 as follows:

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 as follows:

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 as follows:

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 in-doubt 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 in-doubt 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 in-doubt UORs to pass to DBRC.

This utility control statement has a fixed format using the positions described as follows:

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.

NOREADBACK Statement

Use the NOREADBACK statement for batch input logs to buffer the DB update log records, regardless of origin, during the initial forward read of the input and suppress the backward read of the input. NOREADBACK allows Batch Backout to use both dataspace storage and local storage for this buffering.

The NOREADBACK statement is not used for online input logs.

Position	Description
1-10	Statement ID Positions 1 through 10 must be the characters NOREADBACK. These characters define the control statement.
11	This position must be blank.
11-80	Positions 11 to 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.

For IMS batch input logs on DASD devices, the READBACK statement is required if reading the logs backwards is desired. The READBACK statement is not needed where the logs are on tape, because those jobs use read backward only. The use of READBACK when reading logs from tape devices might degrade I/O performance.

Attention: The use of the default action when reading very large log data sets might result in running out of virtual storage and consequently abending.

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 for DFSBBO00

The Batch Backout utility provides the following return codes. Each return code causes a message to be printed. The following messages correspond to each return code:

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 the correct ddname 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 of DFSBBO00

The following example shows the JCL to backout of database changes made by a program that uses PSB PLVAPZ12.

```
//*
//EXAMPLE EXEC PGM=DFSRRC00,
//            PARM='DBB,DFSBBO00,PLVAPZ12,008,1,,,,,,,IMSS,,Y,Y,IRLM'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSABEND  DD SYSOUT=A
//IMSACB    DD DSN=IMS.ACBLIB,DISP=SHR
//IEFRDER   DD DSN=LGBKOUTF,DISP=(NEW,KEEP),
//             UNIT=SYSDA,VOL=SER=000000,
//             DCB=(RECFM=VB,BLKSIZE=8192,LRECL=8188,BUFNO=12),
//             SPACE=(CYL,(1,1))
//IMSLOGR   DD DSN=DSHR.OLDSP0,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSQAD
//IMSLOG00  DD DSN=DSHR.OLDSP1,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSQAD
//IMSLOG01  DD DSN=DSHR.OLDSP2,DISP=OLD,
//             UNIT=SYSDA,VOL=SER=IMSQAD
//DFSVSAMP  DD *
VSRBF=512,50
VSRBF=1024,50
VSRBF=2048,50
//DBHVSAM1  DD DSN=DSHR.FJVHSG1K,DISP=SHR
//DBHVSAM2  DD DSN=DSHR.FJVHSG1E,DISP=SHR
//HIDAM     DD DSN=DSHR.FKVHIG1E,DISP=SHR
//HIDAM2    DD DSN=DSHR.FKVHIG2E,DISP=SHR
//XDLBT04I  DD DSN=DSHR.FKVHIIXK,DISP=SHR
//RECON1    DD DSN=IMS.RECON1,DISP=SHR
//RECON2    DD DSN=IMS.RECON2,DISP=SHR
//RECON3    DD DSN=IMS.RECON3,DISP=SHR
```

Chapter 27. 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 a z/OS 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.

The following topics provide additional information:

- “Input and Output for DBFDBDR0” on page 282
- “JCL Requirements for DBFDBDR0” on page 283

Input and Output for DBFDBDR0

The information in Table 16 identifies inputs and outputs for the MSDB Dump Recovery utility.

Table 16. Data Set Requirements for the MSDB Dump Recovery Utility

Input	Output
MSDBCP1, MSDBCP2, MSDBCP3, MSDBCP4, or MSDBDUMP	New MSDBINIT
Control statements	Messages
SLDS	

Figure 78 shows a flow diagram of the data sets used by the MSDB Dump Recovery utility for unloading and reconstructing MSDBs.

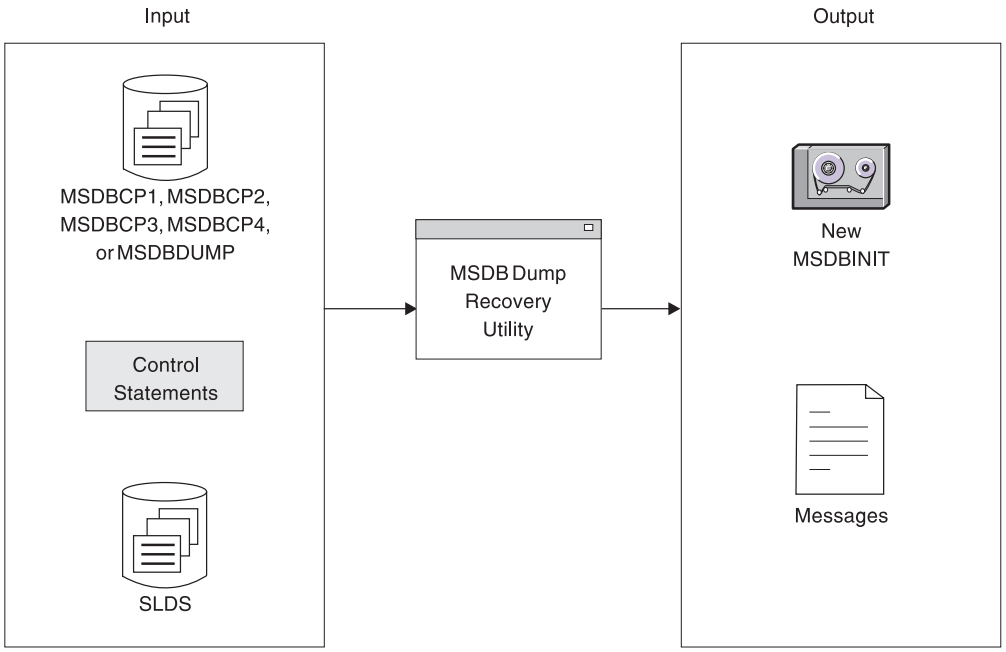


Figure 78. MSDB Dump Recovery Utility Input and Output Data Sets

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 for DBFDBDR0” on page 284.

The primary output is a new MSDBINIT data set. A message data set is also produced.

JCL Requirements for DBFDBDR0

The following statements 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.SDFSRESL, 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.

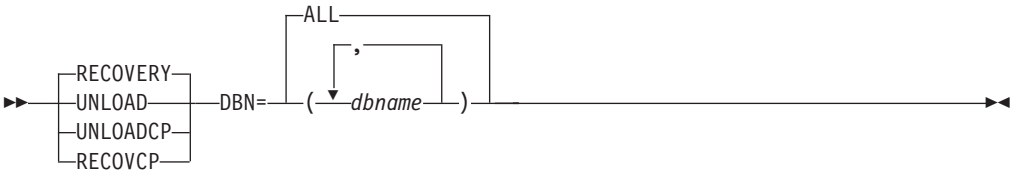
MSDBINIT DD
Describes the data set that contains the unloaded or reconstructed MSDBs after the utility executes.

Utility Control Statements for DBFDBDR0

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



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 a z/OS 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 a z/OS sequential data set (MSDBINIT). No updates from the log are applied.

RECOVCP
Specifies that a failing MSDBCPx data set is to be recreated from the remaining error-free MSDBCPx data set; or, in the case of an XRF-generated IMS system, from the MSDBCPx data set that contains the most recent check-pointed data.

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 for DBFDBDR0

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 of DBFDBDR0

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.SDFSRESL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSN=IMS.MSDBLM02,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBDUMP  DD DSN=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 DSN=IMS.SDFSRESL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSN=IMS.MSDBLM03,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBCP1   DD DSN=IMS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2   DD DSN=IMS.LMCP2,DISP=SHR      CHECKPOINT # 2
//IEFRDER   DD DSN=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 DSN=IMS.SDFSRESL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSN=IMS.MSDBLM02,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBDUMP  DD DSN=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 DSN=IMS.SDFSRESL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSN=IMS.MSDBLM03,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBCP1   DD DSN=IMS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2   DD DSN=IMS.LMCP2,DISP=SHR      CHECKPOINT # 2
//IEFRDER   DD DSN=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 DSN=IMS.SDFSRESL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//MSDBINIT  DD DSN=IMS.MSDBLM03,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=IMSDCL,
//             DCB=(BLKSIZE=13030,RECFM=VBT,LRECL=13026),
//             SPACE=(CYL,1)
//MSDBCP1   DD DSN=IMS.LMCP1,DISP=SHR      CHECKPOINT # 1
//MSDBCP2   DD DSN=IMS.LMCP2,DISP=SHR      CHECKPOINT # 2
//MSDBCP3   DD DSN=IMS.LMCP3,DISP=SHR      CHECKPOINT # 3
//MSDBCP4   DD DSN=IMS.LMCP4,DISP=SHR      CHECKPOINT # 4
//IEFRDER   DD DSN=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 data set using the RECOVCP utility control statement.

```
//RECOVCP   EXEC PGM=DBFDBDR0
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
//MSDBPRT   DD SYSOUT=A
//*               MESSAGE PRINT FILE
//NEWCP     DD DSN=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 DSN=IMS.LMCP2,DISP=SHR     CHECKPOINT DS #2
//MSDBCP2   DD DSN=IMS.LMCP2,DISP=SHR     CHECKPOINT DS #2
//@      IF XRF ENVIRONMENT, UNCOMMENT NEXT TWO LINES
//@MSDBCP3  DD DSN=IMS.LMCP3,DISP=SHR    CHECKPOINT DS #3
//@MSDBCP4  DD DSN=IMS.LMCP4,DISP=SHR    CHECKPOINT DS #4
//IEFRDER   DD DSN=IMS.LMLOG,DISP=SHR     IMS OLDS/SLDS
//*                               RECOVER ALL MSDB'S
//MSDBCTL   DD *
   RECOVCP
/*
```

Chapter 28. 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. 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 a z/OS 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.

The following topics provide additional information:

Restrictions for DBFUMRI0

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 for DBFUMRI0

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 539 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 for DBFUMRI0

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, the system logs, or a combination for recovery purposes.

JCL Requirements for DBFUMRI0

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 “FPUTIL Procedure” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for information on FPUTIL.
- See Appendix A, “Summary of DEDB Utility Commands,” on page 539 for more information on the DEDB online utility commands.

DD Statements

STEPLIB DD
Describes the library that contains the DEDB Data Set Create utility.

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 539 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 of DBFUMRI0

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.

```
//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.

```
//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.

```
//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 29. 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.

The following topics provide additional information:

Restrictions for DBFUMMH0

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 for DBFUMMH0

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 539 for more details on DEDB online utility commands.

The DEDB Area Data Set Compare utility produces the following output:

- A printed dump of up to 10 unmatched records on the SYSPRINT data set
- 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 for DBFUMMH0

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 for DBFUMMH0

The following statements 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:

- See Appendix A, “Summary of DEDB Utility Commands,” on page 539 for more information on the DEDB online utilities commands.
- See “FPUTIL Procedure” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for more information on FPUTIL.

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 539 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 of DBFUMMH0

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.

```
//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.

```
//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.

```
//UTL103 EXEC FPUTIL,DBD=DEDBJN23
//*
//*      DEDBJN23 IS THE TARGET DATABASE
//*
//SYSIN    DD *
```

```
TYPE COMPARE
AREA DB23AR2
GO
/*
```

Example 4

Example 4 shows 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

Example 5 shows a 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 5. Report and Test Utilities

Chapter 30. 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: See Chapter 31, “Interpreting DB-Monitor Reports,” on page 303 for examples of the output of each of these reports.

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 Version 9: 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.

The following topics provide additional information:

Restrictions for DFSUTR30

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 for DFSUTR30

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
```

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, 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 for DFSUTR30

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 of DFSUTR30

Figure 79 shows the JCL for a complete set of reports on the first trace interval from a tape with a serial number of IMSDA1:

```
//*
//STEP1   EXEC PGM=DFSUTR30
//STEPLIB   DD DSNAME=IMS.SDFSRESL,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
/*
```

Figure 79. JCL for Reports on the First Trace Interval

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 shown in Figure 80 on page 301:

```
//ANALYSIS DD  *
DISTRIBUTION
FILE=2
D13 ,,2,4,6,8,10,12
```

Figure 80. Modified Analysis Control Data Set

In the example the first two intervals are not changed, but remain as 0 and 1, respectively.

Chapter 31. Interpreting DB-Monitor Reports

This chapter describes the DB-Monitor reports and how to use them.

The following topics provide additional information:

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 81 on page 304 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 81. 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.

SHARED RESOURCE POOL TYPE
This field indicates whether this subpool contains index (I) or data (D) buffers.

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.

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 input for VSAM and OSAM buffers and options
⋮
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 the four VSAM-Buffer-Pool reports displayed in Figure 82, Figure 83, Figure 84, and Figure 85. The headings in each report display as shown in the examples.

```
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
```

Figure 82. Report #1

```
SHARED RESOURCE POOL ID:     BB
SHARED RESOURCE POOL TYPE:     D
SUBPOOL ID:            2
SUBPOOL BUFFER SIZE:      8192
HIPERSPACE BUFFERS:        20
TOTAL BUFFERS IN SUBPOOL:      4
```

Figure 83. Report #2

```
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
```

Figure 84. Report #3

```
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
```

Figure 85. Report #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 or PHDAM 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.

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 DATA SET 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 86 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 86. 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 or PHDAM database uses a series of RBA calls to search for the dependent segment, one call for each direct-address pointer it follows.

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 or PHIDAM database causes DL/I to issue a retrieve by key call to access the index segment pointing to the requested root segment.

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, HIDAM or PHIDAM 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 and PHIDAM 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 DATA SET 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 319) 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 87 on page 315 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 87. 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 88 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 88. 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 322.

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 89 on page 320 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  2:43:07 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 89. 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” on page 322.

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

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, see Figure 90. Suppose in a DL/I-Call-Summary report, the row of information in Figure 90 appears:

```
                                        (C)
PCB       CALL     LEV NO.       DL/I   IWAITS          DISTRIB.
NAME      FUNC     SEGMENT       CALLS  CALL     ...    NUMBER
----      ----     -------       -----  ------   ----   --------
DHVBT203  DLET     (01)A1111111  11     8.63             11C
```

Figure 90. Example of a Distribution-Appendix Report Format

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 91 on page 324), 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 91 on page 324 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, and so on) or a number and a character (3A, 3B, and so on). 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 ranges are made appropriate to the event. For example, ranges 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, and so on IWAITs.) Ranges such as 0, 1000, 2000, and so on 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, and so on numbers are in microseconds. A microsecond is one millionth of a second (in other words, 2000 microseconds equals 0.002000 second). The ranges have default values, but you can redefine the ranges. See “Redefining the Default Ranges” on page 325 for more information.
- The number below the line is always the number for the event being distributed.

```
IMS MONITOR   ****DISTRIBUTION APPENDIX****    RACE 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          0          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 91. 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 17 shows the events that can be distributed in the Program-I/O and DL/I-Call-Summary reports. Each event has an ID.

Table 17. 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 18 shows, using this ID, what each predefined set of ranges consists of when the default ranges are used.

Table 18. 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 Version 9: 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 92 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 92. 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 32. 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.

Related Reading: You can use Knowledge-Based Log Analysis (KBLA) to build JCL and execute DFSPIRP0. See the information about using KBLA to run a job against IMS log records in *IMS Version 9: Utilities Reference: System* for more detail.

The following topics provide additional information:

- "Input and Output for DFSPIRP0"
- "JCL Requirements for DFSPIRP0" on page 328
- "Utility Control Statement for DFSPIRP0" on page 329
- "Example of DFSPIRP0" on page 330

Input and Output for DFSPIRP0

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 for DFSPIRP0

The following are required:

- A JOBLIB or STEPLIB DD statement
- An EXEC statement
- DD statements specifying input and output data sets

JOBLIB or STEPLIB DD Statement

The JOBLIB or STEPLIB DD statement describes the program library containing the utility program. The format of this statement is:

```
//JOBLIB DD DSNAME=IMS.SDFSRESL,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.

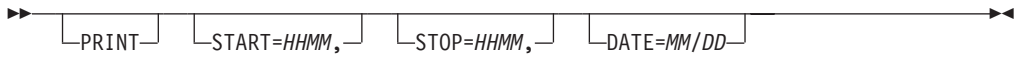
SYSPRINT DD
Describes the system messages data set. The format is:

```
//SYSPRINT DD SYSOUT=A
```

SYSIN DD
Describes the data set containing the optional utility control statement. 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 for DFSPIRP0

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:



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 of DFSPIRP0

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.

```
//JOBLIB    DD DSNAME=IMS.SDFSRESL,DISP=SHR
//*
//        EXEC PGM=DFSPIRP0
//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
/*
```

Figure 93. JCL to Request a Report of All Enqueues

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
```

Figure 94 shows examples that 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
```

Figure 94. Examples of Additional Report Headings

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 33. 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: See "SB Image Capture (SBIC) Control Statement" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for more information about the SBIC control statement.

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 336.

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 336.

Because the SB Test utility can be used to recreate and re-execute 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 re-execution 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: See "Interpreting //DFSSTAT Reports" in *IMS Version 9: Utilities Reference: System* for more information about invalid SB buffers.

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.

The information in Table 19 identifies the inputs and outputs used by the SB Test utility.

Table 19. Input to and output from the SB Test Utility

Input	Output
RECON	DFSSTAT report
SB image capture log records	SYSPRINT output messages
databases	
DBDLIB, PSBLIB, or ACBLIB (depending on region type)	
SB control statements	
Utility control statements	

Figure 95 depicts the data set requirements for the SB Test utility.

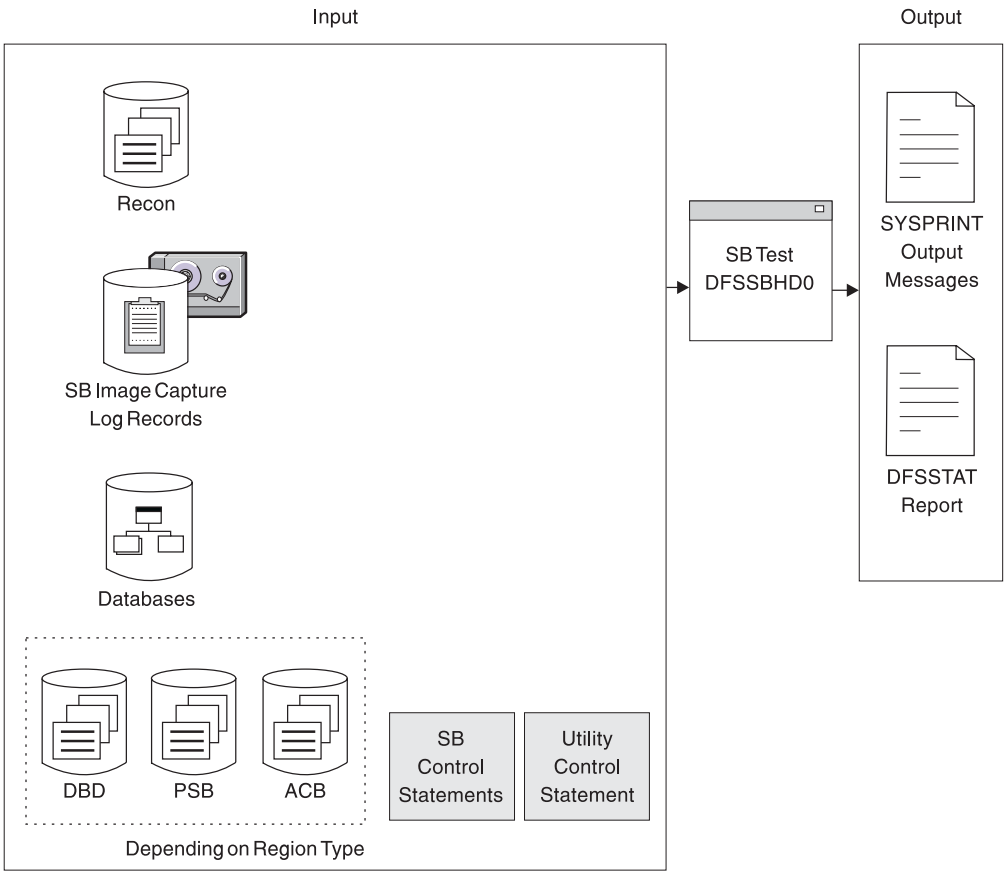


Figure 95. Data Set Requirements for the SB Test Utility

The following topics provide additional information:

- "Restrictions for DFSSBHD0" on page 333
- "Input and Output for DFSSBHD0" on page 333
- "JCL Requirements for DFSSBHD0" on page 333
- "Utility Control Statements for DFSSBHD0" on page 336
- "Example of DFSSBHD0" on page 337

Restrictions for DFSSBHD0

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 for DFSSBHD0

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 for DFSSBHD0” on page 336.
- 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: See “Interpreting //DFSSTAT Reports” in *IMS Version 9: Utilities Reference: System* for a detailed description of DFSSTAT reports.

JCL Requirements for DFSSBHD0

The SB Test utility is executed as a standard z/OS job. The following statements 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 Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on executing a batch processing region.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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 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: See “Specifying the IMS Buffer Pools” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure.

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: See the section about SB control statements in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for more information.

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 for DFSSBHD0

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:

YES
Specifies that the SB buffer handler issues DB Read I/Os.

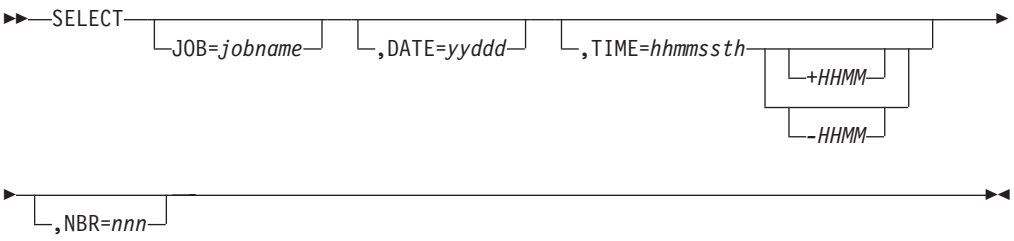
When YES is specified, only those OSAM Read I/O operations issued by the OSAM buffer handler are issued. I/O operations to VSAM database data sets and output DB I/O operations are not issued during execution of this utility.

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 of DFSSBHD0

Figure 96 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
/*
```

Figure 96. JCL for Execution of the SB Test Utility

Part 6. Utility Control Facility

Chapter 34. Utility Control Facility (DFSUCF00)

The Utility Control Facility (UCF) 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 free-form 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 for DFSUCF00” on page 343.)

The following topics provide additional information:

Restrictions for DFSUCF00

The following restrictions apply to the Utility Control Facility:

- The UCF does not support HALDB databases.
- 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, and so on). Replace "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 for DFSUCF00

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 statement 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.

The control statements are executed within a function 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 standalone utilities statistics and summary reports are part of the UCF output. Where options to suppress statistics exist within the utilities, the REQUEST keyword values STATS and NOSTATS determine whether or not statistics are printed.

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 for DFSUCF00

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 contains a 4-word parameter list as follows:
                      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 contains the 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 from 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 non-sequenced 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 check-pointed 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 check-pointed 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 for DFSUCF00" on page 341.

For information about writing an exit routine, see the *IMS Version 9: Customization Guide*.

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. One complete database record must be written before DFSUCP90 will honor a checkpoint request by the user exit routine. The presence of a second database record indicates that the first record is complete, and DFSUCP90 can be run successfully.

The initial load exit routine is called only once per database record after the last segment in the record has been inserted. Code other than checkpoint logic must be on a database record boundary so that the next record can be inserted. If a restart occurs, it must be from data written on the record boundary.

The interface to the exit routine is:

```
Register 1 contains a 3-word parameter list as follows:
                      1st word = common area defined by DSECT UCFCMVEC
                      2nd word = DFSPRINT data set
                      3rd word = PST
```

The common area includes fields U7CURCKP and U7CKPTNK. The U7CURCKP field contains the checkpoint intervals currently in effect. The U7CKPTNK field serves as a 4-byte counter and contains the number of records to be processed before a checkpoint is taken.

You can synchronize (in whatever manner desired) check-pointing of individual data sets with checkpoints currently in effect for the Initial Load program by doing one of the following:

- Monitoring the count and check-pointing data sets when the counter (U7CKPTNK) reaches zero
- Forcing a checkpoint by clearing the counter to hexadecimal zeros and check-pointing 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.

The Initial Load Exit Routine is only valid for VSAM data sets. If a user exit is specified for OSAM, then the exit routine is ignored.

End of General-use programming interface

Termination/Error Processing for DFSUCF00

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 z/OS 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 of DFSUCF00” on page 387.

Restart Processing for DFSUCF00

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 for DFSUCF00

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 data addressed by the first parameter is an image copy record. On the first call, the exit is called with the IC header record.

Considerations for coding your user exit routine:

- To address the data, adjust for the RDW because it is not part of the data.
- To adjust the length of the file record, change the RDW.
- Your user exit routine will be called in 24 bit addressing mode, and its resident mode must be below the 16 MB line.

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.

Considerations for return code 8:

- For HD unload, return code 8 has no meaning and is ignored.
- To return to HD reload having entered at offset 0, process the unload record pointed to by register 1 before the current record. In this case, register 1 should point to the first data byte of the record (the first byte after the RDW).

  Recommendation: Do not modify the record passed to the user exit. This will result in user abend 0805. On the next call, HD reload will enter the exit routine at the same offset with the same record that was returned to it on the previous call.
- To return to HD reload having entered at offset 4, process normally.

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.

Related Reading: For information specific to a FUNCTION=IL (initial load) exit routine, see "Initial Load Exit Routine" on page 345.

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 356 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: See *z/OS MVS Diagnosis: Tools and Service Aids* for further information on SPZAP.

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 to Table 20 for a summary of when zaps occur.

Table 20. 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 Figure 97 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
```

Figure 97. Example of the Write-to-Operator-with-Reply Function

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. See Table 21 for a summary of information that you can enter.

Table 21. Control and Option Replies 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 388 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 for DFSUCF00

The Utility Control Facility is executed as a standard z/OS job. The following JCL statements 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 of DFSUCF00" on page 387.

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: See "The System Definition Process" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for further information on specifying PARM options.

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules. If STEPLIB is unauthorized by having unauthorized libraries concatenated to IMS.SDFSRESL, 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 and is always defined as DISP=(,KEEP). 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. For restart processing information, see DFSOCDS DD.

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 or OSAM data sets.

Related Reading: See “Tailoring the IMS System to Your Environment” in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for additional information on control statement format and buffer pool structure. 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 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 22 on page 355 summarizes the use of the FUNCTION keyword on the various DD statements.

Table 22. JCL DD Statements Summary Table

	FUNCTION= Keywords Used on Utility Control Statements																
DD 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
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 for DFSUCF00

The UCF control statements can be coded in a free-form 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 this 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

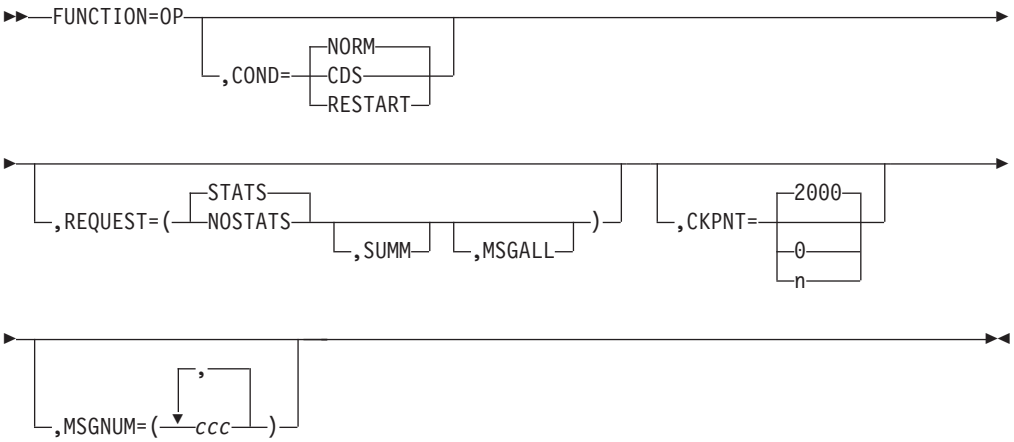
ZM Module Zaps

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:



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.

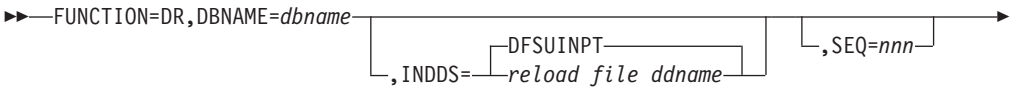
The FUNCTION=DR Statement

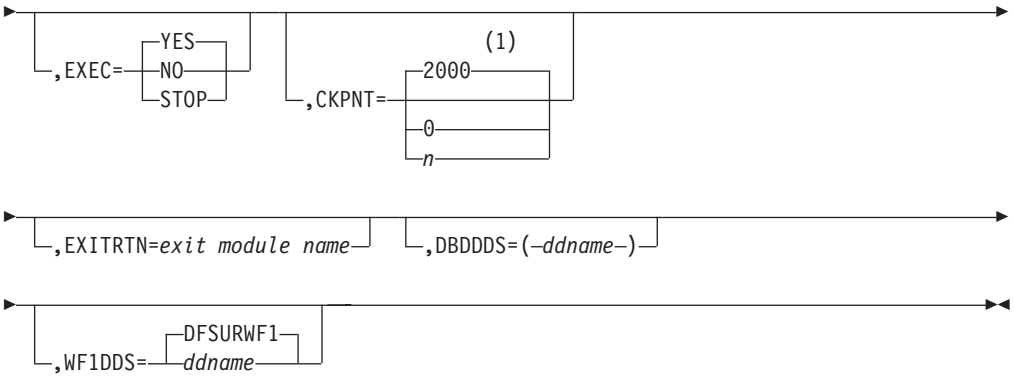
This control statement causes the UCF to execute the HD Reorganization Reload utility (DFSURGL0) to reload an HDAM or HIDAM database from a data set created by the HD Reorganization Unload utility (DFSURGU0).

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:





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:

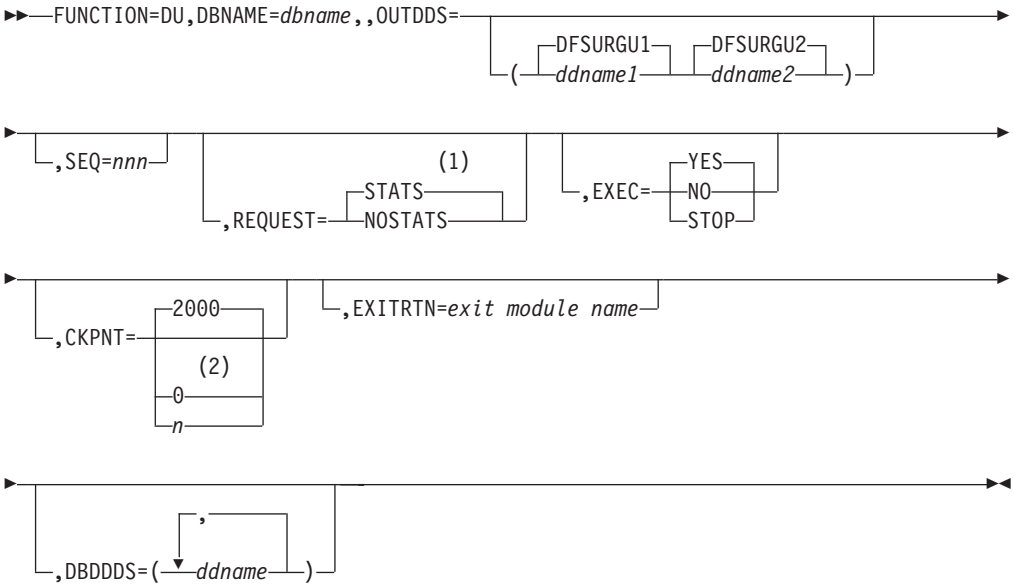
- Either counter, LT, or LP pointers are changed.
- The level of segments involved in logical relationships change from adding or deleting segments.
- The DBD name is changed and the DBD contains logical relationships.

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



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 for DFSUCF00” on page 341.

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.

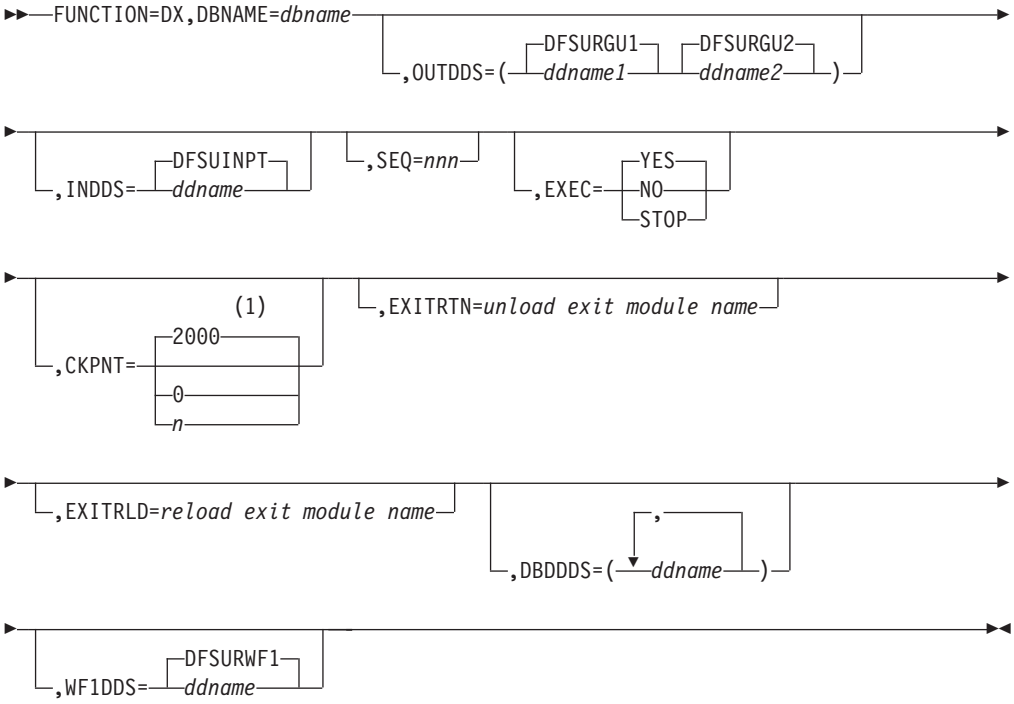
DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

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:



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 for DFSUCF00” on page 341.

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.

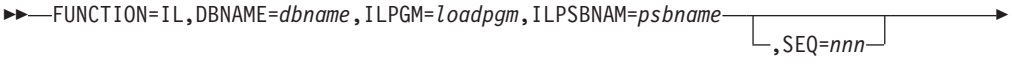
WF1DDS=
Defines the output work data set that is supplied as an input to the Prefix Resolution utility. The data set can reside on a tape or a direct-access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

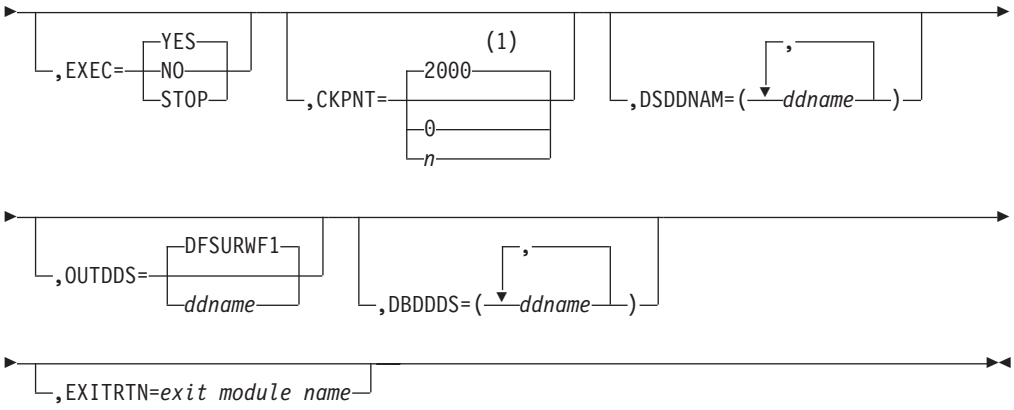
The FUNCTION=IL Statement

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 for DFSUCF00” on page 344.

Do not specify more than 20 ddnames for this function.

The format of the FUNCTION=IL control statement for your Initial Load program is:





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.

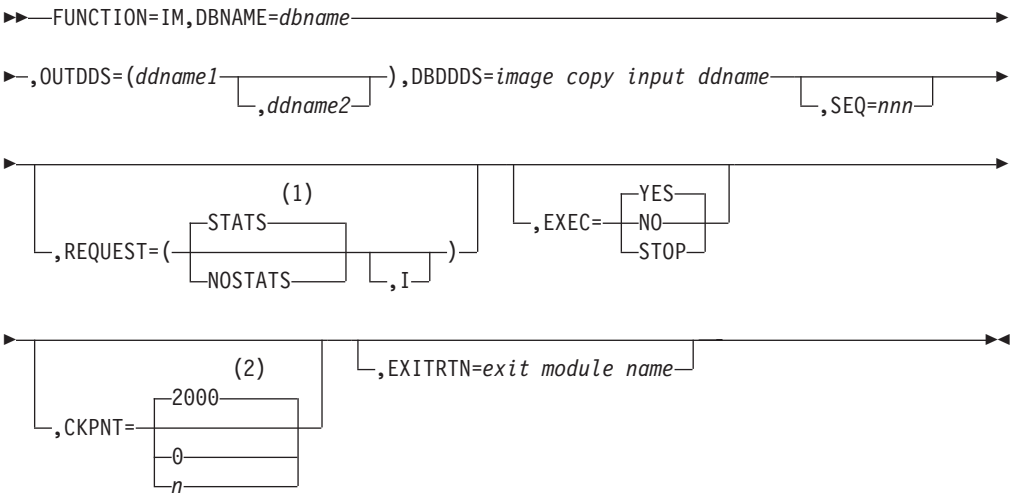
EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

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:



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 ddname of the database data set that is input to the image copy process. 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.

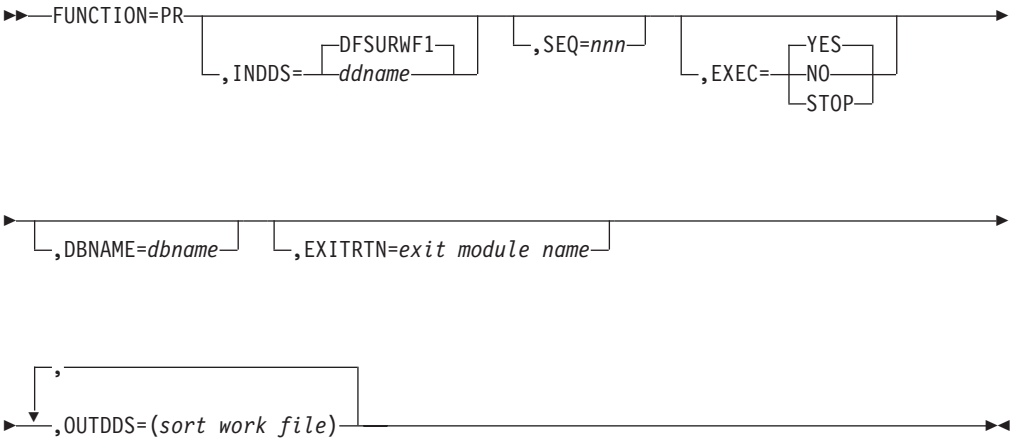
The content of this data set can be different when compared to the same data set generated by the Prefix Resolution utility run without UCF. This can occur if you have logically related databases and database records which are logical parents without logical children. In this case, some pointers will not require resolution and those records will be discarded. Since the non-UCF controlled utility run might require the coding of the DBIL control statement for the Prereorganization utility (DFSURPR0), the number and type of records discarded can 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.

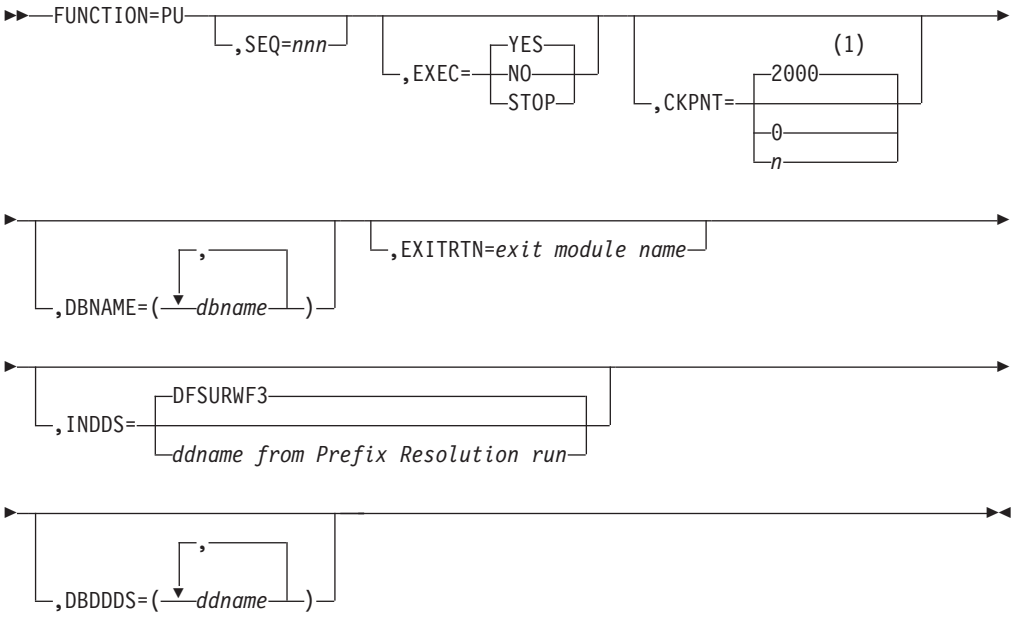
The FUNCTION=PU Statement

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:



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).

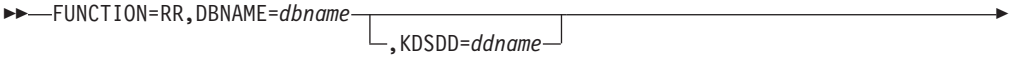
DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart processing begins to terminate the UCF.

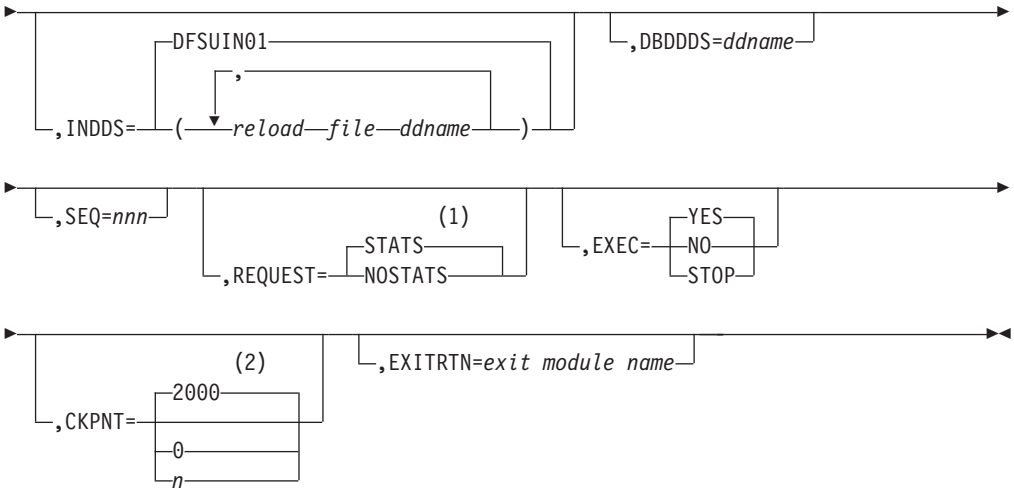
The FUNCTION=RR Statement

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:





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.

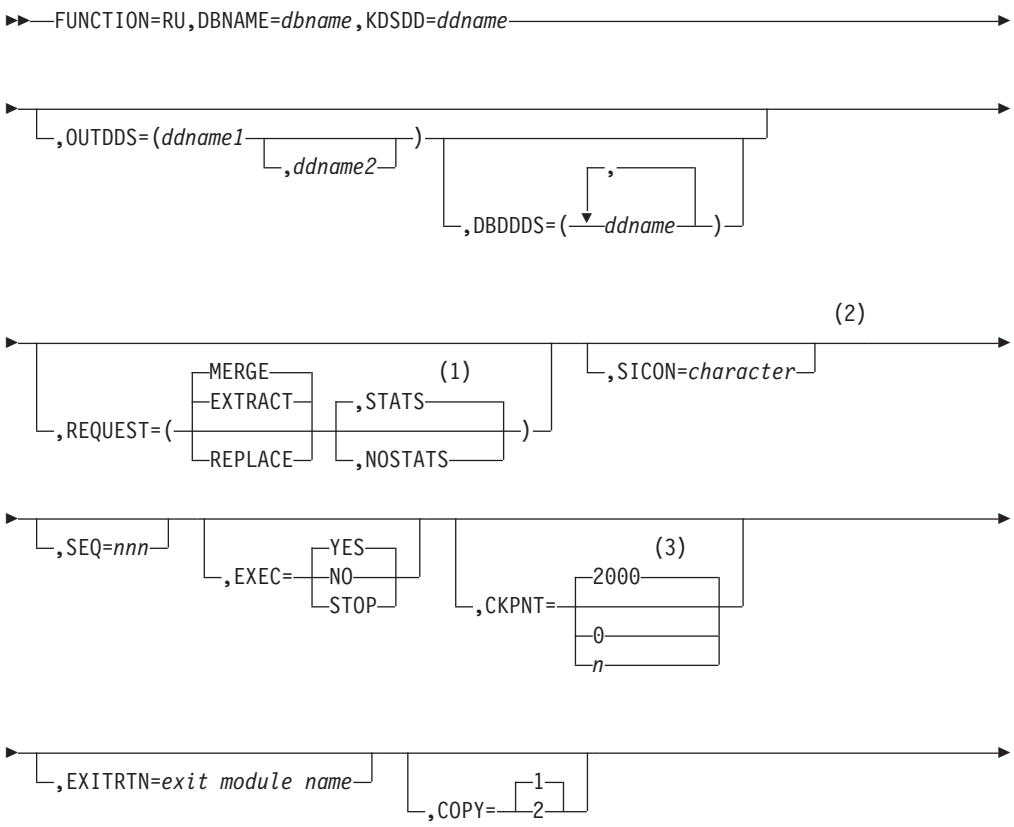
EXITRTN=
Specifies an exit routine to be given control in order to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

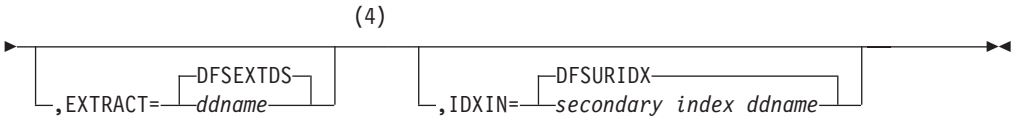
The FUNCTION=RU Statement

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

3 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.

4 Can only be specified with REQUEST=EXTRACT.

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.

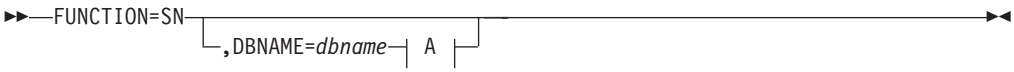
The FUNCTION=SN Statement

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.

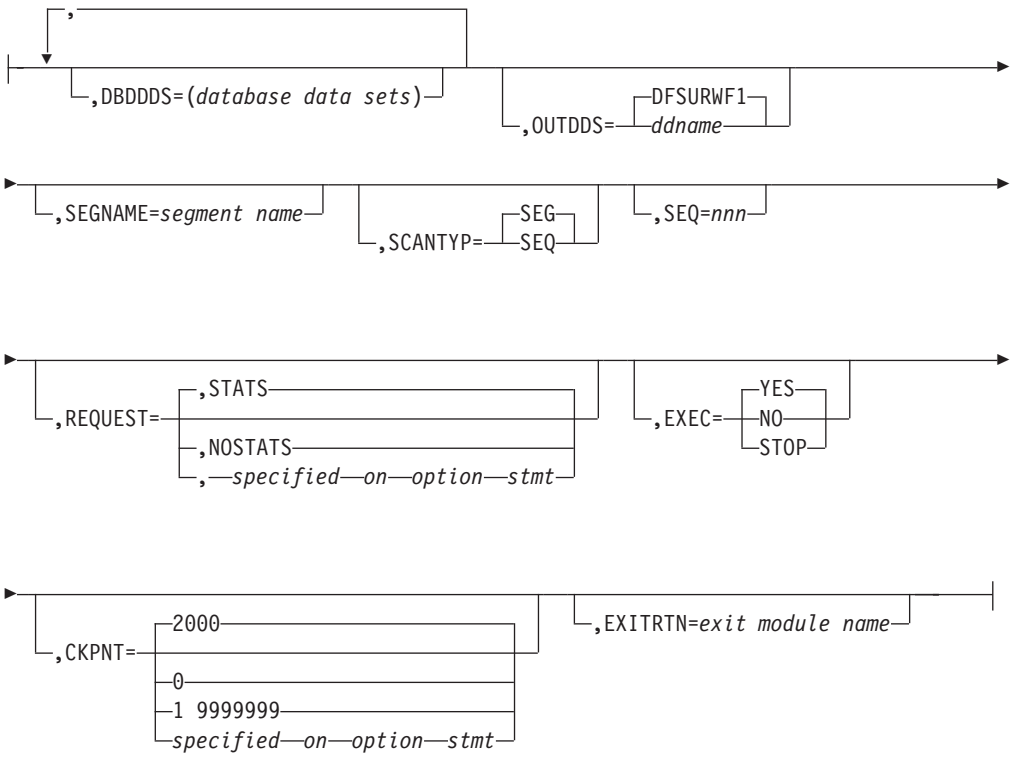
The SCAN function is executed for initial loads and HD Reloads. When SCAN is not required, the DFSURWF1 DD statement must be present unless EXEC=NO is specified on the FUNCTION=SN statement.

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:



A:



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.

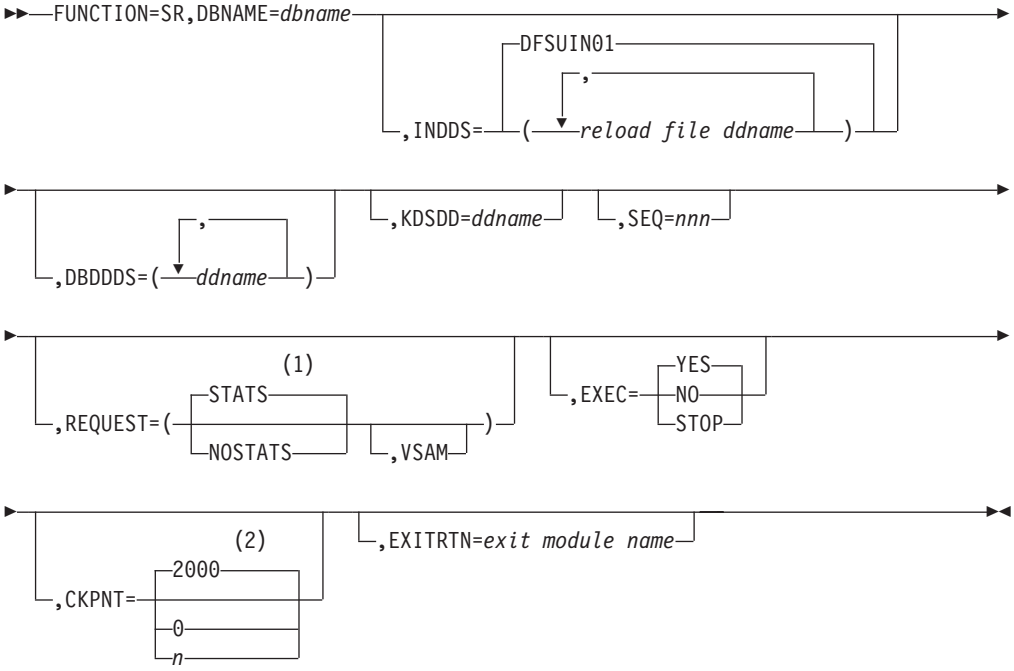
Reorganization of HISAM databases is significantly faster using the HISAM Unload/Reload utilities instead of the HD Unload/Reload utilities.

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:



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 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 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.

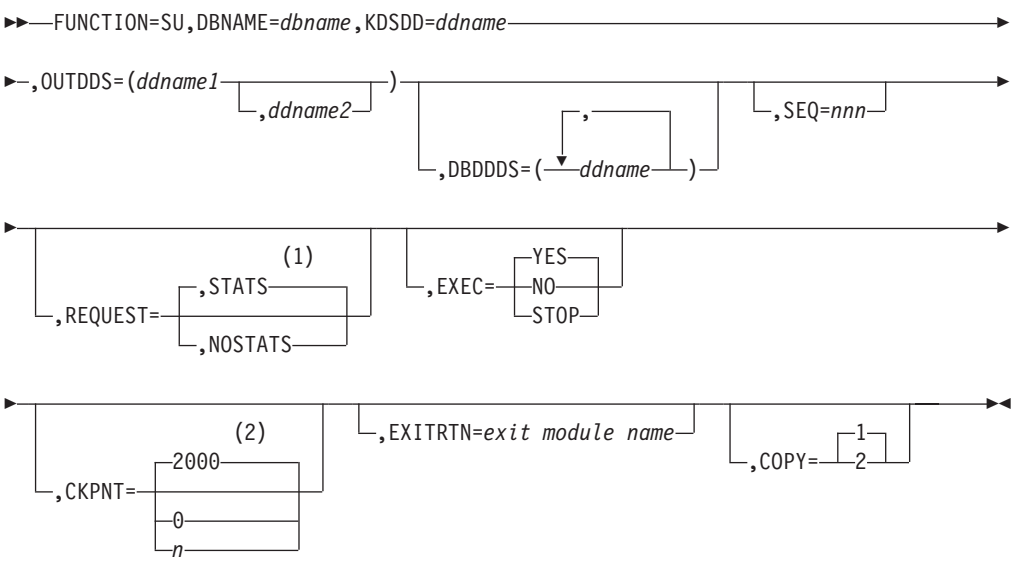
EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

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:



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 for DFSUCF00” on page 341.

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.

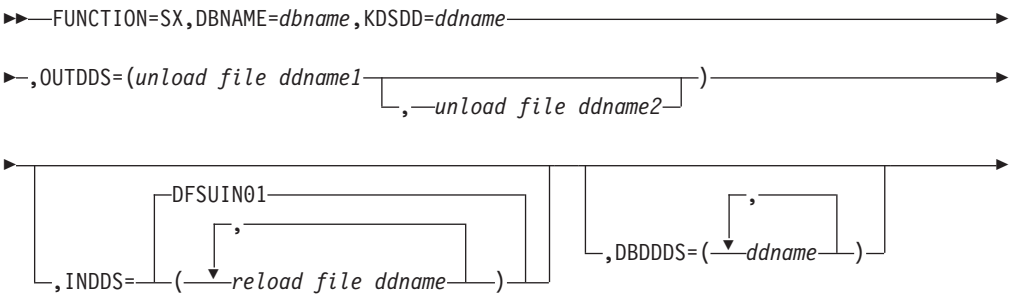
COPY=
Provides for multiple copies of output data sets. COPY can be specified only once for each control statement. COPY=1 produces only one copy and is the default. COPY=2 produces an additional output copy. Specify it if a ddname-2 is specified for the OUTDDS keyword. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.)

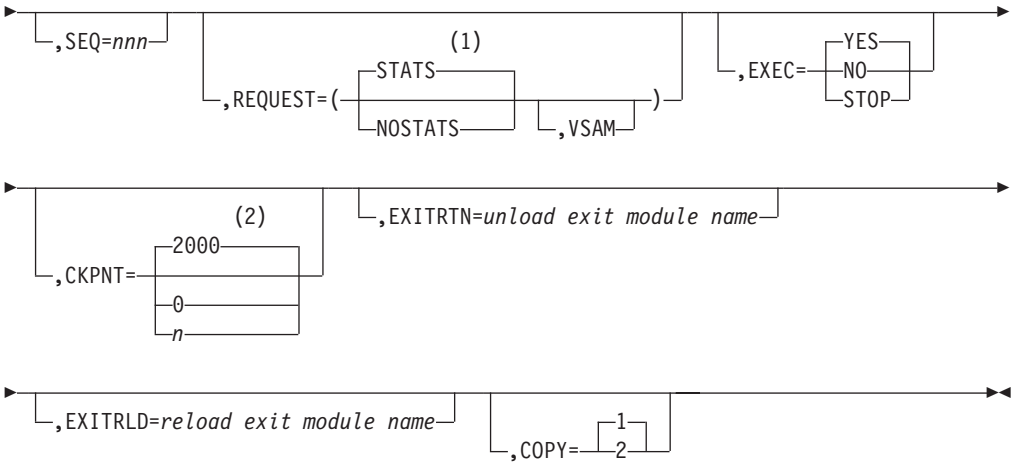
The FUNCTION=SX Statement

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:





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 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 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 for DFSUCF00” on page 341.

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 service aids SPZAP program are supported.

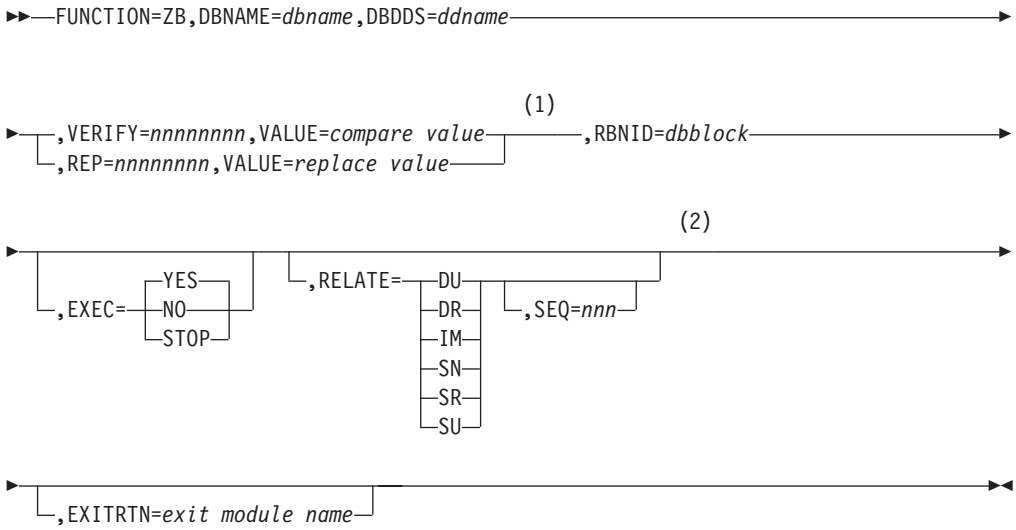
Restriction: This utility function should be used only by an IBM Field Engineering Program Support Representative or by someone under that person's direction.

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:



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.

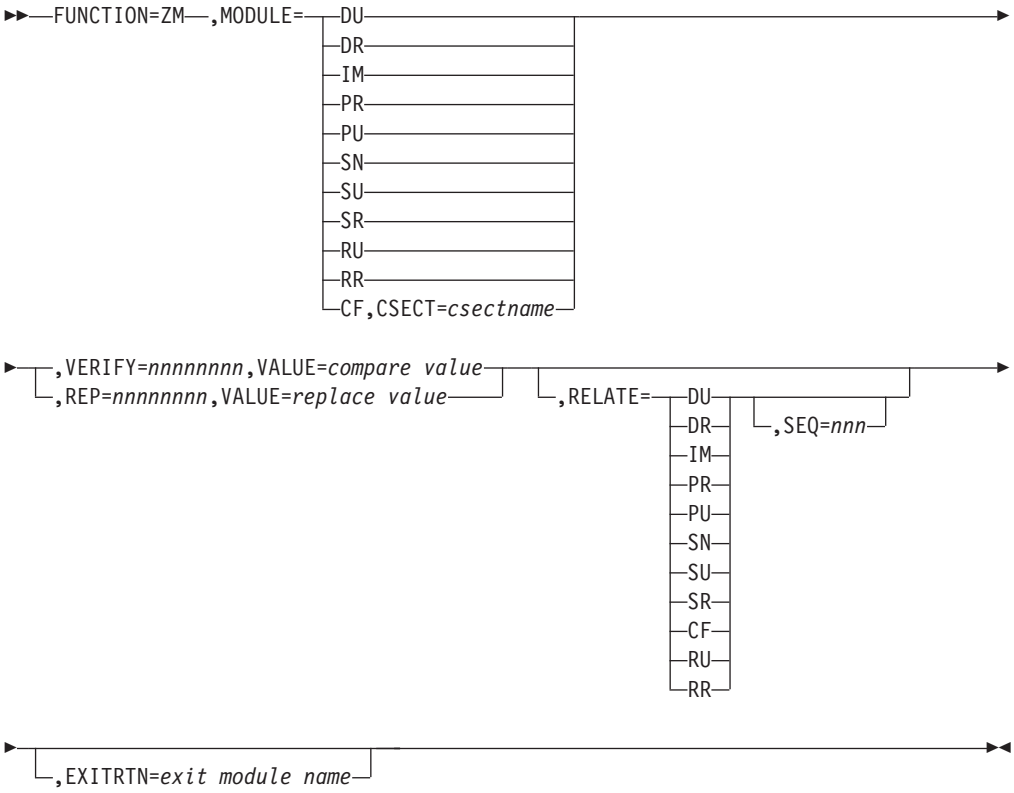
EXITRTN=
Use to obtain control at the time the block is in storage to verify that the zap was performed, and, if desired, to compile statistics on the block.

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 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:



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 system definition 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 for DFSUCF00

Table 23 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 23. UCF FUNCTION Summary Table

		Function															
Keyword	**Times 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	
COPY	N										1			1	1		
EXTRACT	N										1						
IDXIN	N										1						
ILPSBNAM	N					1											
MODULE	N																1
CSECT	N																1
VERIFY	N															E	E

Table 23. 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
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 24 shows the minimum keywords that you must specify when constructing each type of control statement.

Table 24. UCF-Required Keywords by Function

Keyword	Function															
	OP	DR	DU	DX	IL	IM	PR	PU	RR	RU	SN	SR	SU	SX	ZB	ZM
FUNCTION	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DBNAME		X	X	X	X	X			X	X	X	X	X	X	X	
KDSDD										X			X	X		
ILPGM					X									X		
OUTDDS						X				X			X			
INDDS														X		
LOGIN																
SEGNAME											X					
DBDDDS						X									X	
ILPSENAM					X											
MODULE																X
VERIFY															X	X
REP															X	X
VALUE															X	X
RBNID															X	

Return Codes for DFSUCF00

Upon termination of the UCF, the return codes in Table 25 are passed to register 15.

Table 25. UCF Return Codes

Code	Meaning	Action
0	Processing was normal.	DFSNJRNL and DFSNCDS data sets can be scratched.

Table 25. UCF Return Codes (continued)

Code	Meaning	Action
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 of DFSUCF00

The following examples show the use of the Utility Control Facility.

Example 1

Figure 98 on page 388 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.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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=4008,LRECL=4000)
//DFSOJRNL  DD DUMMY
//DFSNCDS   DD DSNAME=NCDS,DISP=(,KEEP),
//             UNIT=SYSDA,SPACE=(CYL,(2,2)),DCB=BLKSIZE=1600
//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.
```

Figure 98. JCL Required to Execute the UCF

Example 2

Figure 99 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.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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)),DCB=(RECFM=B,BLKSIZE=4008,LRECL=4000)
//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.
```

Figure 99. JCL to Execute the UCF in a Restart Situation

Example 3

Figure 100 on page 389 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.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSNAME=IMS.SDFSRESL,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)),DCB=(RECFM=B,BLKSIZE=4008,LRECL=4000)
//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.
```

Figure 100. JCL to Execute the UCF in a Restart Situation using Control Statements

Part 7. Generation Utilities

Chapter 35. MFS Language Utility (DFSUPAA0)

Use the MFS Language utility (DFSUPAA0) to create and store the message format service (MFS) control blocks. The intermediate text block (ITB) form of the control blocks is placed in the IMS.REFERAL library. The control blocks are placed in the IMS.FORMAT library for use during normal IMS operation.

Definition: One format and all the messages that refer to it in their SOR= operand make up a *format set*.

The MFS Language utility has three modes of operation: standard, test, and batch. In all three modes, the utility is executed offline, accepts the same control statements, and produces the same kinds of ITBs and control blocks. The modes differ in their use of the MFS libraries. Accordingly, they use different procedures.

In standard mode, ITBs written in IMS.REFERAL are converted to control blocks and placed in the staging library, IMS.FORMAT, by the MFSUTL procedure. Because the control blocks are placed in the staging library and not the active library, the standard mode can run concurrently with the online IMS control region.

Batch mode differs from standard mode, in that the MFSBTCH1 procedure places the created control blocks in a special library, IMS.MFSBATCH, for later transfer by the MFSBTCH2 procedure (in another job) to the staging library, IMS.FORMAT.

In test mode, the MFSTEST procedure creates control blocks and places them in a separate IMS.TFORMAT library. The control blocks can be tested without interfering with online operation and can operate concurrently with the online IMS control region.

The MFSTEST procedure should not be executed concurrently with itself or any other program or procedure that utilizes the MFS libraries. To test the control blocks in IMS.TFORMAT, the terminal operator enters the `/TEST MFS` command. Then, test control blocks from IMS.TFORMAT (as well as online control blocks from the active format library, if necessary) are read into a buffer for test operation. After successful testing, the control blocks can be placed in the staging IMS.FORMAT library by recompiling the source statements using the MFSUTL procedure.

Stage 2 of IMS system definition generates the following procedures:

MFSUTL A two-step standard mode execution procedure of the MFS Language utility for creating MFS online control blocks and placing these blocks into the IMS.FORMAT library.

MFSBTCH1 A one-step batch mode execution procedure of the MFS Language utility for creating and accumulating MFS online blocks.

MFSBTCH2 A one-step batch mode execution procedure of the MFS Language utility for placing the accumulated MFS online control blocks (from MFSBTCH1) into the IMS.FORMAT library.

MFSBACK A two-step execution procedure to back up the MFS libraries. If the optional MFSTEST facility is used, MFSBACK contains an additional step to back up the test library.

MFSREST A two-step execution procedure to restore the MFS libraries. If the optional MFSTEST facility is used, MFSREST contains an additional step to restore the test library.

MFSRVC A one-step execution procedure for maintaining the MFS libraries.

If MFSTEST mode is selected during system definition, an additional procedure is generated:

MFSTEST A two-step test mode execution procedure of the MFS Language utility for creating MFS online blocks and placing them into the IMS.TFORMAT library.

In addition to the procedures for creating new or replacement control blocks, the MFS Language utility includes MFSBACK and MFSREST procedures for backup and restore operations in MFS libraries.

Delete and listing operations are performed by the service utility.

Related Reading:

- MFS control blocks are described in *IMS Version 9: Administration Guide: Transaction Manager*.
- The MFSRVC procedure is described in Chapter 37, “MFS Service Utility (DFSUTSA0),” on page 503.
- The Delete and listing operations performed by the service utility, are described in Chapter 37, “MFS Service Utility (DFSUTSA0),” on page 503.

The following topics provide additional information:

Standard Mode for DFSUPAA0

This section describes the two-step mode of operation using the MFSUTL procedure.

Figure 101 on page 395 shows an overview of the two-step standard mode of operation using the MFSUTL procedure. For the JCL for the MFSUTL procedure see “JCL Requirements” on page 398.

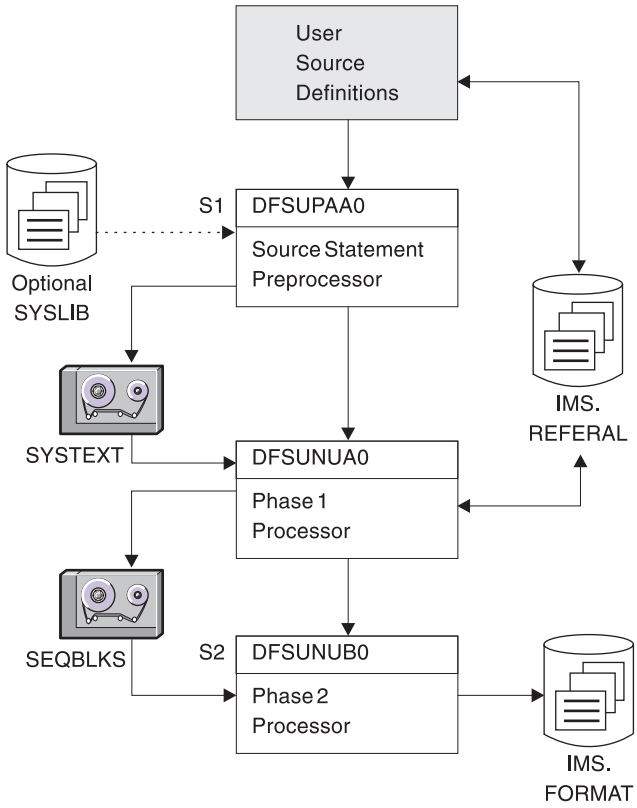


Figure 101. Overall Flow with the MFSUTL Procedure

The following sections discuss the details of Step 1 (S1) and Step 2 (S2).

MFSUTL Procedure Description

Step 1 (S1)

Source Statement Preprocessor: The MFS Language utility preprocessor provides a nonassembler, nonmacro preprocessor for the MFS Language source statements.

Related Reading: See information on utility control statements in *IMS Version 9: Administration Guide: Transaction Manager*.

The execution of the preprocessor generates intermediate text blocks (ITBs), which are then processed by the remaining utility phase to generate message (MSG) and format (FMT) control blocks. IMS uses the control blocks to format messages for device display and application program presentation.

The primary function of the preprocessor is to perform syntax and relational validity checks on user specifications. The preprocessor generates ITBs for each MSG, FMT, partition descriptor block (PDB), and TABLE source definition processed, and stores them in IMS.REFERAL.

The IMS.REFERAL partitioned data set (PDS) directory contains an entry for each MSG, FMT, PDB, and TABLE ITB. Additionally, the interrelationships between all known FMT and MSG ITBs that have been placed in the IMS.REFERAL are recorded in the PDS directory.

The preprocessor executes in the following order:

1. The preprocessor constructs a control table representing the interrelationships between all known FMT and MSG ITBs that have been placed in IMS.REFERAL. If you request the compress function in the EXEC statement, the preprocessor compresses IMS.REFERAL.
2. The preprocessor adds the user-supplied FMT and MSG definitions to the control table and resolves them against the table to ensure that, when a given definition is supplied for processing, all control blocks in the format set are reprocessed by phase 1. This resolution also allows you to compile source for only the message or format that requires change—not the entire format set.
3. After the resolution function has been accomplished to determine the definitions to be processed, the preprocessor places the control statements for these FMT and MSG definitions on the SYSTEXT data set for phase 1 processing.

If you change a PDB definition, each format set referencing that PDB might need to be recompiled.

The following actions cause phase 1 reprocessing of the format set ITBs to create new FMT and MSG control blocks for the IMS.FORMAT library:

- Modification of a FMT definition that already exists as a control block in IMS.FORMAT.
- Modification of a MSG definition that already exists as a control block in IMS.FORMAT.
- Addition of a new MSG to the format set.
- Reassignment of a MSG definition to a different FMT. This reassignment causes the old format set and the new format set to be reprocessed.

If a MSG definition refers to a FMT name (through the SOR= keyword) that has not yet been supplied to the MFS Language utility, the preprocessor stores the MSG ITB in the IMS.REFERAL library. The MSG control block is not created until a FMT definition is supplied. The format set is then processed to create the new MSG and FMT control blocks for IMS.FORMAT.

Similarly, if a FMT definition is supplied to the MFS Language utility and no MSGs refer to it, the FMT ITB is stored in IMS.REFERAL. The FMT control block is not created until at least one MSG definition is supplied. The format set is then processed to create the new MSG and FMT control blocks for IMS.FORMAT.

Each IMS error message is accompanied by a return code of 4, 8, 12, 16, or 20 to indicate increasing severity of error.

Related Reading: Refer to *IMS Version 9: Messages and Codes, Volume 1* for more information.

If an error condition is detected during the processing of statements that would create an FMT, MSG, PDB, or TABLE ITB, the highest such severity code associated with the message stating the error is kept and used to determine if the ITB is to be written to the IMS.REFERAL library. The preprocessor maintains the highest return code issued for each definition processed. You can specify a compare value (the lowest unacceptable return code) in the STOPRC parameter of the EXEC statement. If the return code is greater than or equal to the STOPRC value, the ITB is not written in IMS.REFERAL. For example, a STOPRC of 4

permits only ITBs that have a return code of 0 to be written. If no STOPRC is specified, a value of 8 is assumed, and only ITBs having a return code of 0 or 4 are written.

The preprocessor also maintains the highest return code for all ITBs processed during a job. Phase 1 is not given control by the preprocessor if the highest return code is greater than or equal to 16, or if no ITBs were written in IMS.REFERAL. If the return code is 16 or greater, the preprocessor returns control to z/OS with a completion code equal to the return code.

Phase 1 Processor: The preprocessor invokes the phase 1 processor. Initially, the phase 1 processor uses the control statements placed by the preprocessor on the SYSTEXT data set to construct a module table representing all of the FMT and MSG ITBs to be processed in this run. After constructing the module table, the phase 1 processor reads in ITBs from IMS.REFERAL and builds control blocks for each MSG and FMT definition. If a TABLE of control functions is requested by an input format definition, the phase 1 processor obtains the TABLE ITB from IMS.REFERAL and builds functions into the device input format (DIF).

When a format definition requests a HALDB partition set, the phase 1 processor gets the PDB ITB from IMS.REFERAL and builds the partitioning control functions into the device output format (DOF).

The phase 1 processor places the newly constructed control blocks on the SEQBLKS data set. Each member processed has a control record placed on the SEQBLKS data set identifying the member, its size, and the date and time of creation. This control record is followed by the image of the control block as constructed by the phase 1 processor.

If an error is detected during control block building, an error control record is placed on the SEQBLKS data set for the definition in error, identifying the member in error, and the date and time the error control record was created. In addition, the phase 1 processor returns a completion code of 12 to z/OS. If execution of step 2 is forced, the phase 2 processor deletes control blocks with build errors.

The phase 1 processor maintains a high return code for all ITBs processed during an execution of the MFS Language utility. Before returning to z/OS, the phase 1 processor compares its high return code to the preprocessor's high return code. The highest of the two is passed to z/OS as the completion code for step 1.

Step 2 (S2)

Phase 2 Processor: The phase 2 processor receives control as a job step when the phase 1 processor is finished. The phase 2 processor operates in a two-pass mode to place the new control blocks into the IMS.FORMAT library. On the first pass, the phase 2 processor reads the SEQBLKS data set and creates an internal table that contains the name of every MOD, MID, DOF, and DIF created by the phase 1 processor. The name of the format or message description that had build errors during the phase 1 processor's execution is also added to this internal table. Control blocks in the IMS.FORMAT library that are to be replaced in IMS.FORMAT, or had build errors during phase 1, are deleted from IMS.FORMAT.

If you request the compress function, the phase 2 processor compresses the IMS.FORMAT library. This ensures maximum available library space for adding control block members and, due to the reprocessing of all related members by the

phase 1 processor, allows the grouping of related control blocks for seek time reductions when fetching the control blocks during online execution.

In the second pass, the SEQBLKS data set is reprocessed, together with the module table, to write the new control blocks into IMS.FORMAT and STOW them for a directory update.

If control blocks with entries in the main-storage index directory, $$IMSDIR, were deleted and not replaced, the index entries should be deleted. This update can be done automatically, but it is inefficient for a large format library with a relatively small number of blocks deleted. To avoid this, the following parameter can be used for both the MFSUTL and MFSBTCH2 procedures:
DIRUPDT=UPDATE|NOUPDATE. The default is UPDATE. If UPDATE is specified or defaulted, $$IMSDIR is updated automatically. If NOUPDATE is specified, updating is bypassed; and you must delete the blocks from $$IMSDIR with the MFS Service Utilities

If index entries are to be added to $$IMSDIR for new control blocks created in this run, the INDEX function of the MFS service utility must be used.

The phase 2 processor passes a completion code to z/OS for step 2 based on all the control block maintenance to IMS.FORMAT for a given execution of the MFS Language utility.

JCL Requirements

The JCL for the MFSUTL is shown in Figure 102 on page 399. Refer to “JCL Parameter Descriptions for DFSUPAA0” on page 410 for details on the EXEC parameters and DD statements.

```
//         PROC RGN=4M,SOUT=A,SYS2=,
//            SNODE='IMS',
//            SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,COMPR2=COMPRESS,
//            LN=55,SN=8,DEVCHAR=0,COMPR3=NOCOMPREND,
//            DIRUPDT=UPDATE
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//            &COMPR,'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR')
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DISP=SHR,
//         DSN=&SNODE..&SOR.(&MBR)
//REFIN    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFRD    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY    DD DISP=SHR,
//         DSN=IMS.&SYS2.PROCLIB(REFCPY)
//UTPRINT  DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT&SOUT
//SEQBLKS  DD DSN=&&BLKS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
//            PARM=(&COMPR2,&COMPR3,&DIRUPDT,
//            'DEVCHAR=&DEVCHAR'),COND=(8,LT,S1)
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=IMS.&SYS2.FORMAT,DISP=SHR
//DUMMY    DD DISP=SHR,
//         DSN=IMS.&SYS2.PROCLIB(FMTCPY)
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Figure 102. MFSUTL Procedure

The DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported.

REFCPY Control Statement: The MFSUTL procedure uses this control statement to compress REFERAL.

```
COPY  INDD=REFOUT,OUTDD=REFOUT
```

FMTCPY Control Statement: The MFSUTL procedure uses this control statement to compress FORMAT.

```
COPY  INDD=FORMAT,OUTDD=FORMAT
```

Batch Mode for DFSUPAA0

This section describes the two procedures for batch mode: MFSBTCH1 and MFSBTCH2.

Batch mode provides the ability to batch the message descriptors and device formats into an accumulation data set, IMS.MFSBATCH. This data set can then be applied to the MFS staging library, IMS.FORMAT, with a separate job. The batch accumulation data set requires you to allocate and catalog an IMS system data set, IMS.MFSBATCH, large enough to hold all the control blocks that are to be accumulated before they are placed into IMS.FORMAT. See Figure 103 for an overview of the batch mode.

Two procedures are required for batch execution: MFSBTCH1 and MFSBTCH2. For the JCL of the MFSBTCH1 procedure, see Figure 104 on page 401. For the JCL of the MFSBTCH2 procedure, see Figure 105 on page 402.

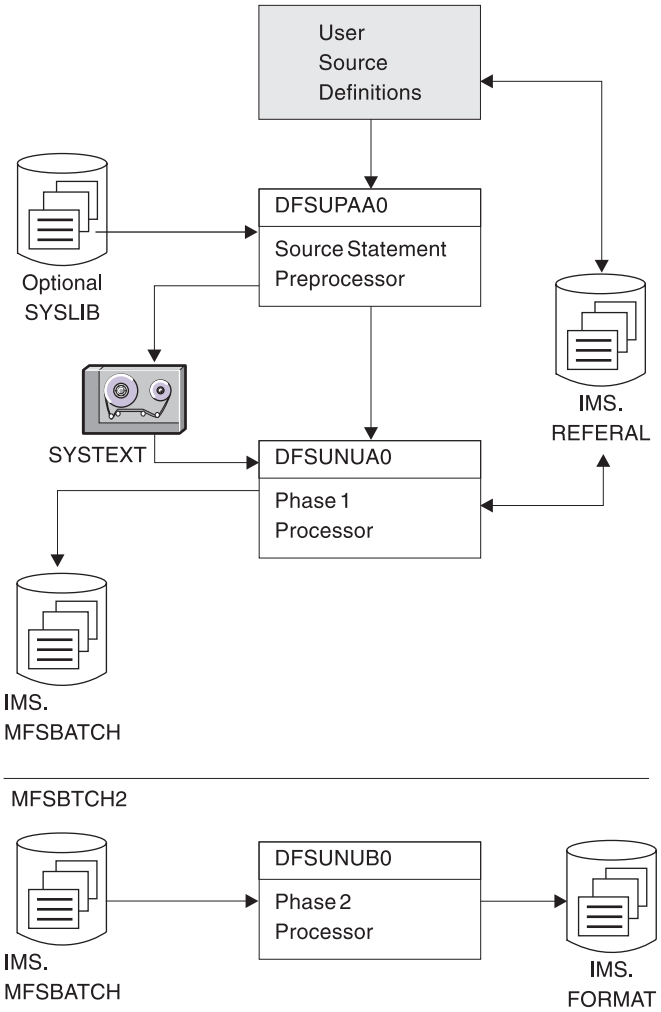


Figure 103. Overall Flow with the MFSBTCH1 and MFSBTCH2 Procedures

MFSBTCH1 Procedure Description

This procedure is identical to step 1 of the MFSUTL procedure in “Step 1 (S1)” on page 395, except that the newly constructed control blocks or error control records, or both, are added to the SEQBLKS accumulation data set, IMS.MFSBATCH.

JCL Requirements

The JCL for the MFSBTCH1 procedure is shown in Figure 104. Refer to “JCL Parameter Descriptions for DFSUPAA0” on page 410 for details on the EXEC parameters and DD statements.

```
//         PROC RGN=4M,SOUT=A,SYS2=,
//            SNODE='IMS',
//            SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,LN=55,SN=8,DEVCHAR=0
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//            &COMPR, 'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR')
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DISP=SHR
//            DSN=&SNODE..&SOR.(&MBR),
//REFIN    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFRD    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//            SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY    DD DISP=SHR
//         DSN=IMS.&SYS2.PROCLIB(REFCPY),
//UTPRINT  DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS  DD DISP=(MOD,KEEP)
//         DSN=IMS.&SYS2.MFSBATCH
```

Figure 104. MFSBTCH1 Procedure

REFCPY Control Statement: The MFSBTCH1 procedure uses this control statement to compress REFERAL.

```
COPY  INDD=REFOUT,OUTDD=REFOUT
```

MFSBTCH2 Procedure Description

This procedure is identical to step 2 of the MFSUTL procedure in “Step 2 (S2)” on page 397, except as noted in the following paragraphs.

If control blocks with duplicate names are found, only the last one found is recorded. This ensures that if the control block with the same name was processed more than once by the MFSBTCH1 procedure, the control block created last is added to IMS.FORMAT. If the control blocks are to be replaced in IMS.FORMAT, they are first deleted from IMS.FORMAT. Consequently, if the control block created last had build time errors, and a block with the same name existed in IMS.FORMAT, the block is deleted from IMS.FORMAT.

On the second pass, IMS.MFSBATCH is reprocessed together with the module table to write the new control blocks, and last occurrences of the duplicate control blocks, into IMS.FORMAT and STOW them for a directory update.

At the end of this step, the SEQBLKS data set is emptied for subsequent use by the MFSBTCH1 procedure.

JCL Requirements

The JCL for the MFSBTCH2 procedure is shown in Figure 105. Refer to "JCL Parameter Descriptions for DFSUPAA0" on page 410 for details on the EXEC parameters and DD statements.

```
//         PROC RGN=4M,SOUT=A,COMPR2=COMPRESS,
//            COMPR3=NOCOMPREND,DIRUPDT=UPDATE,SYS2=
//S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
//            PARM='&COMPR2,&COMPR3,&DIRUPDT'
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DISP=(OLD,KEEP),
//         DSN=IMS.&SYS2.MFSBATCH
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=IMS.&SYS2.FORMAT,DISP=SHR
//DUMMY    DD DISP=SHR,
//         DSN=IMS.&SYS2.PROCLIB(FMTCPY)
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Figure 105. MFSBTCH2 Procedure

FMTCPY Control Statement: The MFSBTCH2 procedure uses this control statement to compress FORMAT.

```
COPY  INDD=FORMAT,OUTDD=FORMAT
```

Test Mode for DFSUPAA0

This section describes the test mode of operation using the MFSTEST procedure.

Figure 106 on page 403 shows an overview of the test mode of operation using the MFSTEST procedure. For the MFSTEST procedure JCL, see "JCL Requirements" on page 404.

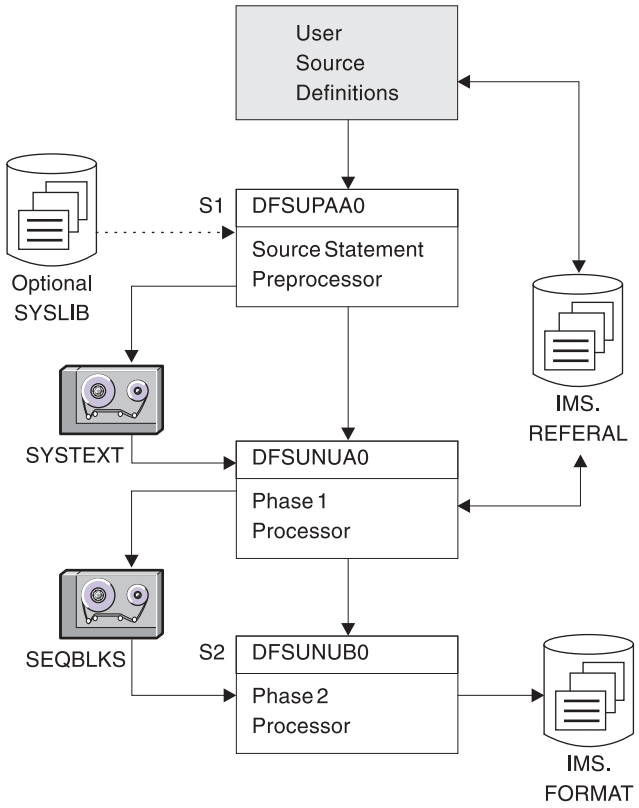


Figure 106. Overall Flow with the MFSTEST Procedure

MFSTEST Procedure Description

The MFSTEST procedure can be used to create message descriptors and device formats while an IMS control region is active. You can check the control blocks created by MFSTEST, without disrupting online production activity, by using the `/TEST MFS` command. Control blocks that have been tested can be placed into the staging library using the MFSUTL procedure. The MFSTEST procedure does not alter the staging library, and all of the ITBs and control blocks remain stable.

This section describes how the two-step execution of the MFS Language utility differs when the MFSTEST procedure is used.

Step 1 (S1)

Source Statement Preprocessor: The source statement preprocessor operates in the same manner as the MFSUTL procedure with the exception that the ITBs are placed into a temporary library. The contents of the historical reference library, IMS.REFERAL, are not changed to reflect new MSG, FMT, PDB, or TABLE ITBs, or new relationships that result from this test mode execution. The IMS.REFERAL library is used only in a read-only manner to perform the resolution function that ensures that all required MSG and FMT ITBs are processed.

Phase 1 Processor: The phase 1 processor operates identically to the operation with the MFSUTL procedure, except that the ITBs for the required MSGs and FMTs are read in from the concatenated temporary library created by the preprocessor and from the IMS.REFERAL library.

The phase 1 processor obtains all MSGs, FMTs, PDBs, and TABLEs defined by this execution from the temporary library created by the preprocessor. Additional blocks, if related ITBs are present, are obtained from IMS.REFERAL.

Step 2 (S2)

Phase 2 Processor: The phase 2 processor operates against a special format library, IMS.TFORMAT, which is used by the IMS control region to access MFS control blocks when terminals are in MFSTEST mode. The phase 2 processor deletes control blocks from this library if new versions are created during this execution or if errors are detected during this execution. The phase 2 processor then inserts the new control blocks created during this execution into the library which will be available for online testing.

IMS.TFORMAT is not compressed, since the IMS control region might be concurrently reading from it.

Recommendation: It is recommended that you periodically compress this data set when the IMS control region is **not** executing (use DISP=OLD for IMS.TFORMAT).

The test procedure deletes $$IMSDIR, if one exists on the test format data set.

JCL Requirements

The JCL for the MFSTEST procedure is shown in Figure 107 on page 405. Refer to “JCL Parameter Descriptions for DFSUPAA0” on page 410 for details on the EXEC parameters and DD statements.

```
//         PROC RGN=4M,SOUT=A,SYS2=,
//            SNODE='IMS',
//            SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,LN=55,SN=8,DEVCHAR=0
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//            &COMPR,'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR')
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DISP=SHR,
//         DSN=&SNODE..&SOR.(&MBR)
//REFIN    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=&&TEMPPDS,
//         DCB=IMS.&SYS2.REFERAL,
//         UNIT=SYSDA,SPACE=(CYL,(5,1,10))
//REFRD    DD DSN=*.REFOUT,VOL=REF=*.REFOUT,
//         DISP=(OLD,DELETE)
//         DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS  DD DSN=&&BKLS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
//             PARM='TEST,DEVCHAR=&DEVCHAR',
//             COND=(8,LT,S1)
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=IMS.&SYS2.TFORMAT,DISP=SHR
```

Figure 107. MFSTEST Procedure

The DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported.

MFS Library Backup and Restore Operations

This section describes the two procedures, MFSBACK and MFSREST, which are provided to perform utility library backup and restore operations.

Attention: When you use these procedures, make sure that the IMS.REFERAL and IMS.FORMAT libraries are dumped and restored at the same level, that is, at the same time. It is important to do this because of the relational information in the IMS.REFERAL PDS directory which describes the contents of the libraries. To ensure that all libraries are restored to the same level, scratch and reallocate all MFS data sets prior to performing the restore operation. If the libraries are not restored to the same level, unpredictable operation can occur.

MFSBACK Procedure

Figure 108 on page 407 shows the JCL for the MFSBACK procedure and includes the optional MFSTEST facility. All DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported in the statements that require DISP=OLD.

The block size for the IMS.REFERAL library, if specified, must be 800.

Refer to “JCL Parameter Descriptions for DFSUPAA0” on page 410 for details on the EXEC parameters and DD statements.

```
//       PROC  NODE='IMS',
//             TAPE=MFSDBS,SOUT=A,DSN=FORMAT,SYS2=
//*
//*************************************************
//*                                               *
//*  PROCEDURE KEYWORDS FOR // EXEC STATEMENT:    *
//*                                               *
//*    NODE=   PREFIX LEVEL TO BE USED FOR        *
//*            ACCESS TO IMS MFS LIBRARIES.       *
//*                                               *
//*    SYS2=   SECOND PREFIX LEVEL TO BE USER FOR *
//*            ACCESS TO IMS MFS LIBRARIES.       *
//*                                               *
//*    TAPE=   BACKUP TAPE SERIAL NUMBER.         *
//*                                               *
//*    SOUT=   SPECIFIES THE PRINT OUTPUT CLASS   *
//*            TO BE USED FOR PRINTED OUTPUT      *
//*            DURING THE BACKUP OPERATION.       *
//*                                               *
//*************************************************
//*
//MOVE1  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//REFERAL  DD DSN=&NODE..&SYS2.REFERAL,DISP=OLD
//TAPEOUT  DD UNIT=2400,LABEL=(1,SL),DISP=(NEW,PASS),
//            VOL=(,RETAIN,SER=&TAPE),
//            DSN=&NODE..&SYS2.REFERAL,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//*
//*************************************************
//*                                               *
//*  //MOVE1.SYSIN  DD  *  MUST BE SUPPLIED BY THE *
//*  USER WITH THE APPROPRIATE COPY CONTROL       *
//*  STATEMENT AS SHOWN BELOW:                    *
//*                                               *
//*   COPY OUTDD=TAPEOUT,INDD=REFERAL             *
//*                                               *
//*************************************************
//*
//MOVE2  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS2.&DSN,DISP=OLD
//TAPEOUT  DD UNIT=2400,LABEL=(2,SL),
//            VOL=(,RETAIN,REF=*.MOVE1.TAPEOUT),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DSN=&NODE..&SYS2.&DSN,
//            DISP=(OLD,KEEP)
//*
//*************************************************
//*
//*  //MOVE2.SYSIN  DD  *  MUST BE SUPPLIED WITH   *
//*  APPROPRIATE COPY CONTROL STATEMENT           *
//*  AS SHOWN BELOW:                              *
//*                                               *
//*    COPY OUTDD=TAPEOUT,INDD=FORMAT             *
//*                                               *
//*************************************************
//*
```

Figure 108. MFSBACK Procedure (Part 1 of 2)

```
//MOVE3  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS2.TFORMAT,DISP=SHR
//TAPEOUT  DD UNIT=2400,LABEL=(3,SL),
//            VOL=REF=*.MOVE1.TAPEOUT,
//            DSN=&NODE..&SYS2.TFORMAT,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DISP=(OLD,KEEP)
//*
//*************************************************
//*  //MOVE3.SYSIN  DD  *  MUST BE SUPPLIED WITH     *
//*  THE APPROPRIATE COPY CONTROL STATEMENT          *
//*  AS SHOWN BELOW:                                 *
//*                                                  *
//*   COPY OUTDD=TAPEOUT,INDD=FORMAT                 *
//*                                                  *
//*************************************************
//*
```

Figure 108. MFSBACK Procedure (Part 2 of 2)

MFSREST Procedure

Figure 109 on page 409 shows the JCL for the MFSREST procedure and includes the optional MFSTEST facility. All DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported in the statements that require DISP=OLD.

Refer to “JCL Parameter Descriptions for DFSUPAA0” on page 410 for details on the EXEC parameters and DD statements.

```
//       PROC  NODE='IMS',
//             TAPE=MFSDBS,SOUT=A,DSN=FORMAT,SYS2=
//*
//*************************************************
//*                                               *
//*  PROCEDURE KEYWORDS FOR // EXEC STATEMENT:    *
//*                                               *
//*    NODE=   PREFIX LEVEL TO BE USED FOR        *
//*            ACCESS TO IMS MFS LIBRARIES.       *
//*                                               *
//*    SYS2=   SECOND PREFIX LEVEL TO BE USED FOR *
//*            ACCESS TO IMS MFS LIBRARIES.       *
//*                                               *
//*    TAPE=   RESTORE TAPE SERIAL NUMBER.        *
//*                                               *
//*    SOUT=   SPECIFIES THE PRINT OUTPUT CLASS   *
//*            TO BE USED FOR PRINTED OUTPUT      *
//*            DURING THE RESTORE OPERATION.      *
//*                                               *
//*************************************************
//*
//MOVE1  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//REFERAL  DD DSN=&NODE..&SYS2.REFERAL,DISP=OLD
//TAPEIN   DD UNIT=2400,LABEL=(1,SL),DISP=(OLD,KEEP),
//            VOL=(,RETAIN,SER=&TAPE),
//            DSN=&NODE..&SYS2.REFERAL,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//*
//*************************************************
//*                                               *
//*  //MOVE1.SYSIN  DD  *  MUST BE SUPPLIED BY THE *
//*  USER WITH THE APPROPRIATE COPY CONTROL        *
//*  STATEMENT AS SHOWN BELOW:                    *
//*                                               *
//*   COPY OUTDD=REFERAL,INDD=TAPEIN              *
//*                                               *
//*************************************************
```

Figure 109. MFSREST Procedure (Part 1 of 2)

```
//*
//MOVE2  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS2.&DSN,DISP=OLD
//TAPEIN   DD UNIT=2400,LABEL=(2,SL),
//            VOL=(,RETAIN,REF=*.MOVE1.TAPEIN),
//            DSN=&NODE..&SYS2.&DSN,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DISP=(OLD,KEEP)
//*
//*****************************************************
//*  //MOVE2.SYSIN  DD  *  MUST BE SUPPLIED WITH     *
//*  THE APPROPRIATE COPY CONTROL STATEMENT          *
//*  AS SHOWN BELOW:                                 *
//*                                                  *
//*    COPY OUTDD=FORMAT,INDD=TAPEIN                 *
//*                                                  *
//*****************************************************
//*
//MOVE3  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS1.TFORMAT,DISP=SHR
//TAPEIN   DD UNIT=2400,LABEL=(3,SL),
//            VOL=REF=*.MOVE1.TAPEIN,
//            DSN=&NODE..&SYS2.TFORMAT,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DISP=(OLD,KEEP)
//*
//*****************************************************
//*  //MOVE3.SYSIN  DD  *  MUST BE SUPPLIED WITH     *
//*  THE APPROPRIATE COPY CONTROL STATEMENT          *
//*  AS SHOWN BELOW:                                 *
//*                                                  *
//*   COPY OUTDD=FORMAT,INDD=TAPEIN                  *
//*                                                  *
//*****************************************************
//*
```

Figure 109. MFSREST Procedure (Part 2 of 2)

JCL Parameter Descriptions for DFSUPAA0

This section describes the JCL parameters for MFSUTL, MFSBTCH1, and MFSTEST.

When Step 1 (S1) executes (in the MFSUTL, MFSBTCH1, and MFSTEST procedures), the following parameters can be specified in the PARM keyword of the EXEC statement.

PXREF= NOXREF | XREF
Specifies whether (XREF) or not (NOXREF) a sorted cross reference listing should be provided. The default value is NOXREF. A sorted cross reference listing includes a list of all labels and related references.

PCOMP= NOCOMP | COMP
Specifies whether (COMP or COMPOSITE) or not (NOCOMP) the composite or final version of the statement, after error recovery or substitution has modified it, is printed. The default value is NOCOMP. The composite statement reflects syntactic assumptions made during error recovery. Semantic assumptions do

not appear in the composite statement but are reflected in the intermediate text blocks. If the repetitive generation function for MFLD/DFLD statements is used, COMP also causes the generated statements to be printed; NOCOMP suppresses this printing.

PSUBS= NOSUBS | SUBS
Specifies whether (SUBS or SUBSTITUTE) or not (NOSUBS) the substitution variable and its equated value are printed when the substitution variable is encountered in the operand field of a statement. The default value is NOSUBS.

PDIAG= NODIAG | DIAG
Specifies whether (DIAG or DIAGNOSTIC) or not (NODIAG) the XREF, COMP, and SUBS options should all be set on. In addition, diagnostic information is printed. The default value is NODIAG, which has no effect on the XREF, COMP, and SUBS options but suppresses printing of the diagnostic information.

COMPR= NOCOMPRESS | COMPRESS
Specifies whether (COMPRESS) or not (NOCOMPRESS) the IMS.REFERAL library is to be compressed before new ITBs are added. The default value is NOCOMPRESS.

LN= 55 | *nn*
Specifies how many lines per page should be printed. The default value is 55.

SN= 08 | *nn*
Specifies the severity code compare value. MSG, FMT, and TABLE blocks whose error severity equals or exceeds this value are not written to the IMS.REFERAL library. The default value is 08.

DEVCHAR= 0 | *x*
Specifies the alphanumeric suffix character (x) to be appended to DFSUDT0. The name DFSUDT0 identifies the desired device characteristics table. This suffix character (x) corresponds to the value specified in the SUFFIX= keyword of the IMSGEN macro. The default is zero (0).

In the execution of the MFSRVC procedure, one parameter can be specified. The DEVCHAR=0 or x parameter specifies the alphanumeric suffix character (x) to be used for the device characteristics table, when no suffix is specified in the LIST control statement parameter DEVCHAR. The default is zero.

In the execution of Step 2 (S2) in the MFSUTL and MFSBTCH2 procedures, three parameters can be specified in the EXEC statement's PARM keyword:

COMPR2= COMPRESS | NOCOMPRESS
Specifies whether (COMPRESS) or not (NOCOMPRESS) the IMS.FORMAT library is to be compressed before new control blocks are added. The default value is COMPRESS.

COMPR3= COMPREND | NOCOMPREND
Specifies whether (COMPREND) or not (NOCOMPREND) the data set with the ddname of FORMAT is compressed after all format blocks have been added/replaced and the index directory ($$IMSDIR) has been updated.

DIRUPDT= UPDATE | NOUPDATE
Specifies whether (UPDATE) or not (NOUPDATE) the special index directory ($$IMSDIR) is automatically updated after a block has been deleted from a format library. You can bypass the $$IMSDIR update by specifying NOUPDATE. The default is UPDATE.

In the execution of Step 2 (S2) in the MFSTEST procedure, the PARM='TEST' parameter must be specified.

Other EXEC statement parameters that can be specified are:

RGN=
Specifies the region size for this execution. The default is 360K.

SOUT=
Specifies the SYSOUT class. The default is A.

SNODE=
Specifies the node that can be assigned to the MFS utility data set name. The default value is IMS.

SOR=
Specifies the library name that can be assigned to the MFS utility library for SYSIN or SYSLIB. The default value is NOLIB.

MBR=
Specifies the member name that can be assigned to the MFS utility member for SYSIN. The default is NOMBR.

DDNAMES Used in MFS Procedures

The data set names used in the MFSUTL, MFSBTCH1, MFSBTCH2, and MFSTEST procedures fit installation needs. The ddnames used and the data sets they refer to are:

REFIN
REFOUT
REFRD
Refers to the MFS reference library, except when used in the MFSTEST procedure. In MFSTEST, REFIN and REFRD refer to the MFS reference library; REFOUT is a temporary data set.

FORMAT
Refers to the MFS control block library. In MFSTEST, this ddname refers to the MFS test control block library.

SYSLIB
Refers to an optional user library from which input can be copied.

SYSIN
Refers to the input data set, which can be a sequential data set or a member of a partitioned data set.

DUMMY
Refers to the IMS procedure library, which contains control statements used to compress the MFS reference and control block libraries.

SYSUT3
SYSUT4
Are ddnames for data sets used during the data set compression as work data sets.

DUMMY, SYSUT3, and SYSUT4 can all be omitted if neither the MFS reference library nor the MFS control block library is to be compressed.

UTPRINT
Is used for messages during the compression of the MFR reference library, and is used for MFS error and status messages during MFS Language utility Phase 2 processing.

The following ddnames refer to data sets used in the MFSRVC procedure. The data set names can be altered to fit installation needs.

REFIN
Refers to the MFS reference library.

FORMAT
Refers to the MFS control block library.

SYSIN
Refers to the input data set, which can be a sequential data set or a member of a partitioned data set.

SYSSNAP
Refers to a data set that is used to receive the output from a SNAP macro if certain severe errors are detected.

SYSPRINT
Refers to the destination of the output. If output is to be sent to a data set (instead of SYSOUT=), use DISP=MOD for the data set.

MFSUTL and MFSTEST Region Parameter Estimate

The following steps help you estimate the main storage requirements that you should specify in the RGN= parameter of the EXEC statement invoking the MFSUTL and MFSTEST procedures.

1. Calculate statement base count. For the input to the MFS Language utility, determine the largest (number of statements) device format to be processed and the largest message descriptor related to the format. Add the total number of statements contained in these two control blocks to obtain the statement base count.

   For the processing of a specific user-supplied MSG or FMT ITB, the utility reprocesses all related MSG or FMT ITBs saved from the IMS.REFERAL data set to ensure compatible linkage between all related online blocks. These reprocessed ITBs must be analyzed as well for the process of obtaining the statement base count.
2. Estimate Region Requirements. Multiply the statement base count by 214 and add 300000 to the result. Round the resulting value to the next highest multiple of 2048. The result is an estimate of the main storage requirements which should be specified in the RGN= parameter of the EXEC statement invoking the MFSUTL and MFSTEST procedures.

   Complex formats with a large number of literal DFLD statements in relation to the statement base count can exceed the estimate.

MFS Language Utility Control Statements

The control statements used by the MFS Language utility are divided into two major categories:

- *Definition statements* are used to define message formats, device formats, partition sets, and operator control tables.
- *Compilation statements* are those used to control the compilation and SYSPRINT listings of the definition statements.

Use the definition and compilation control statements to identify a particular function performed by the utility and to specify various options.

The definition and compilation control functions are:

- SYSPRINT LISTING CONTROL

  The following parameters are provided to format the compilation listing: XREF, SUBS, COMP, DIAG, and LINECNT.
- SYSIN and SYSLIB RECORD STACKING and UNSTACKING

  Control statements are provided to allow one or more SYSIN or SYSLIB records to be processed and kept in processor storage for reuse later in the compilation. These statements are an alternative to the COPY facility for groups of statements that are repeated.

  MFLD and DFLD statements can be repetitively generated if preceded by a DO statement and followed by an ENDDO statement. Repetitive DFLD generation supports increments to line and column position information.
- ALPHA CHARACTER GENERATION

  The ALPHA statement allows specification of additions to the set of characters as alphabetic.
- COPY

  The COPY statement allows members of partitioned data sets to be copied into the input stream of the utility preprocessor.

The following topics provide additional information:

- "Control Statement Syntax"
- "Summary of Control Statements" on page 417
- "EXEC Statement Parameters" on page 418
- "Message Definition Statements" on page 420
- "Format Definition Statements" on page 432
- "Partition Set Definition Statements" on page 483
- "Table Definition Statements" on page 486
- "Compilation Statements" on page 488

Control Statement Syntax

The control statements are written in assembler-like language with the following standard format:

label *operation* *operand* *comments*

Figure 110. Control Statement Syntax for MFS Language Utility

label
: Identifies the statement; if it is shown as optional, it can be omitted. When included, the name must begin in the first position of the statement (column 1) and must be followed by one or more blanks. It can contain from one to eight alphanumeric characters (one to six, for the FMT label), the first of which must be alphabetic.

operation
: Identifies the type of control statement. It normally begins in column 10 and must be preceded and followed by one or more blanks.

operand
: Is made up of one or more parameters, which can be positional or keyword parameters. A positional parameter in MFS control statements always appears in the first position of the operand, normally starting in column 16. The position of a keyword parameter is not important. The parameters within one operand

are separated by commas. In the syntactical description of the control statements, parameters preceded by commas are thus identified as keyword parameters. The operand field itself must be preceded and followed by one or more blanks.

comments
Can be written in a utility control statement, but they must be separated from the last parameter of the operand field by one or more blanks. (If the statement does not include an operand, the comment should be separated from the statement by at least one blank.) A comment line begins with an asterisk in column 1.

Continuation is accomplished by entering a nonblank character in column 72. If the current line is a comment, then the continuation line can begin in any column.

Other considerations are as follows:

- There is no limit on the number of continuation lines.
- There is no limit on the number of characters in the operand field. Individual operand items cannot exceed 256 characters, excluding trailing and embedded second quote characters.
- If a nonstandard character (for definition, see “ALPHA Statement” on page 488) is detected in a literal, a severity 4 warning message is issued. The nonstandard character is retained in the literal.
- If the current line is a control statement, the continuation line must begin in column 16.
- A single ampersand is needed to generate one ampersand character in the literal.

In addition to the definition and compiler statement specifications, several parameters can be specified in the EXEC statement PARM keyword to control the current compilation for the preprocessor and phase 1; one parameter can be specified for phase 2.

Five Special Rules

The five special rules that follow use actual MFS code as examples.

1. If you code a statement such that an equal sign or a left parenthesis immediately precedes a comma, you can omit the comma.

   ,FTAB=(,FORCE) could be coded as ,FTAB=(FORCE)
2. If you code a statement such that an equal sign immediately precedes a single item enclosed in parentheses, you can omit the parentheses.

   ,FTAB=(,FORCE) could be coded as ,FTAB=,FORCE
3. You can apply both Rule 1 and Rule 2, in either order, to a single item.

   ,FTAB=(,FORCE) could be coded as ,FTAB=FORCE
4. Under no condition can you specify a keyword without specifying at least one parameter immediately after that keyword.

   Neither ,FTAB= nor ,FTAB=,LDEL='**' is permitted.
5. Blanks are required between labels and statement type names, and between statement type names and their parameters; they are not permitted elsewhere unless explicitly represented by the symbol ƀ.

   DEV ,PAGE is correct, but DEV,PAGE and ,FTAB= (,MIX) are incorrect.

Syntax Errors

The MFS Language utility attempts to recover from syntax errors in source statements. No guarantee exists for the correctness of the assumptions made in the recovery, and these assumptions can differ in different releases of IMS. Assumptions made during recovery are based on (1) what is expected when the incorrect item is encountered; (2) what could appear to the right of the item preceding the incorrect item; and (3) what could appear to the left of the incorrect item.

During the process of error recovery, the following notation can be used in the diagnostic messages:

; indicates that the end of the source statement was encountered. The position marker points to the position immediately following the last source item scanned.

L refers to a literal operand item.

V refers to an identifier operand item (alphabetic character optionally followed by alphanumeric characters).

I refers to a numeric operand item.

A refers to an alphanumeric operand item (numeric character optionally followed by alphanumeric characters).

D refers to a delimiter operand item.

Most error recovery messages have a severity code of 4, indicating a warning level error. When an item is deleted, or the syntax scan is aborted, the statement cannot be validly processed and a severity code of 8 is generated.

Invalid Sequence of Statements

The language utility preprocessor routines that process MSG, FMT, PDB, or TABLE definition statements are organized hierarchically. A routine for a given level processes a statement at that level, reads the next statement, then determines which routine will next receive control.

If the statement just read is the next lower level statement (for example, a DIV statement following a DEV statement), the next lower level routine (for example, the DIV statement processor) is called.

If the statement just read is not the next lower level statement, control can be passed to one of the following three routines:

1. The next lower level routine to assume the missing statement (for example, the DIV processor if a DEV statement is followed by a DPAGE statement)
2. The same level routine if the statement just read is of the same level as the processor (for example, a series of DFLD statements)
3. The next higher level routine (the calling routine) if the statement just read is not the same or the next lower level (for example, a DEV statement following a DFLD statement, an invalid statement, or a statement out of sequence)

Thus, if the hierarchic structure of a MSG, FMT, PDB, or TABLE definition is invalid or a statement operator is misspelled, case (3) will result in control being returned to successively higher level routines. At the highest level, only a FMT, MSG, TABLE, PDB, or END statement will be accepted by the preprocessor. Therefore, all statements before the next FMT, PDB, MSG, TABLE or END statement will be flushed (that is, not processed) and flagged with the appropriate error message.

Summary of Control Statements

The definition of message formats, device formats, partition sets, and operator control tables is accomplished with separate hierarchic sets of definition statements.

Message Definition Statement Set
is used to define message formats. It includes the following statements:

MSG	Identifies the beginning of a message definition.
LPAGE	Identifies a related group of segment/field definitions.
PASSWORD	Identifies a field or fields to be used as an IMS password.
SEG	Identifies a message segment.
DO	Requests iterative processing of the subsequent MFLD statements.
MFLD	Defines a message field. Iterative processing of MFLD statements can be invoked by specifying DO and ENDDO statements. To accomplish iterative processing, the DO statement is placed before the MFLD statements and the ENDDO after the MFLD statements.
ENDDO	Terminates iterative processing of the preceding MFLD statements.
MSGEND	Identifies the end of a message definition.

Format Definition Statement Set
is used to define device formats. It consists of the following statements:

FMT	Identifies the beginning of a format definition.
DEV	Identifies the device type and operational options.
DIV	Identifies the format as input, output, or both.
DPAGE	Identifies a group of device fields corresponding to an LPAGE group of message fields.
PPAGE	Identifies a group of logically related records that can be sent to a remote application program at one time.
DO	Requests iterative processing of the subsequent RCD or DFLD statements.
RCD	Identifies a group of related device fields that are sent to a remote application program as a single record.
DFLD	Defines a device field. Iterative processing of DFLD statements can be invoked by specifying DO and ENDDO statements. To accomplish iterative processing, the DO statement is placed before the DFLD statements and the ENDDO after the DFLD statements.
ENDDO	Terminates iterative processing of the previous RCD or DFLD statements.
FMTEND	Identifies the end of a format definition.

Partition Definition Statement Set
is used to define partition sets (Partition Descriptor Blocks). It consists of the following statements:

PDB	Identifies the beginning of a partition set definition and allows the specification of several parameters that describe it.
PD	Defines a Partition Descriptor, which contains the parameters necessary to describe a partition.
PDBEND	Identifies the end of a partition set definition.

TABLE Definition Statement Set
is used to define operator control tables. It includes the following statements:

TABLE	Identifies the beginning of a table definition.
IF	Defines a conditional test and resulting action.
TABLEEND	Identifies the end of a table definition.

Compilation Statements
are used for variable functions. Compilation statements that are supported by the MFS Language utility are listed in alphabetic order:

ALPHA	Defines a set of characters to be considered alphabetic for the purpose of defining field names and literals.
COPY	Copies a member of the partitioned data set represented by the SYSLIB DD statement into the input stream of the preprocessor.
DO	Requests iterative processing of MFLD or DFLD definition statements.
EJECT	Ejects SYSPRINT listing to the next page.
END	Defines the end of data for SYSIN processing.
ENDDO	Terminates iterative processing of MFLD, RCD, or DFLD definition statements.
EQU	Equates a symbol with a number, alphanumeric identifier, or literal.
PRINT	Controls SYSPRINT options.
RESCAN	Controls EQU processing.
SPACE	Skips lines on the SYSPRINT listing.
STACK	Delineates one or more SYSIN or SYSLIB records that are to be kept in processor storage for reuse.
TITLE	Provides a title for the SYSPRINT listing.
UNSTACK	Retrieves previously stacked SYSIN or SYSLIB records.

Compilation statements are inserted at logical points in the sequence of control statements. For example, TITLE could be first, and EJECT could be placed before each MSG, FMT, or TABLE statement.

EXEC Statement Parameters

EXEC statement parameters supported by the MFS Language utility have variable compilation control functions. Parameters can be specified on the EXEC statement for the preprocessor and phase 1 to:

- Control the printed output
- Compress the reference library (IMS.REFERAL)
- Request diagnostic information

- Indicate which MFS device characteristics table is to be used
- Prevent control blocks with a specified level of error from being written in IMS.REFERAL

Parameters can also be specified on the EXEC statement for phase 2 to specify whether IMS.FORMAT and IMS.REFERAL should be compressed and whether $$IMSDIR should be automatically updated after deletions.

The DEVCHAR parameter specifies the suffix of the MFS device characteristics table to be used. The device characteristics table is accessed only if DEV TYPE=3270-An (where n is 1 to 15) is coded as input to the MFS Language utility. See “MFS Device Characteristics Table” in *IMS Version 9: Application Programming: Transaction Manager* for more information.

The EXEC statement parameters supported by the MFS Language utility have variable compilation control functions. The parameters that can be specified are:

NOXREF|XREF
Specifies whether (XREF) or not (NOXREF) a sorted cross-reference listing should be provided. A sorted cross-reference listing includes a list of all the labels and related references. The default is NOXREF.

NOCOMP|COMP
Specifies whether (COMP or COMPOSITE) or not (NOCOMP) the composite or final version of the statement, after error recovery or substitution has modified it, will be printed. A composite statement reflects syntactic assumptions made during error recovery. Semantic assumptions do not appear in composite statements but are reflected in the intermediate text blocks. The default is NOCOMP.

NOSUBS|SUBS
Specifies whether (SUBS or SUBSTITUTE) or not (NOSUBS) any statement containing a substitution variable (EQU operand) is printed. The default is NOSUBS.

NODIAG|DIAG
Specifies whether (DIAG or DIAGNOSTIC) or not (NODIAG) the XREF, COMP, and SUBS options should be set on and diagnostic information be printed. The default is NODIAG, which has no effect on the setting of the XREF, COMP, and SUBS options but suppresses printing of the diagnostic information.

NOCOMPRESS|COMPRESS
Specifies whether (COMPRESS) or not (NOCOMPRESS) the IMS.REFERAL library is to be compressed before new ITBs are added. The default is NOCOMPRESS.

DIRUPDT= UPDATE|NOUPDATE
Specifies whether (UPDATE) or not (NOUPDATE) the special index directory ($$IMSDIR) will be automatically updated after one or more blocks have been deleted from a format library. You can bypass the $$IMSDIR update by specifying NOUPDATE. The default is UPDATE.

LINECNT=*nn*
Specifies how many lines per page should be printed. The default is 55.

STOPRC=*nn*
Specifies the severity code compare value. MSG, FMT, and TABLE blocks whose error severity equals or exceeds this value will not be written to the IMS.REFERAL library. The default is 08.

DEVCHAR=*n*
Specifies the alphanumeric suffix character (x) used as the final character of the name of the device characteristics table DFSUDT0x loaded when DEV TYPE=3270-An is encountered. The default is zero (DFSUDT00).

The remainder of this chapter describes, in detail, the utility control statements. The definition statements are described in the sequence shown, with the DO and ENDDO compilation statements where they would normally be coded—before and after the MFLD or DFLD statements. The compilation statement formats are sequenced according to related function (if any)—ALPHA; COPY; EQU and RESCAN (equate processing); STACK and UNSTACK (stacking SYSIN/SYSLIB records); TITLE, PRINT, SPACE, and EJECT (SYSPRINT listing control); and END.

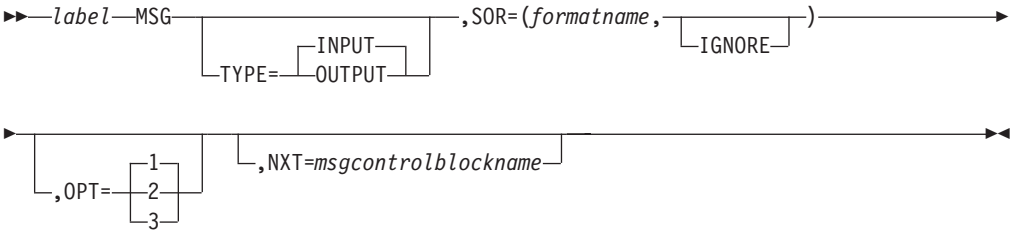
Message Definition Statements

Message definition statements include the MSG statement, the LPAGE statement, the PASSWORD statement, the SEG statement, the DO statement, printing generated MFLD statements, the MFLD statement, the ENDDO statement, and the MSGEND statement. Details of each are provided in the information that follows.

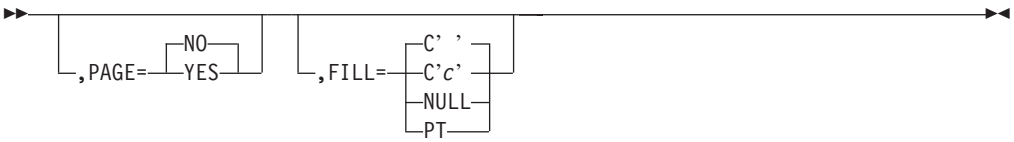
MSG Statement

The MSG statement initiates and names a message input or output definition.

Format for MSG TYPE=INPUT or OUTPUT:



Format for MSG TYPE=OUTPUT Only:



Parameters:

label
A one- to eight-character alphanumeric name must be specified. This label can be referred to in the NXT operand of another message descriptor.

TYPE=
Defines this definition as a message INPUT or OUTPUT control block. The default is INPUT.

SOR=
Specifies the source name of the FMT statement which, with the DEV statement, defines the terminal or remote program data fields processed by this message descriptor. Specifying IGNORE for TYPE=OUTPUT causes MFS to use data fields specified for the device whose FEAT= operand specifies

IGNORE in the device format definition (see “DEV Statement” on page 432). For TYPE=INPUT, IGNORE should be specified only if the corresponding message output descriptor specified IGNORE. If you use SOR=IGNORE, you must specify IGNORE on both the message input descriptor and the message output descriptor.

OPT=
Specifies the message formatting option used by MFS to edit messages. The default is 1. Options 1, 2, and 3 are described in *IMS Version 9: Application Programming: Transaction Manager*.

NXT=
Specifies the name of a message descriptor to be used to map the next expected message as a result of processing a message using this message descriptor. If TYPE=INPUT, NXT= specifies a message output descriptor. If TYPE=OUTPUT, NXT= specifies a message input descriptor. For ISC output, NXT= becomes the RDPN in the ATTACH FM header.

If TYPE=OUTPUT and the *formatname* specified in the SOR= operand contains formats for 3270 or 3270P device types, the *msgcontrolblockname* referred to by NXT= must use the same *formatname*.

PAGE=
Specifies whether (YES) or not (NO) operator logical paging (forward and backward paging) is to be provided for messages edited using this control block. This operand is valid only if TYPE=OUTPUT. The default is NO, which means that only forward paging of physical pages is provided.

FILL=
Specifies a fill character for output device fields. This operand is valid only if TYPE=OUTPUT. The default is C' '. The fill specification is ignored unless FILL=NONE is specified on the DPAGE statement in the FMT definition. For 3270 output when EGCS fields are present, only FILL=PT or FILL=NULL should be specified. A FILL=PT erases an output field (either a 1- or 2-byte field) only when data is sent to the field, and thus does not erase the DFLD if the application program message omits the MFLD. For DPM-Bn, if OFTAB is specified, FILL= is ignored and FILL=NULL is assumed.

C'*c*'
Character '*c*' is used to fill device fields. For 3270 display devices, any specification with a value less than X'3F' is changed to X'00' for control characters or to X'40' for other nongraphic characters. For all other devices, any FILL=C'*c*' specification with a value less than X'3F' is ignored and defaulted to X'3F' (which is equivalent to a specification of FILL=NULL).

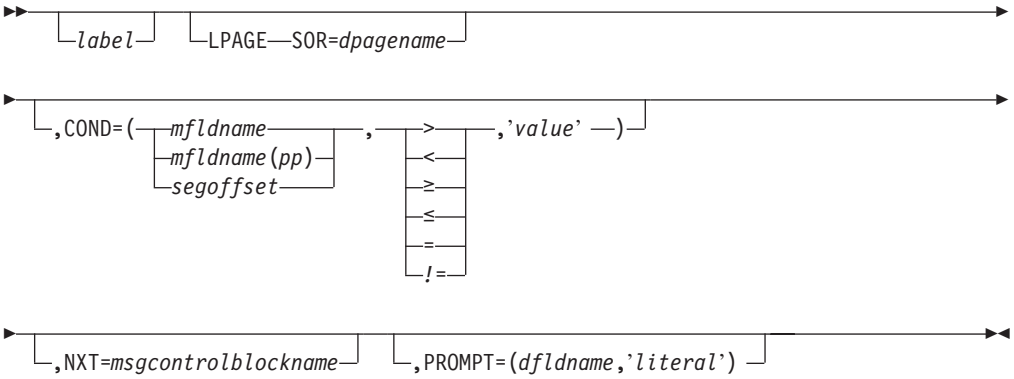
NULL
Specifies that fields are not to be filled. For devices other than 3270 and SLU 2 display, 'compacted lines' are produced when message data does not fill device fields.

PT
Is identical to NULL except for 3270 and SLU 2 display. For 3270 and SLU 2 display, PT specifies that output fields that do not fill the device field (DFLD) are followed by a program tab character to erase data previously in the field.

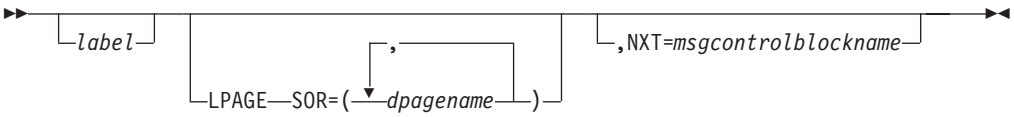
LPAGE Statement

The optional LPAGE statement defines a group of segments comprising a logical page.

Format for MSG TYPE=OUTPUT:



Format for MSG TYPE=INPUT:



Parameters:

label
A one- to eight-character alphanumeric name can be specified to uniquely identify this statement.

SOR=
Specifies the name of the DPAGE statement that defines the device format for this logical page. If TYPE=INPUT and more than one DPAGE can be used as a source of data to create an input message, more than one *dpagename* can be specified.

COND=
Describes a conditional test that, if successful, specifies that the segment and field definitions following this LPAGE are to be used for output editing of this logical page. The specified portion of the first segment of a logical page is examined to determine if it is greater than (>), less than (<), greater than or equal to (≥), less than or equal to (≤), equal to (=), or not equal to (≠) the specified literal value to determine if this LPAGE is to be used for editing. COND= is not required for the last LPAGE statement in the MSG definition.

The area examined can be defined by a field name (*mfldname*), an offset in a field (*mfldname(pp)* where *pp* is the offset in the named field), or an offset in the segment (*segoffset*). If the *mfldname(pp)* form is used, *pp* must be greater than or equal to 1. The length of the compare is the length of the specified literal. If OPT=3 is specified on the previous MSG statement, the area to be examined must be within one field as defined on an MFLD statement.

If *segoffset* is used, it is relative to zero, and the specification of that offset must allow for LLZZ of the segment (that is, the first data byte is at offset 4).

If *pp* is used, the offset is relative to 1 with respect to the named field (that is, the first byte of data in the field is at offset 1, not zero).

If the *mfldname* specified is defined with ATTR=YES, the *pp* offset must be used. The minimum offset specified must be 3. That is, the first byte of data in the field is at offset 3, following the two bytes of attributes.

If ATTR=*nn* is specified, the minimum offset must be one plus twice *nn*. Thus, if ATTR=2 is specified, *pp* must be at least 5, and, if ATTR=(YES,2) is specified, *pp* must be at least 7.

If the conditional tests for all LPAGEs fail, the last LPAGE in this MSG definition is used for editing.

If LPAGE selection is to be specified using the command data field, that is, /FORMAT*modname...(data)*, the MFLD specified in the LPAGE COND=*mfldname* parameter should be within the first 8 bytes of the associated LPAGEs of the MOD.

NXT=
Specifies the name of the message descriptor to be used to map the next message if this logical page is processed. This name overrides any NXT=*msgcontrolblockname* specified on the preceding MSG statement.

PROMPT=
Specifies the name of the DFLD into which MFS should insert the specified literal when formatting the last logical page of an output message. If FILL=NULL is specified once the prompt literal is displayed, it can remain on the screen if your response does not cause the screen to be reformatted.

PASSWORD Statement

The PASSWORD statement identifies one or more fields to be used as an IMS password. When used, the PASSWORD statement and its associated MFLDs must precede the first SEG statement in an input LPAGE or MSG definition. Up to 8 MFLD statements can be specified after the PASSWORD statement but the total password length must not exceed 8 characters. The fill character must be X'40'. For option 1 and 2 messages, the first 8 characters of data after editing are used for the IMS password. For option 3 messages, the data content of the first field after editing is used for the IMS password.

A password for 3270 input can also be defined in a DFLD statement. If both password methods are used, the password specified in the MSG definition is used.

Format:

Parameters:

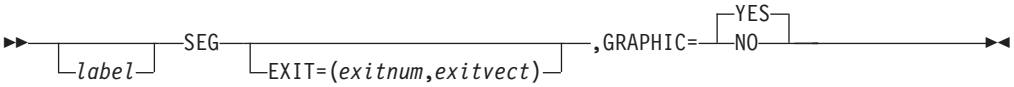
label
A one- to eight-character alphanumeric name can be specified to uniquely identify this statement.

SEG Statement

The SEG statement delineates message segments and is required only if multisegment message processing is required by the application program. Output message segments cannot exceed your specified queue buffer length. Only one segment should be defined for TYPE=INPUT MSGs when the input message destination is defined as a single segment command or transaction. If more than

one segment is defined, and the definition is used to input a single segment command or transaction, care must be used to ensure that your input produces only one segment after editing.

Format for MSG TYPE=INPUT:



Format for MSG TYPE=OUTPUT:

Parameters:

label
A 1- to 8-character name can be specified to uniquely identify this statement.

EXIT=
Describes the segment edit exit routine interface for this message segment. *exitnum* is the exit routine number and *exitvect* is a value to be passed to the exit routine when it is invoked for this segment. *exitnum* can range from 0 to 127. *exitvect* can range from 0 to 255. Unless NOSEGEXIT is specified on the DIV statement (for DPM devices only), the SEG exit is invoked when processing completes for the input segment.

GRAPHIC=
Specifies for MSG TYPE=INPUT whether (YES) or not (NO) IMS should perform upper case translation on this segment if the destination definition requests it (see the EDIT= parameter of the TRANSACT or NAME macro). The default is YES. If input segment data is in nongraphic format (packed decimal, EGCS, binary, and so forth), GRAPHIC=NO should be specified. When GRAPHIC=NO is specified, FILL=NULL is invalid for MFLDs within this segment.

The following list shows the translation that occurs when GRAPHIC=YES is specified and the input message destination is defined as requesting upper case translation:

Before Translation	After Translation
a through z	A through Z
X'81' through X'89'	X'C1' through X'C9'
X'91' through X'99'	X'D1' through X'D9'
X'A2' through X'A9'	X'E2' through X'E9'

If FILL=NULL is specified for any MFLD in a segment defined as GRAPHIC=YES, the hexadecimal character X'3F' is compressed out of the segment. If GRAPHIC=NO and FILL=NULL are specified in the SEG statement, any X'3F' in the non-graphic data stream is compressed out of the segment and undesirable results might be produced. Non-graphic data should be sent on output as fixed length output fields and the use of FILL=NULL is not recommended in this case.

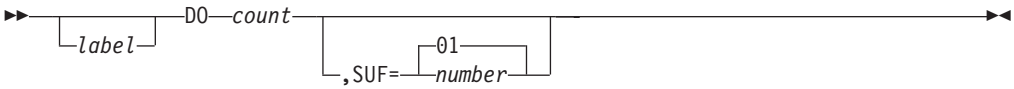
For MSG TYPE=OUTPUT, the GRAPHIC= keyword applies only for DPM. It specifies whether (YES) or not (NO) nongraphic control characters (X'00' to X'3F') in the data from the IMS application program are to be replaced by blanks. The default value is YES. If NO is specified, MFS allows any bit string received from an IMS application program to flow unmodified through MFS to the remote program.

Restriction: When GRAPHIC=NO is specified, IMS application programs using Options 1 and 2 cannot omit segments in the middle of an LPAGE, or truncate or omit fields in the segment using the null character (X'3F').

DO Statement

The DO statement causes repetitive generation of MFLD statements between the DO and ENDDO statements. DO is optional, but a message that includes a DO must include a subsequent ENDDO.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

count
Specifies how many times to generate the following MFLD statements. The maximum *count* that can be specified is 99; if more than 99 is specified, the 2 rightmost digits of the specified *count* are used (for example, 03 would be used if 103 were specified) and an error message is issued.

SUF=
Specifies the 1- or 2-digit suffix to be appended to the MFLD *label* and *dfldname* of the first group of generated MFLD statements. The default is 01. MFS increases the suffix by 1 on each subsequent generation of statements.

If the specified suffix exceeds 2 digits, MFS uses the rightmost 2 digits.

If the specified *count* is such that the generated suffix eventually exceeds 2 digits, MFS reduces the *count* to the largest legitimate value. For example, if *count* equals 8 and SUF=95, invalid suffixes of 100, 101, and 102 would result. In this instance, MFS reduces *count* to 5, processes the statement, and issues an error message.

Printing Generated MFLD Statements

The generated MFLD statements can be printed in a symbolic source format by specifying COMP in the parameter list of the EXEC statement. This provides a means of seeing the results of the MFLD statement generation without having to interpret the intermediate text blocks.

The following items are printed for each generated MFLD statement:

- The generated statement sequence number followed by a + (plus sign) to indicate that the MFLD statement was generated as a result of DO statement processing.
- The MFLD statement label, if present, including the appended suffix.

- The statement operator, MFLD.
- *dfldname*, if present, including the appended suffix.
- For ECGS literals, the G, SO, and SI is not present. Literals are truncated if there is insufficient room to print all specifications. Truncation is indicated by a portion of the literal followed by an ellipsis (...) representing the truncated portion.
- The system literal name, if present.

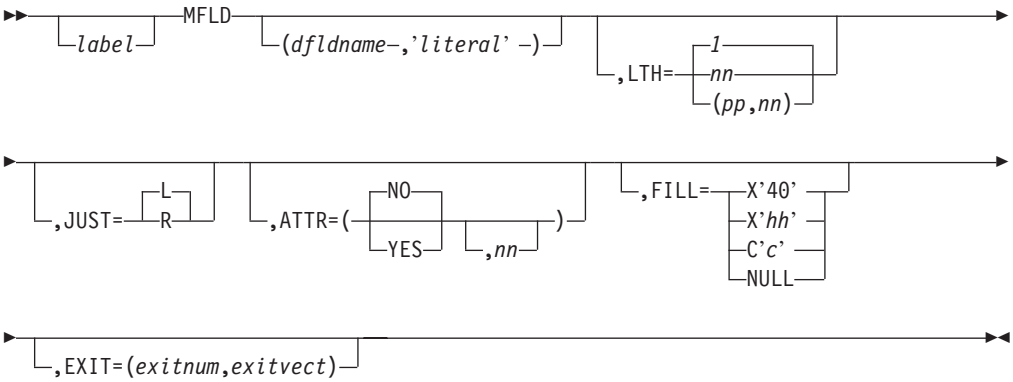
  If both *dfldname* and a literal are present, they are enclosed in parentheses.
- (,SCA), if present.
- The field length, in the form LTH=*nnnn* (or LTH=(*pppp,nnnn*), if present).
- JUST=L or R, if present.
- ATTR=YES, if present.
- ATTR=*nn*, if present.

No other operands are printed, even if specified on the source MFLD statement.

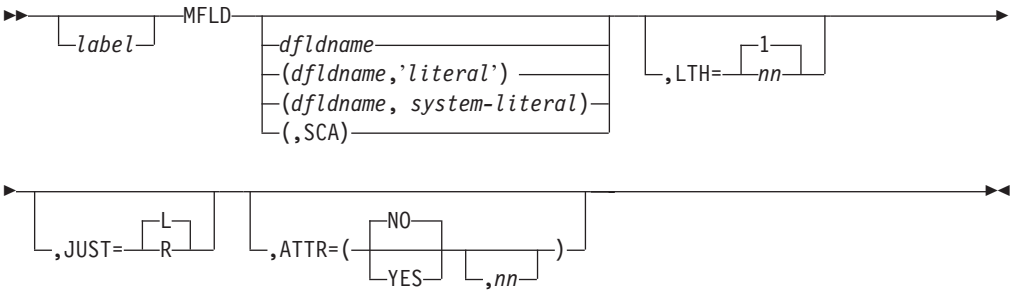
MFLD Statement

The MFLD statement defines a message field as it will be presented to an application program as part of a message output segment. At least one MFLD statement must be specified for each MSG definition.

Format for MSG TYPE=INPUT:



Format for MSG TYPE=OUTPUT:



Parameters:

label

A one-to eight-character alphanumeric name can be specified. *label* is required if it is referred to in the COND operand of the previous LPAGE statement. It can

be used to uniquely identify this statement. If the MFLD is between the DO and ENDDO statements, *label* is restricted to 6 characters or less. DO statement processing appends a 2-digit suffix (a sequence number, 01 to 99) to the label and prints the label as part of the generated MFLD statement. If *label* is more than 6 characters and iterative generation is used, the label is truncated at 6 characters, and the 2-digit sequence number is added to make the 8-character name. No error message is issued if this occurs.

dfldname
Specifies the device field name (defined using the DEV or DFLD statement) from which input data is extracted or into which output data is placed. If this parameter is omitted when defining a message output control block, the data supplied by the application program is not displayed on the output device. If the repetitive generation function of MFS is used (DO and ENDDO statements), *dfldname* should be restricted to 6 characters maximum length. When each repetition of the statement is generated, a 2-character sequence number (01 to 99) is appended to *dfldname*. If the *dfldname* specified here is greater than 6 bytes and repetitive generation is used, *dfldname* is truncated at 6 characters and a 2-character sequence number is appended to form an 8-character name. No error message is provided if this occurs. This parameter can be specified in one of the following formats:

dfldname
Identifies the device field name from which input data is extracted or into which output data is placed.

'literal'
Can be specified if a literal value is to be inserted in an input message.

(*dfldname***,***'literal'***)**
If TYPE=OUTPUT, this describes the literal data to be placed in the named DFLD. When this form is specified, space for the literal must not be allocated in the output message segment supplied by the application program.

If TYPE=INPUT, this describes the literal data to be placed in the message field when no data for this field is received from the device. If this *dfldname* is used in the PFK parameter of a DEV statement, this literal is always replaced by the PF key literal or control function. However, when this *dfldname* is specified in the PFK parameter, but the PF key is not used, the literal specified in the MFLD statement is moved into the message field. When physical paging is used, the literal is inserted in the field but is not processed until after the last physical page of the logical page has been displayed.

In both cases, if the LTH= operand is specified, the length of the literal is truncated or padded as necessary to the length of the LTH= specification. If the length of the specified literal is less than the defined field length, the literal is padded with blanks if TYPE=OUTPUT and with the specified fill character (FILL=) if TYPE=INPUT. If no fill character is specified for input, the literal is padded with blanks (the default). The length of the literal value cannot exceed 256 bytes.

(*dfldname***,***system-literal***)**
Specifies a name from a list of system literals. A system literal functions like a normal literal except that the literal value is created during formatting prior to transmission to the device. The LTH=, ATTR=, and JUST= operands

cannot be specified. When this form is specified, space for the literal must not be allocated in the output message segment supplied by the application program.

The system literals and their associated lengths and formats are shown in Table 26.

Table 26. Lengths and Formats of System Literals

	Produces Literal of:		
System Literal Name	**Length**	**Format**	**Notes**
LTSEQ	5	*nnnnn*	1
LTNAME	8	*aaaaaaaa*	1
TIME	8	HH:MM:SS	
DATE1 or YYDDD	6	YY.DDD	
DATE2 or MMDDYY	8	MM/DD/YY	
DATE3 or DDMMYY	8	DD/MM/YY	
DATE4 or YYMMDD	8	YY/MM/DD	
DATE1Y4 or YYYYDDD or DATEJUL	8	YYYY.DDD	
DATE2Y4 or MMDDYYYY or DATEUSA	10	MM/DD/YYYY	
DATE3Y4 or DDMMYYYY or DATEEUR	10	DD/MM/YYYY	
DATE4Y4 or YYYYMMDD or DATEISO	10	YYYY/MM/DD	
LPAGENO	4	*nnnn*	2
LTMSG	14	MSG WAITING Q*x*	3

Table 26. Lengths and Formats of System Literals (continued)

System Literal Name	Produces Literal of: Length	Format	Notes

Notes:

1. LTSEQ is the output message sequence number for the logical terminal. The value created is the logical terminal dequeue count plus 1. The first output message after an IMS cold start or `/NRESTART BUILDQ` has a sequence number of 00001. Certain IMS-created messages do not change this number.

   LTNAME is the logical terminal (LTERM) name of the LTERM for which this message is being formatted.

   Messages generated by the IMS control region in response to terminal input (error messages, most command responses) do not have an LTSEQ or an LTNAME. These messages use the IMS message output descriptor DFSMO1. In these instances, the values provided are 00000 and blanks, respectively.
2. LPAGENO specifies that the current logical page number of the message be provided as a system literal. This number corresponds to the page number you entered for an operator logical page request. The literal produced is a 4-digit number with leading zeros converted to blanks.
3. LTMSG specifies that when this output message is sent to the terminal, the literal 'MSG Waiting Q*x*' (where *x* is message queue number 1, 2, 3, or 4) is sent in the LTMSG field if there are messages in the queue for the terminal. If there are no messages in the queues, other than the current queue, blanks are sent in the LTMSG field.

   Usually the message waiting is sent when the current message is dequeued. If the message is waiting in Q1, it is sent. If the message is in Q2 and the terminal is in exclusive mode, it is sent (when any other messages from Q1 are sent). If the message is in Q2 and conversational status does not prevent it from being sent or if the message is in Q3 or Q4 and the exclusive or conversational status does not prevent it from being sent, it is sent. If a message is waiting to be sent on another queue and the terminal is in conversation, the conversation can be held to view the message; if the terminal is in exclusive mode, the message can be viewed when the terminal is taken out of exclusive mode. If you are entering response mode transactions, the message can be viewed before entering response mode transaction input from the terminal.

   This system literal is recommended for conversational mode. It is not recommended for ISC subsystems.

(,SCA)
: Defines this output field as the system control area which is not displayed on the output device. There can be only one such field in a logical page (LPAGE) and it must be in the first message segment of that page. If no logical pages are defined, only one SCA field can be defined and it must be in the first segment of the output message. This specification is valid only if TYPE=OUTPUT was specified on the previous MSG statement.

LTH=
: Specifies the length of the field to be presented to an application program on input or received from an application program on output. Default or minimum value is 1. Maximum value is 8000. (The maximum message length must not exceed 32767.)

  The form (*pp*,*nn*) can be used when defining an input field; however, a field name must be specified in the first positional parameter if the (*pp*,*nn*) form is used. The value supplied for *pp* specifies which byte in the input data field is to be considered the first byte of data for the message field. For example, a *pp* of 2 specifies that the first byte of input data is to be ignored, and the second byte

becomes the first byte of this field. The value of *pp* must be greater than or equal to 1. The value supplied for *nn* specifies the length of the field to be presented to an application program.

If (,SCA) is specified in the positional parameter, the specified LTH= value must be at least 2.

LTH= can be omitted if a literal is specified in the positional operand (TYPE=INPUT), in which case, length specified for *literal* is used. If LTH= is specified for a literal field, the specified literal is either truncated or padded with blanks to the specified length. If the MFLD statement appears between a DO and an ENDDO statement, a length value is printed on the generated MFLD statement, regardless of whether LTH= is specified in the MFLD source statement.

JUST=
Specifies that the data field is to be left-justified (L) or right-justified (R) and right- or left- truncated as required, depending upon the amount of data expected or presented by the device format control block. The default is L.

ATTR=
Specifies whether (YES) or not (NO) the application program can modify the 3270 attributes and the extended attributes (*nn*).

If YES, 2 bytes must be reserved for the 3270 attribute data to be filled in by the application program on output and to be initialized to blanks on input. These 2 bytes must be included in the LTH= specification.

The value supplied for *nn* is the number of extended attributes that can be dynamically modified. The value of *nn* can be a number from 1 to 6. An invalid specification will default to 1. Two additional bytes per attribute must be reserved for the extended attribute data to be filled in by the application program on output and to be initialized to blanks on input. These attribute bytes must be included in the MFLD LTH= specification.

The following example shows valid specifications for ATTR= and the number of bytes that must be reserved for each different specification:

MFLD ,ATTR=(YES,*nn*)
$2 + (2 \times nn)$

MFLD ,ATTR=(NO,*nn*)
$2 \times nn$

MFLD ,ATTR=(*nn*)
$2 \times nn$

MFLD ,ATTR=YES
2

MFLD ,ATTR=NO
0

ATTR=YES and *nn* are invalid if a literal value has been specified through the positional parameter in an output message.

The attributes in a field sent to another IMS ISC subsystem are treated as input data by MFS regardless of any ATTR= specifications in the format of the receiving subsystem. For example, a message field (MFLD) defined as ATTR=(YES,1),LTH=5 would contain the following:

```
00A0C2F1C8C5D3D3D6
```

If the MFLD in the receiving subsystem is defined as LTH=9 and without ATTR=, the application program receives:

```
00A0C2F1C8C5D3D3D6
```

If the MFLD in the receiving subsystem is defined as LTH=13 and ATTR=(YES,1), the application program receives:

```
4040404000A0C2F1C8C5D3D3D6
```

If the MFLD in the receiving subsystem is defined as LTH=5 and ATTR=(YES,1), the application program receives:

```
4040404000A0C2F1C8
```

The input SEG statement should be specified as GRAPHIC=NO to prevent translation of the attribute data to uppercase.

FILL=

Specifies a character to be used to pad this field when the length of the data received from the device is less than the length of this field. This character is also used to pad when no data is received for this field (except when MSG statement specifies option 3.) This operand is only valid if TYPE=INPUT. The default is X'40'.

X'*hh*'

Character whose hexadecimal representation is *hh* is used to fill fields. FILL=X'3F' is the same as FILL=NULL.

C'*c*'

Character *c* is used to fill fields.

NULL

Causes compression of the message segment to the left by the amount of missing data in the field. Refer to *IMS Version 9: Application Programming: Transaction Manager* for more information.

EXIT=

Describes the field edit exit routine interface for this message field. The exit routine number is specified in *exitnum*, and *exitvect* is a value to be passed to the exit routine when it is invoked for this field. The value of *exitnum* can range from 0 to 127. The value of *exitvect* can range from 0 to 255. The address of the field as it exists after MFS editing, (but before NULL compression for option 1 and 2), is passed to the edit exit routine, along with the vector defined for the field. (If NOFLDEXIT is specified for a DPM device, the exit routine will not be invoked.) The exit routine can return a code with a value from 0 to 255. MFS maintains the highest such code returned for each segment for use by the segment edit routine. EXIT= is invalid if '*literal*' is specified on the same MFLD statement.

ENDDO Statement

The ENDDO statement terminates the group of MFLD statements that are to be repetitively generated. The generated MFLD statements are printed immediately following the ENDDO statement. ENDDO is required when a DO statement has been specified.

label

A one- to eight-character alphanumeric name can be specified. It is not used.

MSGEND Statement

The MSGEND statement terminates a message input or output definition and is required as the last statement in the definition. If this is the end of the job submitted, it must also be followed by an END compilation statement.

label
a one- to eight-character alphanumeric name can be specified. It is not used.

Format Definition Statements

Format definition statements include the FMT statement, the DEV statement, the DIV statement, the DPAGE statement, the PPAGE statement, the DO statement, the RCD statement, the DFLD statement, the ENDDO statement, and the FMTEND statement. Details of each are provided in the information that follows.

FMT Statement

The FMT statement initiates and names a format definition that includes one or more device formats differing only in the device type and features specified in the DEV statement. Each device format included in the format definition specifies the layout for data sent to or received from a device or a remote program.

Format:

Parameters:

label
A required one- to six-character alphanumeric name that is referred to by message descriptors in the SOR= operand of MSG statements.

The name specified for *label* becomes part of the member name used for the resulting device output format and device input format blocks that are stored in the IMS.FORMAT library.

If DEV TYPE=DPM-An, and DIV OPTIONS=MSG, the name specified for *label* is sent to the remote program as the data name in the output message header.

If DEV TYPE=DPM-Bn, and DIV OPTIONS=(MSG,DNM), the name specified for *label* is sent to the other subsystem as the data structure name in the DD FM header.

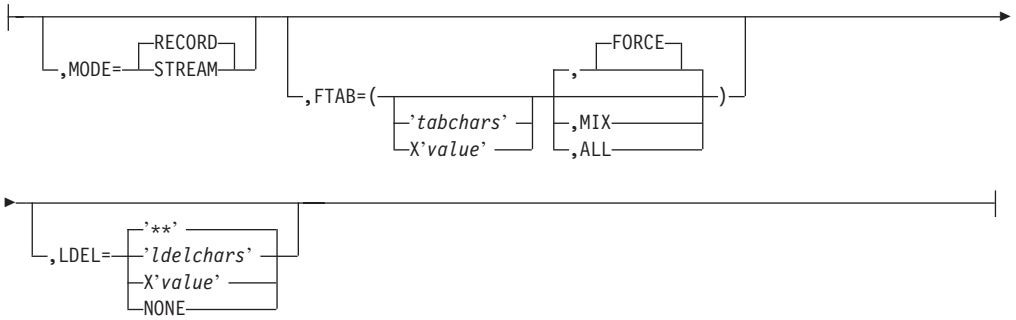
DEV Statement

The DEV statement defines device characteristics for a specific device or data formats for a specific device type. The DFLD statements following this DEV statement are mapped using the characteristics specified until the next DEV or FMTEND statement is encountered. For DPM devices, the DEV statement specifies the DPM program type number and (optionally) a feature set number.

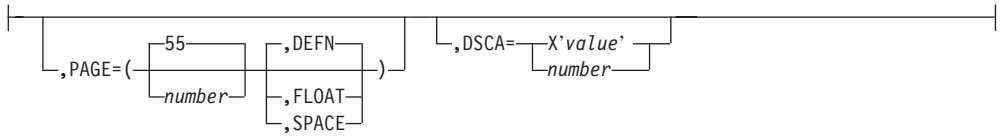
Recommendation: Read the TYPE= operand description before using the DEV statement.

Format for 2740 or 2741:

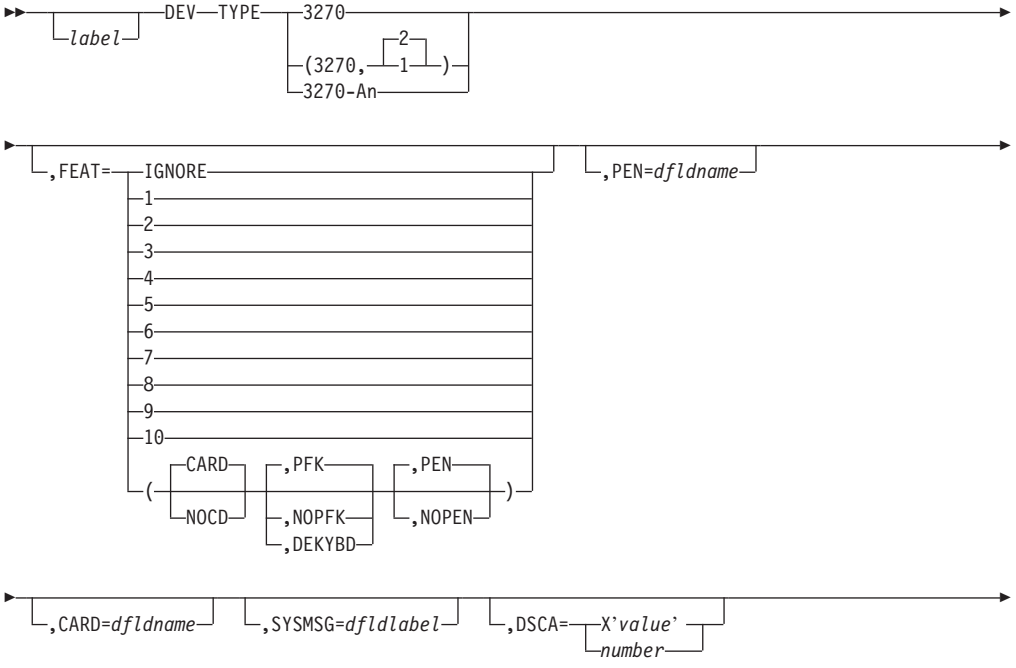
<FOR INPUT>:

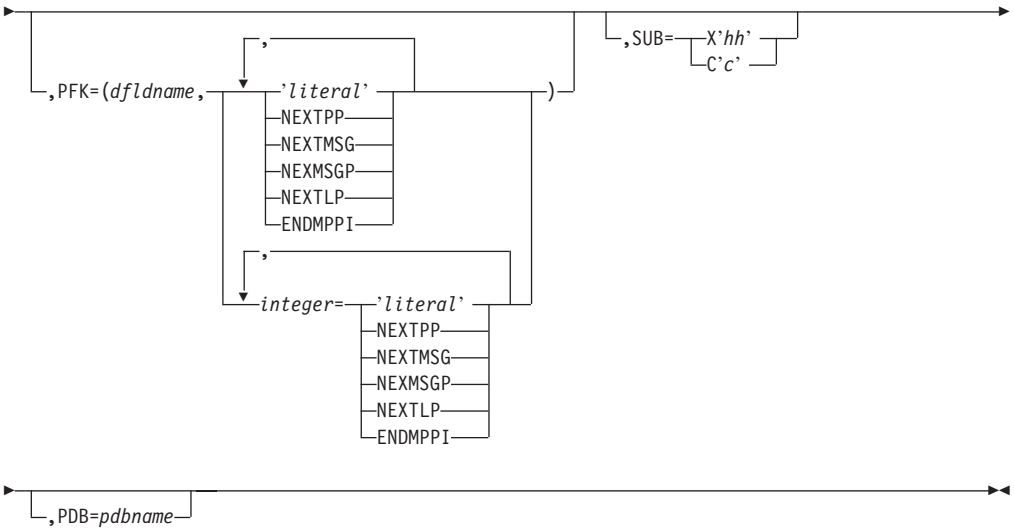


<FOR OUTPUT>:

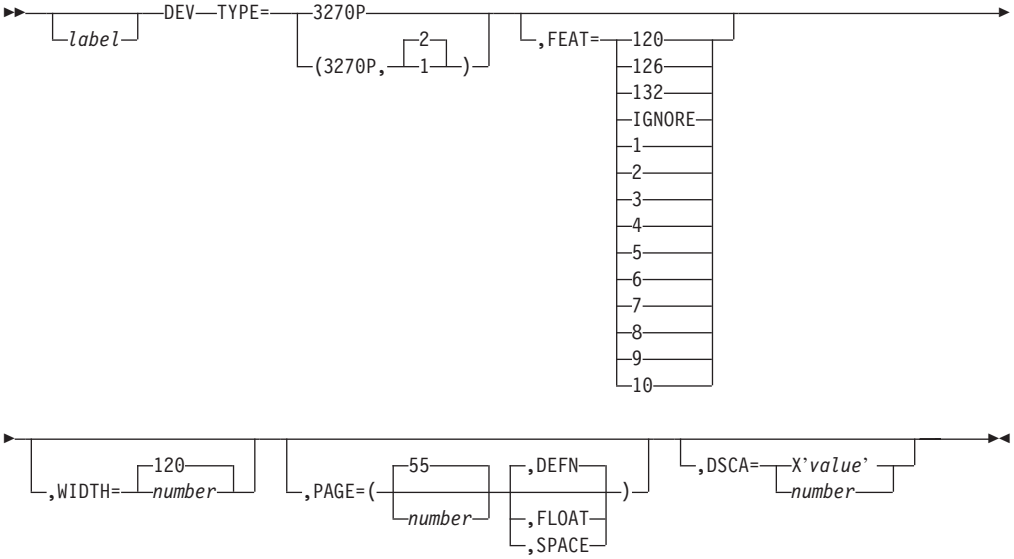


Format for 3270 Display:

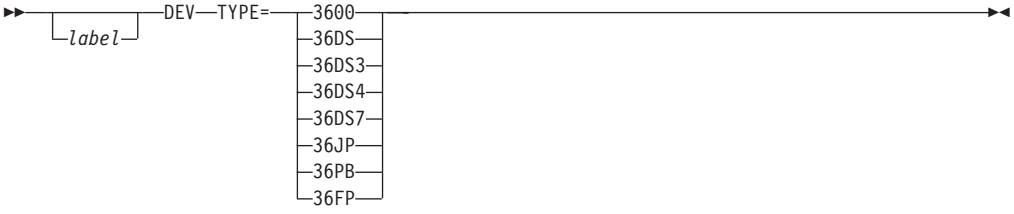




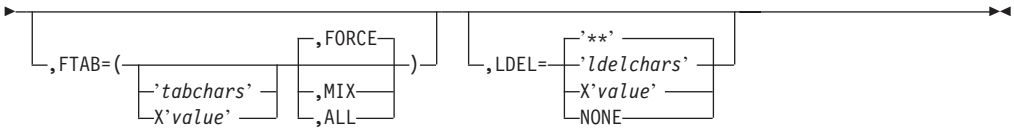
Format for 3270 Printers:



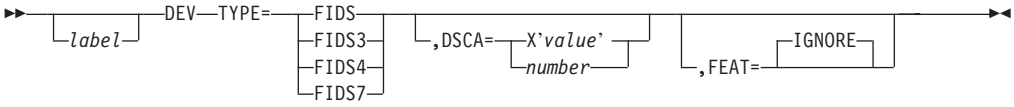
Format for Finance Workstations (3600 OR 4700):



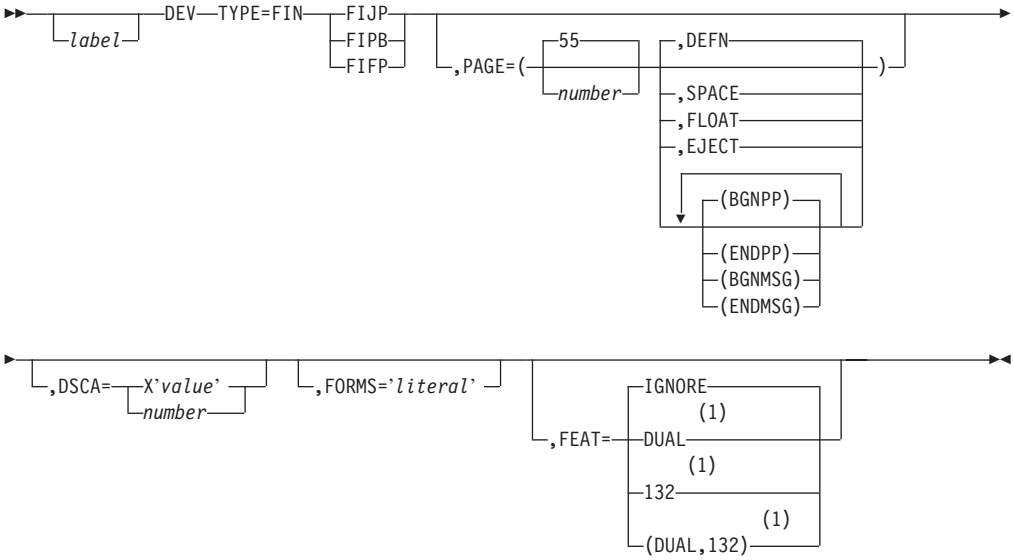
Format for DEV TYPE=FIN:



Format for DEV TYPE=FIDS, FIDS3, FIDS4, FIDS7:



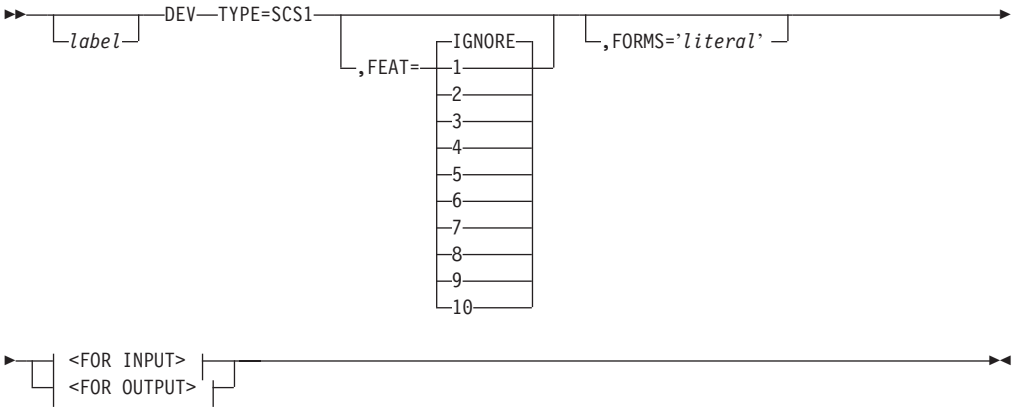
Format for DEV TYPE=FIJP, FIPB, FIFP:



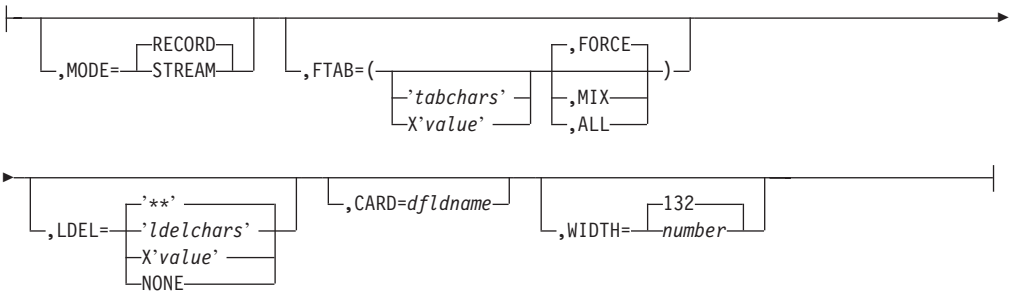
Notes:

1 FIFP only

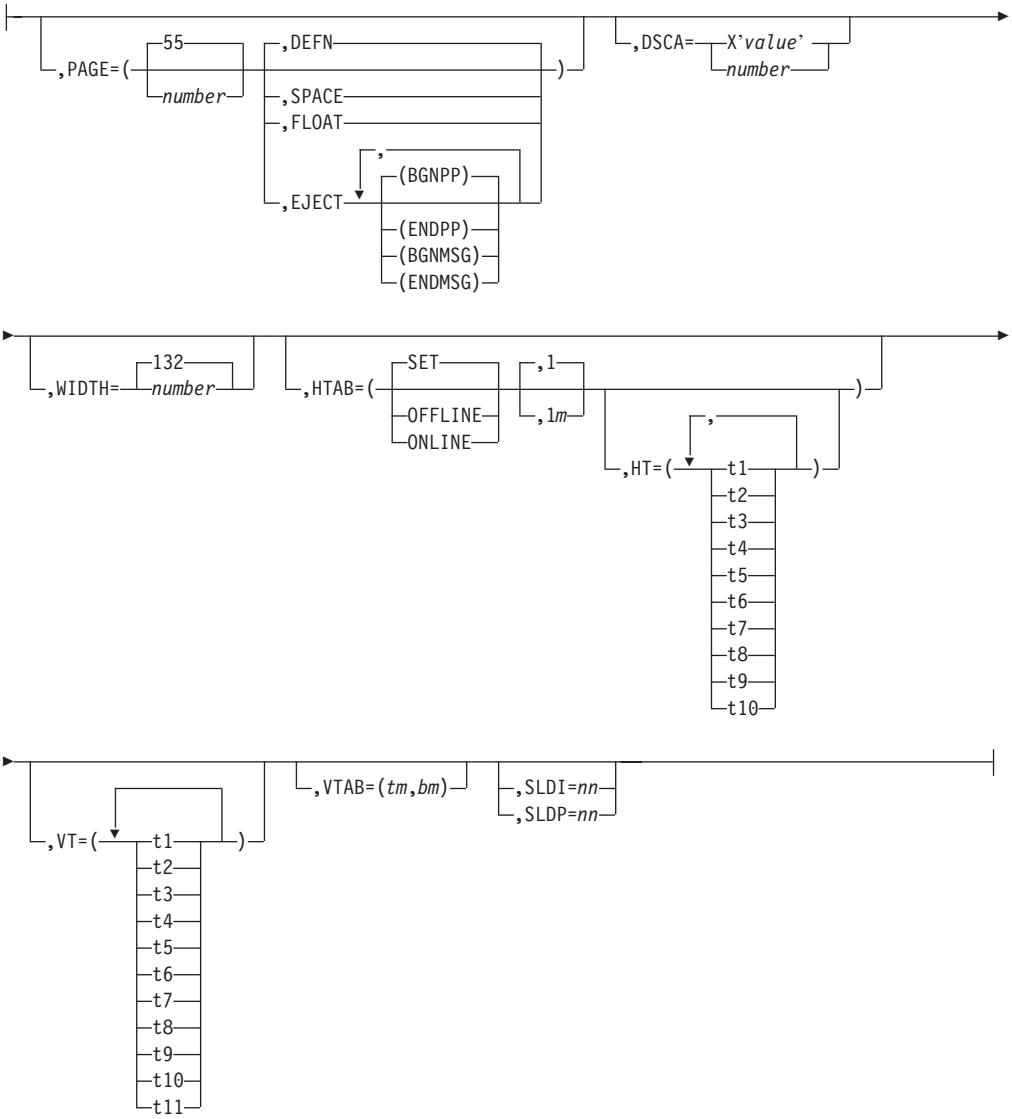
Format for SCS1:



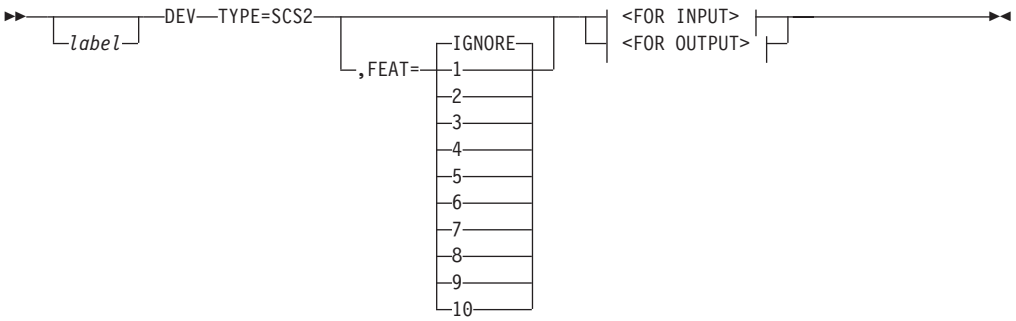
<FOR INPUT>:



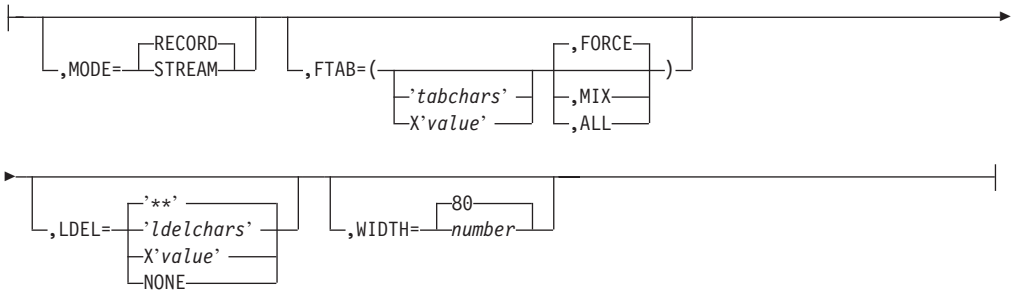
<FOR OUTPUT>:



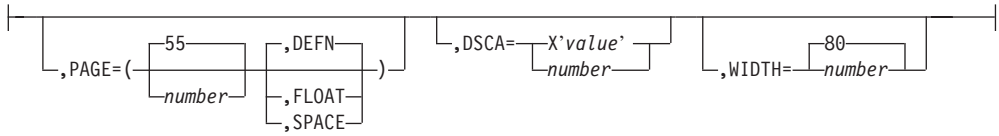
Format for SCS2:



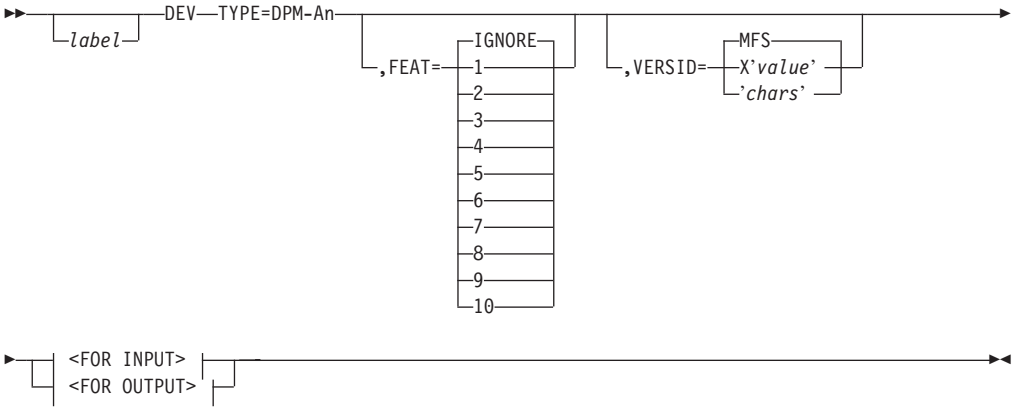
<FOR INPUT>:



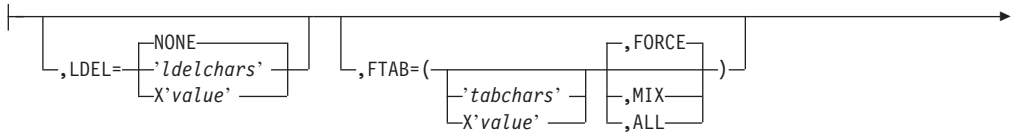
<FOR OUTPUT>:



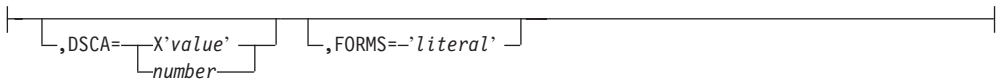
Format for DPM-An:



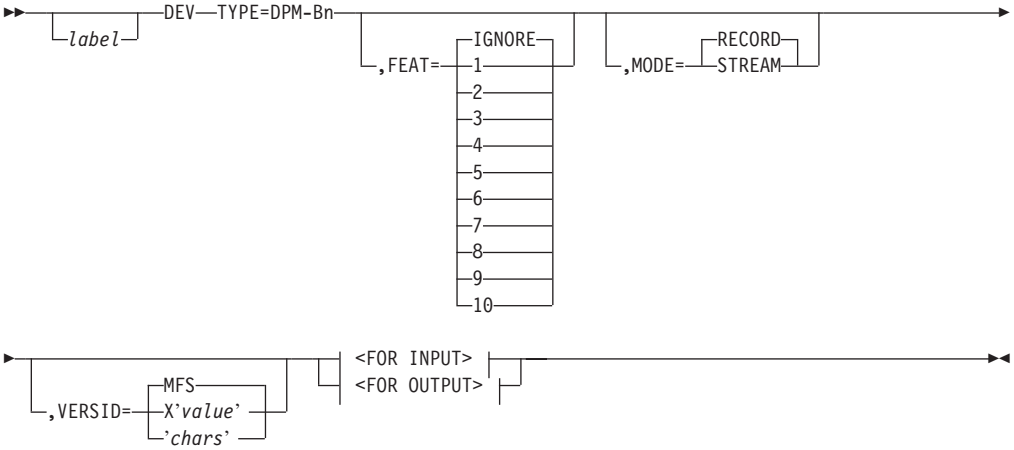
<FOR INPUT>:



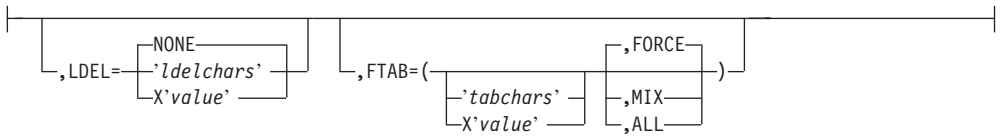
<FOR OUTPUT>:



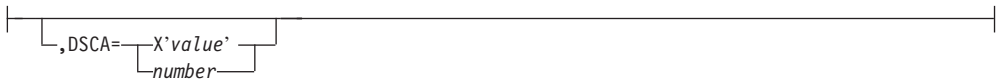
Format for DPM-Bn:



<FOR INPUT>:



<FOR OUTPUT>:



Parameters:

label
An optional one- to eight-character alphanumeric name that uniquely identifies this statement.

TYPE=
Specifies the device type and model number of a device using this format description. The 3284-3 printer attached to a 3275 is supported only as TYPE=3270P. The model number specified when defining a format for a 3284-3 is the model number of the associated 3275.

TYPE=3270-An specifies a symbolic name for 3270 and SLU 2 displays with the screen size defined during IMS system definition, feature numbers *n*=1-15.

This specification causes the MFS Language utility to read the MFS device characteristics table (DFSUDT0x) to extract the screen size.

TYPE=DPM-Bn specifies the device as an ISC node. The device type specified by *n* must agree with the specification of the component (COMPT=) on the system definition TERMINAL macro.

Based on the device and model used, specify:

TYPE=	Device-Model
274X	2740-1, 2741-1, or 2740-2
3270, 1	3275-1 3276-1,11 (defined at IMS system definition as 3270 model 1) 3277-1 3278-1 (defined at IMS system definition as 3277 model 1) SLU 2 (480 characters)
3270,2	3275-2 SLU 2 (1920 characters) (any display defined during IMS system definition as 'mod 2' with screen area of 1920 characters)
3270-An	3270-An (applies to any 3270 or SLU 2 display defined as TYPE=3270-An during IMS system definition) Examples of 3270 devices that can be defined as 3270-An and the recommended standard of associating screen sizes with the device type symbolic name follow:

Device	Screen size and definition
3180	24×80 screen size defined as 3270-A2
327X-1,11	12×80 screen size defined as 3270-A1
327X-2,12	24×80 screen size defined as 3270-A2
327X-3,13	32×80 screen size defined as 3270-A3
327X-4,14	43×80 screen size defined as 3270-A4
3278-5	27x132 screen size defined as 3270-A7
3290	62x160 screen size defined as 3270-A8 or 24×80 screen size defined as 3270-A2
5550	3270 Kanji Emulation or 3270 PC with 24×80 screen size defined as 3270-A2

3270P,1	3284-1 3286-1 3287 (with 480 character print feature and not attached as SLU 1 or SLU 4) 3289 (with 480 character print feature and not attached as SLU 1 or SLU 4)
3270P,2	3284-2 3286-2

3287 (with 1920 character print feature and not attached as SLU 1 or SLU 4)

3289 (with 1920 character print feature and not attached as SLU 1 or SLU 4)

FIN Finance application program (input only)

FIDS Finance display component (6×40; for example, 3604-1 or -2)

FIDS3 Finance display component (12×40; for example, 3604-3)

FIDS4 Finance display component (16×64; for example, 3604-4)

FIDS7 Finance display component (24×80; for example, 3604-7)

FIJP Finance journal printer

FIPB Finance passbook printer

FIFP Finance administrative printer

SCS1 The following console keyboard printers:

NTO
3771
3773
3774
3775
3776
5553
5557
SLU 1 (with a print data set or bulk printer)
SLU 4
3289 and 3287 when attached to IMS as SLU 1

SCS2 3521 card punch

3501 card reader

2502 card reader

SLU 1 (transmit data set)

SLU 4

DPM-An SLU P (n is value 1-15)

DPM-Bn ISC (n is value 1-15)

MODE=

Specifies the manner in which field scanning is to occur. Default value is RECORD. MODE= is valid for DPM-An input only, and for DPM-Bn input and output. For DPM-Bn, if the input and output modes are not the same, each DIV statement must be preceded by a DEV statement.

RECORD

Specifies that fields are defined as occurring within specific records (a line from a device, a transmission from a remote program) that is transmitted from the device or program. For DPM-Bn, Record mode must be specified for variable length, variable blocked (VLVB) format records.

STREAM

Specifies that fields are defined as a contiguous stream of fields—record

boundaries do not affect the MFS scan. Fields can be split across records and fields can be entered from any record provided they are entered in the defined sequence. For DPM-Bn, Stream mode must be specified for chained request/response units (RUs).

FTAB=

Specifies the field tab (FTAB) characters that you or a remote program can use to terminate an input field when either the length of the data entered is less than the defined field length, or no data for the field exists:

- For FIN, DPM-An, and DPM-Bn, a maximum of eight FTAB characters or 16 hexadecimal digits can be specified, and at least one character (or two hexadecimal digits) should be specified.
- For SCS1, up to four FTAB characters or eight hexadecimal digits can be specified; the characters NL, LF, HT, and VT are always FTAB characters and do not need to be specified.
- For SCS2, up to three FTAB characters or six hexadecimal digits can be specified. The characters NL, CR, LF, HT, and VT are always FTAB characters and do not have to be specified; however, they are received by MFS only if the Hollerith code is punched in the card if the input is from the card reader.

If no FTAB characters are defined, each device input field is considered to be of its defined length. In Record mode, when the end of a record is reached, the current field is terminated and all subsequent fields defined for that record are processed with no device data (message fill). In Stream mode, all transmissions that comprise the input message are treated as a stream of data fields unaffected by transmission boundaries. If FTABs are not defined or are not used for DPM input, each input field is considered to be of defined length except when NULL=DELETE is specified. With NULL=DELETE, if trailing nulls are encountered in a field or an entire field is null, the field is padded to defined length using the message fill character.

If FTAB characters are defined in this operand, either FORCE, MIX, or ALL can also be specified. The default is FORCE.

FORCE

Specifies that an FTAB is not required until you or a remote program enters an FTAB character. In record mode, if an FTAB is used for one field, the remaining fields of the current record must be terminated with an FTAB, regardless of length. In stream mode, if an FTAB is used for one field, the remaining fields in the message must be terminated with an FTAB.

MIX

Specifies that an FTAB is never required but can be used to terminate any input field when data is less than the defined field length.

ALL

Specifies that an FTAB must be used to terminate all fields, regardless of length, except for certain mode (MODE=) dependent conditions. In record mode, an FTAB is not required for the last field defined or entered in the record. In stream mode, an FTAB is not required for the last field defined or entered in the message.

LDEL=

Specifies two characters or four hexadecimal digits, which, if entered as the last two characters of a record of input data, cause the record to be discarded. A specification of NONE causes IMS to bypass record delete processing, except

for the first record, which is always deleted if the last two characters are asterisks (**). NONE is the default for DPM devices. For other devices, the default is **.

PAGE=
Specifies output parameters as follows:

number
For printer devices, number defines the number of print lines on a printed page; for card devices, number defines the number of cards to be punched per DPAGE or physical page (if *pp* parameter is used in the DFLD statements). This value is used for validity checking. The number specified must be greater than or equal to 1 and less than 256. The default is 55.

If VTAB= is specified for SCS1 printers, then the minimum value for PAGE= is 3.

DEFN
Specifies that lines/cards are to be printed/punched as defined by DFLD statements (no lines/cards are to be removed or added to the output page).

SPACE
Specifies that each output page contains the exact number of lines/cards specified in the *number* parameter.

FLOAT
Specifies that lines/cards with no data (all blank or NULL) after formatting are to be deleted.

For 3270P and SCS1 devices, some lines having no data (that is, all blank or null) must not be deleted under the following circumstances:

- The line contains one or more set line density (SLDx=) specifications.
- A field specified as having extended attributes spans more than one line.

EJECT
Specifies that a forms eject operation should be performed for printer devices. EJECT is valid only when TYPE=FIJP, FIPB, FIFP, or SCS1. If EJECT is specified for SCS1, MFS assumes the Vertical Forms Control feature is present. The default for the sublist is BGNPP.

The sublist specifies when ejects are to be performed:

BGNPP
Specifies that an eject is to be performed before each physical page of output.

ENDPP
Specifies that an eject is to be performed after each physical page is printed.

BGNMSG
Specifies that an eject is to be performed before any data in the message is printed.

ENDMSG
Specifies that an eject is to be performed after all message data is printed.

DSCA=
Specifies a default system control area (DSCA) for output messages using this device format. The DSCA supersedes any SCA specified in a message output descriptor if there are conflicting specifications. Normally, the functions specified

in both SCAs are performed. If the DSCA= operand is specified for SCS1 or SCS2, it is ignored. If the DSCA= operand is specified for 3270P, it is ignored, except for the bit setting for “sound device alarm”. If this bit is specified on the DSCA/SCA option, it is sent to the device. For TYPE=DPM-An or DPM-Bn, DSCA/SCA information is sent to a remote program or ISC subsystem only if a DFLD definition requests it.

The value specified here must be a decimal number not exceeding 65535 or X'*hhhh*'. If the number is specified, the number is internally converted to X'*hhhh*'.

The two bytes of the DSCA field should be defined as shown in Table 27 or Table 29 on page 444.

Table 27 shows the DSCA bit settings for 3270 display or SLU 2 devices or TYPE=DPM-An or DPM-Bn.

Table 27. Bit Settings for DSCA Field. For 3270 Display, SLU 2 Devices, TYPE=DPM-An, or DPM-Bn

Byte	Bit	
0	0-7	Should be 0.
1	0	Should be 1.
	1	Force format write (erase device buffer and write all required data).
	2	Erase unprotected fields before write.
	3	Sound device alarm.
	4	Copy output to candidate pointer. Bits 1-4 are ignored for DPM-Bn.
	5	B'0'- For 3270, protect the screen when output is sent. For DPM, demand paging can be performed. B'1'- For 3270, do not protect the screen when output is sent. For DPM-B, autopaging can be performed.
	6-7	Should be 0, except for the 3290 in partitioned format mode.

If byte 1 bit 5 is set to B'1' (unprotect screen option) for a 3275 display, and both input and output occur simultaneously (contention), the device is disconnected. For non-3275 devices, the SCA option is ignored. If byte 1 bit 5 is set to B'0', the application program can request autopaged output by setting the SCA value to B'1'. This request is honored only if present in the first segment of the first LPAGE of the output message.

If a nonzero value is specified for byte 0, or for bit 6 or 7 in byte 1, MFS overrides the specified value with zero, except for the 3290 in partitioned format mode.

For the 3290 in partitioned format mode, byte 1 bit 6 has special significance. If the DOF of the output message is the same as the DOF of the last message, then byte 1 bit 6 of the DSCA is checked for the erase/not erase partitions option before the output message is sent. Meanings of the bit 6 settings are shown in Table 28.

Table 28. 3290 Partitioned Format Mode Bit Setting

Byte	Bit	Setting	Meaning
1	6	B'1'	Erase all partitions before sending output message.
		B'0111'	Do not erase existing partitions.

The default is B'0' (do not erase). If bit 6 is defined, all existing partitions are erased and the output is sent according to the specified partition paging option. See “Application Programming Using MFS” in *IMS Version 9: Application Programming: Transaction Manager*. If bit 6 is not defined, the output is sent according to the specified partition paging option and partitions that do not receive output remain in the state they were in before output was sent.

Table 29 shows the DSCA bit settings for TYPE=FIDS, FIDS3, FIDS4, FIDS7, FIJP, FIPB, or FIFP.

Table 29. Bit Settings for DSCA Field. For TYPE=FIDS, FIDS3, FIDS4, FIDS7, FIJP, FIPB, or FIFP

Byte	Bit	
0	0-7	Should be 0.
1	0	Should be 1.
	1-2	Not applicable for FIN output devices.
	3	Set 'device alarm' in output message header.
	4	Not applicable for FIN output devices.
	5-7	Should be 0.

For FIN devices, if a nonzero value is specified for byte 0, or for bits 1, 2, 5, 6, or 7 in byte 1, MFS overrides the specified value with zero.

Bits 1, 2, and 4 in byte 1 only function for 3270 and SLU 2 and are therefore not applicable to FIN. If set on, and the message is edited for an FIN output device, they are ignored.

For 3270 and FIN devices, the function specified is performed. For DPM devices, the specification is supplied to the remote program in a user-defined device field (DFLD).

FEAT=
Specifies features for this device or program group.

IGNORE
Specifies that device features are to be ignored for this device.

120|126|132
Specifies line length for 3284, and 3286 device types (TYPE=3270P).

CARD
Specifies that the device has a 3270 operator identification card reader. NOCD specifies the absence of the CARD feature.

DEKYBD
Specifies data entry keyboard feature. This feature implies PFK feature; therefore, PFK is invalid if DEKYBD is specified. NOPFK implies the absence of PFK and DEKYBD features.

PFK
Specifies that the device has program function keys. NOPFK specifies the absence of the PFK and DEKYBD features.

PEN
Specifies the selector light pen detect feature. NOPEN specifies the absence of the PEN feature.

DUAL
Specifies that the FIFP device has the dual independent forms feed feature.

132
Specifies that the FIFP device has the expanded print line feature.

1|2|3|4|5|6|7|8|9|10
Specifies customer-defined features for the SCS1, SCS2, 3270P, DPM-An, or DPM-Bn device type.

For SCS1, SCS2, and 3270P devices, FEAT= allows grouping of devices with special device characteristics. For example, FEAT=1 could group devices with a maximum of 80 print positions and no VFC, and FEAT=2 could group devices with 132 print positions and the VFC feature. FEAT=IGNORE should be specified to group together devices with a minimum set of device capabilities. For 3270P devices, when WIDTH= is specified, FEAT=(1...10) must also be specified. If FEAT=(1...10) is specified but WIDTH= is not specified, WIDTH= defaults to 120.

For DEV TYPE=DPM-An or DPM-Bn, FEAT= specifies a user-defined group of device formats so that programs with common features and dependencies can be selected together.

When IGNORE is specified, no other values should be coded in the FEAT= operand. When FEAT=IGNORE is not specified in the TERMINAL macro during system definition, the MSG statement must specify IGNORE in the SOR= operand for the device format with the IGNORE specification. Unless FEAT=IGNORE is used, FEAT= must specify **exactly** what was specified in the TERMINAL macro during IMS system definition. If it does not, the DFS057 error message is issued. When FEAT=IGNORE or 1-10 is specified for 3270 devices, the operands PEN=, CARD=, and PFK= can still be specified. When TYPE=3270P and FEAT=IGNORE, MFS allows a line width of 120 characters.

CARD, PFK, DEKYBD, and PEN feature values are valid only for 3270 displays. DUAL is valid only if TYPE=FIFP. If the FEAT= operand is omitted, the default features are CARD, PFK, and PEN for 3270 displays; the default line width is 120 for TYPE=3270P and 80 for TYPE=FIFP.

1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 are valid values only for 3270, 3270P, 3270-An, SCS1, SCS2, DPM-An, and DPM-Bn (for DEV TYPE=). For 3270 displays, the FEAT= specifications of 1 to 5 can be used to group devices with specific features or hardware data stream dependencies.

Restriction: This keyword is optional and cannot be used with any other feature specification for 3270 displays.

When using the same format for both the 3290 and the 3180, you must specify a different value on the FEAT= operand for each device type. The FEAT parameter values selected for each device must also be specified on the TERMINAL macro in the IMS system definition.

For 274X, FIN, FIDS, FIDS3, FIDS4, FIDS7, FIJP, and FIPB, FEAT is always IGNORE. For FIFP, IGNORE is used unless 132 and DUAL are specified.

Feature operand values can be specified in any order, and only those values desired need be specified. The underlined values do not have to be specified because they are defaults. Only one value in each vertical list can be specified.

Examples: Some of the uses of the FEAT= specification are:

- TYPE=DPM-A1,FEAT=1 could group device formats with DPAGE paging option and simulated attributes.
- TYPE=DPM-A5,FEAT=2 could group device formats with no paging option and bit string attributes (which are not interpreted by MFS).
- TYPE=DPM-B1,FEAT=IGNORE could identify device formats with PPAGE paging option and a minimum set of program requirements.

PFK=

Defines an input field name to contain program function key literal or control function data (first subparameter) and, in positional or keyword format, either the literal data to be placed in the specified field, or the control function to be performed when the corresponding function key is entered (remaining subparameters).

The name of the first subparameter (the input field name that will contain the program function key literal or control function data) can be referred to by an MFLD statement and must *not* be used as the label of a DFLD statement within this DEV definition. The remaining subparameters can be specified in positional or keyword format. If the subparameters are in keyword format, the integer specified must be from 1 to 36, inclusive, and not duplicated. Only one PFK= operand format (positional or keyword) can be specified on a DEV statement. This operand is valid only for 3270 displays. At the time the actual format blocks are created, each literal is padded on the right with blanks to the length of the largest literal in the list. The maximum literal length is 256 bytes.

If the device supports the IMS copy function, then PFK12 invokes the copy function and the definition of PFK12 in the DEV statement is ignored; otherwise, the definition of PFK12 is honored.

If FEAT=NOPFK is specified, it is changed to PFK. The maximum number of user-defined PFKs is 36.

Control functions that can be specified are:

NEXTPP—PAGE ADVANCE

Specifies a request for the next physical page in the current output message. If no output message is in progress, no explicit response is made.

NEXTMSG—MESSAGE ADVANCE

Specifies a request to dequeue the output message in progress (if any) and to send the next output message in the queue (if any).

NEXTMSGP—MESSAGE ADVANCE PROTECT

Specifies a request to dequeue the output message in progress (if any), and send the next output message or return an information message indicating that no next message exists.

NEXTLP—NEXT LOGICAL PAGE

Specifies a request for the next logical page of the current message.

ENDMPPI—END MULTIPLE PAGE INPUT

Specifies the end of a multiple physical page input message.

PEN=

Defines an input field name to contain literal data when an immediate light pen detection of a field with a space or null designator character occurs. The literal data is defined on the DFLD statement with the PEN= operand. (See PEN= operand on the DFLD statement.) This name can be referred to by an MFLD statement and must not be used as the label of a DFLD statement within this

DEV definition. The PEN= operand is valid only for 3270 displays. If FEAT=NOPEN is specified, it is changed to PEN.

If an immediate detect occurs on a field defined with a space or null designator character, and either another field has been selected or modified or has the MOD attribute, or the PEN= operand is not defined for the DFLD, a question mark (?) is inserted in the PEN= field name.

If no immediate detection occurs or the immediate detect occurs on a field defined with an ampersand (&) designator character, the PEN= operand is padded with the fill specified in the MFLD statement.

CARD=

Defines the input field name to receive operator identification card data when that data is entered. This name can be referenced by an MFLD statement and must not be used as the label of a DFLD statement within this DEV definition. This operand is valid only if a 3270 display or SCS1 is specified. If FEAT=NOCD is specified for a 3270 display, it is changed to CARD. All control characters are removed from magnetic card input before the data is presented to the input MFLD that refers to this card field name.

For 3270 displays, an unprotected field large enough to contain the magnetic card data and control characters must be defined through a DFLD statement. Position the cursor to this field and insert the card in the reader to enter card information. The card data is logically associated with the CARD= field name, not the name used in the DFLD statement.

For device TYPE=SCS1, only card data with the operator ID (OID) character is associated with this field name. Cards with the OID character can be entered at any time during data entry. MFS treats data without the OID character as if it were data entered from the keyboard.

SYSMSG=

Specifies the label of the DFLD statements that define the device field in which IMS system messages are to be displayed. This operand is valid only if a 3270 display is specified. A DFLD with this label should be defined for each physical page within each DPAGE defined within this DEV definition. DFLDs for SYSMSG should be at least LTH=79 to prevent message truncation. The referenced DFLD can also be referenced by an MFLD statement.

FORMS=

Specifies a 1- to 16-byte literal. For the FIN, this literal is included in the output message header for each message sent to the device using this FMT. The data can be used by the FIN application program to ensure that special forms required for a given message are mounted on the device and that page size and forms alignment are established.

For SCS1 output to SLU 1 print data set components or SLU 4, this literal names the data set to receive IMS output. For 3770 programmable models defined to IMS as SLU 1 or SLU 4, however, the literal is ignored by the terminal and all print data set output goes to the SYS.INTR data set. For all SCS1 output to 3770 (nonprogrammable), SLU 1 non-PDS components or SLU 4, the literal is ignored.

For DEV TYPE DPM-An, this literal is included in the output message header. If the DPAGE or PPAGE paging option is specified, the literal is part of the special forms output message header sent as a separate transmission, followed (after a paging request from the remote program) by the DPAGE or PPAGE output message header and data records. If the default MSG option is selected, the output message header with literal is sent as the first record, followed by data

records. See *IMS Version 9: Application Programming: Transaction Manager* for a discussion of output headers with the forms literal.

WIDTH=
Specifies the maximum line width for this DEV type as one of:

- Number of print positions per line of input or output data
- Number of punch positions per card of input or output data
- Card width for card reader input data

The defaults are 132 for SCS1 input and output, 80 for SCS2 input and output, and 120 for 3270P output. A specified number cannot exceed 255 for SCS1 and 249 for SCS2. Line width is specified relative to column 1, regardless of whether a left margin value is specified in the HTAB= keyword (SCS1 and SCS2 only). The width specified must be greater than or equal to 1.

For 3270P devices, if WIDTH is specified, then FEAT=(1...10) must also be specified. If FEAT=(1...10) is specified, and WIDTH= is not specified, WIDTH= defaults to 120.

HTAB=
Specifies when TYPE=SCS1:

- Where on the device MFS should set horizontal tab stops
- Whether and when MFS should insert tab control characters in the output message to cause horizontal tabbing
- Where on the device MFS should position the left margin

If HTAB= is not specified, no horizontal tabbing is done and the left margin position is assumed to be column 1.

SET|ONLINE|OFFLINE
Specifies that MFS should set horizontal formatting controls for the device. When MFS sets horizontal format controls for the device, the following characteristics are established: maximum line width, left and right margins, and horizontal tab stops. The default is SET when the HTAB= keyword is present.

SET
Specifies that MFS should set horizontal tab stops but should not insert tab control characters into the output message. You can then use horizontal tabbing on subsequent input.

ONLINE
Specifies that MFS should set horizontal tab stops at the specified (HT=) locations and insert tab control characters during online processing.

OFFLINE
Specifies that MFS should set horizontal tab stops at the specified (HT=) locations and insert tab control characters during offline compilation of the format.

1|lm (left margin)
Specifies the column position of the left margin. The default is 1. The value specified must be less than the WIDTH= value.

HT=
Specifies from 1 to 10 horizontal tab stop locations. The values specified must be relative to position 1, equal to or greater than the left margin value, and less than the WIDTH= value.

VT=

Specifies that MFS should insert tab control characters at the specified locations. From 1 to 11 vertical tab stop locations can be specified. If VTAB= is specified, the VT= values specified must be relative to line 1 and equal to or less than the bottom margin specified on the VTAB= keyword. If VTAB= is not specified, the VT= values must be equal to or less than the page depth specified in the PAGE= keyword. The maximum value is 255. If a value greater than 255 is specified, 255 is assumed and no error message is generated. VT= is valid only when TYPE=SCS1. If PAGE=(n,FLOAT) is specified, VT= is invalid.

X'00' is accepted as a valid tab stop only if VTAB= is also specified.

Together with VTAB= and PAGE=, VT= comprises a data stream to set the vertical format of the page. *tm* on the VTAB= keyword must be greater than or equal to 1 and less than *t1* on the VT= keyword. *bm* on the VTAB= keyword must be greater than or equal to *t11* on the VT= keyword and less than or equal to the maximum page length specified on the PAGE= keyword.

VTAB=

For SCS1 printers, specifies top (*tm*) and bottom (*bm*) page margins. Together with VT= and PAGE=, VTAB= comprises a data stream to set the vertical format of the page. *tm* must be greater than or equal to 1 and less than *t1* on the VT= keyword. The maximum *tm* is 253.

bm must be greater than or equal to *t11* on the VT= keyword and less than or equal to the maximum page length specified on the PAGE= keyword. *bm* must be at least two greater than *tm*. If VTAB= is specified, then the PAGE= value must be 3 or greater.

A form feed (FF) is inserted after the set vertical format (SVF) data stream if the top margin (*tm*) specified on the VTAB= keyword is not equal to 1.

If PAGE=(n,FLOAT) is specified, VTAB= is invalid.

SLDI=

For SCS1 printers, specifies the line density for an output message in lines per inch. (See also SLDP=). SLDI= can also be specified on the DFLD statement. The SLDI= value must be from 1 through 72 and consistent with the architecture of the device for which it is specified (see the appropriate device or component manual).

If SLDI= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDI= specification is encountered, but after any vertical tabs and new line characters.

Restriction: You cannot specify both SLDI= and SLDP= on the DEV statement.

The SLDI= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

SLDP=

For SCS1 printers, specifies the line density for an output message in points per inch. (See also SLDI=). SLDP= can also be specified on the DFLD statement. The SLDP= value must be from 1 through 72 and consistent with the architecture of the device for which it is specified (see the appropriate device or component manual).

If SLDP= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to

set the line density. The second is sent within the message, just prior to the field on which the SLDP= specification is encountered, but after any vertical tabs and new line characters.

Restriction: You cannot specify both SLDP= and SLDI= on the DEV statement.

The SLDP= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

Recommendation: Be careful when defining set line density (SLDx) keywords to ensure that forms alignment is maintained. If SLDx= is improperly defined, loss of forms alignment can occur.

VERSID=
Specifies any two-character or 2-byte hexadecimal value as the version ID. If MFS is specified or if the VERSID keyword is not specified, MFS calculates the version ID. MFS is the default.

The version ID is calculated by MFS and is based on the date and time stamp that an FMT definition has compiled. The value is printed on the MFS Language utility output so you can refer to it in format definitions.

SUB=
Specifies the character used by MFS to replace any X'3F' characters in the input data stream. No translation occurs if this parameter is specified as X'3F' or this parameter is not specified, or the input received bypasses MFS editing. The specified SUB character should not appear elsewhere in the data stream; therefore, it should be nongraphic.

X'*hh*'
Character whose hexadecimal representation is '*hh*' replaces all X'3F' in the input data stream.

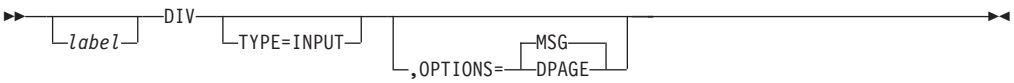
C'*c*'
Character '*c*' replaces all X'3F' in the input data stream.

PDB=
(For the 3290 or 3180 in partitioned format mode) specifies the name of the Partition Descriptor Block that is used to describe the partition set for an output or input message. This parameter is valid only for DEV statements that specify TYPE=3270-An.

DIV Statement

The DIV statement defines device formats within a DIF or DOF. The formats are identified as input, output, or both input and output, and can consist of multiple physical pages. For DEV TYPE=274X, SCS1, SCS2, or DPM-AN, two DIV statements can be defined: DIV TYPE=OUTPUT and DIV TYPE=INPUT. For all other device types, only one DIV statement per DEV is allowed.

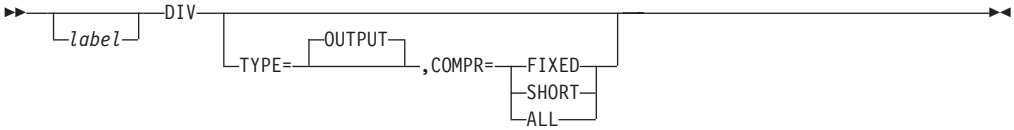
Format for DEV TYPE=274X, SCS1, or SCS2 and DIV TYPE=INPUT:



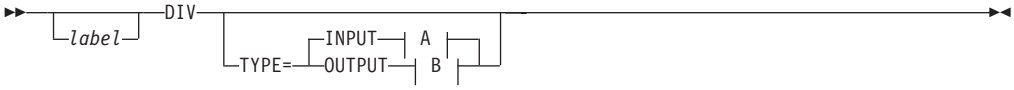
Format for DEV TYPE=3270 or 3270-An:

Format for DEV TYPE=FIN:

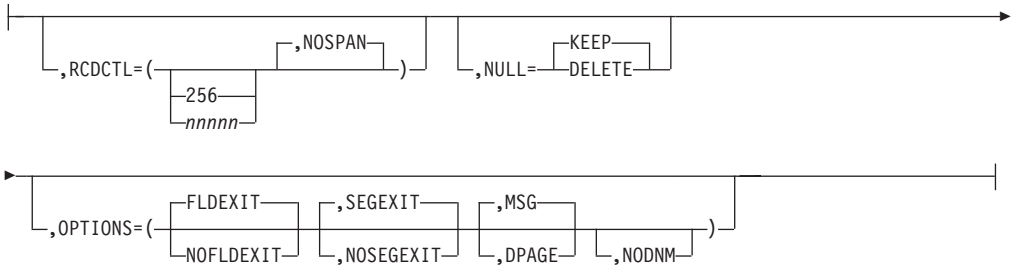
Format for DEV TYPE=274X, SCS1, SCS2, 3270P, FIDS, FIDS3, FIDS4, FIDS7, FIJP, FIPB, or FIFP and DIV TYPE=OUTPUT:



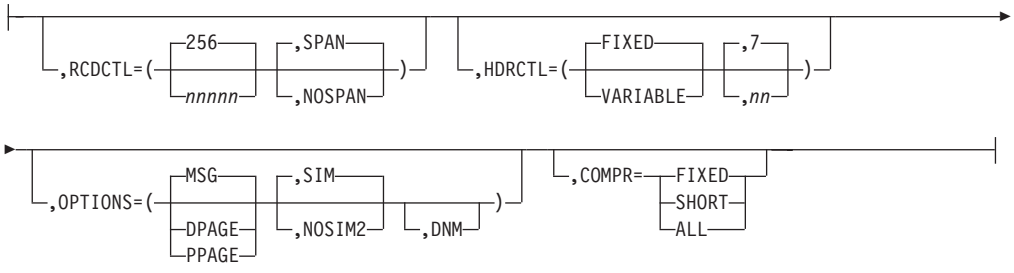
Format for DEV TYPE=DPM-An:



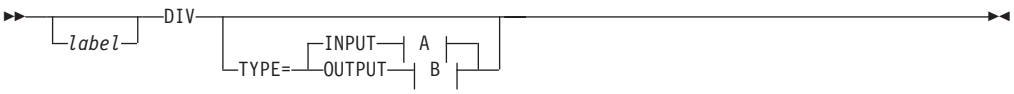
A:



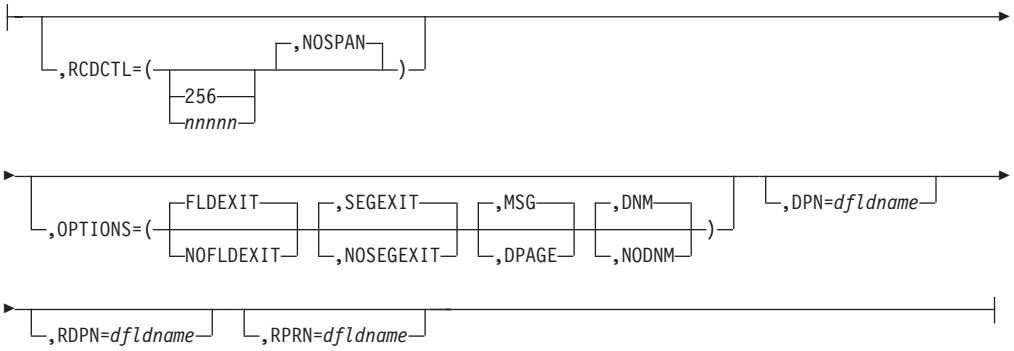
B:



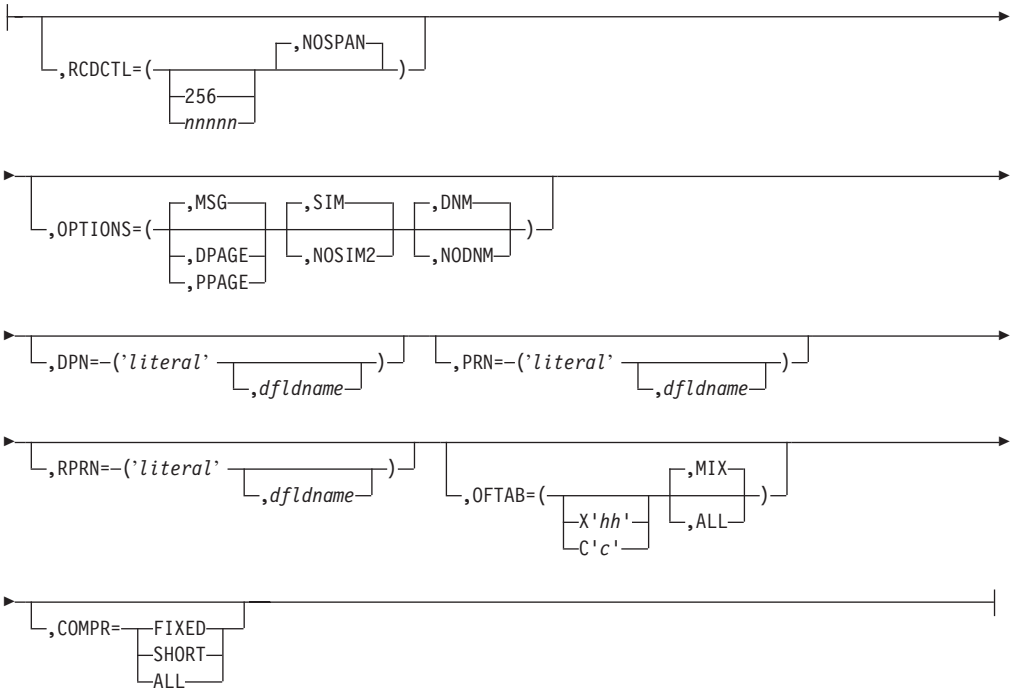
Format for DEV TYPE=DPM-Bn:



A:



B:



Parameters:

label
A one- to eight-character alphanumeric name can be specified to uniquely identify this statement.

TYPE=
Describes an input only format (INPUT), an output only format (OUTPUT), or both (INOUT).

If DIV TYPE=OUTPUT or TYPE=INPUT is specified, certain DEV statement keywords are applicable.

For example, specifying WIDTH=80 for DEV TYPE=SCS1 indicates that fields can be printed in columns 1 through 80 on output and received from columns 1 through 80 on input. Specifying WIDTH=80 for DEV TYPE=SCS2 indicates that both the card reader and card punch have the same number of punch positions. Specifying WIDTH=80 and HTAB=(SET,5) for DEV TYPE=SCS1 indicates that fields can be printed in columns 5 through 80 and received from columns 5 through 80 on input. In this case DFLD POS=(1,5) or POS=5 on input is the same as if you specified column 1 and a left margin position at 1. You enter data the same way, regardless of where the left margin is currently set.

RCDCTL=

This parameter is valid only if MODE=RECORD is specified on the DEV statement. For DEV TYPE=DPM-An or DPM-Bn only, RCDCTL specifies the maximum length of an input or output transmission record. For DPM-An, RCDCTL specifies whether (SPAN) or not (NOSPAN) fields can span records. The RCDCTL number cannot be larger than 32000 and should not be less than the length of the message output header (For DPM-An, see HDRCTL discussion.) The default value is 256. RCDCTL creates record definitions even if RCD statements are used in the same format definition.

- For DEV TYPE=DPM-An or TYPE=Bn and DIV TYPE=INPUT

  For an input format definition, fields must *not* span record boundaries, and therefore must be within the length specified by the RCDCTL value. NOSPAN is the default.

- For DEV TYPE=DPM-An or Bn and DIV TYPE=OUTPUT

  The RCDCTL size specified should be less than or equal to the output buffer size specified in the OUTBUF= macro at IMS system definition. If the RCDCTL size is greater than the OUTBUF value specified, one record might require multiple output transmissions and might produce undesirable results in the remote program. If fields do not exactly fit in the defined records, and NOSPAN has been specified, records might not be completely filled.

  The RCDCTL also specifies whether (SPAN) (for DPM-An only) or not (NOSPAN) a field can span record boundaries. If SPAN is specified (for DPM-An only), some fields can span a record boundary (but never a PPAGE boundary), and the remote program must include logic to associate the partial fields or deal with them separately.

  If NOSPAN is specified, every field is entirely contained within a record and no field will have a length greater than the RCDCTL value specified.

  The first data field is the first field of the message for OPTIONS=MSG. The first data field is the first field of the DPAGE or PPAGE for OPTIONS=DPAGE and PPAGE respectively. If the first data field does not fit in the same record as the output message header, and if OPTIONS=DPAGE or PPAGE has been specified, the first data record will be sent in the next transmission. The output message header will be transmitted by itself (as is always the case for OPTIONS=MSG).

NULL=

For DEV TYPE=DPM-An and DIV TYPE=INPUT only, NULL= specifies whether MFS is to ignore (KEEP) or search for and replace (DELETE) trailing nulls in fields. If NULL=DELETE is specified, MFS searches input message fields for trailing nulls or for fields that are all nulls, and replaces the nulls with the fill character specified in the message definition. See “Application Programming Using MFS” in *IMS Version 9: Application Programming: Transaction Manager* for a discussion of the effects of NULL=DELETE.

OPTIONS=

For DIV TYPE=INPUT, the OPTIONS keyword specifies the exit routines to be

called, the type of paging or delivery requested, and, for DPM-Bn only, the selection of the DPAGE data name to be used to map data.

For DIV TYPE=OUTPUT, the OPTIONS= keyword specifies the type of paging or delivery requested, the type of attribute processing requested, and, for DPM-Bn only, the selection of the DPAGE data name to be used to map data.

For DPM output messages, the option selection determines how records are constructed for transmission to the remote program or ISC subsystem and effects the distribution of processing and logic between the IMS application program and the remote program or ISC subsystem.

- For DEV TYPE=DPM-An or TYPE=DPM-Bn and DIV TYPE=INPUT

  FLDEXIT|NOFLDEXIT
  SEGEXIT|NOSEGEXIT
  Input data from this device type can be partially edited by the remote program before it is sent to IMS. For input format definitions, this parameter specifies whether (FLDEXIT and SEGEXIT) or not (NOFLDEXIT and NOSEGEXIT) exit routines specified in the MSG definition MFLD and SEG statements, respectively, are to be called for this DPM format. If NOFLDEXIT or NOSEGEXIT is specified, the corresponding exit routine is bypassed. FLDEXIT and SEGEXIT are the defaults.

  MSG
  Specifies that an input message can be created from a single DPAGE.

  DPAGE
  Specifies that an input message can be created from multiple DPAGEs. If multiple DPAGE input is not requested in MFS definitions, messages can *not* be created from more than one DPAGE. In this case:

  If a single DPAGE is transmitted and contains more data than defined for the DPAGE selected, the input message is rejected and an error message is issued.

  If multiple DPAGEs are transmitted, the input message is rejected and an error message is issued.

  NODNM (DPM-An only)
  DNM/NODNM (DPM-Bn only)
  When a data name (DNM) is specified or defaulted to (DPM-Bn only), a specific DPAGE is selected to map the current or only data transmission when:

  The DPAGE data name is supplied as the DSN parameter in the message header, and

  The DPAGE data name matches a defined DPAGE data name.

  If these conditions are not met, the last defined DPAGE name is used to map the data, unless the DPAGE is defined as conditional.

  When no data name (NODNM) is specified (for either DPM-An or -Bn) MFS selects a specific DPAGE by performing a conditional test on the data received and the COND= parameter.

- For DEV TYPE=274x, SCS1, SCS2, FIN, and DIV TYPE=INPUT

  MSG
  Specifies that an input message can be created from a single DPAGE.

DPAGE
Specifies that an input message can be created from multiple DPAGEs. If multiple DPAGE input is not requested in MFS definitions, messages can *not* be created from more than one DPAGE. In this case:

- If a single DPAGE is transmitted and contains more data than defined for the DPAGE selected, the input message is rejected and an error message is sent to the other subsystem.
- If multiple DPAGEs are transmitted, the input message is rejected and an error message is sent to the other subsystem.

- For DEV TYPE=DPM-An or TYPE=DPM-Bn and DIV TYPE=OUTPUT

MSG
Is the default and specifies that IMS will transmit all the DFLDs within a message together as a single message group. The message is preceded by an output message header. All DFLDs are transmitted. For DPM-Bn, the data structure name is optional in the header.

DPAGE
Specifies that IMS will transmit all DFLDs that are grouped in one logical page together. The logical page will be transmitted in one or more records. If PPAGE statements are defined with the DPAGE, each PPAGE statement begins a new record. An additional logical page will be sent when a paging request is received from the remote program. Each logical page is preceded by an output message header, and the label on the DPAGE is placed in the header. For DPM-Bn, the data structure name is optional in the DD header and depends on the specification of DNM/NODNM.

PPAGE
Specifies that IMS will transmit the DFLDs that are grouped in one presentation page (PPAGE) together in one chain. The presentation page will be transmitted in a group of one or more records. An additional presentation page will be sent when a paging request is sent to IMS from the remote program. Each presentation page is preceded by an output message header, and the label on the PPAGE statement is placed in the header. For DPM-Bn, the data structure name is optional in the DD header and depends on the specification of DNM/NODNM.

SIM/NOSIM2
Specifies whether (SIM) or not (NOSIM2) MFS is to simulate attributes. SIM, the default, indicates that MFS is to simulate the attributes specified by the IMS application program and place the simulated attributes in corresponding DFLDs that are defined with ATTR=YES or YES,nn. The first byte of the field is used for the simulated attributes. If the MFLD does not supply 3270 attribute information (by means of the ATTR=YES or YES,nn operand) for the corresponding DFLD specifying ATTR=YES or YES,nn, a blank is sent in the first byte of the field. The application designer of the remote program or ISC subsystem is responsible for interpreting the simulated attribute within the remote program or ISC subsystem.

If NOSIM2 is specified, MFS sends a 2-byte bit string to the remote program or subsystem. This bit string is sent exactly as received from the IMS application program. 3270 extended bytes, if any (ATTR=YES,nn), are always sent as received from the application program and follow the 2-byte string of 3270 attributes. If the MFLD does not supply attribute information, binary zeros are sent in the two bytes preceding the data for the field.

See ATTR= on the DFLD statement for additional information.

DNM (DPM-An only)
Can be used with the FORMS= keyword on the DEV statement to specify a literal in the message header. This parameter is optional. See 447 for more information on the FORMS= keyword.

DNM/NODNM (DPM-Bn only)
If DNM is specified or defaulted to, MFS includes the following in the DD header:

- The FMT name, if OPTIONS=MSG
- The DPAGE name, if OPTIONS=DPAGE
- The PPAGE name, if OPTIONS=PPAGE

If NODNM is specified, no data structure name (DSN) is supplied in the DD header.

HDRCTL=
Specifies, for DEV TYPE=DPM-An and DIV TYPE=OUTPUT only, the characteristics of the output message header.

FIXED
Specifies that a fully padded output message header is to be sent to the remote program. The structure of the fixed output message header is the same for all DPM output messages built using this FMT definition. See "Application Programming Using MFS" in *IMS Version 9: Application Programming: Transaction Manager* for an example of the content in the output message header. The base DPM output message header has a length of 7, and includes the version ID.

VARIABLE
Specifies that MIDNAME and DATANAME will have trailing blanks omitted and their length fields adjusted accordingly. If MIDNAME is not used, neither the MIDNAME field nor its length is present.

nn Specifies the minimum length of the header, that is, the base header without MFS fields, as shown in an example in "Application Programming Using MFS" in *IMS Version 9: Application Programming: Transaction Manager*. The default is 7, which is the length of the base message header for DPM. Specifying other than 7 might cause erroneous results in the remote program.

The parameters referenced as RDPN=, DPN=, PRN=, and RPRN= refer to both the ISC ATTACH function management header and the equivalent ISC SCHEDULER function management header.

RDPN=
For DIV TYPE=INPUT, the dfldname specification permits the suggested return destination process name (RDPN) to be supplied in the input message MFLD referencing this dfldname. If dfldname is not specified, no RDPN is supplied in the input message.

DPN=
For DIV TYPE=OUTPUT, the 'literal' specification requests MFS to use this literal as the DPN in the output ATTACH message header. The literal cannot exceed 8 characters. If the dfldname is also specified, the data supplied in the MFLD referencing this dfldname is used as the DPN in the output ATTACH message header. If no output message MFLD reference to the dfldname exists, the 'literal' is used. If the data in the MFLD referencing the dfldname is greater than 8 characters, the first 8 characters are used.

PRN=

For DIV TYPE=INPUT, the dfldname specification permits the suggested primary resource name (PRN) to be supplied in the input message MFLD referencing this dfldname. If the dfldname is not specified, no PRN is supplied in the input message to the application program.

For DIV TYPE=OUTPUT, the 'literal' specification requests MFS to use this literal as the PRN in the output ATTACH message header. The literal cannot exceed 8 characters. If the dfldname is also specified, the data supplied in the MFLD referencing this dfldname is used as the PRN in the output ATTACH message header. If no output message MFLD reference to the dfldname exists, the 'literal' is used. If the data in the MFLD referencing the dfldname is greater than 8 characters, the first 8 characters are used.

RPRN=

For DIV TYPE=INPUT, the dfldname specification permits the suggested return primary resource name (RPRN) to be supplied in the input message MFLD referencing this dfldname. If dfldname is not specified, no RPRN is supplied in the input message to the application program.

For DIV TYPE=OUTPUT, the 'literal' specification requests MFS to use this literal as the suggested return primary resource name (RPRN) in the output ATTACH message header. The literal cannot exceed 8 characters. If the dfldname is also specified, the data supplied in the MFLD referencing this dfldname is used as the RPRN in the output ATTACH message header. If no output message MFLD reference to the dfldname exists, the 'literal' is used. If the data in the MFLD referencing the dfldname is greater than 8 characters, the first 8 characters are used.

OFTAB=

Directs MFS to insert output field tab separator characters in the output data stream for the message. If OPTIONS=DNM and OFTAB, then the OFTAB character is placed in the DD header and an indicator is set to MIX or ALL. If OPTIONS=NODNM, then no DD header is sent.

X'*hh***'**

Character whose hexadecimal representation is "hh" is used as the output field tab separator character. Specification of X'3F' or X'40' is invalid.

C"c"

Character "c" is used as the output field tab separator character. Specification of C""b is invalid.

Restriction: The character specified cannot be present in the data stream from the IMS application program. If it is present, it is changed to a blank (X'40').

If an output field tab separator character is defined, either MIX or ALL can also be specified. Default value is MIX.

MIX

Specifies that the output field tab separator character is to be inserted into each individual field with no data or with less data than the defined DFLD length.

ALL

Specifies that the output field tab separator character is to be inserted into all fields, regardless of data length.

COMPR=

Requests MFS to remove trailing blanks from short fields, fixed-length fields, or all fields presented by the application program.

For DPM-AN devices, trailing blanks are removed at the end of a segment if all of the following conditions are true:

1. FILL=NULL or FILL=PT is specified.
2. GRAPHIC=YES is specified for the current segment being mapped.
3. OPT=1 or OPT=2 is specified in the MSG segment.

If conditions 1, 2, and 3 are met, replacement of trailing blanks occurs as follows:

FIXED
Specifies that trailing blanks from fixed-length fields are to be replaced by nulls.

SHORT
Specifies that trailing blanks fields shortened by the application program are to be replaced by nulls.

ALL
Specifies that trailing blanks from all fields are to replaced by nulls.

The trailing nulls are then compressed at the end of the record. See the description of the FILL= operand for additional information.

For DPM-BN devices, trailing blanks are removed if all of the following conditions are true:

1. OFTAB is specified on the current DIV statement, or FILL=NULL or FILL=PT is specified.
2. GRAPHIC=YES is specified for the current segment being mapped.
3. OPT=1 or OPT=2 is specified in the MSG segment.

If conditions 1, 2, and 3 are met, the removal of trailing blanks occurs as follows:

FIXED
Specifies that trailing blanks are to be removed from fixed-length fields.

SHORT
Specifies that trailing blanks are to be removed from fields shortened by the application program.

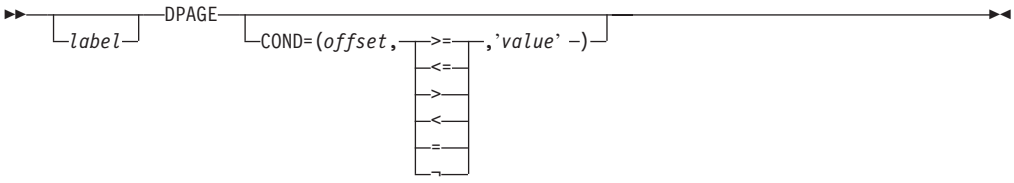
ALL
Specifies that trailing blanks are to be removed from all fields.

Related Reading: For additional information about blank compression for DPN-BN devices, see “Application Programming Using MFS” in *IMS Version 9: Application Programming: Transaction Manager*.

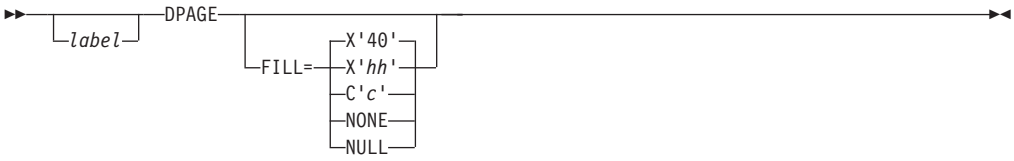
DPAGE Statement

The DPAGE statement defines a logical page of a device format. This statement can be omitted if none of the message descriptors referring to this device format (FMT) contains LPAGE statements and if no specific device option is required.

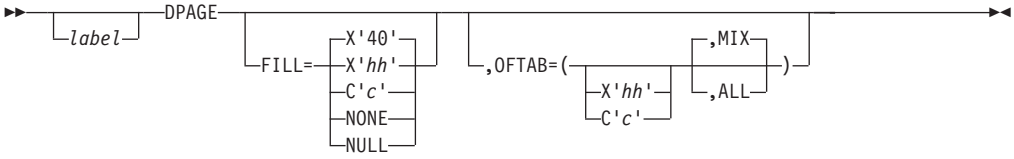
Format for DEV TYPE=274X, DPM-An, or DPM-Bn AND DIV TYPE=INPUT:



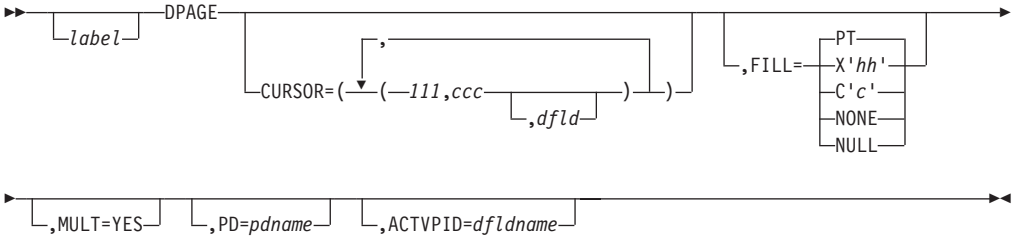
Format for DEV TYPE=274X or DPM-An AND DIV TYPE=OUTPUT:



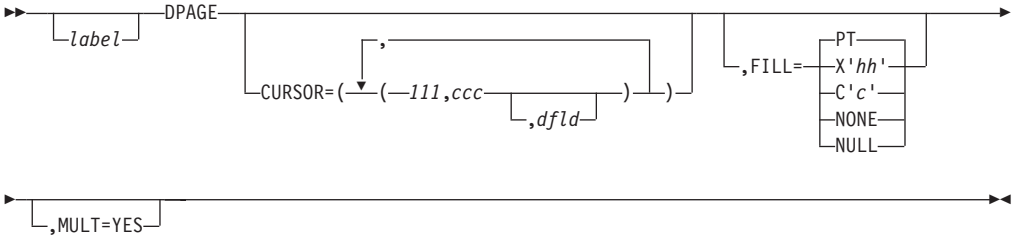
Format for DEV TYPE=DPM-Bn AND DIV TYPE=OUTPUT:



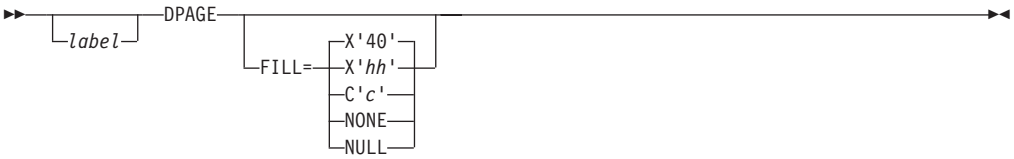
Format for DEV TYPE=3270-An:



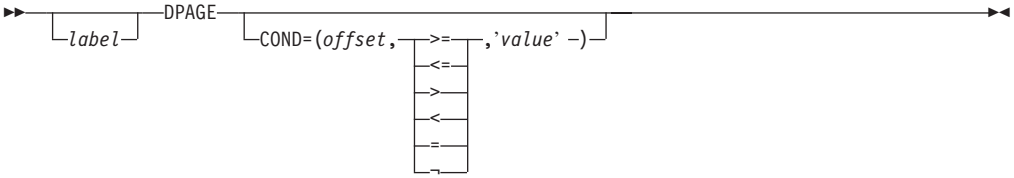
Format for DEV TYPE=3270:



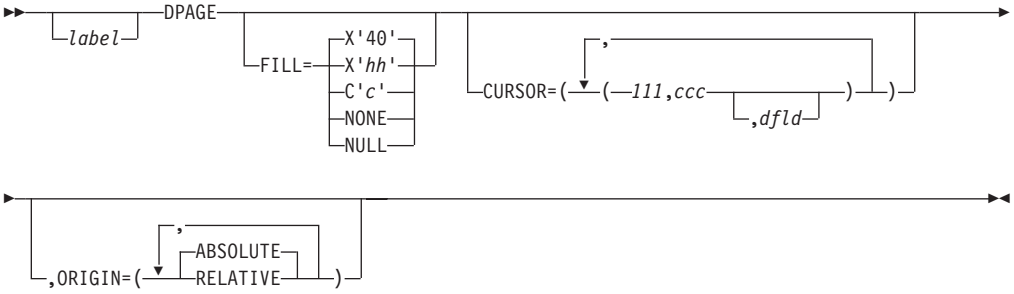
Format for DEV TYPE=3270P:



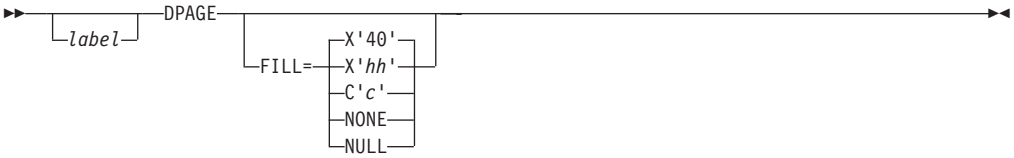
Format for DEV TYPE=FIN:



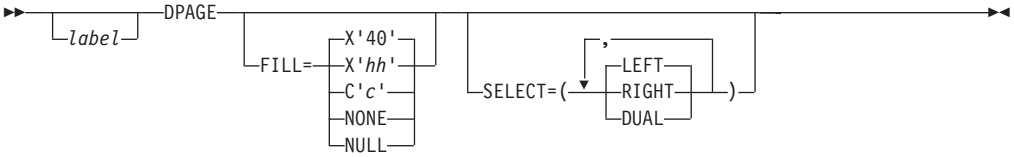
Format for DEV TYPE=FIDS, FIDS3, FIDS4, or FIDS7:



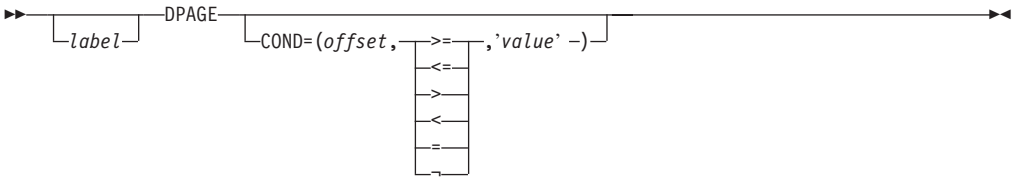
Format for DEV TYPE=FIJP or FIPB:



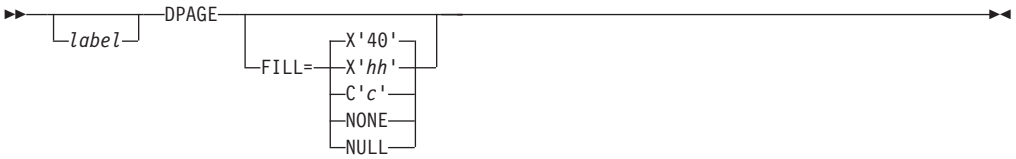
Format for DEV TYPE=FIFP:



Format for DEV TYPE=SCS1 or SCS2 AND DIV TYPE=INPUT:



Format for DEV TYPE=SCS1 or SCS2 AND DIV TYPE=OUTPUT:



Parameters:

label

A 1- to 8-byte alphanumeric name can be specified for this device format that contains LPAGE SOR= references, or if only one DPAGE statement is defined for the device. If multiple DEV statements are defined in the same FMT definition, each must contain DPAGE statements with the same label.

For device type DPM-An and DIV statement OPTIONS=DPAGE, this name is sent to the remote program as the data name in the output message header. If the label is omitted, MFS generates a diagnostic name and sends it to the remote program in the header. If the DPAGE statement is omitted, the label on the FMT statement is sent in the output message header. If OPTIONS=DNM, the label on the FMT statement is sent as the DSN in the DD header.

COND=

Specifies a conditional test to be performed on the first input record. The offset specified is relative to zero. The specification of the offset must allow for the LLZZ field of the input record (for example, the first data byte is at offset 4). If the condition is satisfied, the DFLDs defined following this DPAGE will be used to format the input. When no conditions are satisfied, the last defined DPAGE will be used only if the last defined DPAGE does not specify COND=. If the COND= parameter is specified for the last DPAGE defined and the last defined DPAGE condition is not satisfied, the input message will be rejected. Multiple LPAGE definitions are allowed in message input definitions.

If this keyword is specified, and OPTIONS=NODNM is specified on the DIV statement, this specification is used for DPAGE selection. If this keyword is specified and OPTIONS=DNM is specified on the DIV statement, the COND= specification is ignored and the data structure name from the DD header is used for DPAGE selection.

Lowercase data entered from 274X, Finance, SCS1, or SCS2 keyboards is not translated to uppercase when the COND= comparison is made. Therefore, the literal operand must also be in lowercase.

FILL=

Specifies a fill character for output device fields. Default value for all device types except the 3270 display is X'40'; default for the 3270 display is PT. For 3270 output when EGCS fields are present, only FILL=PT or FILL=NULL should be specified. A FILL=PT erases an output field (either a 1- or 2-byte field) only when data is sent to the field, and thus does not erase the DFLD if the

application program message omits the MFLD. For DPM-Bn, if OFTAB is specified, FILL= is ignored and FILL=NULL is assumed.

NONE
Must be specified if the fill character from the message output descriptor is to be used to fill the device fields.

X'*hh*'
Character whose hexadecimal representation is 'hh' will be used to fill the device fields.

C'*c*'
Character 'c' will be used to fill the device fields.

NULL
Specifies that fields are not to be filled. For devices other than the 3270 display, 'compacted lines' are produced when message data does not fill the device fields.

For DPM-An devices, trailing nulls (X'3F') are removed from all records transmitted to the remote program or subsystem. Trailing nulls are removed up to the first non-null character. Null characters between non-null characters are transmitted. If the entire record is null, but more data records follow, a record containing a single null is transmitted to the remote program. If the entire record is null and more records follow, if OPTIONS=MSG or DPAGE, or in a PPAGE, if OPTIONS=PPAGE, then all null records are deleted to the end of that DPAGE or PPAGE.

PT
Is identical to NULL except for the 3270 display. For the 3270 display, specifies that output fields that do not fill the device field (DFLD) are followed by a program tab character to erase data previously in the field; otherwise, this operation is identical to FILL=NULL.

For 3270 display devices, any specification with a value less than X'3F' is changed to X'00' for control characters or to X'40' for other nongraphic characters. For all other devices, any FILL=X'hh' or FILL=C'c' specification with a value less than X'3F' is ignored and defaulted to X'3F' (which is equivalent to a specification of FILL=NULL).

MULT=YES
Specifies that multiple physical page input messages will be allowed for this DPAGE.

CURSOR=
Specifies the position of the cursor on a physical page. Multiple cursor positions might be required if a logical page or message consists of multiple physical pages. The value lll specifies line number, ccc specifies column; both lll and ccc must be greater than or equal to 1. The cursor position must either be on a defined field or defaulted. The default lll,ccc value for 3270 displays is 1,2. For Finance display components, if no cursor position is specified, MFS will not position the cursor—the cursor is normally placed at the end of the output data on the device. For Finance display components, all cursor positioning is absolute, regardless of the ORIGIN= parameter specified.

The **dfld** parameter provides a method for supplying the application program with cursor information on input and allowing the application program to specify cursor position on output.

Recommendation: Use the cursor attribute facility (specify ATTR=YES in the MFLD statement) for output cursor positioning.

The **dfld** parameter specifies the name of a field containing the cursor position. This name can be referenced by an MFLD statement and must *not* be used as the label of a DFLD statement in this DEV definition. The format of this field is two binary halfwords containing line and column number, respectively. When this field is referred to by a message input descriptor, it will contain the cursor position at message entry. If referred to by a message output descriptor, the application program places the desired cursor position into this field as two binary halfwords containing line and column, respectively. Binary zeros in the named field cause the specified lll,ccc to be used for cursor positioning during output. During input, binary zeros in this field indicate that the cursor position is not defined. The input MFLD referring to this dfld should be defined within a segment with GRAPHIC=NO specified or should use EXIT=(0,2) to convert the binary numbers to decimal.

ORIGIN=
Specifies page positioning on the Finance display for each physical page defined. Default value is ABSOLUTE.

ABSOLUTE
Erases the previous screen and positions the page at line 1 column 1. The line and column specified in the DFLD statement will become the actual line and column of the data on the screen.

RELATIVE
Positions the page starting on column 1 of the line following the line where the cursor is positioned at time of output. Results might be undesirable unless all output to the device is planned in a consistent manner.

OFTAB=
Directs MFS to insert the output field tab separator character specified on this DPAGE statement for the output data stream of the DPAGE being described.

X'*hh***'**
Character whose hexadecimal representation is 'hh' is used as the output field tab separator character. Specification of X'3F' or X'40' is invalid.

C'*c***'**
Character 'c' is used as the output field tab separator character. Specification of C' ' is invalid.

Restriction: The character specified cannot be present in data streams from the IMS application program. If it is present, it is changed to a blank (X'40').

If the output field tab separator character is defined, either MIX or ALL can also be specified. Default value is MIX.

MIX
Specifies that an output field tab separator character is to be inserted into each individual field with no data or with data less than the defined DFLD length.

ALL
Specifies that an output field tab separator character is to be inserted into all fields, regardless of data length.

SELECT=
Specifies carriage selection for a FIFP device with FEAT=DUAL specified in the previous DEV statement. It is your responsibility to ensure that proper forms are mounted and that left margins are set properly. Default value is LEFT.

LEFT
Causes the corresponding physical page defined in this DPAGE to be directed to the left platen.

RIGHT
Causes the corresponding physical page defined in this DPAGE to be directed to the right platen.

DUAL
Causes the corresponding physical page defined in this DPAGE to be directed to both the left and right platens.

PD=
(for the 3180 and 3290 in partition formatted mode) Specifies the name of the partition descriptor of the partition associated with the DPAGE statement. This is the parameter that maps a logical page of a message to or from the appropriate partition. The name of the PD must be contained within the PDB statement specified in the DEV statement.

ACTVPID=
(for the 3290 in partition formatted mode) Specifies the name of an output field in the message containing the partition identification number (PID) of the partition to be activated. This dfldname must be referenced by an MFLD statement and must not be used as the label of a DFLD statement in the DEV definition. The application program places the PID of the partition to be activated in this field. The PID must be in the format of a two byte binary number ranging from X'0000' to X'000F'.

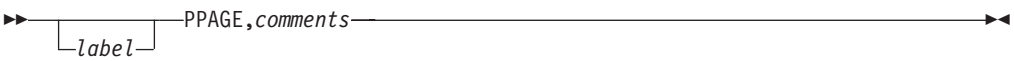
Do not specify this operand for the 3180. Because only one partition is allowed for this device, you need not specify an active partition.

PPAGE Statement

The PPAGE statement, valid only for device types of DPM-An or DPM-Bn, defines the beginning of a presentation page. A presentation page is the unit of data delivered to the remote program in response to a paging request when OPTIONS=PPAGE has been specified in the DIV statement for this definition. For DPM-Bn MODE=RECORD only, if OPTIONS=MSG or DPAGE has been specified, paging is as described for those options under the DIV statement, and the PPAGE statement then defines the beginning of a new record (that is, it is equivalent to a RCD statement).

For an input DPAGE, only one PPAGE statement is allowed, and it must be placed between the DPAGE statement and the first DFLD statement. For an output DPAGE, if two consecutive PPAGE statements appear in the DPAGE for a message defined with OPTIONS=PPAGE, only an output message header with the PPAGE label as its data name is sent to the remote program, except OPTIONS=(PPAGE,DNM) for DPM-Bn. For DPM-Bn, a PPAGE statement without a DFLD statement is not allowed when OPTIONS=(PPAGE, NODNM) is specified for DIV TYPE=OUTPUT. A warning message is issued, and the PPAGE statement is ignored. For OPTIONS=MSG or DPAGE, consecutive PPAGE statements are ignored.

Format:



Parameters:

label
A one- to eight-character alphanumeric name should be specified. For OPTIONS=PPAGE, this label is sent as the data name for DPM-An or as the data structure name for DPM-Bn in the message output header or DD header to identify the data structure of this presentation page to the remote program. If no label is specified, MFS generates a diagnostic label that is sent to the remote program in the header.

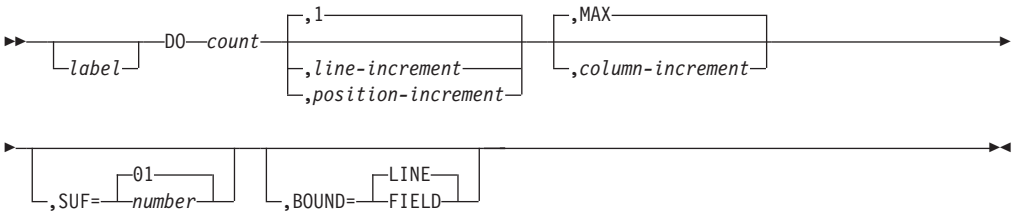
Recommendation: Specify a user-defined label because the MFS-generated name can change whenever the MFS definitions are recompiled.

The label specified should be unique, at least within a given FMT definition, and preferably within an IMS system if the remote program uses this label to identify the appropriate DSECT for formatting the data included in this presentation page.

DO Statement

The DO statement causes repetitive generation of DFLD and RCD statements between the DO and ENDDO statements. When DO is used, there are restrictions in the naming of DFLDs (refer to “DFLD Statement” on page 468).

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

count
Specifies how many times to generate the statements.

line-increment
Specifies how much to increase the line position after the first cycle. The first cycle uses the *lll* value specified in the POS= keyword of the DFLD statement. The default is 1. This parameter is not specified for DEV type DPM-An or DPM-Bn.

position-increment
Specifies how much to increase the position parameter after the first cycle. The first cycle uses the *nnn* value specified in the POS= operand of the DFLD statement. The position increment is used for an input device format when MODE=STREAM is specified. This parameter is not specified for DEV type DPM-An or DPM-Bn.

MAX
Specifies that the line increment to be used at the end of each cycle and the column values in the DFLDs are to remain the same for each cycle. This parameter is not used if MODE=STREAM is specified for the device format or if DEV type is DPM-An or DPM-Bn; if present, it is ignored.

column-increment
Specifies how much to increase the column position after the first cycle. The first cycle uses the *ccc* value specified in the POS= keyword of the DFLD statement. The default is MAX. This parameter is not used for DEV type DPM-An or DPM-Bn, or when MODE=STREAM is specified for the device format, because it is ignored.

SUF=
Specifies the 2-digit suffix to be appended to the *dfldname* of the first group of generated DFLD statements. The default is 01. MFS increments the suffix by one on each subsequent generation of statements.

If the specified suffix exceeds 2 digits, MFS uses the rightmost 2 digits.

If the specified count is such that the generated suffix eventually exceeds 2 digits, MFS reduces the count to the largest legitimate maximum value. For example, if count equals 8 and SUF=95, invalid suffixes of 100, 101, and 102 would result. In this instance, MFS reduces the count to 5, processes the statement, and issues an error message.

BOUND=
Specifies when updates to line position and column position are to occur. The default is LINE. This parameter is not used if MODE=STREAM is specified for the device format or if DEV type is DPM-An or DPM-Bn; if present, it is ignored.

LINE
Specifies that all fields be inspected **before** the repetition is performed. If the column increment would cause any field in the group of DFLD statements to not fit on a line, the column position value for all fields is reset to the initial value, and the line position values are increased by the line-increment value.

FIELD
Specifies that each time the statement is repeated, the column position value is increased by the column-increment value. If MAX is specified, or the new column position value reaches device line length capacity, the line position value is increased by the line-increment value and the column position value is reset to its initial value.

Printing Generated DFLD Statements: The generated DFLD statements can be printed in a symbolic source format by specifying COMP in the parameter list of the EXEC statement. This provides a means of seeing the results of the DFLD statement generation without having to interpret the intermediate text blocks.

The following items are printed for each generated DFLD statement:

- The generated statement sequence number followed by a plus sign (+) to indicate that the DFLD statement was generated as a result of DO statement processing.
- The DFLD statement label, if present, including the appended suffix.
- The statement operator, DFLD.
- For EGCS literals, the G, SO, and SI are not present. Literals are truncated if there is insufficient room to print all specifications. Truncation is indicated by a portion of the literal with three periods (...), representing the truncated portion.
- ATTR=(YES,*nn*), if present.
- ATTR=YES, if present.
- ATTR=*nn*, if present.
- ATTR=(...), if attributes are present.

- EATTR=(...), if present.
- The RECORD or STREAM form of the POS= keyword, with the line and column or stream position updated by the respective increments. This is not printed if DEV type is DPM-An or DPM-Bn.
- SCA, if present.
- The field length, in the form of LTH=*nnnn*.

No other operands are printed, even if specified on the source DFLD statement.

For device type DPM-An or DPM-Bn, the RCD statement can appear between a DO and ENDDO statement. If it does, a new record boundary is created for each repetitive generation of the DFLD field following the RCD statement. For example, the following sequence causes the DFLDs A01, B01, and C01 to be in record 1, while A02, B02, and C02 are in record 2, and A03, B03, and C03 are in record 3.

```
    DO 3
    RCD
A   DFLD    LTH=10
B   DFLD    LTH=10
C   DFLD    LTH=10
    ENDDO
```

Alternatively, the RCD statement can immediately precede the DO statement. If it does, a new record boundary begins with the first DFLD after the DO statement and does not end until the ENDDO statement (or the maximum record length) is reached. For example, the following sequence causes the DFLD D01 to begin a new record, in which E01, D02, and E02 also occur.

```
    RCD
    DO 2
D   DFLD    LTH=10
E   DFLD    LTH=10
    ENDDO
```

RCD Statement

The RCD statement, valid for DEV TYPE=DPM-An or DPM-Bn only, can be used to influence the placement of DFLDs in records. The RCD statement precedes a DFLD statement and initiates a new transmission record for delivery to a remote program. DFLDs following the RCD statement are included into the transmission record until the next RCD statement or the maximum record length is reached (or, if NOSPAN is specified, until a field will not be fully contained in the current record).

The RCD statement can be placed after the PPAGE, DO, DFLD, or ENDDO statements. (The effects of placing RCD before and after a DO statement are discussed in “DO Statement” on page 465.) If a RCD statement is immediately followed by another, only the first one is effective.

The RCD statement is invalid for STREAM mode.

Format:

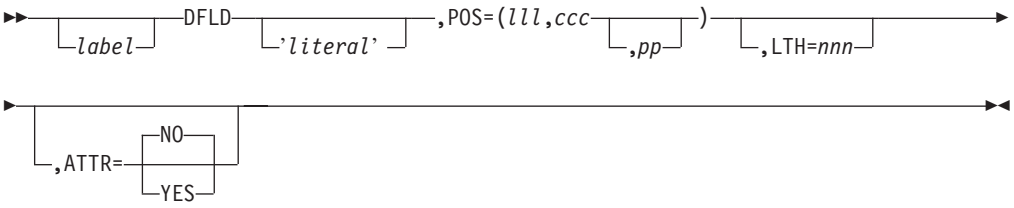
Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

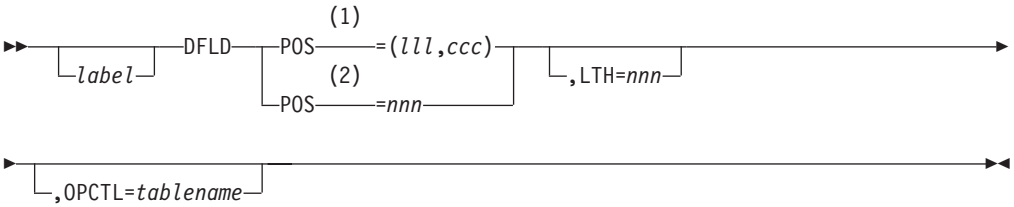
DFLD Statement

The DFLD statement defines a field within a device format which is read from or written to a terminal or remote program. Only those areas which are of interest to the IMS or remote application program should be defined. Null space in the format does not need to be defined.

Format for DEV TYPE=274X AND DIV TYPE=OUTPUT:



Format for DEV TYPE=274X AND DIV TYPE=INPUT:

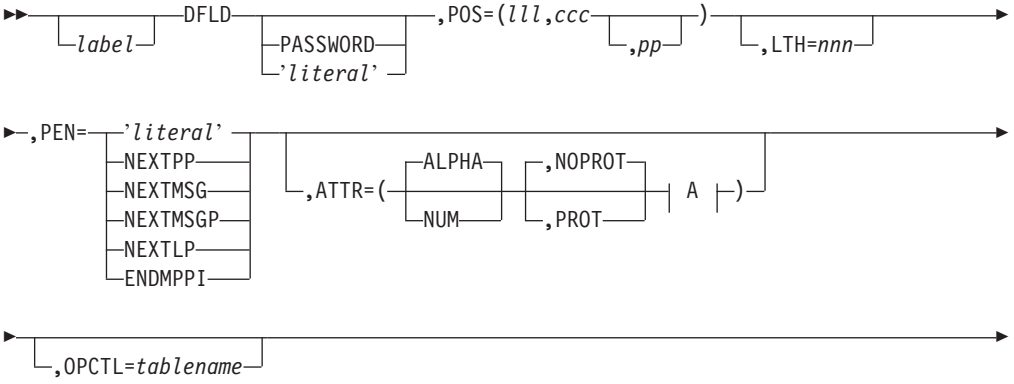


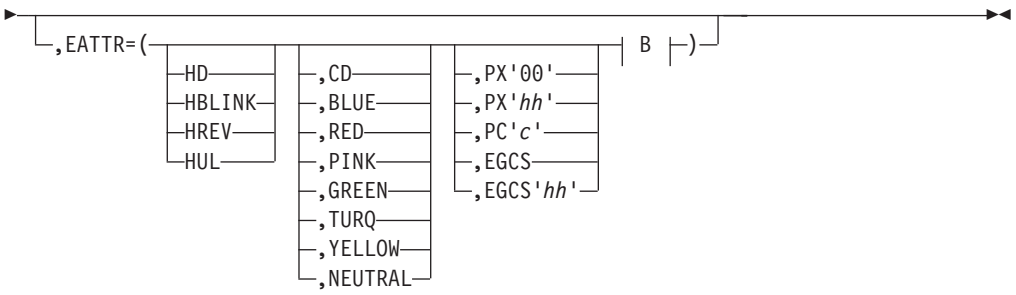
Notes:

1 Used for MODE=RECORD only.

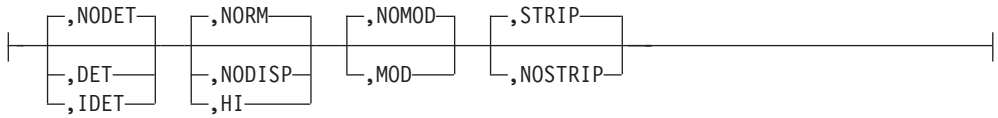
2 Used for MODE=STREAM only.

Format for DEV TYPE=3270 or 3270-An:

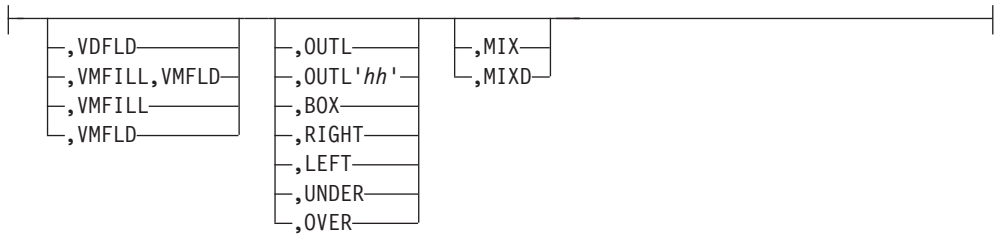




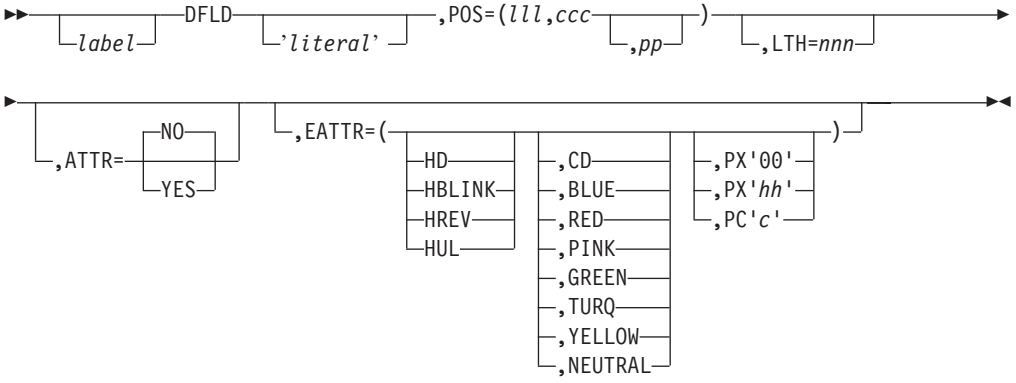
A:



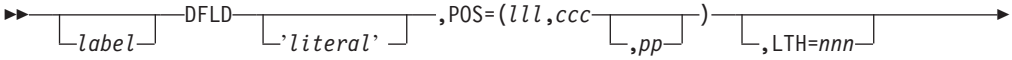
B:



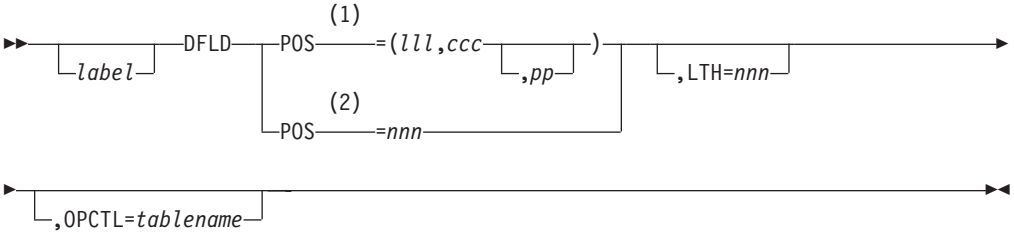
Format for DEV TYPE=3270P:



Format for DEV TYPE=FIDS, FIDS3, FIDS4, FIDS7, FIFP, FIJP, and FIPB:



Format for DEV TYPE=FIN:

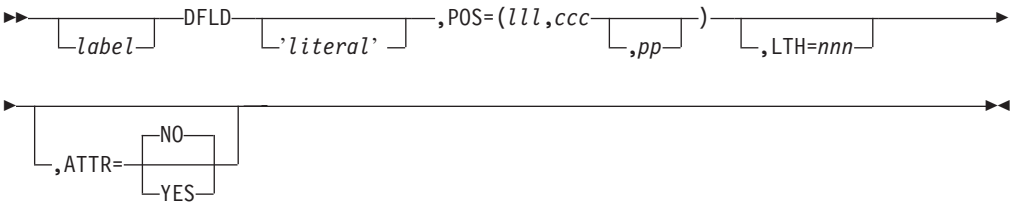


Notes:

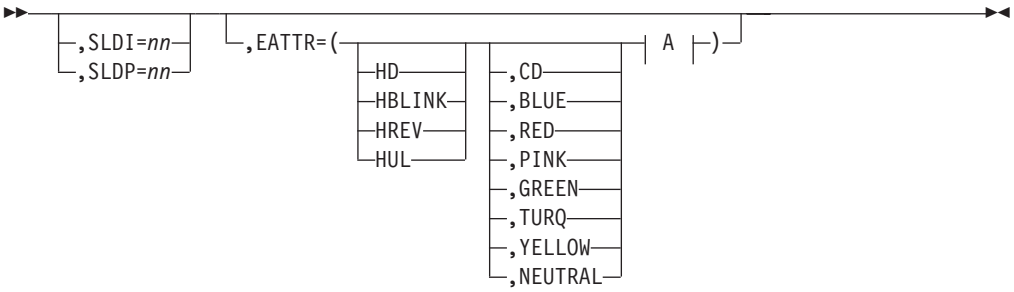
1 MODE=RECORD only

2 MODE=STREAM only

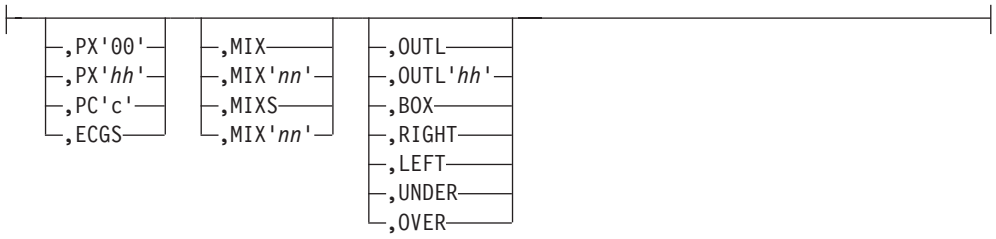
Format for DEV TYPE=SCS1 OR SCS2 AND DIV TYPE=OUTPUT:



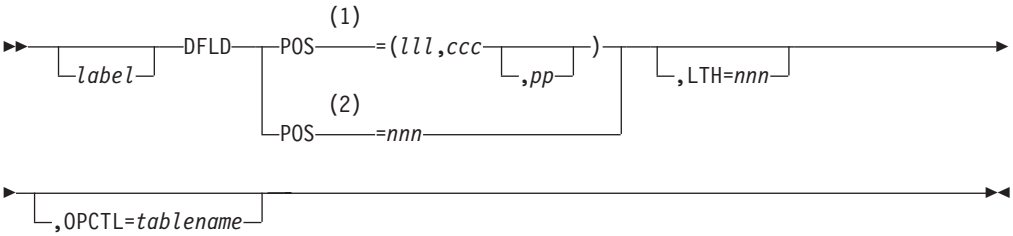
Format for SCS1 ONLY:



A:



Format for DEV TYPE=SCS1 or SCS2 AND DIV TYPE=INPUT:

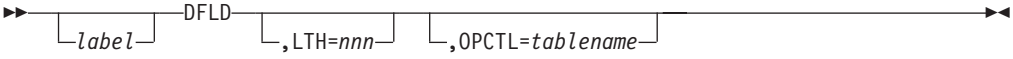


Notes:

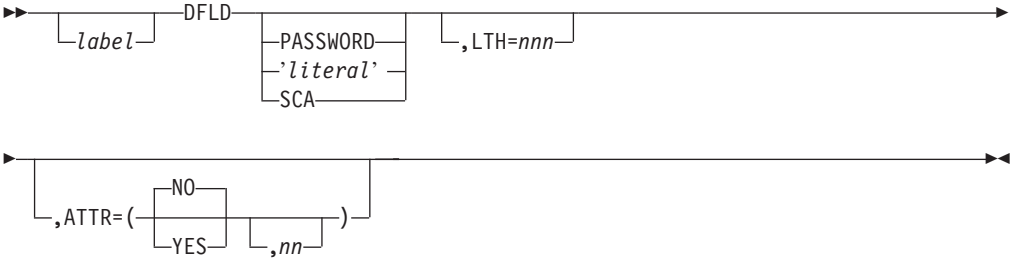
1 MODE=RECORD only

2 MODE=STREAM only

Format for DEV TYPE=DPM-An or DPM-Bn AND DIV TYPE=INPUT:



Format for DEV TYPE=DPM-An or DPM-Bn AND DIV TYPE=OUTPUT:



Parameters:

label

A one- to eight-character alphanumeric name can be specified. This label (dfldname) can be referred to by a message descriptor in transferring data to and from a terminal or remote program. If the repetitive generation function of MFS is used (DO and ENDDO statements), this dfldname should be restricted to 6 characters maximum length. When each repetition of the statement is generated, a 2-digit sequence number (01 to 99) is appended to the label. If the label specified here is greater than 6 characters and repetitive generation is

used, the label is truncated at 6 characters, and a 2-digit sequence number is appended to form the 8-character name. No error message is provided if this occurs.

If PASSWORD, SCA, or '*literal*' is specified, label is not valid, and specification of a label will result in an error message. If a DPN, PRN, RDPN, or RPRN dfldname is specified on the DIV statement, the *dfldname* cannot be used as a DFLD label for the current DIV statement.

PASSWORD
Identifies this field as the location of the IMS password field for input messages.

Recommendation: Use the PASSWORD capability in the input message definition. If you specify PASSWORD you cannot refer to the field described by this DFLD statement with a message descriptor. Additionally, if you specify PASSWORD you must omit *label*.

'literal'
Specifies a literal character string to be presented to the device. The length of *literal* cannot exceed 256 bytes for 3270 display devices, 40 bytes for FIDS and FIDS3, 64 bytes for FIDS4, 80 bytes for FID57, 132 bytes for 274X, 256 bytes for 3270P, and line width for all printer and punch devices. For DPM, the length of *literal* cannot exceed the value specified in the RCDCTL operand.

For 3270 displays, literal fields have the PROT attribute whether specified or not; the NUM attribute is assumed if ALPHA is not specified.

Restriction: If you specify *literal* you cannot refer to the field described by this DFLD statement with a message descriptor. Additionally, if you specify *literal* you must omit *label*.

SCA
Specifies, for DPM definitions only, that SCA information, when sent by the IMS application program or specified in the DSCA, is to be sent in this DFLD.

If SCA is specified, *label* must not be specified.

POS=
Defines the first data position of this field in terms of line (lll), column (ccc), and physical page (pp) of the display format. If pp is omitted, 1 is assumed.

For DEV TYPE=274X, FIN,FIDS,FIDS3,FIDS4, FIDS7,FIJP,FIPB,FIFP,SCS1, or SCS2

lll,ccc
Specifies the record number and position within the record of this field. This form is required if MODE=RECORD. lll and ccc must be greater than or equal to 1.

nnn
Specifies the starting position of this field in STREAM mode input. If not specified, this field starts immediately following the preceding field, or at the left margin if this is the first field. If MODE=STREAM has been specified, and POS= is specified, this form is required. *nnn* must be greater than or equal to 1.

lll,ccc,pp
Specifies the line, column, and optionally, the physical page number for an output field. lll, ccc, and pp must be greater than or equal to 1.

For DEV TYPE=3270, 3270-An, or 3270P

lll,*ccc*,*pp*
Specifies the line, column, and optionally, the physical page number for an output field. *lll*, *ccc*, and *pp* must be greater than or equal to 1.

For 3270 displays, POS=(1,1) must not be specified. Fields must not be defined such that they wrap from the bottom to the top.

Restriction: On some models of 3270s, the display screen cannot be copied when a field starting on line 1, column 2, has both alphabetic and protect attributes.

For DEV TYPE=DPM-An or DPM-Bn

For DPM devices
The POS= keyword is ignored.

LTH=
Specifies the length of the field. This operand should be omitted if *'literal'* is specified in the positional parameter, in which case the length of *literal* is used as the field length. Unpredictable output formatting can occur if this operand is used in conjunction with a *'literal'* and the two lengths are different. The specified LTH= cannot exceed the physical page size of the device.

The maximum allowable length for all devices except 3270, 3604 display, and DPM with RCDCT=NOSPAN is 8000 characters. For 3270 displays, the maximum length is one less than screen size. For example, for a 480-character display, the maximum length is 479 characters. For a FIDS display component, the maximum length is 240 characters; for a FIDS3, the maximum length is 480 characters; for a FIDS4, the maximum length is 1024 characters; for a FIDS7, the maximum length is 1920. A length of 0 must not be specified. For DPM, if RCDCT=NOSPAN is specified, the length must be less than or equal to the RCDCTL value, if RCDCTL is less than 8000. If SCA and LTH= are both specified, LTH must be 2.

POS= and LTH= do not include the attribute character position reserved for a 3270 display device or a DFLD with ATTR=YES specified. The inclusion of this byte in the design of display/printer formats is necessary because it occupies the screen/printed page position preceding each displayed/printed field even though it is not accessible by an application program.

When defining DFLDs for 3270 printers, a hardware ATTRIBUTE character is not used. Therefore, fields must be defined with a juxtaposition that does not allow for the attribute character unless ATTR=YES is specified. However, for printers defined as 3270P the last column of a print line (based on FEAT=, WIDTH=, or the device default width) cannot be used. The last column of the line is reserved for carriage control operations performed by IMS. Thus, if the print line specifies 120 (FEAT=120) and the DFLD specifies POS=(1,1),LTH=120 then 119 characters are printed on line 1 and one character on line 2.

For DPM definitions, if OPTIONS=NOSIM2 is specified on the DIV statement, and ATTR=YES or YES,*nn* is specified, 2 bytes plus the extended attributes are added to the length of the DFLD. The first two bytes are reserved for the binary 3270 attribute, (protect, numeric, and so forth.) If OPTIONS=SIM is specified, 1 byte or 1 byte plus the extended attributes is added to the length of the DFLD with ATTR=YES or YES,nn. The first byte of the field is thus reserved for the simulated attribute.

Detectable fields (DET or IDET) must include four positions in POS and LTH for a 1-byte detection designator character and 3 pad characters, unless the detectable field is the last field on a display line, in which case only one position for the detection designator character is required. The detection designator character must precede field data, and pad characters (if required) follow field data. Detection designator and required pad characters must be supplied by the application program or MFLD literal with the field data. Pad characters can also be required in the preceding field on the device.

ATTR=

Defines the display attributes of this field for each of the listed DEV TYPE, DIV TYPE combinations:

- For DEV TYPE=3270 or 3270-An

  Attribute keywords can be specified in any order and only those desired need be specified. The underlined keywords do not have to be specified, because they are defaults.

  When two user-defined fields are separated by two or more characters, MFS generates an undefined field to represent that space in the display buffer. The display attributes for an undefined field are NUM, PROT, and NODISP.

  ALPHA|NUM

  specifies whether the field should have the numeric attribute. The numeric attribute specifies that the Numeric Lock feature (automatic upshift of data entry keyboard) will be used by the 3275/3277 or 3276/3278. If NUM and PROT are specified for the field, the auto-skip feature is used. That is, upon entry of a character into the last character location of an unprotected field, the cursor automatically skips the field with the NUM and PROT attribute specifications and is positioned to the first character location of the next unprotected field. If an undefined field, as described in the ATTR= parameter, follows the filled unprotected field, the auto-skip feature is used. This parameter, in conjunction with the PROT parameter, is used to lock the COPY function. See ″PROT″ for details.

  NOPROT|PROT

  Specifies whether the field is protected from modification by you. For literal fields, PROT is used and specification of NOPROT is ignored.

  The IMS copy function on remote 3270 terminals can be locked by setting the attribute value of protect and alpha for an attribute byte in line 1 and column 1 of a display. When the copy function is locked, it cannot be used to copy the contents of a display to a printer. For more information, see “IMS Support of Devices” in *IMS Version 9: Operations Guide*. The “Local Copy Function” available on the 3274 and 3276 control units is not locked by the attribute setting. The “Local Copy Function” is invoked by the print key.

  NODET|DET|IDET

  Specifies the detectability of the field through light pen operations. DET specifies a deferred detectable field, while IDET indicates an immediately detectable field. You must provide appropriate designator and pad characters as discussed under the LTH= operand. Note that the 3270 display devices place restrictions on the number of detectable or mixed detectable and nondetectable fields that can precede that last detectable field on a given line.

NORM|NODISP|HI
Specifies the field's display intensity as normal (NORM), high intensity (HI), or nondisplayable (NODISP). If NODISP is specified, DET or IDET cannot be specified.

When defining a high-intensity (HI) field, including a detection designator character as the first data byte causes the high-intensity (HI) field to be detectable.

NOMOD|MOD
defines whether or not the field-modified-attribute byte should be assumed for this field. MOD causes the terminal to assume the field has been modified by you even though it was not (that is, the modified data tag (MDT) is set in the field-modified-attribute byte). This should not be confused with the PROT attribute which prevents modification by you. MOD is ignored for literal fields.

When MOD is specified, each time MFS sends output for this physical page, the modified attribute is set (unless overridden by dynamic attribute modification).

Related Reading: For a description of when IMS resets modified data tags, see *IMS Version 9: Operations Guide*.

STRIP|NOSTRIP
Specifies whether the pen detect designator byte preceding the input field should be stripped (STRIP) before presentation to the application program. If an EGCS attribute is defined for a light-pen-detectable field, you should specify ATTR=NOSTRIP on the DFLD statement and design the application program to bypass or remove the two designator characters from the input data. If ATTR=STRIP is specified or defaulted, MFS will only remove the first designator character and the last character in the field could be lost (truncated).

- For DIV TYPE=OUTPUT and DEV TYPE=274X, 3270P, FIDS, FIDS3, FIDS4, FIDS7, FIFP, FIJP, FIPB, FIS1, or SCS2

  Attribute keywords specify whether (YES) or not (NO) the first byte of this field will be used to display attribute information when the output message includes attribute information for the field. The default is NO. If ATTR=YES is specified, the LTH= and POS= keywords do not have to allow for the simulated attribute byte because the MFS preprocessor adjusts the keyword values internally. The action taken when ATTR=YES is specified is:

CURSOR	(FIDS, FIDS3, FIDS4, and FIDS7 ABSOLUTE output only). The cursor will be positioned to the first position of this field.
NODISP	No data sent regardless of other attributes
HI	An asterisk (*) is placed in the first byte
MODIFIED	An underscore character (_) is placed in the first byte
HI and MODIFIED	An exclamation point (!) is placed in the first byte

  If attribute information is not provided from the output message, the first byte is a blank.
- For DIV TYPE=OUTPUT, DEV TYPE=DPM-An, and DEV TYPE=DPM-Bn, 3270P, FIDS, FIDS3, FIDS4, FIDS7, FIFP, FIJP, FIPB, FIS1, or SCS2

Attribute keywords specify whether (YES) or not (NO) the first one or two bytes of this field carries existing 3270 attributes and whether extended attributes (nn) are present. The keywords can be used in various combinations as follows:

YES
Specifies that the first one or two bytes of this field are used to convey the existing 3270 attributes (in simulated or binary form depending upon the specification of SIM or NOSIM2 respectively on the DIV statement) from the IMS application program to the remote program. (SIM causes MFS to simulate an attribute. NOSIM2 causes MFS to pass the bits exactly as entered.)

Thus, if ATTR=YES is specified and OPTIONS=SIM or OPTIONS= is not specified, one byte is added to the length of the DFLD. If OPTIONS=NOSIM2, two bytes are added to the length of the DFLD. These bytes are reserved as the attribute bytes to be transmitted to the remote program.

NO
Specifies that the first one or two bytes of this field will **not** be used to convey the existing 3270 attributes (in simulated or binary form respectively) from the IMS application program to the remote program. This is the default.

nn Is the number of extended attributes that can be dynamically modified, and is a number from 1 to 4. An invalid specification is defaulted to 1. Two additional bytes are added to the length of the DFLD for each attribute specified (2 x *nn*). The additional bytes, which just precede the data, either can (YES) or must not (NO) follow the bytes reserved for the existing 3270 attribute bytes. These bytes are used to convey the extended attributes (in binary form) from the IMS application program to the remote program. The attributes are always transmitted as presented from the IMS application program. They are never simulated or validated.

When used in combination, YES,*nn* specifies that both attributes and extended attributes are to be transmitted. In this case, and depending upon the specification of SIM and NOSIM2 as described:

YES,*nn*
When specified with SIM, specifies that 3270 simulated attributes (1 byte) plus extended attributes (2 x *nn* bytes) of this field are to be transmitted from the IMS application program to the remote program. The total number of bytes used to convey all of these attributes to the remote program is 1 + (2 x *nn*).

When specified with NOSIM2, specifies that 3270 attributes in binary form (2 bytes) plus extended attributes (2 x *nn* bytes) of this field are to be transmitted from the IMS application program to the remote program. The total number of bytes used to convey all of these attributes, which are all in binary form, to the remote program is 2 + (2 x *nn*).

When used in combination, NO,*nn* specifies that only extended attributes are transmitted. Thus, the number of bytes transmitted, in binary form, is (2 x *nn*) only.

Valid specifications and the number of bytes which must be reserved are:

```
For  DIV ,OPTION=NOSIM2  then:
DFLD    ,ATTR=(YES,nn)     2 + (2 × nn)
DFLD    ,ATTR=(NO,nn)      2 × nn
DFLD    ,ATTR=(,nn)        2 × nn
DFLD    ,ATTR=YES          2
DFLD    ,ATTR=NO           0
For  DIV ,OPTION=SIM or not specified  then:
DFLD    ,ATTR=(YES,nn)     1 + (2 × nn)
DFLD    ,ATTR=(NO,nn)      2 × nn
DFLD    ,ATTR=YES          1
DFLD    ,ATTR=NO           0
```

EATTR=

Is valid for output DFLDs only and defines the extended attributes of this field for DEV TYPE=3270, 3270-An, 3270P, or SCS1.

Not all extended attributes apply to all device types. To ensure that your specifications for your device types are correct, refer to the component description manual for your device.

The operands specify:

- Additional field highlighting
- Field color
- Field outlining
- Input control
- Validation to be performed
- Local ID of the programmed symbol buffer

Characters are selected from the programmed symbol buffer and placed in the field. These operands can be specified in any order. When the device default value is selected for an operand, it is used to hold a place in the data stream to permit application program modification of the attribute so specified.

For details on modifying these attributes, see *IMS Version 9: Application Programming: Transaction Manager*.

To specify the additional highlighting for the field use the following:

HD	device default
HBLINK	blink
HREV	reverse video
HUL	underline

To specify the field's color use the following:

- **BLUE**
- **RED**
- **PINK**
- **GREEN**
- **TURQ(uoise)**
- **YELLOW**
- **CD**
- **NEUTRAL**

The last two operands are used as follows:

CD Used to specify the default.

NEUTRAL Used to specify device-dependent. The particular color displayed for NEUTRAL is device-dependent. In general, NEUTRAL is white on displays and black on printers with single-plane programmed symbols and as multicolored on displays or printers with tri-plane programmed symbols.

The following five operands—PX'00', PX'*hh*', PC'*c*', EGCS, and EGCS'*hh*'—are mutually exclusive. That is, a field can be specified as having one of these characteristics, but not a combination thereof. For all 3270 devices, MFS does not verify that any specified character set has been properly loaded. The programmed symbol buffers can be loaded by an IMS application program using the MFS bypass.

PX'00'|PX'*hh***'|PC'***c***'**
Specifies a value that must correspond to the local ID specified for a programmed symbol buffer already loaded or to the EGCS programmed symbol buffer.

PX'00'
Is the same as no specification, except that it allows an application program to specify a programmed symbol buffer for the field through dynamic modification of the programmed symbol attribute.

PX'*hh***'**
Is a hexadecimal character in the range X'40' through X'FE'.

PC'*c***'**
Is a hexadecimal character within the range X'40' through X'FE'.

EGCS|EGCS'*hh***'**
Is valid only on output DFLDs for the 3270 display. SCS1 device types can specify EGCS only and not EGCS '*hh*'.

When an extended graphic character set literal is specified on a DFLD statement, the extended graphic character set attribute is forced—that is, you do not have to code EATTR=EGCS'*hh*' for 3270 displays or EATTR=EGCS for SCS1 device types. For 3270 displays, a programmed symbol value of X'F8' is set.

Restriction: The IMS application program cannot modify the SCS1 DFLD extended graphic character set attribute.

When defining an EGCS field for a 3283 Model 52, the length must be an even number. If the EGCS field spans device lines, WIDTH= and POS= should be specified so that an even number of print positions are reserved on each of the device lines.

EGCS
Specifies the field attribute for the field as Extended Graphic Character Set. Also specifies the field attribute for the field as Double Byte Character Set.

EGCS'*hh***'**
'*hh*' is the programmed symbol value that is used. The value for '*hh*' can be any hexadecimal value from X'40' through X'FE' or X'00'. If '*hh*' is omitted from the extended graphic character set specification for a 3270 display, a programmed symbol value of X'F8' is assumed. '*hh*' is ignored if specified for an SCS1 device.

To define an EBCDIC field that can be dynamically modified by the IMS application program to accept extended graphic character set data, the programmed symbol attribute should be specified as EGCS'00'.

VDFLD|VMFILL|VMFLD|VMFILL,VMFLD
Defines the type of validation for the field as follows:

VDFLD	default
VMFILL	mandatory fill
VMFLD	mandatory field
VMFILL,VMFLD	a combination of mandatory fill and mandatory field

If a field is defined as protected (ATTR=PROT) or if it is a literal with validation attributes specified, then the validation attribute specifications are reset and a message is issued.

The following are used to specify field outlining:

OUTL'*hh*'	Field outlining with field outlining value '*hh*'
OUTL	Device default
BOX	Box
RIGHT, LEFT, UNDER, OVER	Lines that can be specified individually or in combination

Field outlining value '*hh*' is a two-digit hexadecimal number between X'00' and X'0F'. If any other value is specified, the device default, X'00', is assumed. Table 30 shows the values for the field outlining patterns:

Table 30. Field Outlining Values

Value	UNDER	RIGHT	OVER	LEFT
00				
01	X			
02		X		
03	X	X		
04			X	
05	X		X	
06		X	X	
07	X	X	X	
08				X
09	X			X
0A		X		X
0B	X	X		X
0C			X	X
0D	X		X	X
0E		X	X	X
0F	X	X	X	X

Field outlining for 3270 displays and SCS1 printers can be dynamically modified by code in an application program. The position of left, right, over, and underlines differ according to the device.

The following is a brief description of field outlining for the IBM 5550 family (as 3270) of devices.

3270 display Left and right lines are printed in the position of the 3270 basic attribute byte. The overline of the current line and the underline of the preceding line are the same line.

The underline for the 24th line is the same line as the line separating the application program area and your message area.

SCS1 printer Left and right lines are printed in the byte reserved by MFS before and after the current field. The overline of the current line and the underline of the preceding line are the same line. When an underline is specified in the last line of the page, an underline is drawn in the last line of the page, and an overline is drawn on the first line of the next page.

If one byte space exists between two adjacent fields, the right line of the first field is the same line as the left line of the second field.

MIX|MIXD|MIX'*nn***'|MIXS|MIXS'***nn***'**
Specify a DBCS/EBCDIC mixed field.

3270 display

MIX DBCS/EBCDIC mixed field

MIXD device default

Input control for the 3270 display can be dynamically modified by the application program. Refer to *IMS Version 9: Application Programming: Transaction Manager* for more information on dynamic modification.

SCS1 printer

MIX DBCS/EBCDIC mixed field with SO/SI blank print option.

MIXS DBCS/EBCDIC mixed field with SO/SI blank print suppress option.

MIX'*nn*' '*nn*' is the maximum number of SO/SI pairs. DBCS/EBCDIC mixed field with SO/SI blank print option.

MIXS'*nn*' '*nn*' is the maximum number of SO/SI pairs. DBCS/EBCDIC mixed field with SO/SI blank print suppress option.

The '*nn*' is buffer information used by MFS message editor and must be a two-digit decimal number between 01 and 31. If MIX or MIXS is specified, the MFS default is calculated as follows:

MIX DFLD length divided by 5 plus 1, or 31, whichever is smaller.

MIXS DFLD length divided by 3 plus 1, or 31, whichever is smaller.

When a field spans continuation lines, the number '*nn*' obtained from the field length with either of the methods plus 1, is assigned to each line.

With the SCS1 printer, when DBCS/EBCDIC mixed data spanning across continuation lines is split at a DBCS character, MFS replaces the last character with a blank and places that character at the beginning of the next line. As a result, one print position is lost.

PEN=

Specifies a literal to be selected or an operator control function to be performed when this field is detected. If (1) *'literal'* is specified, (2) the field is defined as immediately detectable (ATTR= operand), and (3) contains the null or space designator character, the specified literal is placed in the field referred to by the PEN operand of the preceding DEV statement when the field is detected (if no other device fields are modified). If another field on the device is modified, a question mark (?) is provided instead of the literal. Literal length must not exceed 256 bytes.

If (1) a control function is specified, (2) the field is defined as immediately detectable (ATTR= operand), and (3) contains the null or space designator character, the specified control function is performed when the field is detected and no other device fields are modified. If another field on the device is modified, a question mark (?) is provided and the function is not performed. Control functions that can be specified are:

NEXTPP—PAGE ADVANCE

Specifies a request for the next physical page in the current output message. If no output message is in progress, no explicit response is made.

NEXTMSG—MESSAGE ADVANCE

specifies a request to dequeue the output message in progress (if any) and to send the next output message in the queue (if any).

NEXTMSGP—MESSAGE ADVANCE PROTECT

Specifies a request to dequeue the output message in progress (if any), and send the next output message or return an information message indicating that no next message exists.

NEXTLP—NEXT LOGICAL PAGE

Specifies a request for the next logical page of the current message.

ENDMPPI—END MULTIPLE PAGE INPUT

Specifies the end of a multiple physical page input message.

ENDMPPI is valid only if data has been received and will not terminate multiple page input (MPPI) in the absence of data entry.

OPCTL=

Specifies the name of a table, defined by a TABLE statement, that is to be checked for operator control requests when this device field is received. OPCTL processing occurs when the input device data is processed. If a control function is selected, in most cases the control function is performed immediately; no IMS input message is created.

SLDI=

For SCS1 printers, specifies the line density for an output message in lines per inch. (See also SLDP=.) SLDI= can also be specified on the DEV statement. SLDI= is validated for a value from 1 through 72. The value specified must be consistent with the architecture of the device for which this value is specified (see the appropriate device or component manual).

If SLDI= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDI= specification is encountered, but after any vertical tabs and new line characters. The SLDI= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

SLDP=
For SCS1 printers, specifies the line density for an output message in points per inch. (See also SLDI=.) SLDP= can also be specified on the DEV statement. SLDP= is validated for a value from 1 through 72. The value specified must be consistent with the architecture of the device for which this value is specified (see the appropriate device or component manual).

If SLDP= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDP= specification is encountered, but after any vertical tabs and new line characters. The SLDP= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

Recommendation: Be careful, when defining set line density (SLDx) keywords, to ensure that forms alignment is maintained. If SLDx= is improperly defined, the forms might not align properly. Also, note that SLDI= and SLDP= are mutually exclusive. Neither SLDI= nor SLDP= can occur on a DFLD statement between a DO and an ENDDO statement.

ENDDO Statement

The ENDDO statement terminates the group of DFLD statements that are to be repetitively generated. The generated DFLD statements are printed immediately following the ENDDO statement. An ENDDO statement is required for each DO statement entered in this definition.

Format:

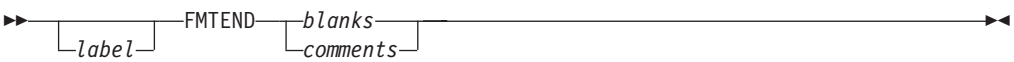
Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

FMTEND Statement

The FMTEND statement terminates a device format definition and is required as the last statement in the device format definition. If this is the end of the input to SYSIN processing, the FMTEND statement must be followed by an END compilation statement.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Partition Set Definition Statements

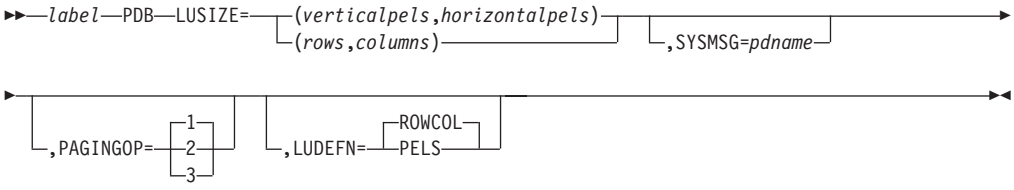
Partition set definition statements include the PDB statement, the PD statement, and the PDBEND statement. Details of each are provided in the information that follows.

PDB Statement

This statement initiates and defines a partition set (a Partition Descriptor Block) for 3290 and 3180 devices in partitioned format mode. The PDB statement contains several parameters that describe certain characteristics of the entire partition set. Its name is referenced by the PDB keyword of a DEV statement if a partition set is required to format logical pages of a message.

At least one PD statement must be specified within each PDB. Note, however, that for a 3180 in partitioned format mode, **only one** PD statement should be specified within each PDB. This is because only one partition can be specified for the 3180. There are additional differences in specifications that can be made for the partitioned 3180 and 3290 which are described in the following section.

Format:



Parameters:

label
A one- to eight-character alphanumeric name (pdbname) for the PDB must be specified.

LUSIZE=
Describes the physical size of the Logical Unit display for which the PDB is defined. If LUDEFN=PELS, the size is specified in picture elements (pels). If LUDEFN=ROWCOL, the size is specified in rows and columns (this is the default value). For the 3180, LUSIZE must be specified in terms of rows and columns.

SYSMSG=
Specifies the partition name (*pdname*) for displaying system messages. The system message partition should have only one field defined. This DFLD should be defined as at least LTH=79 so the system message is not truncated.

If the current PDB defines a system message partition, then all system messages are directed to this partition. If a system message partition is not defined, but a SYSMSG field is defined in the current DOF, the system message is directed to the system message field of the active partition. Finally, if the current PDB does not define a partition for system messages and the DOF does not define a field for that purpose, a system message destroys the current partitioned format mode and the 3290 returns to standard format mode.

PAGINGOP=
Specifies the option number (1, 2, or 3) for the partition page presentation algorithm. These three algorithms specify different ways of presenting the initial pages of the message to the partitions of the partition set. They also specify what paging actions result when you enter paging requests from the 3290 device.

The default of 1 must be accepted or specified on this operand for 3180 formats.

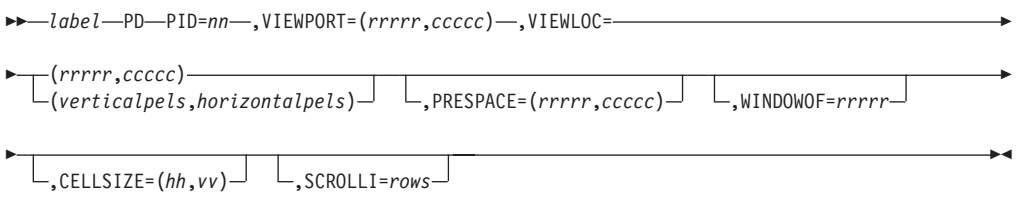
LUDEFN=
Indicates whether the LUSIZE parameter in the PDB statement and the VIEWLOC parameter in the PD statements are specified in rows and columns or in pels. LUDEFN is optional if all the PD statements use the same cell size and the default (ROWCOL) is acceptable. Note that ROWCOL must be specified or accepted as the default for 3180 formats.

If two or more PD statements within the same PDB specify different cell sizes, PELS **must** be chosen.

PD Statement

The Partition Definition statement defines one partition and its presentation space. Every partition set described by a PDB statement must contain at least one PD statement. Note, however, that for a 3180 in partitioned format mode, **only one** PD statement should be specified within each PDB.

Format:



Parameters:

label
A one- to eight-character alphanumeric name (pdname) must be specified. This name is referenced by the DPAGE statement to associate a logical page with its appropriate partition.

PID=
Specifies a partition identifier number for the partition. Values 00 through 15 are valid for 3290 formats. Each partition must have a unique PID. A value of 00 must be specified for 3180 formats, because only one partition need be identified.

VIEWPORT=
Specifies the size of the viewport for the partition. *rrrrr* indicates rows and *ccccc* indicates columns. For the 3180 device, the following restrictions apply:

- If the number of columns is greater than or equal to 80, then the number of rows must be less than or equal to 43.
- If the number of columns is greater than 80 and less than or equal to 132, then the number of rows must be less than or equal to 27.

VIEWLOC=
Specifies the location of the viewport on the display screen, in terms of the

distance offset from the top left of the screen. When the LUDEFN parameter of the PDB statement is ROWCOL, the distance is expressed in rows and columns. *rrrrr* indicates rows and *ccccc* indicates columns. When the LUDEFN parameter is PELS, the distance is expressed in the number of pels from the top of the screen and the number of pels from the left of the screen. When defining formats for the 3180, VIEWLOC must be expressed in rows and columns.

PRESPACE=
Indicates the size of the presentation space buffer in rows and columns. *rrrrr* indicates rows and *ccccc* indicates columns. If this parameter is not specified, the default is the size of the viewport specified on the VIEWPORT parameter. When this parameter is specified, the columns parameter is optional and defaults to the columns specification on the VIEWPORT parameter. If columns are specified, they must be the same as the columns specified in the VIEWPORT parameter.

When specifying this operand for 3180 formats, the product of the number of rows times the number of columns might not be greater than 7680.

WINDOWOF=
Indicates the initial offset in rows of the top edge of the view window from the top of the presentation space. The window maps the portion of the presentation space to be displayed onto the viewport on the screen. During interactive processing, change the offset by scrolling. The default value of WINDOWOF is zero.

CELLSIZE=
Indicates the number of horizontal and vertical pels in a character cell. Note that this specification is in an unusual order for MFS. That is, the **width** of the character cell is specified first, **then the height**. This is the reverse of the usual MFS order.

For the 3290, the default is 6 X 12 PEL (for a small character). Valid values for the 3290 are 6 X 12 to 12 X 31, or the value 00 X 00. If the value is 00 X 00, the 3290 device will select a cell size for optimum readability. This prevents MFS from making validity checks on the viewport locations and possible overlaps. Therefore, be careful to choose viewport size and location specifications accurately.

For the 3180, this operand should be specified according to usable screen area size as follows:

- CELLSIZE=(12,12)
  - 24 x 80
  - 32 x 80
  - 43 x 80
- CELLSIZE=(10,16)
  - 27 x 132

SCROLLI=
Indicates the number of rows that are scrolled when the scrolling function is used. The default scrolling increment is one row. If the scrolling increment is larger than the viewport size, part of the presentation space is not viewable on the screen. Specifying 0 as the scrolling increment disables the scrolling function.

PDBEND Statement

The PDBEND statement terminates a partition set definition (a partition descriptor block) and is required as the last statement of the definition. If this is the end of the input to SYSIN processing, the PDBEND statement must be followed by an END compilation statement.

Format:

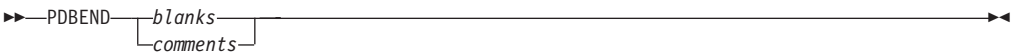


Table Definition Statements

Table definition statements include the TABLE statement, the IF statement, and the TABLEEND statement. Details of each are provided in the information that follows.

TABLE Statement

The TABLE statement initiates and names an operator control table that can be referred to by the OPCTL keyword of the DFLD statement (“DFLD Statement” on page 468). For a discussion of the use of the table, see “Message Formatting Functions” in *IMS Version 9: Application Programming: Transaction Manager*. The TABLE statement, and the IF and TABLEEND statements that follow, must be outside of a MSG or FMT definition.

Format:

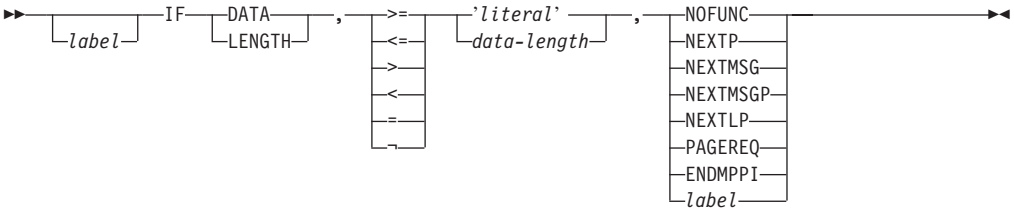
Parameters:

tablename
A 1- to 8-byte alphanumeric name for the table must be specified.

IF Statement

The IF statement defines an entry in the table named by the previous TABLE statement. Each IF statement defines a conditional operation and an associated control or branching function to be performed if the condition is true.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. This label is required if a previous IF statement contained a branch function.

DATA
Specifies that the conditional operation is to be performed against the data received from the device for the field.

LENGTH
Specifies that the conditional operation is testing the number of characters entered for the field. The size limit for this field is the same as for DFLDs (see“Message Formatting Functions” in *IMS Version 9: Application Programming: Transaction Manager*).

=, <, >, ¬, ≤, ≥
Specify the conditional relationship that must be true to invoke the specified control function.

'literal'
Is a literal string to which input data is to be compared. The compare is done before the input is translated to upper case. If '*literal*' is specified, DATA must be specified in the first operand. If the input data length is not equal to the *literal* string length, the compare is performed with the smaller length, unless the conditional relationship is ¬ and the data length is zero, in which case the control function is performed. If the input is in lowercase, the ALPHA statement should be used and the literal coded in lowercase.

data-length
Specifies an integer value to which the number of characters of input data for the field is compared.

NOFUNC
Specifies that conditional function testing is to be terminated.

NEXTPP—PAGE ADVANCE
Specifies a request for the next physical page in the current output message. If no output message is in progress, no explicit response is made.

NEXTMSG—MESSAGE ADVANCE
Specifies a request to dequeue the output message in progress (if any) and to send the next output message in the queue (if any).

NEXTMSGP—MESSAGE ADVANCE PROTECT
Specifies a request to dequeue the output message in progress (if any), and either send the next output message or return an information message indicating that no next message exists.

NEXTLP—NEXT LOGICAL PAGE
Specifies a request for the next logical page of the current message.

PAGEREQ—LOGICAL PAGE REQUEST
Specifies that the second through last characters of input data are to be considered as a logical page request.

ENDMPPI—END MULTIPLE PAGE INPUT
Specifies the end of multiple physical page input (this input is the last for the message being created).

label
Specifies that testing is to continue with the IF statement bearing the label (branch). The label must be placed on an IF statement that follows the current statement in the TABLE definition (that is, it must be a forward branch function).

TABLEEND Statement

The TABLEEND statement establishes the end of a table definition. If this is the end of the input to SYSIN processing, the TABLEEND statement must be followed by an END compilation statement.

Format:

Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Compilation Statements

Compilation statements include the ALPHA statement, the COPY statement, the EQU statement, the RESCAN statement, the STACK statement, the UNSTACK statement, the TITLE statement, the PRINT statement, the SPACE statement, the EJECT statement, and the END statement. Details of each are provided in the information that follows.

ALPHA Statement

The ALPHA statement specifies a set of characters to be considered alphabetic by the MFS language utility for the purpose of defining valid field names and literals.

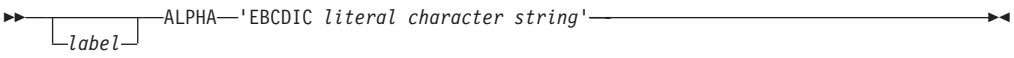
Restriction: The following characters cannot be made alphabetic using ALPHA.

```
b ¢ * < ( +  !! * ) ; ¬ .
- / , % _ > ? :
' = ”
0 through 9
```

The characters A through Z, &; (X'50'), #, $, and @ are always considered alphabetic by the MFS language utility.

All the characters referred to are known as standard characters. Therefore, all other characters are referred to as nonstandard characters.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

'*literal character string*'
Specifies the characters to be considered alphabetic by the MFS language utility. The use of an EGCS literal in an ALPHA statement causes an ERROR message.

COPY Statement

The COPY statement invokes a copy of a member of the partitioned data set represented by the SYSLIB DD statement. The copied member can request the nested copy of another member. The member to be copied cannot already exist at a higher level in a nested chain of copy requests. The nesting level available for copy is limited only by the amount of storage available to the language utility preprocessor. The level of the COPY statement is indicated to the right of each printed COPY record.

Format:

Parameters:

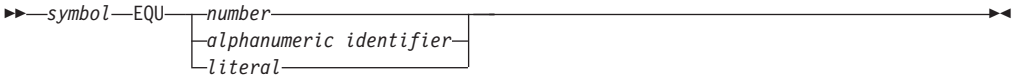
label
A one- to eight-character alphanumeric name can be specified. It is not used.

member-name
Specifies the name of the partitioned data set member to be copied into the input stream of the utility preprocessor.

EQU Statement

The EQU statement defines a symbol as a substitution variable. All subsequent occurrences of the symbol in the operand field of a statement is replaced by the value specified in the operand field of the EQU statement.

Format:



Parameters:

symbol
Specifies the symbol to be equated to the value specified in the operand field. The symbol must be a one- to eight-character alphanumeric identifier, the first character of which must be alphabetic.

number
Specifies the value to be represented by the symbol, and consists of 1 to 256 decimal digits.

alphanumeric identifier
Specifies the value to be represented by the symbol, and consists of 1 to 256 alphanumeric characters, the first of which must be alphabetic.

literal
Specifies the value to be represented by the symbol, and consists of 1 to 256 valid characters (not counting embedded second quotes), enclosed in quotes. The characters within the leading and trailing quotes replace the symbol when substitution occurs. An EGCS literal cannot be equated if any hexadecimal value within the literal is a X'7D' (a single quote character).

A symbol used in an equate (EQU) statement can be re-equated to another value.

There are no reserved words that cannot be used as symbols on the EQU statement. However, when defining symbols do not use a symbol as one of the words used by the MFS statement operands. Otherwise, the intended function of the MFS word cannot be used.

Example: Consider the following equate statement:

```
NOPROT EQU PROT
```

Then if one DFLD specifies ATTR=NOPROT and another DFLD specifies ATTR=PROT, both DFLDs would generate the protect attribute (PROT).

Restriction: Once an MFS word is equated, it cannot be restored to its original symbol. In other words, a symbol cannot be equated to itself.

Concatenated EQU Statements

A period (.) can be used to concatenate two equated values or one value and specific data, providing that at the point of concatenation a delimiter exists.

Example: Consider the following EQU statements:

```
A  EQU ATTR

AE EQU 'ATTR='

P  EQU '(PROT,NUM)'

EP EQU '=(PROT,NUM)'
```

The following all produce the same results:

```
ATTR=(PROT,NUM)

ATTR=P

AE.P

A.EP

A=P
```

RESCAN Statement

The RESCAN statement controls the operation of EQU statements during replacement mode.

Format:

Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

OFFON
Specifies whether (ON) or not (OFF) replacement text should be rescanned for further substitution. The default is OFF unless a number is specified.

If ON is specified, replacement text can invoke further substitution within the substituted text up to a maximum number of occurrences.

5number
Specifies how many times further substitution is allowed in a single rescan substitution. The default is 5. If recursive substitutions are attempted beyond the 'number', an error message is issued and substitution terminates. RESCAN ON,0 will be interpreted as RESCAN OFF.

STACK Statement

The STACK statement is used to delineate one or more SYSIN or SYSLIB records, and to request that those records, once processed, be kept (stacked) in processor storage for reuse at a later time. A stack of SYSIN/SYSLIB records must not contain STACK and UNSTACK statements. The letter S to the right of each printed record indicates that it is being stacked for future use.

Format:

Parameters:

label
A 1- to 8-character name can be specified. It is not used.

ON
Specifies the beginning of a stack of SYSIN/SYSLIB records. ON is the default, and it does not have to be specified to begin stacking.

OFF
Specifies the end of a stack of SYSIN/SYSLIB records.

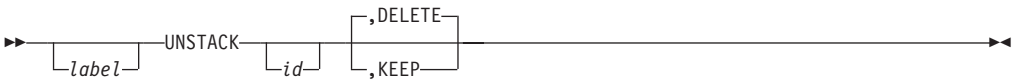
id Specifies the one-to eight-character alphanumeric name for the record stack. If the compilation only uses one stack, no ID is required; MFS assigns an ID of eight blanks to the stack.

When multiple stacking operations are requested, all stacks should be uniquely identified; one unnamed stack is permitted.

UNSTACK Statement

The UNSTACK statement requests retrieval of a previously processed stack of SYSIN/SYSLIB records and specifies whether the retrieved stack should be deleted after processing. The letter U to the right of each printed record indicates that it is being read from the processor storage stack for processing.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

id Specifies the 1- to 8-character identifier of the stack to be retrieved and processed. If no ID is specified, MFS retrieves the stack identified by eight blanks.

DELETE|KEEP
Specifies whether (KEEP) or not (DELETE) the stack should be retained after retrieval and processing. The default is DELETE.

TITLE Statement

The TITLE statement is used to specify the heading to appear on the SYSPRINT listing.

Format:

Parameters:

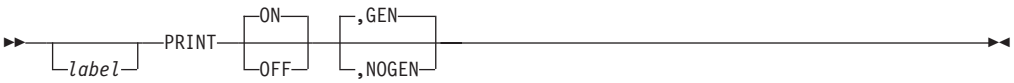
label
A one- to eight-character alphanumeric name can be specified. It is not used.

literal
Specifies the heading to be printed on the output listing. The heading can be specified as an EGCS literal. An EGCS literal of more than 108 bytes causes an error message.

PRINT Statement

The PRINT statement provides printing specifications for the SYSPRINT listing.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

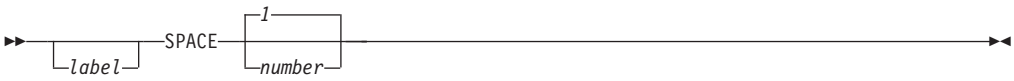
ON|OFF
Specifies whether (ON) or not (OFF) a listing should be printed. The default is ON.

GEN|NOGEN
Specifies whether (GEN) or not (NOGEN) the intermediate text blocks (ITBs) should be printed in hexadecimal following the statement at the left margin. If PRINT GEN is used following the ENDDO statement, all definitions generated for the iterative DO group are printed. The default is GEN.

SPACE Statement

The SPACE statement specifies the number of lines to skip when output is printed. The SPACE statement is printed.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

1*number*
Specifies how many lines to skip after this statement is encountered. The default is 1.

EJECT Statement

The EJECT statement is used to eject a page in an output listing. The EJECT statement is printed.

Format:

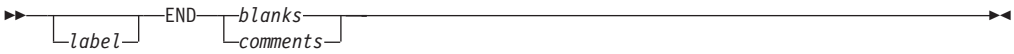
Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

END Statement

The END statement is used to define the end of the input to SYSIN processing. If this statement is omitted, one is provided and an error message is issued.

Format:



Parameters:

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Chapter 36. MFS Device Characteristics Table Utility (DFSUTB00)

The Message Format Service Device Characteristics Table (MFSDCT) utility (DFSUTB00) defines new screen sizes in a descriptor member of the IMS.PROCLIB library without performing an IMS system definition. These new screen size definitions are added to the screen sizes that were previously defined.

The MFSDCT (DFSUTB00) utility procedure consists of the following steps:

1. The DFSUTB00 program is executed to initiate several functions. The DFSUTB00 program:
   - Reads one or two descriptor members from PROCLIB and uses only the new device descriptors as input.
   - Builds DCTENTRY statements for each device descriptor.
   - Optionally loads an existing device characteristics table from JOBLIB/STEPLIB data sets (usually from the IMS.SDFSRESL library) and then builds DCENTRY statements for each DCT entry.
   - Invokes the assembler, passing the DCTENTRY statements and the DCTBLD and MFSINIT macros as input.
   - Readies the output from the assembler as an updated or new device characteristics table and as a new set of default MFS format definitions. (This output is split into separate files for later processing.)
2. The assembler is invoked to assemble the new device characteristics table.
3. The linkage editor is invoked to link-edit the new device characteristics table into the IMS.SDFSRESL.
4. Phase 1 of the MFS Language utility generates new default MFS format control blocks.
5. Phase 2 of the MFS Language utility puts the new default MFS format control blocks into the IMS.FORMAT library.

The following topics provide additional information:

- “Restrictions for DFSUTB00”
- “MFSDCT Procedure for DFSUTB00”

Restrictions for DFSUTB00

The following restrictions apply to this utility:

- The utility ignores all other descriptors while reading the one or two descriptor members from PROCLIB.
- At least one device descriptor must be specified or the utility terminates.

MFSDCT Procedure for DFSUTB00

This section explains the JCL for the MFSDCT procedure, the procedure statement, the EXEC statement, the DD statements, the MFS device descriptions, and error processing.

Figure 111 on page 496 shows the JCL for the procedure used to invoke the MFS Device Characteristics Table utility.

```
//         PROC RGN=4M,SOUT=A,SYS2=,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,COMPR2=COMPRESS,
//            LN=55,SN=8,DEVCHAR=0,COMPR3=NOCOMPREND,
//            DIRUPDT=UPDATE,DCTSUF=,
//            DSCTSUF=,DSCMSUF=,FMTMAST=N
//S1       EXEC PGM=DFSUTB00,REGION=&RGN,
//         PARM=('DCTSUF=&DCTSUF,DSCTSUF=&DSCTSUF',
//         'DSCMSUF=&DSCMSUF,DEVCHAR=&DEVCHAR'
//         'FMTMAST=&FMTMAST')
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SYSLIB   DD DSN=IMS.OPTIONS,DISP=SHR
//         DD DSN=IMS.ADFSMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//         DD DSN=SYS1.SDFSMAC,DISP=SHR
//PROCLIB  DD DSN=IMS.&SYS2.PROCLIB,DISP=SHR
//SYSIN    DD DSN=&&SYSIN,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSPUNCH DD DSN=&&SYSPUNCH,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//DCTIN    DD DSN=&&DCTIN,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//DEFLTS   DD DSN=&&DEFLTS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//DCTLNK   DD DSN=&&DCTLNK,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(TRK,(1,1)),
//         DCB=BLKSIZE=800
//S2       EXEC PGM=ASMA90,REGION=& RGN,
//            PARM='OBJECT,NODECK,NOLIST',
//            COND=(0,LT)
 //SYSLIB   DD DSN=IMS.OPTIONS,DISP=SHR
//         DD DSN=IMS.ADFSMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSLIN   DD DSN=&&DCT,DISP=(NEW,PASS)
//         UNIT=SYSDA,SPACE=(CYL,(1,1)),
//         DCB=BLKSIZE=800<
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         SPACE=(CYL,(15,15))
//SYSIN    DD DSN=&&DCTIN,DISP=(OLD,DELETE)
//S3       EXEC PGM=IEWL,
//            PARM=('SIZE=(880K,64K)',NCAL,LET,REUS,
//            XREF,LIST),
//            REGION=&RGN,
//            COND=(0,LT)
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605),
//         SPACE=(605,(10,10),RLSE,,ROUND)
```

Figure 111. MFSDCT Procedure (Part 1 of 2)

```
//SYSLMOD  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSPUNCH)),
//         SPACE=(CYL,(10,1))
//SYSLIN   DD DSN=&&DCTLNK,DISP=(SHR,DELETE)
//DCT      DD DSN=&&DCT,DISP=(SHR,DELETE)
//S4       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,;
//            &COMPR,'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR'),COND=(0,LT)
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DSN=&&DEFLTS,DISP=(OLD,DELETE)
//REFIN    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//REFRD    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY    DD DISP=SHR,
//         DSN=IMS.&SYS2.PROCLIB(REFCPY)
//UTPRINT  DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS  DD DSN=&&BLKS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//S5       EXEC PGM=DFSUNUB0,REGION=&RGN,
//            PARM=(&COMPR2,&COMPR3,&DIRUPDT,;
//            'DEVCHAR=&DEVCHAR'),COND=((0,LT,S1),
//            (0,LT,S2),(0,LT,S3),(8,LT,S4))
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=IMS.&SYS2.FORMAT,DISP=SHR
//DUMMY    DD DISP=SHR,
//         DSN=IMS.&SYS2.PROCLIB(FMTCPY)
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Figure 111. MFSDCT Procedure (Part 2 of 2)

Procedure Statement

The procedure statement must be in the form shown in Figure 112:

Figure 112. Procedure Statement Format

```
PROC RGN=4M,SOUT=A,SYS2=,PXREF=NOXREF,
PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
COMPR=NOCOMPRESS,COMPR2=COMPRESS,
LN=55,SN=8,DEVCHAR=0,COMPR3=NOCOMPREND,
DIRUPDT=UPDATE,DCTSUF=,
DSCTSUF=,DSCMSUF=
```

In addition to the optional keyword parameters described in "JCL Parameter Descriptions for DFSUPAA0" on page 410, the following parameters can be specified. (*x* is the alphanumeric suffix character that you are appending to the member name.)

DCTSUF=*x*
Specifies the suffix character to be appended to DFSUDT0. The name DFSUDT0*x* identifies the device characteristics table to which new definitions are added. This suffix character corresponds to the value specified in the SUFFIX= keyword of the IMSGEN macro. If a suffix character is not specified, a completely new device characteristics table is built from just the device descriptors.

DSCTSUF=*x*
Specifies the suffix character to be appended to DFSDSCT. The name DFSDSCT*x* identifies a descriptor member. This suffix character corresponds to the value specified in the IMS procedure DSCT= keyword. This parameter is required if DSCMSUF= is not specified.

DSCMSUF=*x*
Specifies the suffix character to be appended to DFSDSCM. The name DFSDSCM*x* identifies a descriptor member. This suffix character corresponds to the value specified in the SUFFIX= keyword of the IMSGEN macro. This parameter is required if DSCTSUF= is not specified.

DEVCHAR=0 | *x*
Specifies the suffix character to be appended to DFSUDT0. The name DFSUDT0*x* identifies the updated or new device characteristics table. Any existing device characteristics table with the same name is replaced in IMS.SDFSRESL by step 3 of this utility. The default is 0.

FMTMAST=Y/N
Specifies whether (Y) or not (N) the IMS-provided support for MFS on the master terminal is to be used.

EXEC Statement

The EXEC statement determines that a device characteristics table is created. It also specifies the name for the desired descriptor member and the name of the updated or new device characteristics table. Each of the five steps in this procedure names a different program for execution.

The format for each step is shown is Figure 113 on page 499.

```
S1      EXEC PGM=DFSUTB00,REGION=&RGN,
        PARM=('DCTSUF=&DCTSUF,DSCTSUF=&DSCTSUF'
        'DSCMSUF=&DSCMSUF,DEVCHAR=&DEVCHAR')
S2      EXEC PGM=ASMA90,REGION=&RGN,
        PARM=('OBJECT,NODECK,NOLIST',
        COND=(0,LT)'
S3      EXEC PGM=IEWL,
        PARM=('SIZE=880K,64K),NCAL,LET,REUS,XREF,LIST',
        REGION=&RGN,
        COND=(0,LT)
S4      EXEC PGM=DFSUPAA0,REGION=&RGN,
        PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,&COMPR,;
        'LINECNT=&LN,STOPRC=&SN,DEVCHAR=&DEVCHAR'),
        COND=(0,LT)
S5      EXEC PGM=DFSUNUB0,REGION=&RGN,
        PARM=(&COMPR2,&COMPR3,&DIRUPDT,;
        'DEVCHAR=&DEVCHAR'),COND=((0,LT,S1),
        (0,LT,S2),(0,LT,S3),(8,LT,S4))
```

Figure 113. Five Steps of the MFSDCT (DFSUTB00) Utility

In addition to the optional keyword parameters described in “JCL Parameter Descriptions for DFSUPAA0” on page 410, you can also specify the parameters described in “Procedure Statement” on page 497.

DD Statements

The following ddnames are used in step 1 of the MFSDCT procedure.

DCT
Defines the temporary data set for the updated or new device characteristics table as output from the assembler with the ddname SYSLIN (step 2) and as input to the Linkage Editor with ddname DCT (step 3).

DCTIN
Defines a temporary data set for the device characteristics table as input to the assembler (step 2).

DCTLNK
Defines the temporary data set for the link-edit control statements for step 3.

DEFLTS
Defines the temporary data set for the default MFS format definitions for MFS Language utility input (step 4).

PROCLIB
Defines the libraries containing the descriptor members DFSDSCM*x* and DFSDSCT*x*.

STEPLIB
Defines the libraries containing the program DFSUTB00 and the device characteristics table specified in the DCTSUF= parameter.

SYSIN
Defines the temporary file containing the generated DCENTRY statements.

SYSLIN
Defines the temporary data set for the updated or new device characteristics table as output from the assembler (step 2) and as input to the Linkage Editor with ddname DCT (step 3).

SYSLIB
Defines the libraries containing IMS and z/OS macros.

SYSPRINT
Defines the data set for all of the printed output from step 1, including error messages and output from steps 2 and 3.

SYSPUNCH
Defines the temporary file containing the object module output from the assembler. The output is the device characteristics table, followed immediately by the default MFS format definitions.

SYSUT1
Defines an assembler and linkage editor work data set.

SYSLMOD
Defines the IMS.SDFSRESL data set to contain the new or modified device characteristics table.

MFS Device Descriptors

MFS device descriptors are used by the MFS Device Characteristics Table utility to update screen size in the DCT and generate new MFS default formats without system definition.

Related Reading: See "Message Formatting Functions" in *IMS Version 9: Administration Guide: Transaction Manager* for more information about how to use ETO descriptors.

MFS Device Descriptor Format

The format for an MFS device descriptor is:

```
D descriptor name parm1 parm2 parm3
```

D Is the descriptor type.

descriptor name
Is ignored.

parm (1...3)
Are the keywords TYPE=, SIZE=, and FEAT=.

Related Reading: See "Macros" in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for information about the values for SIZE, TYPE, and FEAT on the TERMINAL macro.

Error Processing

Return codes are based on the error message.

Related Reading: See *IMS Version 9: Messages and Codes, Volume 1* for explanations of the return code and error message.

Part 8. Service Utilities

Chapter 37. MFS Service Utility (DFSUTSA0)

The MFS Service (MFSRVC) utility (DFSUTSA0) helps to control and maintain MFS intermediate text blocks and control blocks once they have been processed and stored by the MFS Language utility. An intermediate text block (ITB) is a message, format, partition set, or table source language definition stored in the IMS.REFERAL library. A control block is a message or format definition stored in the IMS.FORMAT, IMS.FORMATA, IMS.FORMATB, or IMS.TFORMAT library.

The service utility performs the following functions:

- INDEX creates a special directory for faster access to IMS.FORMAT control blocks.
- DELETE deletes specified contents of the special index directory ($$IMSDIR).
- SCRATCH scratches specified contents of the IMS.FORMAT and IMS.REFERAL libraries and their directories (SCRATCH also operates on IMS.TFORMAT).
- RELATE produces an interpreted listing of the contents of the IMS.REFERAL library.
- LIST produces an interpreted listing of either:
  - The contents of the IMS.FORMAT or IMS.TFORMAT library
  - The contents of the special index directory in the IMS.FORMAT library
  - The contents of the MFS device characteristics table (DFSUDT0x) in the IMS.SDFSRESL library.

The error message `ERR TYPE x IN REFERAL LIBRARY, FUNCT=RELATE` is issued if a problem exists in the MFS REFERAL library. The error types are:

1 Directory block length error

2 User appendage length error

3 Unknown block error

4 Duplicate DUMMY FORMAT error

5 Duplicate FORMAT error

6 TABLE block error

7 Message input block error

8 REFIN OPEN error

9 REFIN OPEN SYNAD taken

A PDB block error

B FORMAT block error

C Message output block error

If an error is detected in the SYSIN, SYSPRINT, or REFIN <REFERAL> file, a return code is set to 4, 8 or 12 respectively.

If an error is detected on the RELATE function, the subsequent functions are ignored.

The following topics provide additional information:

- “Restrictions for DFSUTSA0” on page 504

Restrictions for DFSUTSA0

The following restrictions apply to this utility:

- Do not run the MFS Service utility concurrently with the MFS Language utility (MFSUTL procedure) if they are accessing the same data set.
- Do not run the MFS Service utility concurrently with the IMS online control region if they are both accessing the active format library.

MFSRVC Procedure

This section describes the JCL for the utility, the procedure statement, the EXEC statement, the DD statements, and invoking the procedure.

Figure 114 shows a one-step procedure for maintaining the MFS libraries. It is placed in the IMS.PROCLIB by Stage 2 of system definition.

```
//        PROC DEVCHAR=0,SYS2=,SOUT=A
//MFSRVC  EXEC PGM=DFSUTSA0,REGION=4M,PARM='DEVCHAR=&DEVCHAR'
//STEPLIB  DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//*
//*        PRINT FILES
//*
//SYSPRINT DD SYSOUT=&SOUT
//*           DCB=(RECFM=VBA,LRECL=137)
//SYSSNAP  DD SYSOUT=&SOUT
//*           DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//SYSUDUMP DD SYSOUT=&SOUT
//*
//*        REFERAL LIBRARY
//*
//REFIN    DD DSN=IMS.&SYS2.REFERAL,DISP=OLD
//*
//*        ON-LINE FORMAT LIBRARY
//*
//FORMAT   DD DSN=IMS.&SYS2.FORMAT,DISP=SHR
//*
//*
//*         //SYSIN DD * MUST BE SUPPLIED BY
//*         USER WITH INPUT CONTROL CARD STREAM
//*
//*         ALL DISP=OLD SPECIFICATIONS OF THIS
//*         PROCEDURE ARE REQUIRED .......
//*
//*
```

Figure 114. MFSRVC Procedure

Procedure Statement

The procedure statement must be in the form:

```
DEVCHAR=0,SYS2=,SOUT=A
```

DEVCHAR=
Specifies the device characteristics table. The default is 0.

SOUT=
Specifies the SYSOUT class. The default is A.

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as “Optional Replicate” in an XRF complex. When specified, the operand must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'.

EXEC Statement

The EXEC statement must be in the form:

```
PGM=DFSUTSA0,REGION=250K,PARM='DEVCHAR=&DEVCHAR'
```

REGION=
Specifies the region size for the execution of the MFS Service utility. The default is 4 MB.

PARM=
The PARM= field must be in the form:

```
PARM='DEVCHAR=&DEVCHAR'
```

where &DEVCHAR is the device characteristics table to be listed.

DD Statements

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

SYSPRINT DD
Defines the output message data set. It can be a printer or it can be routed through the output stream. If DISP=(MOD,...) is specified, it can be a tape volume or a direct-access device. The output can be blocked as a multiple of 121.

SYSSNAP
Refers to a data set that is used to receive the output from a SNAP macro if certain severe errors are detected.

Invoking the Procedure

The JCL statements to invoke the MFSRVC procedure are shown in Figure 115.

```
//MFSRVC   JOB  MSGLEVEL=1
//        EXEC  MFSRVC
//SYSIN   DD     *
     END
/*
```

Figure 115. JCL to Invoke the MFSRVC Procedure

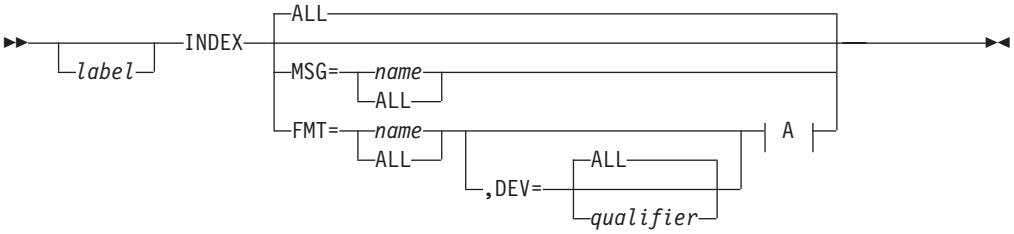
SYSIN DD
Defines the input control statement data sets.

Function Descriptions for DFSUTSA0

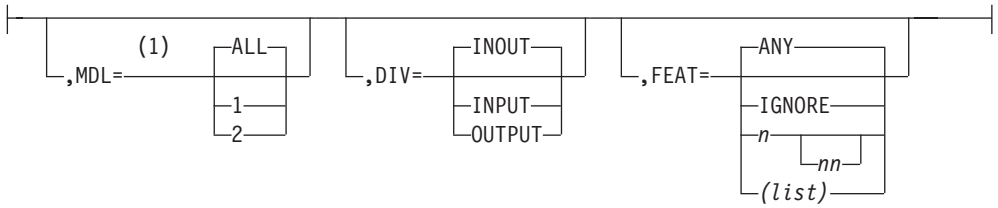
This section includes syntax diagrams and explanations for the INDEX, DELETE, SCRATCH, RELATE, and LIST functions.

INDEX

The INDEX function places the specified names of control blocks (or sets of related control blocks) in a special index directory ($$IMSDIR), that is used to provide quick online access to the control blocks.



A:



Notes:

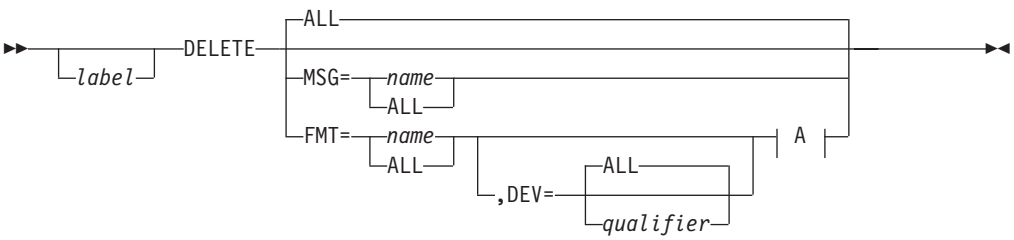
1 MDL=only applies to 3270 and 3270P devices.

If $$IMSDIR has been created by the INDEX function, it is read into the MFS buffer pool during initialization of the online IMS control region and made permanently resident there. If a requested control block is not found in the MFS buffer pool, the MFS pool buffer manager next looks for an entry in $$IMSDIR. The entry, if found, allows the MFS pool manager to issue a direct read for the control block.

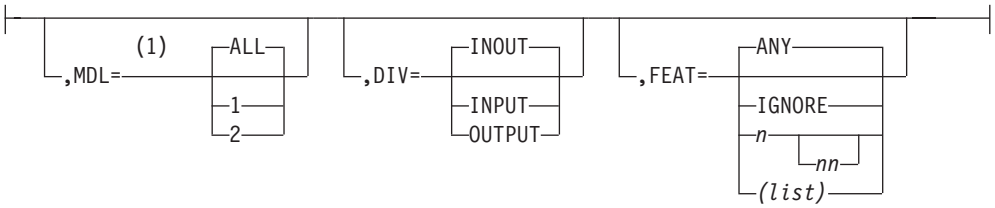
Using the special index directory to replace two direct-access storage reads with one is a performance advantage, but this advantage must be weighed against the storage cost in the online control region (14 bytes per control block indexed in the MFS buffer pool). Indexing only the most frequently used control blocks, which might be a small percentage of the total, can be advisable.

DELETE

The DELETE function specifies the names of a control block, or a set of control blocks, which are to be deleted from the special index directory ($$IMSDIR) used by the online MFS pool manager.



A:

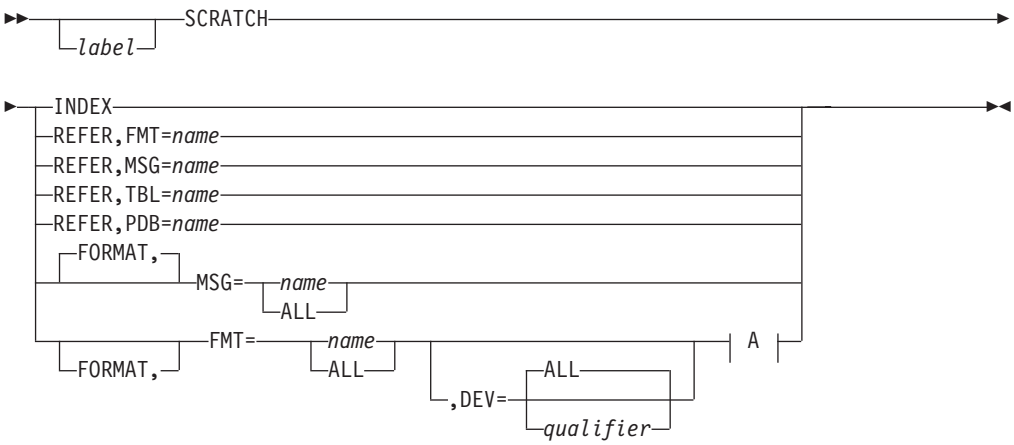


Notes:

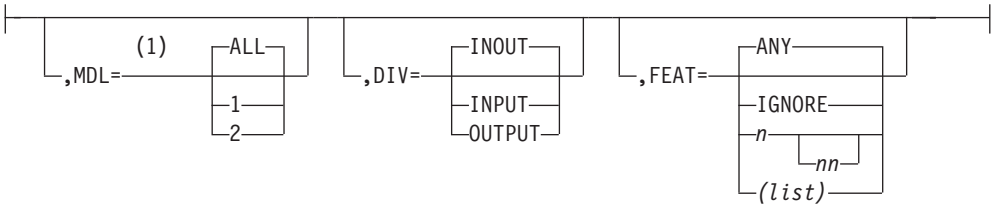
1 MDL=only applies to 3270 and 3270P devices.

SCRATCH

The SCRATCH function scratches a message, format, partition set, or table ITB from IMS.REFERAL. Using this function, you can also scratch a message descriptor, device format, or an index directory from IMS.FORMAT or IMS.TFORMAT.



A:



Notes:

1 MDL=only applies to 3270 and 3270P devices.

Effective use of the SCRATCH function requires an understanding of the relationship between the ITBs in IMS.REFERAL and the control blocks in IMS.FORMAT or IMS.TFORMAT. As shown in Figure 116 on page 508, a format set consists of a format and all associated messages where the SOR= parameter specifies the format as a source. In Figure 116, DFSDF2 is a format that is specified as the source for messages DFSMI2, DFSMO2, and DFSMO3. The ITB form of the format set is stored in IMS.REFERAL; the control block form is stored in

IMS.FORMAT. The format ITB can correspond to a number of device formats for different device types and features.

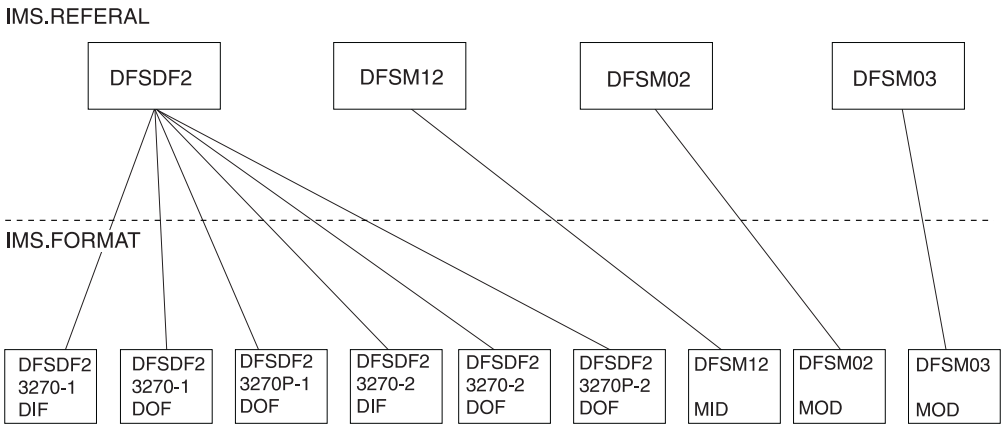


Figure 116. Relationships between ITBs in IMS.REFERAL and Control Blocks for 3270 Format DFSDF2 and Its Format Set

When a control block is scratched from IMS.FORMAT, the action is temporary; that is, the control block is restored to the staging library the next time any member of its format set is processed by the language utility. Thus, if the DFSDF2 DIF for 3270-1 were scratched from IMS.FORMAT, and DFSDF2 or one of its associated messages was later processed by the language utility, the DIF for 3270-1 would be placed in IMS.FORMAT again.

Total elimination of a control block requires the removal of its ITB as well. The 3270-1 control blocks for DFSDF2 could be deleted by recompiling the DFSDF2 format definition without the 3270-1 source included.

Another method is to use the MFS Service utility and scratch all of the DFSDF2 ITBs from the referral data set.

You can scratch the index directory using either DELETE ALL or SCRATCH INDEX.

Exception: The SCRATCH function does not apply to the MFS device characteristics table.

RELATE

The RELATE function provides a listing of all FMT, MSG, PDB, and TABLE ITBs in the IMS.REFERAL library.

FMT and MSG ITBs are related. Following each FMT ITB name, and indented three spaces, are the names of the MSG ITBs which include a SOR= parameter referencing that FMT. The MSG ITB entries indicate whether the message is INPUT or OUTPUT. A FMT ITB entry that does not exist in IMS.REFERAL, but is referred to by one or more MSG ITBs, is designated as **NOT DEFINED to the right of the entry. The PDB and TABLE ITBs are listed following the FMT and MSG ITBs.

Sample Output

Figure 117 shows the contents of IMS.REFERAL and the relationships between the FMT and MSG ITBs. For example, the first entry shows DFSDF1 as a FMT name which has two MSGs associated with it: DFSMI1 (an input MSG) and DFSMO1 (an output MSG).

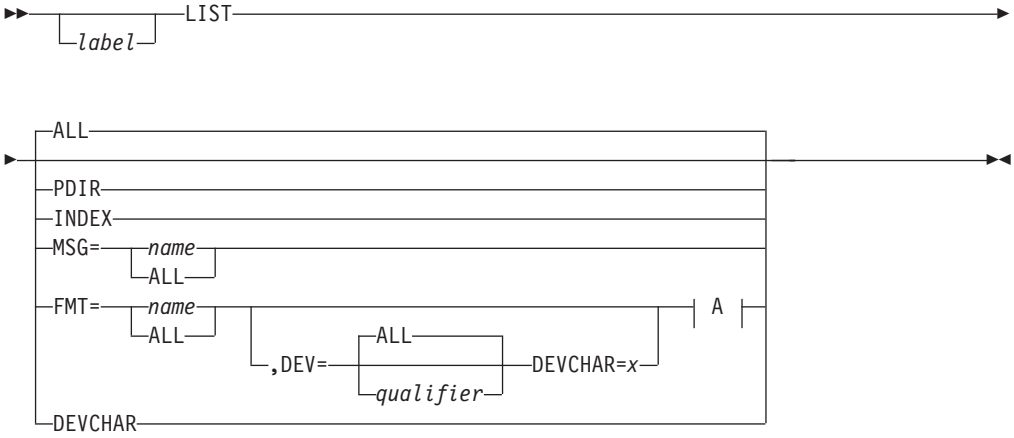
```
   RELATE
DFSDF1
   DFSMI1        INPUT
   DFSMO1        OUTPUT
DFSDF2
   DFSDSPO1      OUTPUT
   DFSMI2        INPUT
   DFSMO2        OUTPUT

   DFSMO3        OUTPUT
   DFSM05        OUTPUT
DFSDF3
   DFSMI4        INPUT
   DFSMO4        OUTPUT
DSFDF5
   DFSMSTRI      INPUT
       DFS1209I  PROCESSING TERMINATED BY EOD ON SYSIN
```

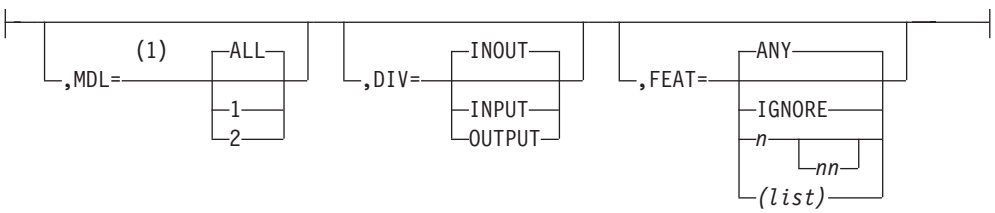
Figure 117. Example of a RELATE Output Listing

LIST

The LIST function provides an interpreted listing of contents of the test library IMS.TFORMAT, the staging library IMS.FORMAT, or the device characteristics table DFSUDT0X.



A:



Notes:

1 MDL=only applies to 3270 and 3270P devices.

For the staging library and the test library you can select the contents to be listed by qualifying the LIST control statement as follows:

ALL
Specifies both the PDS directory and the index directory. The default is ALL.

PDIR
Specifies the PDS directory.

INDEX
Specifies the index directory, $$IMSDIR.

MSG
Specifies the PDS directory entries for one or all message descriptors.

FMT
Specifies the PDS directory entries for one or more device formats.

A special case arises when the following chain of events occurs:

1. The Service Utility is started when the Index Directory ($$IMSDIR) does not exist.
2. The Index Directory is then created by the INDEX function.
3. LIST PDIR or LIST ALL is invoked.

In this case, the output containing the contents of the PDS directory does not include an entry for the Index Directory. This case occurs because the Index Directory is maintained in storage and is not written to the format library until the Service Utility ends.

The output listing contains a line for each control block with the following fields:

NAME
Represents the name of control block specified in the MSG or FMT statement.

TYPE
Represents the type of control block:

DIF
Device input format

DOF
Device output format

MSG
Message input or output control block

DEV
For FMT control blocks, represents the device type to which the control block applies.

MDL
For FMT control blocks, represents the model of the device type to which the control block applies. This field is used with 3270 and 3270P devices.

TTR
Represents the location of the control block in IMS.FORMAT (hexadecimal). Valid only for entries from IMS.FORMAT. Not valid for LIST INDEX.

SIZE
Represents the size of the control block in bytes (hexadecimal).

FEAT
For FMT control blocks, represents the hexadecimal uninterpreted representation of the device type, model, and specific features for which the control block applies. Because this information is appended to the FMT name in the library, this number determines the collating sequence of the output listing.

FEATURES
For FMT control blocks, represents the specific device features for which the control block applies.

Sample Output

Figure 118 shows an example of the LIST function output listing for the control blocks described in Figure 116 on page 508.

```
NAME      TYPE   DEV    MDL TTR        SIZE       FEAT   INTERPRETED FEATURES
-------------------------------------------------------------------------------
DFSDF2    DIF    3270     1  000705     000001AE   007F   IGNORE
DFSDF2    DOF    3270     1  000703     00000279   007F   IGNORE
DFSDF2    DOF    3270P    1  00070B     000001A5   017F   IGNORE
DFSDF2    DIF    3270     2  000709     000000B6   027F   IGNORE
DFSDF2    DOF    3270     2  000707     00000296   027F   IGNORE
DFSDF2    DOF    3270P    2  00070D     00000195   037F   IGNORE
DFSMI2    MSG                00070F     0000005E
DFSMO2    MSG                000711     00000006
DFSMO3    MSG                000713     000000F3
```

Figure 118. Example of a LIST Function Output Listing

LIST the MFS Device Characteristics Table

For the device characteristics table DFSUDT0x created during IMS system definition, the LIST DEVCHAR or DEVCHAR=0 or x, where x is the table suffix, provides an interpreted listing of all the entries of the MFS device characteristics table indicated. The listing shows the device symbolic name of the 3270 or SLU 2 devices, screen size, and features for each entry in the table. If the suffix is not specified in either the DEVCHAR parameter on the LIST statement or the PARM operand of the EXEC statement, table DFSUDT00 is listed. If the suffix is not specified in the DEVCHAR parameter or the LIST statement, but is specified in the EXEC statement, table DFSUDT0x (where x is the table suffix specified in the EXEC statement) is listed. The examples inFigure 119 and Figure 120 on page 512 are provided for the LIST DEVCHAR function.

Example 1

```
//MFSRVC  EXEC  PGM=DFSUTSA0, PARM='DEVCHAR=3'
//SYSIN   DD     *
          LIST  DEVCHAR
          LIST  DEVCHAR=7
```

Figure 119. Example of a LIST DEVCHAR Function with DEVCHAR=

The first LIST statement lists the contents of the device characteristics table DFSUDT03.

The second LIST statement lists the contents of the device characteristics table DFSUDT07.

Example 2

```
//MFSRVC   EXEC   PGM=DFSUTSA0
//SYSIN    DD     *
           LIST   DEVCHAR
           LIST   DEVCHAR=7
```

Figure 120. Example of a LIST DEVCHAR Function

The first LIST statement lists the contents of the device characteristics table, DFSUDT00.

The second LIST statement lists the contents of the device characteristics table, DFSUDT07.

LIST DEVCHAR Output

The output listing contains a line for each entry in the specified device characteristics table, which provides the following fields:

Symbolic Name
Symbolic name defined for the 3270 display in the TYPE= operand of the IMS system definition TYPE or TERMINAL macro.

Screen Size Screen lines and columns of the 3270 display defined in the SIZE= operand of the IMS system definition TYPE or TERMINAL macro.

Device Features
Features specified in the FEAT= operand of the IMS system definition TYPE or TERMINAL macro.

Sample output

Figure 121 is an example of a LIST DEVCHAR output.

```
SYMBOLIC            SCREEN SIZE           DEVICE
  NAME              LINES  COLS          FEATURES
-------------------------------------------------------------
3270-A01             12     80            CARD,PFK,PEN
3270-A02             24     80            IGNORE
3270-A03             32     80            CARD,DEKYBD,PEN
```

Figure 121. Example of a LIST DEVCHAR Output

Utility Control Statements for DFSUTSA0

The control statements used by the MFS Service utility program utilize the same syntax and many of the same keywords as the source statement input to the preprocessor.

Exception: An exception to this occurs when comments are placed on a statement or portion thereof.

Requirement: Because of the extreme range of allowable operations with a FMT block or blocks, the utility requires that comments on a statement be started with /* (for example, LIST ALL /* THIS STATEMENT...).

Related Reading: See “Writing DL/I Calls for Transaction Management” in *IMS Version 9: Administration Guide: Transaction Manager* for a discussion of the meaning and use of keywords.

Positional Parameters

ALL
Where allowed, implies invocation of all functions supported for the associated operator. For example, INDEX ALL implies the insertion of all MID, MOD, DIF, and DOF names that exist in IMS.FORMAT into the special index directory ($$IMSDIR) to be used by the online control region for direct block access.

PDIR
Implies invocation of the associated operation against the PDS directory entries of IMS.FORMAT or, when used with LIST or SCRATCH, the PDS directories of IMS.TFORMAT. For example, LIST PDIR causes the contents of IMS.FORMAT or IMS.TFORMAT to be listed in interpreted format.

INDEX
Directs the invocation of an operation to the special index directory ($$IMSDIR) used by the online control region.

REFER
Directs the invocation of an operation to IMS.REFERAL used by the MFS language utility as an historical intermediate text storage library.

FORMAT
Directs the invocation of an operation to IMS.FORMAT, or, when used with LIST or SCRATCH, to IMS.TFORMAT.

DEVCHAR
Valid only with the LIST function. It causes the device characteristics table identified by the suffix (DEVCHAR=suffix) in the EXEC parameter to be printed. If the EXEC parameter is not specified, the contents of DFSUDT00 are printed.

Keywords

MSG=*name* **| ALL**
Directs the invocation of a function

name
Directs the invocation of a function to a specific message control block, MID or MOD. Name must be specified as a 1- to 8-character alphanumeric value, the first character of which must be alphabetic.

ALL
Directs the invocation of an operation to all message descriptors.

FMT=*name* **| ALL**
Directs the invocation of an operation

name
Directs the invocation of an operation to a specific device format.

ALL
Directs the invocation of an operation to all device formats. Unless further qualified, the operation proceeds against all FMT control blocks in IMS.FORMAT.

The FMT control block can consist of multiple FMT control blocks (DIFs and the DOFs) in IMS.FORMAT and, unless further qualified, the operation proceeds

against all FMT control blocks with the same root name. You can specify "name" as a 1- to 6-character alphanumeric value, the first character of which must be alphabetic.

TBL=name
Directs the SCRATCH function to scratch a TBL ITB from the IMS.REFERAL library. The keyword is valid only on the SCRATCH utility control statement and only if the REFER positional parameter is specified. The names of all the TBLs that reside in IMS.REFERAL can be obtained through the Service Utility RELATE function.

PDB=name
Directs the function to scratch a PDB ITB from the IMS.REFERAL library. The keyword is valid only on the SCRATCH utility control statement and only if the REFER positional parameter is specified. The names of all the PDBs that reside in IMS.REFERAL can be obtained through the Service Utility RELATE function.

DEVCHAR=*x*
Causes the device characteristics table identified by the suffix following the "=" to be printed. This parameter is only valid with the LIST function.

DEV=
Qualifies a specified FMT control block as applying either to a particular device, a secondary logic unit type, or a remote program or to all (ALL).

Specify	Device
3270	3270 or SLU 2 display station
3270-A	For all 3270 or SLU 2 display stations that have been defined during IMS system definition using the device symbolic name.
3270-A1,...,A15	For a single 3270 or SLU 2 display station that has been defined during IMS system definition using the specific device symbolic name.
3270P	3270 printer
274X	2740/2741 terminals
FIN	Finance application program
FIDS	Finance display component (6x40) (for example, 3604-1 or 3604-2)
FIDS3	Finance display component (12x40) (for example, 3604-3)
FIDS4	Finance display component (16x64) (for example, 3604-4)
FIDS7	Finance display component (24x80) (for example, 3604-7)
FIJP	Finance journal printer
FIPB	Finance passbook printer
FIFP	Finance administrative printer
DPM-A	For all SLU P devices that have been defined during system definition using this device symbolic name.
DPM-B	For all ISC nodes that have been defined during system definition using this device symbolic name.
DPM-A1,...,A15	SLU P

DPM-B1,...,B15
ISC nodes

SCS1 The following console keyboard/printers: 3767; NTO; 3771; 3773; 3774; 3775; 3776; 3777; and SLU 1 (print data set) or SLU 4.

SCS2 3521 card punch, 3501 card reader, 2502 card reader; and SLU 1 (transmit data set) or, SLU 4

ALL All specified devices

MDL=1 | 2 | ALL
Determines FMT control block operation.

1 Restricts a FMT control block operation to control blocks for Model 1 3270/3270P stations.

2 Restricts a FMT control block operation to control blocks for Model 2 3270/3270P stations.

ALL
Directs FMT control block operation to control blocks for both Model 1 and Model 2 3270/3270P stations.

This keyword applies only to 3270 and 3270P devices.

DIV=INPUT | OUTPUT | INOUT
Determines FMT control block operation.

INPUT
Restricts a FMT control block operation to DIFs.

OUTPUT
Restricts a FMT control block operation to DOFs.

INOUT
Indicates a FMT control block operation is to proceed for both DIFs and DOFs.

FEAT=IGNORE | n[*nn*] | (list) | ANY
Determines FMT control block operation.

IGNORE
Restricts a FMT control block operation to control blocks for which FEAT=IGNORE was specified on the DEV statement to the MFS language utility.

n[*nn*]
With either:

- A print line of 120, 126, or 132
- User-defined features 1—10

(list)
Restricts a FMT control block operation to control blocks with a specific feature combination. The specifications allowed for “list” are as follows.

For DEV=FIFP:

```
DUAL
132
(DUAL,132)
```

For DEV=3270 or DEV=3270-An:

```
PEN      ,PFK      ,CARD
NOPEN    DEKYBD    NOCD
         NOPFK
```

Enter commas only where required to separate specifications that are actually coded. Feature specifications do not depend on position. You must code at least one alternative. The same feature value results, whether one, two, three, or none of the NOPEN, NOPFK, or NOCD parameters are specified.

ANY
Directs FMT control block operation to all control blocks without restrictions on the feature specification.

Chapter 38. Multiple Systems Verification Utility (DFSUMSV0)

The Multiple Systems Verification utility (DFSUMSV0) verifies the consistency and compatibility of system definitions for IMS systems in a multisystem environment. Use this utility with IMS Multiple Systems Coupling (MSC) when MTM, CTC, or VTAM® is used. The use of this utility points out errors that can prevent the IMS systems from performing properly. If you do not use this utility, you must manually verify the compatibility of system definitions.

Recommendation: The Multiple Systems Verification utility should be run before attempting online executions to verify all defined links and routing paths.

If the MSVERIFY parameter is specified with the SYSTEM keyword on the IMSCTRL macro, only the IMS multisystem control block and verification utility are generated.

Related Reading: See the IMSCTRL macro in *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for the syntax to specify the MSVERIFY parameter.

The following topics provide additional information:

Restrictions for DFSUMSV0

The following restrictions apply to this utility:

- It cannot detect errors associated with Intersystem Communication (ISC). Refer to *IMS Version 9: Administration Guide: System* for information on ISC.
- It does not support CICS.
- It cannot detect errors associated with directed routing.
- It cannot verify compatibility of the log write-ahead option between systems, because the option is specified when IMS is started. From 2 to 255 IMS systems can be verified in any one execution of the verification utility.
- It cannot be used to verify IMS subsystems within a shared queues group. MSC links between IMS subsystems in the same shared queues group are not supported.
- If the remote logical terminals (LTERMs) are incorrectly defined, the utility recognizes and issues error messages only for the remote LTERMs, leaving the local LTERMs in the target area undetected.
- Only MSC descriptors associated with the MSC links defined within the IMS system being initialized are processed by IMS. All other MSC descriptors are ignored.
- When verifying IMS systems with different release levels, use the utility from the latest release level.

Before executing this utility, you must resolve all IMS definition errors in all the multisystem control blocks to be included in the total multisystem configuration. After all system definitions are complete, you must link-edit all the multisystem control blocks from all the IMS multisystem definitions into IMS.SDFSRESL or some other user-specified library. The utility then has access to the multisystem control blocks. For problem determination, assembly listings of all the IMS multisystem control blocks should be available when the utility is executed. Without the assembly listings, it will be difficult for you to resolve inconsistencies and incompatibilities displayed as a result of the multisystem verification process.

Prerequisites for DFSUMSV0

This section describes the prerequisites for using the Multiple Systems Verification utility.

Before the verification utility can be executed, the multisystem control block modules for all systems to be verified must be loaded into IMS.SDFSRESL or some other user-specified library. Figure 122 provides a sample job stream that can load the multisystem control block modules into IMS.SDFSRESL.

```
//STEP1   EXEC PGM=IEWL
              PARM='SIZE=(200K),NCAL,LET,REUS,XREF,LIST'
//SYSPRINT  DD SYSOUT=A
//SYSLMOD   DD DSNAME=IMS.SDFSRESL,DISP=OLD
//SYSUT1    DD UNIT=SYSDA,SPACE=(1024,(100,10))
//SYSLIN    DD *
  PLACE OBJECT MODULES HERE
  NAME DFSMSxxx(R)
/*
```

Figure 122. Sample Job Stream

You must provide two or more object modules. A NAME statement identifying the preceding multisystem control block module (replace *xxx* with the 3-digit control block name suffix) must follow each object deck. If a library other than IMS.SDFSRESL is used, modify the //SYSLMOD statement accordingly.

IMSMSV Procedure

This section describes the JCL for the IMSMSV procedure, the procedure statement, the EXEC statement, and invoking the procedure.

Figure 123 shows an example of the procedure used to execute the Multiple System Verification utility. IMSMSV is created at system definition and is placed in the IMS.PROCLIB by Stage 2 of SYSDEF.

```
//          PROC    DSN='IMS.SDFSRESL',
//                  REG=32K,CLASS=A,ALL=ALL,
//                  UNIT=SYSDA,SER=,DSM='IMS.MODBLKS'
//MSVERIFY EXEC     PGM=DFSUMSV0,PARM='&ALL',REGION=&REG
//STEPLIB  DD       DSN=&DSM,DISP=SHR,UNIT=&UNIT,VOL=SER=&SER
            DD      DSN=&DSN,DISP=SHR,UNIT=&UNIT,VOL=SER=&SER
//SYSOUT   DD       SYSOUT=&CLASS
```

Figure 123. Multiple System Verification Utility Procedure

Procedure Statement

The procedure statement must be in the form:

```
PROC    DSN='IMS.SDFSRESL',
        REG=32K,CLASS=A,ALL=ALL,
        UNIT=SYSDA,SER=,DSM='IMS.MODBLKS'
```

ALL=
Specifies that all messages, including the information message DFS2327I, are to be printed. The default is ALL.

CLASS=
Specifies the SYSOUT class. The default is A.

DSM=
Specifies the data set name of IMS.MODBLKS.

DSN=
Specifies the data set name that contains the verification utility program (DFSMSV00) and its control blocks. The default is IMS.SDFSRESL.

REG=
Specifies the region size for this execution. The default is 32 KB.

SER=
Specifies the volume serial number of the DASD that contains the data set specified by the DSN parameter. SER= need not be specified if the data set is a cataloged data set.

UNIT=
Specifies the STEPLIB UNIT TYPE.

EXEC Statement

The execution statement must be in the form of PGM=DFSUMSV0. The PARM= field must be in the form:

```
PARM='&ALL',REGION=&REG
```

If PARM=ALL is specified in the EXEC statement for the utility execution, the information message DFS2327I is printed as part of the utility output. This message warns you not to do an //ASSIGN of the SYSID/MSNAME to the logical link referenced, because the assignment does not provide a path to the local SYSID at this SYSID level.

Invoking the Procedure

The JCL required to execute the verification utility is shown in Figure 124.

```
//STEP1   EXEC IMSMSV
//SYSIN     DD *
     INPUT FOR IMS MULTISYSTEM VERIFICATION UTILITY
/*
```

Figure 124. JCL Requirements for the MS Verification Utility

Input for DFSUMSV0

This section describes the required input for the Multiple Systems Verification utility.

This utility processes input in two phases:

- Input validation
- Multisystem control block verification

Input Validation

Requirement: The verification utility requires as input one or more control statements within a SYSIN data set. Refer to “Utility Control Statements for DFSUMSV0” on page 524.

After the input has been validated, this phase prints a list of the valid multisystem control block names. The utility then determines if the multisystem control block names are in the IMS.SDFSRESL PDS directory. The utility prints any control block names not found in the directory. If any errors have been detected up to this point, the utility terminates execution with a completion code of 12. If no errors have been found, the utility loads the multisystem control blocks into real storage.

Multisystem Control Block Verification

The utility verifies the following specific portions of each multisystem control block:

- Partner IDs and assigned physical links
- Remote SYSID to Local SYSID Paths
- Remote and local transaction attributes
- Presence of corresponding logical terminals

Partner IDs and Assigned Physical Links

The partner IDs in the logical link definitions are verified to ensure that a partner ID is:

- Not referenced in only one system
- Referenced in only two systems

Each partner ID, as defined with the PARTNER keyword on the MSLINK macro, is checked against every other partner ID in every other multisystem control block. Appropriate messages are printed if any errors are found. Logical links in error are treated as undefined in subsequent steps of the verification process.

When a partner ID is verified and there is also an MSPLINK (physical link) defined for this logical link in both systems, the physical link attributes are verified for type and buffer size. The following are physical link types:

- Real storage-to-real storage
- Channel-to-channel
- VTAM

If any physical link incompatibility is found, the attributes of both physical link definitions are displayed to assist you in determining the error. If the MSPLINK is defined for only one logical link, an information message is printed, indicating that the other end is undefined. In addition to the MSC physical and logical links that are defined to IMS through system definitions, you can identify remote names to IMS through an MSC descriptor. The MSC descriptor relates each remote resource with the link name of a generated MSNAME macro.

Remote SYSID to Local SYSID Paths

The SYSID table entries for a SYSID are verified across all multisystem control blocks for that SYSID number to determine if any path errors exist. A path error is an incomplete path between a multisystem control block in which the SYSID is defined as remote and the multisystem control block in which the SYSID is defined as local.

An incomplete path can occur for the following reasons:

- A SYSID in a multisystem control block for an intermediate system does not contain an address of an MSNAME block (undefined SYSID).
- The MSNAME block is assigned to a logical link block that has a path back to itself without a local SYSID (loop condition).
- The MSNAME block is associated with a logical link block that has an invalid partner ID. The partner ID is invalid if it is not defined in any other multisystem control block or if it is defined in more than one other multisystem control block.
- A SYSID number is defined as local in more than one system.
- A SYSID number is not defined as local in any system.

SYSIDs are scanned in numeric order until all logical link paths are verified. After a path is verified, the utility might display warning messages. These messages identify which logical link numbers this SYSID number-MSNAME should not be assigned to, because an invalid path to the local SYSID would result.

Remote and Local Transaction Attributes

Each multisystem control block is scanned for remote transaction definitions. Each remote transaction definition references a remote and local SYSID. Each remote transaction code is compared with the transaction codes in the system where the remote SYSID is defined as local. If no matching transaction code is found, an error message is printed. If a match is found, the attributes are verified in the two multisystem control blocks.

The following transaction code attributes must be consistent between systems:

- Local
- Remote
- Recoverable
- Nonrecoverable
- Conversational
- Fixed scratch pad area
- Fixed scratch pad area length
- Non-inquiry
- Inquiry
- Single segment
- Multisegment
- Non-Fast Path

If a discrepancy in the transaction attributes occurs, an error message is printed. The attributes specified for the transaction in both systems are displayed.

The return from the local transaction is checked to ensure a path exists out of the local system back to the corresponding remote transaction. If an error exists, a message is printed.

Presence of Corresponding Logical Terminals

Each multisystem control block is scanned for remote LTERM definitions. Each LTERM is associated with an MSNAME block. Each MSNAME block contains remote and local SYSID definitions.

When a remote LTERM is found, the utility checks to ensure that a corresponding LTERM is defined with the same name in the system where the remote SYSID for the MSNAME is defined as local. If no corresponding LTERM definition is found, an error message is printed. If an LTERM is found, verification continues.

For multisystem LTERMs defined during IMS system definition, the remote LTERM definition can be made through the IMS system definition for the remote system or by the Extended Terminal Option (ETO) MSC descriptors for the remote system. When you use ETO MSC descriptors, the remote LTERMs do not exist until the initialization of the ETO feature on the remote IMS system. Therefore, an error message is printed by this utility for the missing remote LTERM indicating that the LTERM might be a dynamic resource.

The utility checks the return path from the destination LTERM to ensure that a path exists out of the local system back to the corresponding remote LTERM. If an error exists, a message is printed.

The return path to that system is the SYSID defined as local in the MSNAME block in the multisystem control block in which the LTERM was defined as remote.

After all verification work is complete, the utility prints a path map as an aid in visualizing the configuration of systems.

Output Messages and Path Map for DFSUMSV0

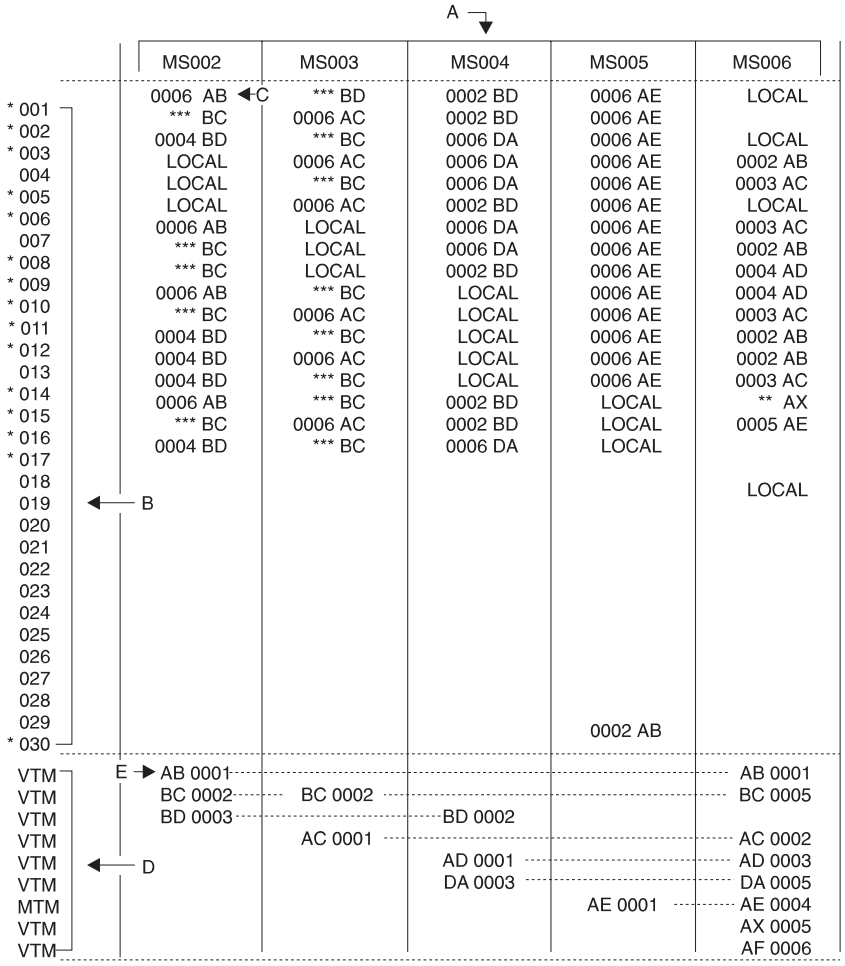
This section describes the utility output and the path map. The verification utility output includes information, warning, and error messages and a path map.

Related Reading: The messages are defined in *IMS Version 9: Messages and Codes, Volume 1*.

Because an error can cause other error conditions, messages with lower numbers should be analyzed and corrected first. The path map is a summary, in matrix format, of the routing paths verified by the utility. It is produced for the first 18 or fewer systems (in ascending sequence by multisystem control block name) being verified. If more than 18 systems are being verified, verification for all systems occurs, appropriate messages for all systems are provided, but the path map is provided only for the first 18 systems.

Figure 125 on page 523 is a sample path map for a configuration of five systems with multisystem control block names of DFSMS002, DFSMS003, DFSMS004, DFSMS005, and DFSMS006. (For the sake of simplicity, these names are used to refer to the specific systems in the rest of this discussion.)

A path map has two sections. The top section relates SYSIDs to specific systems. The bottom section relates partner IDs to logical links between specific systems. The contents of the path map are keyed by letters that are defined in 523.



* AN ERROR WAS DETECTED ON THIS LINE
** NO PARTNER FOR ID
*** MULTI-PARTNERS FOR ID
DFS2399I JOB TERMINATED - RETURN CODE12

Figure 125. Sample Multisystem Path Map

Notes to Figure 125:

Letter	Meaning
A	The top row contains the multisystem control block names (not including the DFS™ prefix) of the systems verified by execution of this utility.
B	The first column of the top section contains all SYSIDs defined in the multisystem configuration. An asterisk preceding the SYSID number indicates an error exists on this line of the matrix.
C	Each entry in the top section relates the SYSID (column B) to the multisystem control block name (row A). • Most entries contain the 4-digit suffix of the multisystem control block name of the system defined as logically linked to the system (row A) and the 2-character partner ID defined for this logical link. • A blank entry, such as SYSID 0002 for DFSMS006, indicates this SYSID was not defined for this system. All entries for SYSIDs

0020 through 0029 are blank; this means these SYSIDs were not specified in any of the multisystem control blocks.

- An entry specifying LOCAL, such as SYSID 0004 for DFSMS002, identifies this system as the system in which this SYSID is defined as local.
- Errors are identified by asterisks. One asterisk preceding the SYSID indicates that one or more errors were found for this printed line. If the suffix portion of an entry is replaced by two asterisks (**), such as SYSID 0015 for DFSMS006, the verification utility found no partner for this system. Three asterisks (***) indicate that more than two partners were found, such as SYSID 0002 for DFSMS002. If a SYSID is defined as local for more than one system, the printed line is identified by one asterisk and more than one entry on that line specifies LOCAL. SYSID 0006 contains an example of this error.

D The first column of the bottom section contains the physical link type:

CTC	Channel-to-channel adapter
MTM	Real storage-to-real storage
VTM	ACF/VTAM session type 6

This column entry is either blank, if no physical link is defined, or assigned depending on the first physical link encountered, for the logical link identified by E. An asterisk preceding the link type (or alone, if no physical link is defined) indicates an error exists for this printed line.

E Identifies the logical link in terms of partner ID and relative logical link number. The partner IDs relate directly to those in the top part of the chart.

The verification utility relates partner systems by connecting them with a dashed line.

In Figure 125 on page 523, partnerships BC and AX are in error. These errors were identified in the top part of the chart but are more clearly demonstrated in the bottom part. Partner ID BC has more than two definitions; AX has just one definition.

Utility Control Statements for DFSUMSV0

This section describes and provides examples of the statement input.

The control statements must contain the 1- to 3-digit suffix supplied on the MSVID keyword of the IMSCTRL macro. Each control statement can contain one or more such suffixes, specified in any sequence. The input statement scan ends when a blank position is encountered; if position 1 is blank, the input statement is treated as a comment statement. If more than one suffix is specified in a control statement, each suffix following the first one must be separated from the preceding suffix by a comma. Only the significant digits of a suffix need be specified.

Each suffix in a control statement must be complete in that statement and cannot be continued in the next control statement.

Sample input data:

- 1,255,6,009,80,02,198 are valid.
- 0,677,040,NYC,1A,0001,5,5 are invalid.

Each invalid entry is printed, and the type of error is identified.

For example:

0	Is not in the range from 1 to 676.
677	Is not in the range from 1 to 676.
0040	Is more than 3 digits.
NYC	Is nonnumeric.
5,5	Is duplicate input data.

Valid input for three systems:

```
STATEMENT    1,5,255
   or
STATEMENT    001,005,255
   or
STATEMENT    255,01,5

   or

STATEMENT  1   DFSMS001  NYC
   and
STATEMENT  255 DFSMS255  LA
   and
STATEMENT  05  DFSMS005  CHICAGO
```

Error Processing for DFSUMSV0

This section explains the condition codes returned by the utility.

These condition codes are:

Code	Meaning
0	Only information and warning messages are printed
12	Errors detected that must be resolved before multisystem execution

Chapter 39. Spool SYSOUT Print Utility (DFSUPRT0)

When a communication line is defined for Spool SYSOUT during system definition, the Spool SYSOUT Print Utility (DFSUPRT0) copies messages produced by the online control program from its set of data sets to a system output device. Both the spool data sets and the system output device are processed using QSAM. Blocking factors for spool data sets are determined by the online control program. System output device blocking can be specified through JCL on the SYSPRINT DD statement.

The following topics provide additional information:

Restrictions for DFSUPRT0

The Spool SYSOUT Print utility does not support CICS.

Input and Output for DFSUPRT0

This section describes the input and output for the utility.

When the print utility is started, it should complete before a `/START LINE` command is issued to make the Spool SYSOUT available.

Output from the print utility includes a page of status information, followed by the contents of the spool data sets indicated as FULL and printed in chronological sequence.

DFSWTnnn Procedure for DFSUPRT0

This section explains the JCL for the DFSWTnnn procedure, the procedure statement, the EXEC statement, and the DD statements.

The DFSWTnnn procedure, shown in Figure 126 on page 528, executes the Spool SYSOUT Print utility program (DFSUPRT0) as an online program for printing data sets created by the Spool SYSOUT option during system definition. The utility program copies messages produced by the online control program from its set of data sets to a system output device. This procedure is created at system definition and is placed in the IMS.PROCLIB procedure library by Stage 2 of SYSDEF. The *nnn* is supplied by system definition.

```
//       PROC  SOUT=A,RGN=30K,SYS1=,SYS2=
//PRINT  EXEC  PGM=DFSUPRT0,REGION=&RGN
//STEPLIB  DD  DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1410
//SYSUDUMP DD SYSOUT=&SOUT
//SPOOLn  DD  DISP=SHR,DSN=IMS.&SYS1.SYSO1
//SPOOLn  DD  DISP=SHR,DSN=IMS.&SYS1.SYSO2
//SPOOLn  DD  DISP=SHR,DSN=IMS.&SYS1.SYSO3
```

Figure 126. DFSWTnnn Procedure

Procedure Statement

This statement must be in the form:

```
PROC  SOUT=A,RGN=30K,SYS1=,SYS2=
```

SOUT=
Specifies the class assigned to SYSOUT DD statements.

RGN=
Specifies the size of the z/OS region to be allocated to the IMS control program.

SYS1=
Specifies an optional second level dsname qualifier for those data sets which are designated as “Mandatory Replicate” in an XRF complex. When specified, the operand must be enclosed in quotes and must include a trailing period; for example, SYS1='IMSA.'.

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as “Optional Replicate” in an XRF complex. When specified, the operand must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'.

EXEC Statement

This statement can be in the form:

```
EXEC   PGM=DFSUPRT0
```

or it can specify a procedure that contains the required JCL statement. A region size of 30KB is usually adequate for execution.

DD statements

STEPLIB DD
Defines the library containing the print utility. This DD statement is usually DSNAME=IMS.SDFSRESL,DISP=SHR.

SYSPRINT DD
Defines the system output device to which output is directed. The record format is VBM. If either no block size or a block size less than 141 is specified, the default block size of 141 is assumed. Any block size valid for QSAM and greater than 141 can be specified. Any logical record length can be specified. If no logical record length is specified, the default is 4 less than the block size specified (137 if block size default of 141 is used).

SPOOLnn DD
Describes the spool data set to be printed (where nn is any valid alphanumeric

identifier). This DD statement is normally DSNAME=IMS.SYSnn, where 'nn' is assigned by system definition. DCB information either should not be coded or, if coded, must specify RECFM=VBM.

IMSWTnnn Job for DFSUPRT0

This section explains the job used to print the data sets created by the utility.

To invoke the DFSWT*nnn* procedure, use a IMSWTnnn job. These jobs print data sets created by the Spool SYSOUT options.

IMSWT*nnn* member job class and message class are determined by the MAXREGN keyword specified on the IMSCTRL macro statement during system definition.

This job executes procedure DFSWT*nnn*, which invokes the Spool SYSOUT utility program (DFSUPRT0) for printing the Spool SYSOUT data set.

```
//SPRT0 JOB 1,IMS,CLASS=A,MSGCLASS=A,MSGLEVEL=1
//        EXEC DFSWT000
```

This job must be copied to the IMS.JOBS data set to run.

Sample Output

Figure 127 shows an example of the Spool SYSOUT Print utility output.

```
    DFSUPRTO - SYSOUT PRINT UTILITY    TIME   9:35:1   DATE    92.056

DDNAME        STATUS      CREATED TIME          DATE       DATASET NAME
SPOOL1         FULL         9:33:239           92.056      IMSTESTL.IMS01.SPOOL1
SPOOL2         INUS          :00:000            0.000      IMSTESTL.IMS01.SPOOL2
SPOOL3         AVAL          :00:000            0.000      IMSTESTL.IMS01.SPOOL3
```

Figure 127. Example of a Spool SYSOUT Print Utility Output

The fields in the report have the following meaning:

DDNAME The user-provided DDNAME

STATUS FULL—if data set is to be printed

INUS—if being filled online

AVAL—if not being used

CREATED TIME
Time of creation (24-hour clock) (HH:MM:SST)

DATE Julian date of creation (YY.DDD)

DATASET NAME
The DSNAME of the assigned data set

System messages included in a spool data set always have unprintable control characters (typically the new-line symbol, X'15').If a UCS printer is used as a SYSOUT device, these messages might print as extraneous alphabetic characters (if fold-mode operation is specified in response to the UCS parameter request).

Error Processing for DFSUPRT0

This section explains the condition codes returned by the utility.

These condition codes are:

Code	Meaning
0	Successful completion
4	No data sets allocated for printing
8	SYSPRINT DD statement missing
12	I/O error on SYSPRINT data set

Chapter 40. Time-Controlled Operations Verification Utility (DFSTVER0)

The Time-Controlled Operations (TCO) Verification utility (DFSTVER0) ensures error-free TCO script members. You should run the verification utility before you execute any script member online. The utility detects any script member error that would be detected during online execution.

Exception: Errors caused by insufficient storage are not detected.

You must add TCO script members to the TCO script library before executing the verification utility. You can verify more than one member at a time by assigning an input control statement for each member you are verifying.

The verification utility generates reports for the following:

- Errors
- Statistics
- Timer elements (time-schedule requests)
- Messages
- Summaries

Related Reading: See "Tools for IMS Operations and Recovery" in *IMS Version 9: Operations Guide* for more information on time-controlled operations.

The following topics provide additional information:

- "TCO Verification Procedure"
- "Output for DFSVER0" on page 533
- "Return Codes for DFSVER0" on page 535

TCO Verification Procedure

This section explains the JCL for the utility, the EXEC statement, and the DD statements.

Figure 128 on page 532 is a sample script member (DFSTCF10). Figure 129 on page 532 shows the JCL for the TCO Verification utility procedure for Figure 128 on page 532.

```
/BRO LTERM CTRL
DFSTCF10 LOADED.                                                    00001500
*TIME      DFSTXITB           S                      ****          00001600
/ASS LTERM LOG27403  TO LINE 31 PTERM 1  ;                          00001700
/START LINE  2 PTERM ALL;                                           00001800
/START LINE 26 PTERM ALL;                                           00001900
/START LINE 18 PTERM ALL;                                           00002000
/STA DB MSDBLM01,MSDBLM02,MSDBLM03,MSDBLM04,MSDBLM05;               00002100
/STA DB MSDBLM06,MSDBLM07,MSDBLM08;                                 00002200
*TIME      DFSTXITB           S                      ****          00002300
/START REGION MSDBMTX3;                                             00002400
/START REGION MSDBMTY3;                                             00002500
/START REGION MSDBMTZ1;                                             00002600
*TIME      DFSTXITB           S                      ****          00002700
PTERM01 BEGIN PTERM1;                                               00002800
PTERM03 BEGIN PTERM3;                                               00002900
/STOP REGION 1;                                                     00003000
*TIME      DFSTXITB           S                      ****          00003100
DFSTCF  LOAD DFSTCF1A;                                              00003200
*TIME      DFSTXITB      0004 S                      ****          00003300
*TIME      DFSTXTIB      0004 S                      ****          00003400
/*
```

Figure 128. Sample Script Member (DFSTCF10)

Figure 129 is an example of the verification of members DFSTCF01, DFSTCF02, and DFSTCF10.

```
//*
//*       THIS JCL IS USED TO VERIFY THE USER SUPPLIED SCRIPTS
//*
//  EXEC  PGM=DFSTVER0
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSTCF   DD DSN=IMS.TCFSLIB,DISP=SHR
//SYSIN    DD *
DFSTCF01 CONT 5
DFSTCF02 CONT 20
DFSTCF10      (ONE OR MORE CARDS SPECIFYING MEMBER NAMES TO BE VERIFIED)
/*
```

Figure 129. TCO Verification Utility Procedure

In this case, the three script members DFSTCF01, DFSTCF02, and DFSTCF10 are in IMS.TCFSLIB, referred to in the DFSTCF DD statement. The verification utility, DFSTVER0, is found in IMS.SDFSRESL, referred to in the STEPLIB DD statement. The reports generated by the verification utility are sent to the device you assign for class A output.

EXEC Statement

The TCO Verification utility is executed as a standard z/OS job. The following are required:

- A JOB statement defined by you
- An EXEC statement
- DD statements

The region size for execution of the utility is acquired in 4 KB increments for time-schedule request sets and message sets. Each run of the utility, therefore,

causes 8 KB to be acquired. Each timing element uses 32 bytes of the 4 KB of storage. The amount of storage messages use varies, depending on the number of segments you specify. In the example job, more time-schedule requests and messages could be added without increasing the storage.

EXEC
Must be in the form:

```
//  EXEC   PGM=DFSTVER0
```

DD Statements

STEPLIB DD
Points to IMS.SDFSRESL, where the TCO modules reside.

SYSPRINT DD
Describes the output data set that contains the reports the verification utility generates.

SYSUDUMP
Defines the dump data set.

SYSIN DD
Describes the input control data set, which contains the 80-character control statements. The TCO script member names you are verifying with the utility are in columns 1 through 8 of each statement.

The CONT keyword with its parameter can be used to change the size of a message segment; the default segment continuation count is 9. The CONT parameter is a 1- or 2-digit number between 1 and 99 that indicates the new segment continuation count for that particular script.

DFSTCF DD
Points to IMS.TCFSLIB, where the TCO script members reside. You can name the data set TCFSLIB or any other valid dsname.

Output for DFSVER0

This section explains and gives examples of the reports generated by the TCO Verification utility.

Error Report

Figure 130 is an example of an error report generated by the TCO Verification utility. In the report, the statement sequenced 00003400 (in the DFSTCF10 script member) had a misspelled exit routine name. This exit routine could not be found in the IMS.SDFSRESL library, so it is reported as an error here. The statement is eliminated from the time-schedule request table, which is in the timer elements report.

```
ERROR REPORT FOR MEMBER DFSTCF10

DFS3360E USER EXIT DFSTXTIB REQUESTED NOT FOUND, SEQUENCE NUMBER= 00003400
```

Figure 130. TCO-Verification-Error Report

Statistics Report

Figure 131 shows an example of a statistics report generated by the TCO Verification utility. The only exit routine specified by any time request statement (that was found in IMS.SDFSRESL) is DFSTXITB.

```
STATISTICS REPORT FOR MEMBER DFSTCF10

PROGRAM EXITS REQUIRED IN IMS.SDFSRESL

DFSTXITB
```

Figure 131. TCO-Verification-Statistics Report

Timer-Elements Report

Figure 132 shows an example of a timer-elements report generated by the TCO Verification utility.

```
TIMER ELEMENTS REPORT

TIME OF ACTIVATION EXIT CALLED  ATTRIBUTES   PARM

 STARTUP              DFSTXITB   RES        **** MSG SET
 STARTUP              DFSTXITB   RES        **** MSG SET
 STARTUP              DFSTXITB   RES        **** MSG SET
 STARTUP              DFSTXITB   RES        **** MSG SET
 0957                 DFSTXITB   RES SNGL   **** MSG SET
```

Figure 132. TCO-Verification-Time-Elements Report

The columns in the report are as follows:

Time of Activation
Indicates either STARTUP or the time the time request is to be processed if this is an actual run.

Exit Called
Indicates the exit to be called for this time request.

Attributes
The following attributes are possible:

RES
Indicates a resident exit routine.

DYN
Indicates a dynamically loaded exit routine.

CONT
Indicates execution each day at the same time.

SNGL
Indicates a single execution.

PARM
Indicates either ****MSG SET, which indicates a message set for this time request, or the 16 bytes of data specified in columns 56 through 71 for this time request in the script member.

Message-Table Report

Figure 133 is an example of a message-table report generated by the TCO-Verification utility. The report indicates that the DFSTCF10 script member has five message sets. The asterisk in column 1 indicates a new message. Each message set is separated by a blank line.

```
 MESSAGE TABLE REPORT

 EACH LINE IS A SEGMENT
 * IN COLUMN 1 SIGNIFIES START OF NEW MESSAGE
 * IN COLUMN 121 SIGNIFIES SEGMENT IS TRUNCATED

 */BRO LTERM CTRL
  DFSTCF10 LOADED.

 */ASS LTERM LOG27403  TO LINE 31 PTERM 1  ;
 */START LINE  2 PTERM ALL;
 */START LINE 26 PTERM ALL;
 */START LINE 18 PTERM ALL;
 */STA DB MSDBLM01,MSDBLM02,MSDBLM03,MSDBLM04,MSDBLM05;
 */STA DB MSDBLM06,MSDBLM07,MSDBLM08;

 */START REGION MSDBMTX3;
 */START REGION MSDBMTY3;
 */START REGION MSDBMTZ1;

 *PTERM01 BEGIN PTERM1;
 *PTERM03 BEGIN PTERM3;
 */STOP REGION 1;

 *DFSTCF  LOAD DFSTCF1A;
```

Figure 133. Example of a Message-Table Report

A message set is composed of one or more messages. A message set is either single-segment or multi-segment. In the sample report:

- The first message set is a single multi-segment message.
- The second, third, and fourth message sets are multiple single-segment messages.
- The last message set is a single single-segment message.

Summary Report

Figure 134 is an example of a summary report generated by the TCO-Verification utility. The summary report lists the number of time-schedule requests and the number of messages found in a script member. It also summarizes the number of exit routines specified in the time-schedule requests in the member being verified and the amount of storage used.

```
SUMMARY REPORT

# ELEMENTS  # MSGS  #EXIT ROUTINES  STORAGE SIZE
  00005      00014      00001          08324
```

Figure 134. TCO-Verification-Summary Report

Return Codes for DFSVER0

This section explains the return codes for the TCO Verification utility.

The TCO Verification utility returns a code that indicates the verification processing status. These return codes are:

Code	Meaning
0	No error found in the scripts being verified. The five output reports are generated for each script.
4	Error in the CONT parameter that was specified on the verification JCL.
6	Syntax errors in the script.
8	One of the following errors occurred: • Unable to get storage • Unable to open SYSIN data set • Unable to open SYSPRINT data set • No DFSTCF DD statement in Verification JCL
10	Error in I/O.
12	Script member not found.
14	Unable to open the script members data set.

If a return code greater than zero is received from the utility, one or more of the scripts being verified has errors.

Verification utility reports are issued for reports that have no errors. Only the error report is issued for scripts that have errors. However, depending on the type of error, it is possible that no error report is generated.

Part 9. Appendixes

Appendix A. Summary of DEDB Utility Commands

Table 31 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 31. Summary of DEDB Utility Commands

COMMAND	OPERATOR	SCAN UTILITY	DELETE UTILITY	REORG UTILITY	CREATE UTILITY	COMPARE UTILITY
ALLFMNEW		O	O	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
EXPANDSEG		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
STARTHEXT		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
STOPAFTERFAIL#		N/A	N/A	O	N/A	N/A
STOPHEXT		O	O	N/A	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
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.

Operands of the scan and delete utilities are area specific. If multiple areas are processed in each run, keywords such as QUITCI, V5COMP, and EXCLUDE must be specified for each area. The applicable value for each of these operands must be coded before each GO command for each AREA command. For example, if V5COMP function is desired for all areas in a utility run, 'V5COMP' must be coded before each GO command for each AREA command.

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 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 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 + 12
```

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 + 12.

See “How the Reorganization Utility Uses the BUFNO Command” on page 175 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 $.

EXPANDSEG
Loads a copy of the compression exit, control block, and dataset with a direct link to user data.

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 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 can 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, except the partner currently owning the high water mark CI, 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.

STARTHEXT
Specifies a start time in hex format.

You must supply 16 hex digits matching the storeclock value desired. The syntax for STARTHEXT is:

```
STARTHEXT x '0123456789abcdef'   /* 16 hex digits required.  */
```

STARTIME
Specifies a specific start time for selecting valid SDEP segments. Used for SCAN.

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.

STARTUOW
Specifies the first unit of work (UOW) to reorganize. This command is optional and is used with the reorganization utility.

STOPAFTERFAIL#
Specifies the number of UOWs that failed to be reorganized due to a lack of available buffers before the utility is abnormally terminated. Maximum value is 999.

If STOPAFTERFAIL# is not specified, the default is taken. The default value is 5. If STOPAFTERFAIL# is specified as 0, then no limit should be imposed and the utility should continue to attempt to reorganize the next UOW until it has reached the end of the DEDB area. Up to 999 failed UOWs will be listed.

STOPHEXT
Specifies a stop time in hex format.

You must supply 16 hex digits matching the storeclock value desired. The syntax for STOPHEXT is:

```
STOPHEXT x '0123456789abcdef'   /* 16 hex digits required.  */
```

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
Specifies 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 32 and Table 33 on page 545 show these abbreviations and synonyms.

Table 32. DEDB Online Utility Command Abbreviations and Synonyms

Commands	Abbreviations/Synonyms
ALLFMNEW	AFN
AREA	A, AREANAME
BUFNO	BUFFERNUMBER, BUFFNO, BUFN

Table 32. DEDB Online Utility Command Abbreviations and Synonyms (continued)

Commands	Abbreviations/Synonyms
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
STARTUOW	FIRSTUOW, FIRSTW, FUOW, FW, STARTW
STOPAFTERFAIL#	SAF#
STOPTIME	
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 33. 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

The Index/ILDS Rebuild utility is needed at the active site (which was once the tracking site) for HALDBs following an RSR takeover (to recover the ILDS and index data sets).

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 HD 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:

  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 can 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:

- Timestamp recovery of one DBDS of a tracked database requires a corresponding timestamp 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 timestamp recovery. Specifically, the log volume boundary requirements no longer apply. Thus, you can use the `NOFEOV` keyword on the `/DBRECOVERY` command when preparing for timestamp 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 HALDB master 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 HALDB master 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), timestamp 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 timestamp recovery.

  For block-level data sharing subsystems tracked at the database readiness level (DLT), timestamp 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.

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 can 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 HD 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 can 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.

Appendix C. Summary of HALDB Utilities

Table 34 lists all of the databases utilities that can be used for HALDBs.

Table 34. Utilities That Can Run Against HALDBs

Utility	Description	Comment
DFSMAID0	HALDB Migration Aid	
DFSPREC0	HALDB Index/ILDS Rebuild	
DFSUPNT0	HALDB Partition Data Set Initialization	
%DFSHALDB	HALDB Partition Definition	Invocation of the utility by a c-list. %DFSHALDB is the TSO invocation of module DSPXPDDU.
DFSURUL0	HISAM Reorganization Unload	
DFSURRL0	HISAM Reorganization Reload	
DFSURGU0	HD Reorganization Unload	Applies to PHDAM, PHIDAM, and PSINDEX
DFSURGL0	HD Reorganization Reload	Applies to PHDAM, PHIDAM, and PSINDEX
DFSURPR0	Prereorganization	
DFSUDMP0	Image Copy	
DFSUICP0	Online Image Copy	
DFSUDMT0	Database Image Copy 2	
DFSUCUM0	Change Accumulation	
DFSURDB0	Database Recovery	
DFSBBO00	Batch Backout	

Database Recovery Control (DBRC) is required for execution of any utility operating on a HALDB. Each utility checks for the presence of DBRC. If DBRC is not present, the utility issues an error message and terminates.

Image copy utilities reject any attempt to image copy HALDB ILDSs or PHIDAM prime index data sets. Recovery utilities reject any attempt to recover HALDB ILDSs or PHIDAM prime index data sets. Both image copy and recovery utilities can only run against a particular data set of a HALDB partition.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This information 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 topic or by the following marking: General-use Programming Interface and Associated Guidance Information.

End of General-use programming interface

This book also documents Product-sensitive Programming Interface Information provided by IMS.

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of IMS. Use of such interfaces creates dependencies on the detailed design or implementation of IMS. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces might need to be changed to run with new product releases or versions, or as a result of service.

Product-sensitive programming interface

Product-sensitive Programming Interface Information is identified where it occurs, either by an introductory statement to a topic, by footnotes in tables, or by the following marking: Product-sensitive Programming Interface Information.

End of Product-sensitive programming interface

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- BookManager
- CICS
- DataPropagator
- DB2
- DFS
- DFSMSdss
- DFSORT
- Enterprise Storage Server
- IMS
- MVS
- NetView
- OS/390
- RAMAC
- Tivoli
- VTAM
- WebSphere
- z/OS

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

This bibliography lists all of the information in the IMS Version 9 library.

DFSMS Access Method Services for Catalogs, SC26-7394

DFSORT Application Programming Guide, SC33-4035

IBM 3990 Storage Control Reference (Models 1, 2, and 3), GA32-0099

z/OS DFSMSdss Diagnosis Guide, LY35-0116

z/OS DFSMSdss Storage Administration Guide, SC35-0423

z/OS DFSMSdss Storage Administration Reference, SC35-0424

z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589

IMS Version 9 Library

Title	Acronym	Order number
IMS Version 9: Administration Guide: Database Manager	ADB	SC18-7806
IMS Version 9: Administration Guide: System	AS	SC18-7807
IMS Version 9: Administration Guide: Transaction Manager	ATM	SC18-7808
IMS Version 9: Application Programming: Database Manager	APDB	SC18-7809
IMS Version 9: Application Programming: Design Guide	APDG	SC18-7810
IMS Version 9: Application Programming: EXEC DLI Commands for CICS and IMS	APCICS	SC18-7811
IMS Version 9: Application Programming: Transaction Manager	APTM	SC18-7812
IMS Version 9: Base Primitive Environment Guide and Reference	BPE	SC18-7813
IMS Version 9: Command Reference	CR	SC18-7814
IMS Version 9: Common Queue Server Guide and Reference	CQS	SC18-7815
IMS Version 9: Common Service Layer Guide and Reference	CSL	SC18-7816
IMS Version 9: Customization Guide	CG	SC18-7817
IMS Version 9: Database Recovery Control (DBRC) Guide and Reference	DBRC	SC18-7818
IMS Version 9: Diagnosis Guide and Reference	DGR	LY37-3203
IMS Version 9: Failure Analysis Structure Tables (FAST) for Dump Analysis	FAST	LY37-3204
IMS Version 9: IMS Connect Guide and Reference	CT	SC18-9287
IMS Version 9: IMS Java Guide and Reference	JGR	SC18-7821
IMS Version 9: Installation Volume 1: Installation Verification	IIV	GC18-7822
IMS Version 9: Installation Volume 2: System Definition and Tailoring	ISDT	GC18-7823
IMS Version 9: Master Index and Glossary	MIG	SC18-7826
IMS Version 9: Messages and Codes, Volume 1	MC1	GC18-7827
IMS Version 9: Messages and Codes, Volume 2	MC2	GC18-7828
IMS Version 9: Open Transaction Manager Access Guide and Reference	OTMA	SC18-7829
IMS Version 9: Operations Guide	OG	SC18-7830
IMS Version 9: Release Planning Guide	RPG	GC17-7831
IMS Version 9: Summary of Operator Commands	SOC	SC18-7832
IMS Version 9: Utilities Reference: Database and Transaction Manager	URDBTM	SC18-7833
IMS Version 9: Utilities Reference: System	URS	SC18-7834

Supplementary Publications

Title	Order number
IMS Connector for Java 2.2.2 and 9.1.0.1 Online Documentation for WebSphere Studio Application Developer Integration Edition 5.1.1	SC09-7869
IMS Version 9 Fact Sheet	GC18-7697
IMS Version 9: Licensed Program Specifications	GC18-7825

Publication Collections

Title	Format	Order number
IMS Version 9 Softcopy Library	CD	LK3T-7213
IMS Favorites	CD	LK3T-7144
Licensed Bill of Forms (LBOF): IMS Version 9 Hardcopy and Softcopy Library	Hardcopy and CD	LBOF-7789
Unlicensed Bill of Forms (SBOF): IMS Version 9 Unlicensed Hardcopy Library	Hardcopy	SBOF-7790
OS/390 Collection	CD	SK2T-6700
z/OS Software Products Collection	CD	SK3T-4270
z/OS and Software Products DVD Collection	DVD	SK3T-4271

Accessibility Titles Cited in This Library

Title	Order number
z/OS V1R1.0 TSO Primer	SA22-7787
z/OS V1R5.0 TSO/E User's Guide	SA22-7794
z/OS V1R5.0 ISPF User's Guide, Volume 1	SC34-4822

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Program Number: 5655-J38

Printed in USA

SC18-7833-00

Spine information:

IMS Utilities Reference: Database and Transaction Manager Version 9